O‘ZBEKISTON RESPUBLIKASI OLIY VA O‘RTA MAXSUS TA’LIM VAZIRLIGI

MUHAMMAD AL-XORAZMIY NOMIDAGI TOSHKENT AXBOROT TEXNOLOGIYALARI UNIVERSITETI

Tursunov Nurmamat Xolmatovich

Ergashev Abdunabi Qarshibayevich

JAVA TILIDA OB’EKTGA YO‘NALTIRILGAN DASTURLASH

FANIDAN

O‘QUV QO‘LLANMA

Ta’lim sohasi: 300000 - Komputer texnologiyalari va informatika

Ta’lim yonalishlari: 5330500 – Komputer injiniring (“Komputer injiniring”, “IT-servis”, “Multimediya texnologiyalari”)
5330600 – Dasturiy injiniring
5330200 – Informatika va axborot texnologiyalari

TOSHKENT-2018

Mualliflar: Tursunov N.X., Ergashev A.Q. “Java tilida ob’ektga yo‘naltirilgan dasturlash” o‘quv qo‘llanma // TATU. Toshkent, 2018. 146 b.

O‘zbekiston Respublikasida innovatsion g‘oyalarnng keng tadbiq etilishi mamlakatimiz iqtisodini rivojlantirishning muhim omillaridan hisoblanadi. Turli yo‘nalishlarda dasturiy maxsulotlar ishlab chiqarish, ishlab chiqarilgan mahsulotlarning amalda joriy etilishi, birinchidan, dasturiy mahsulotlarning iqtisoddagi ulushini oshiradi, ikkinchidan, ishlab chiqarish, boshqarish va boshqa sohalarni zamonaviylashgan xalqaro standartlar asosida yuritish imkonini beradi. Bu o‘z navbatida AT sohasida faoliyat yuritishi kerak bo‘ladigan mutaxassislarga eng ilg‘or, zamonaviy dasturlash vositalarini egallash talabini qo‘yadi.

Xozirgi paytda eng samarali ishlayotgan dasturlash tillaridan biri bu Java dasturlash tilidir. Java dasturlash tilini chuqur o‘zlashtirish va uning imkoniyatlarini turli sohalarga qo‘llay bilish AT mutaxassislarning mehnat bozoridagi mavqiyesini keskin oshiradi. Zamonga mos, raqobatbardosh mutaxassis bo‘lib yetishish imkonini beradi.

Ushbu o‘quv qo‘llanmada aynan shu masalalarga e’tibor berilgan. Qo‘llanmada Java dasturlash tilining o‘ziga xos xususiyatlari, yuqori darajadagi boshqa dasturlash tillaridan farqi va bu dasturlash tilining ma’lumotlar bazasi, tarmoqlar ob’ektlarini dasturlash, hamda zamonaviy mobil vositalar va planshetlar uchun milliy ta’lim resurslarini tayyorlashdagi imkoniyatlari yoritilgan.

Qo‘llanma oliy o‘quv yurtlari professor-o‘qituvchilari, talabalari va barcha boshqa qiziquvchilar uchun mo‘ljallangan.

Taqrizchilar:

Qobulov R.V. - Muxammad Al Xorazmiy nomidagi Toshkent axborot texnologiyalari universiteti ATDT kafedrasi dotsenti, f.-m.f.n., dotsent

Xodjaeva M.S. – Toshkent islom universiteti IAT kafedrasi mudiri, t.f.n., dotsent.

Qo‘llanma Muxammad Al Xorazmiy nomidagi Toshkent axborot texnologiyalari universiteti Kengashi tomonidan tasdiqlangan.

MUNDARIJA

KIRISH

Java nima?

Bu Indoneziya davlati xududidagi Malay orollari majmuasidagi Yava oroli. Bu Java ni yaratuvchilarining sevib ichadigan kofe navi. Jiddiyroq yondashadigan bo‘lsak, bu savolga javob berish juda qiyin, chunki Java ning yaratilish davri aniq chegaraga ega emas, u hozir ham yaratilishda davom etib kelinmoqda. Dastlab Java telefonlarga o‘xshash, ma’ishiy elektron qurilmalarni dasturlash uchun mo‘ljallangan edi (Java ning rasmiy yaratilgan sanasi 1995 yil 23 may hisoblanadi). Keyinchalik Java ni brouzerlarni dasturlash uchun qo‘llay boshlashdi - appletlar yaratildi. Undan keyin Java da to‘la qonli ilovalar yaratish mumkinligi isbotlandi. Ularning grafik elementlarini komponentlar sifatida rasmiylashtirisha boshlashdi — JavaBeans dunyoga keldi. Bu bilan Java tarqoq tizimlar va CORBA texnologiya olami bilan mustahkam bog‘langan oraliq dasturiy ta’minot olamiga kirib keldi. Serverlarni dasturlashga bir qadam qoldi, u ham qilindi servletlar va EJB (Enterprise JavaBeans) yuzaga keldi. Serverlar ma’lumotlar bazasi bilan doimiy muloqatda bo‘lishligi kerak — shuning uchun JDBC (Java DataBase Connection) drayverlari yaratildi. Muloqat muvaffaqiyatli chiqdi. Ko‘pchilik ma’lumotlarni boshqarish tizimlari va hatto operatsion tizimlar ham Java ni o‘z yadrosi ichiga kiritishdi. Masalan, Oracle, Linux, MacOS X, AIX. Java kirib bormagan yo‘nalishni juda kam uchratish mumkin, agar uchragan taqdirda ham u vaqtinchalik, Java u erga ham tezda kirib boradi. Shuning uchun Java ni — texnologiya deb atashadi.

Java ning bunday tez va keng tarqalishi maxsus yaratilgan dasturlash tilining ishlatilishidadir. Bu til ham texnologiyaga mos ravishda Java deb ataladi. Java tili, yaratuvchilarning fikriga ko‘ra, Smalltalk, Pascal, C++ va boshqalarning eng yaxshi tomonlarini olib qolish va yomon tomonlaridan voz kechish asosida yaratilgan. Bu xususida har xil fikrlar bo‘lishiga qaramasdan, Java dasturlash tili o‘rganish uchun qulay, unda yozilgan dastur kodlari oson o‘qiladi va sozlanadi. Java dagi dastlabki dasturni Java ni o‘rganishni boshlagan vaqtdan bir soat o‘tgandayoq yaratish mumkin. Java tili, xuddi Pascal strukturali dasturlashda qanday vazifani bajargan bo‘lsa, ob’ektga yo‘naltirilgan dasturlashni o‘rganishda xuddi shunday vazifani bajarmoqda. Shuning uchun Java da ko‘plab sondagi dasturlar va sinflar kutubxonalari yozilgan.

Hozirgi paytda, Java texnologiya uchun ilovalarni faqat Java tilidagina emas, balki boshqa tillarda ham yaratish imkoniyati bor. Masalan, Pascal va C++ tillarida

shu tillardagi boshlang‘ich kodni Java kodga o‘giruvchi kompilyatorlar mavjud. Biroq Java texnologiyani ishlatishga mo‘ljallangan dasturlarni bari bir Java tilida yozish tavsiya etiladi, chunki unda Java texnologiyaning barcha yo‘nalishlari qamrab olingan, unda kod yozish oson va qulay hisoblanadi. Zamonaviy dasturlashda Java ob’ektga yo‘naltirilgan dasturlashning uslublarini joriy qilishda eng qulay tillardan hisoblanadi.

Java-dasturlarning bajarilishi

Bizga ma’lumki, yuqori darajadagi dasturlash tillaridan birida yozilgan kod boshlang‘ich modul deb ataladi. Yuqori darajali dasturlash tillari jumlasiga Java ham kiradi. Boshlang‘ich kod birdaniga bajarilishi mumkin emas. Uni dastlab kompilyatsiya qilish kerak bo‘ladi, ya’ni mashina buyruqlari ketma ketligi — ob’ekt modulga o‘tkazish kerak. Ob’ekt modul ham hali birdaniga bajarilishga tayyor emas. Uni modulda ishlatilgan funksiyalar kutubxonasi bilan kompanovka qilish, ob’ekt modul seksiyalari orasidagi bog‘lanishlarni hal qilish va natijada yuklanuvchi modulni hosil qilish kerak bo‘ladi. Yuklanuvchi modul bajarilishga tayyor dastur hisoblanadi.

Java da yozilgan boshlang‘ich modul ham bundan mustasno emas. Xuddi shu yerda Java texnologiyaning asosiy xususiyati namayon bo‘ladi. Java da yozilgan dastur mashina buyruqlariga kompilyatsiya qilinadi. Kompilyatsiya aniq bir protsessor buyruqlari emas balki Java virtual mashinasi (JVM, Java Virtual Machine) buyruqlariga kompilyatsiya qilinadi. *Java virtual mashinasi* — bajarilish tizimini ham o‘z ichiga olgan buyruqlar majmuasidir. Mutaxassislar uchun aytish mumkinki, Java virtual mashinasi to‘laligicha stekli bo‘lib, xotirani adreslashning murakkab tizimi va ko‘p sondagi registrlarga ehtiyoji yo‘q. Shuning uchun JVM buyruqlari qisqa, ularning ko‘pchiligining uzunligi 1 bayt, 2 va 3 baytli uzunlikdagi buyruqlarning bo‘lishiga qaramasdan JVM buyruqlari hisoboti *bayt^kodlar* (byte codes) deb ataladi. Statistik tadqiqotlar natijasiga ko‘ra buyruqlarning o‘rtacha uzunligi 1,8 baytni tashkil etadi. JVM arxitekturasi va uning buyruqlarining to‘liq bayoni Java virtual mashinasi xususiyatlari (VMS, Virtual Machine Specification) da keltirilgan. Java virtual mashinasining qanday ishlashi bilan qiziqadiganlar bu xususiyatlar bilan tanishishi mumkin..

Java ning yana bir xususiyati — dastur chaqiradigan barcha standart funksiyalar, bayt kodlarga emas balki, bajarilish jarayonida ulanadi. Mutaxassislar tilida aytilishicha, dinamik komponovka (dynamic binding) amalga oshadi. Bu ham kompilyatsiya qilingan dastur hajmini keskin qisqartiradi.

Demak, birinchi bosqichda, Java tilida yozilgan dastur kompilyator tomonidan bayt-kodlarga o‘giriladi. Bu kompilyatsiya aniq bir protsessor yoki kompyuter arxitekturasiga bog‘liq emas. U dastur yozilgandan keyin bir marta bajarilishi mumkin. Bayt-kodlar bir yoki bir necha fayllarda yoziladi, tashqi xotirada saqlanishi mumkin yoki tarmoq orqali uzatilishi mumkin. Bayt kodli fayllarning xajmi uncha katta bo‘lmagani uchun ularni uzatish qulay. Keyinchalik, kompilyatsiya qilish natijasida xosil bo‘lgan bayt kodlarni JVM ni joriy qila oladigan ixtiyoriy kompyuterda bajarish mumkin. Bunda protsessor turi yoki kompyuter arxitekturasi ahamiyatga ega emas. Java ning "Write once, run anywhere" — "Bir marta yoziladi, xoxlagan joyda bajariladi" tamoyili joriy qilinadi.

Bayt-kodlarni interpretatsiya qilish va dinamik komponovka qilish dastur bajarilishini sezilarli darajada sekinlashtiradi. Bundan bayt kodlarni tarmoq orqali uzatish bundan mustasno, chunki tarmoqda uzatish ixtiyoriy interpretatsiyadan sekin ishlaydi. Boshqa barcha xollarda baquvvat va tez ishlaydigan kompyuter zarur bo‘ladi. Shuning uchun doimiy ravishda interpretatorlarning interpretatsiya qilish tezligini oshirish bo‘yicha izlanishlar olib borilmoqda. Mashina kodlariga interpretatsiya qilingan uchastkalarni eslab qoluvchi *JIT-kompilyatorlar* (Just-In-Time) yaratilgan. *JIT-kompilyatorlar* kompilyatorga qayta murojaat qilinganda bu uchastkalarni avtomatik tarzda bajaradi. Masalan, sikllarni bajarishda bu usul katta samara beradi. SUN firmasi bu yo‘nalishda butun bir Hot-Spot texnologiyasini ishlab chiqdi va o‘zlarining Java virtual mashinasi tarkibiga kiritishdi. Albatta, bu boradagi eng katta tezlikni maxsus protsessor berishi mumkin.

SUN Microsystems firmasi JVM buyruqlar tizimida ishlaydigan PicoJava mikroprotsessorini yaratdi va kuchli Java-protsessorlar qatorini ishlab chiqishni rejalashtirishmoqda. Hozirdayoq boshqa firmalarning Java-protsessorlari mavjud. Bu protsessorlar bevosita bayt-kodlarni bajaradi. Biroq Java dasturlar boshqa protsessorlarda bajarilganda JVM ni aniq protsessor buyruqlariga interpretatsiya qilish kerak bo‘ladi, bundan tashqari, har bir protsessor turiga va kompyuterning har bir arxitekturasiga o‘z interpretatorini yozish zarur bo‘ladi.

Amalda bu masala barcha kompьyuter platformalari uchun xal qilingan. Ularda Java virtual mashinasi joriy qilingan, keng tarqalgan platformalar uchun har xil firmalar tomonidan yaratilgan JVM ning bir necha variantlari joriy qilingan. Ko‘plab operatsion tizimlar va ma’lumotlarni boshqarish tizimlari JVM ni joriy qilishni o‘z yadrosiga kiritishmoqda. Bu borada elektron qurilmalarda ishlatiladigan JavaOS maxsus operatsion tizimi ham yaratilgan. Ko‘pchilik

brouzerlarga, appletlarni bajarish uchun, Java virtual mashinasi ichki joylashtirilgan.

Bayt kodlarni bajaradigan JVM ni joriy qilishdan tashqari bayt kodlar tomonidan chaqiriladigan va bayt kodlar bilan dinamik komponovka qilinadigan funksiyalar to‘plami kerak bo‘ladi. Bu to‘plam bir yoki bir necha paketlardan tuzilgan Java sinflari kutubxonasi sifatida rasmiylashtiriladi. Har bir funksiya bayt kodlar orqali yoziladi, biroq u aniq bir kompyuterda saqlangani uchun, bu funksiyani bevosita ushbu kompyuter buyruqlari tizimida yozish mumkin. Bunda funksiya uchun bayt kodlar interpretatsiyasiga vaqt sarflanmaydi. Bunday funksiyalar "qadrdon" metodlar (native methods) deb ataladi. "Qadrdon" metodlarning qo‘llanilishi dastur bajarilishini tezlashtiradi.

Java texnologiyani yaratuvchisi — SUN Microsystems firmasi — Java dasturlash tili bilan ishlashda kerak bo‘ladigan barcha dasturiy uskunalarni bepul tarqatadi. Bu dasturiy uskuna boy sinflar kutubxonasiga ega bo‘lib, kompilyatsiya, interpretatsiya va sozlash jarayonlarini o‘z ichiga oladi va JDK (Java Development Kit) deb ataladi. Boshqa firmalarning ham dasturiy uskunalari mavjud. Masalan, IBM firmasining JDK si ham keng ommalashgan.

JDK nima?

JDK dasturlar va sinflar majmuasi bo‘lib, quyidagilarni o‘z ichiga oladi:

- JVM ni joriy qilinishini o‘z ichiga oladigan va dasturning boshlang‘ich matnini bayt-kodga o‘giruvchi *javac* kompilyatori;
- engillashtirilgan jre interpretatori (oxirgi versiyalarda yo‘q);
- appletlarni ko‘radigan va brouzerni o‘rnini bosadigan appietviewer dasturi;
- jdt sozlovchi;
- javap dizassembleri;
- arxivlashtirish va zichlashtirish dasturi jar;
- dastur hujjatlarini jamlovchi javadoc dasturi;
- S tilidagi sarlavha fayllarni generatsiya qiluvchi javah dasturi;
- elektron imzoni qo‘shuvchi javakey dasturi;
- binar fayllarni matn fayllarga o‘giruvchi native2ascii dasturi;

- masofadagi ob’ektlar bilan hamkorlik uchun zarur bo‘ladigan rmic va rmiregistry dasturlari;
- sinf versiyasi raqamini aniqlovchi serialver dasturi;
- "Qadrdon" metodlarning kutubxonasi va sarlavha fayllari;
- Java API (Application Programming Interface) sinflari kutubxonasi.

SUN Microsystems kompaniyasi muntazam ravishda JDK ni yangilab boradi va har yili uning yangi yangi versiyasi paydo bo‘ladi.

JRE nima?

Dasturlar va sinflar paketlari JRE o‘z ichida bajarilish uchun zarur bo‘lgan bayt kodlarni mujassamlashtirgan, shu jumladan java interpretator (oldingi versiyalarda engillashtirilgan interpretator jre) va sinflar kutubxonasi ham. Bu JDK ning qismi bo‘lib, ishlanmalar uchun kerak bo‘ladigan vositalar, kompilyatorlar va sozlovchilar bunga kirmaydi. Aynan JRE yoki unga o‘xshashi Java dasturlarni bajara oladigan brouzerlarda, operatsion tizimlarda va ma’lumotlar bazasini boshqarish tizimlarida mavjud.

JRE JDK tarkibiga kirsada, SUN firmasi bu to‘plamni alohida fayl sifatida tarqatadi.

JRE 1.7.0 — bu 8 Mbayt hajmdagi arxiv fayl bo‘lib, diskda ochilganda 20 Mbayt hajm egallaydi.

JDK ni o‘rnatish

JDK to‘plami o‘zi ochiladgan arxiv faylga zichlashtirilgan. Ushbu fayl qandaydir yo‘l bilan olinsa yoki internetdan http://java.sun.com /produsts/jdk/ manzildan ko‘chirilsa, bevosita ishga tushiriladi. Unda o‘rnatish oynasi ochiladi unda boshqa amallar qatori katalogni (directory), masalan, C:\jdkl.7 ni, tanlash taklif etiladi. Agar taklif qilingan katalogga o‘rnatishga rozi bo‘lsangiz boshqa savollarga javob berish ehtiyoji qolmaydi. Agar siz kompyuter taklif qilgan katalogdan boshqa katalogga o‘rnatmoqchi bo‘lsangiz, o‘rnatib bo‘lgandan keyin PATH o‘zgaruvchining qiymatini tekshirib ko‘rish kerak. Buning uchun MS-DOS da Prompt (yoki Windows XR da Command Prompt oynasida) set buyrug‘i teriladi. PATH o‘zgaruvchisi bin katalogga to‘liq yo‘lni o‘z ichiga olishi kerak. Agar bunday bo‘lmasa, bu yo‘lni PATH o‘zgaruvchiga kiritish kerak. Masalan, C:\jdkl.7\bin deb. Bundan tashqari, arxiv fayllar va sinflar kutubxonasini o‘z ichiga oluvchi maxsus o‘zgaruvchi CLASSPATH ni qayta aniqlash kerak. Java 2 ning tizim kutubxonalari CLASSPATH siz, avtomatik ravishda ulanadi.

Bu jarayonda zip- va jar-arxivlarni ochish kerak emas..

Urnatish tugagandan keyin tegishli katalog, masalan, jdkl.7, xosil bo‘ladi. Unda qo‘yidagi qism kataloglar mavjud:

- bin, bajariladigan fayllar uchun;
- demo, dasturlardan namunalar;
- docs, xujjatlar uchun;
- include, “qadrdon” metodlar uchun sarlavha fayllar;
- jre, JRE to‘plami;
- old-include, oldingi versiyalarni tanish uchun;
- lib, sinflar kutubxonasi va xossalar fayllari;
- src, JDK dasturlarining boshlang‘ich kodlari. Yangi versiyalarda katalog o‘rniga o‘ralgan src.jar fayl bo‘ladi.

JDK to‘plami Java da yozilgan o‘z dasturlarini ko‘pchiligining boshlang‘ich kodlarini o‘z ichiga oladi. Bu Java ni o‘rganish uchun juda qulay hisoblanadi.

JDK ni ishlatish

JDK to‘plami MS Windows yoki X Window System ga o‘xshash, grafik muhitda ishlaydigan dasturlarni yaratishga muljallangan bo‘lishiga qaramasdan, u Windows XR da MS-DOS Prompt yoki Windows XR dagi Command Prompt kabi oynalarda ham ishlay oladi. UNIX da matn ko‘rinishida va Xterm oynasida ishlaydi.

Java da dasturni ixtiyoriy matn tahrirchisida, masalan, Notepad, WordPad kabilarda, yozish mumkin. Faqat faylni matn ko‘rinishida saqlab, unga Java kengaymasini berish kerak.

Masalan, fayl nomi MyProgram.java bo‘lsin va fayl joriy katalogda saqlansin.

Bu fayl yaratilgandan keyin buyruqlar satridan javac kompilyator chaqriladi va unga boshlang‘ich fayl parametr sifatida uzatiladi:

javac MyProgram.java

Kompilyator shu katalogda har bir sinf uchun alohida alohida fayl yaratadi va faylga sinf nomini beradi. Yaratilgan faylga class kengaytmasi beriladi. Faraz qilaylik bizning misolimizda MyRrogram deb nomlangan yagona sinf mavjud

bo‘lsin. U xolda biz bayt kodlarni o‘z ichiga olgan MyProgram.class nomli sinfga ega bo‘lamiz.

Agar kompilyatsiya xatosiz tugasa, u holda kompilyator hech qanday xabar bermaydi. Ekranda faqat operatsion tizim taklifi paydo bo‘ladi. Agar kompilyatsiya jarayonida xatolik mavjud bo‘lsa, u holda hato to‘g‘risida xabar ekranga chiqariladi.

Keyingi bosqichda bayt-kodlar interpretatori jaya chaqiriladi va unga parametr sifatida bayt kodli fayl uzatiladi. Bunda kengaytma ko‘rsatilmaydi:

Java MyProgram

Ekranda dastur ishi natijasi paydo bo‘ladi yoki xatolik to‘g‘risida xabar paydo bo‘ladi.

Agar buyruqlar satrida ishlash murakkablik qilsa yoki zerikarli bo‘lsa, u xolda ishlanmalar uchun integrallashgan muhit ishlatiladi.

Java ning integrallashgan muhiti

Java yaratilgan 1996 y. o‘zidayoq Java dasturlarni yaratish imkonini beradigan integrallashgan muhitlar paydo bo‘la boshladi va ularning soni yildan yilga oshib bormoqda. Ularning bazilari JDK ning integrallashtirilgan qobig‘i hisoblanadi. Unda bitta oynada matn tahrirchisi, kompilyator va interpretator ishlaydi. Bunday integrallashgan muhitlar oldindan JDK ning urnatilishini talab etadi. Boshqalarining esa o‘z JDK si bor. Bundan tashqari xususiy kompilyatorga ham ega. Masalan, SUN Microsystems firmasining Java Workshop i, Inprise firmasining Jbuilder i, IBM firmasining Visual Age for Java si va boshqa ko‘plab dasturiy maxsulotlar. Ularni JDK siz ham urnatish mumkin. Shuni alohida ta’kidlash kerakki, aytib o‘tilgan mahsulotlar to‘laligicha Java da yozilgan.

Ko‘pchilik integrallashgan muhitlar vizual dasturlash vositalari hisoblanadi va foydalanuvchi interfeysini tezda yaratish imkonini beradi, ya’ni, RAD (Rapid Application Development) sinfiga tegishli.

Biror bir ishlanma muhitini tanlash, birinchidan, kompyuter imkoniyatiga bog‘liq, chunki vizual muhitlar katta resurslarni talab qiladi, ikkinchidan, shaxsiy ta’b, uchinchidan, dasturiy vosita ichidagi kompilyator imkoniyati sabab bo‘ladi.

I. BOB. JAVA DASTUR KONSTRUKTSIYALARI VA ULARDA MA’LUMOTLAR TURLARINING IFODALANISHI

1.1. Java dasturlarning asosiy konstruktsiyalari

Har bir yangi dasturlash tilini o‘rganishda, ushbu tilda qanday boshlang‘ich ma’lumotlarni qayta ishlash mumkin, bu ma’lumotlarni qanday ko‘rinishda berish kerak va bu ma’lumotlarni qayta ishlash uchun dasturlash tilida qanday standart imkoniyatlar kiritilgan bo‘lishi kerak degan savollarga javob olish qiziqarlidir. Bu zerikarli mashg‘ulot hisoblanib, har bir tilda yangi-yangi ma’lumot turlari va ularni qayta ishlash vositalari mavjud. Biroq bu qoidalarga amal qilmaslik ko‘zga ko‘rinmas xatoliklarni keltirib chiqaradi va bunday xatoliklarni topish oson ish emas.

Java dasturlash tiliga xos qoidalar uning JLS deb ataladigan xususiyatida to‘laligicha bayon qilingan. Ba’zi hollarda, Java ning biror bir konstruksiyasi qanday bajarilishini tushinish uchun xususiyatga murojaat qilishga to‘g‘ri keladi, lekin bunday hollar kam uchraydi, chunki Java tili qoidalari yetarlicha sodda va tabiiy. Ushbu qismda, ma’lumotlarning sodda turlari va ular ustida amallar, boshqaruv operatorlari va ularni ishlatish qoidalari keltirilgan. An’anaga ko‘ra dasturlash tilini o‘rganish eng sodda dasturni yozish orqali boshlanadi.

Java dagi ilk dastur

C tilini o‘zlashtirishga borib taqaladigan an’anaga ko‘ra, dasturlash tilini o‘rgatadigan barcha darsliklar "Hello, World!" dasturi bilan boshlanadi. Biz ham bu an’anani buzmaymiz. 1.1 listingda Java da yozilgan ushbu dastur kodlari keltirilgan.

1.1. **Listing.** Java da yozilgan ilk dastur;

```
2. class HelloWorld {
3. public static void main(String[] args){
4. System.out.println("Hello, XXI Century World!");
5. }
6. }
```

Mana hammasi bo‘lib besh satr! Mana shu soddagina dasturning o‘zidayoq Java dasturlash tilining muhim xususiyatlarini payqash mumkin.

- Har qanday dastur bir yoki bir necha sinflardan iborat bo‘ladi, bu eng sodda dasturimizda faqat bitta sinf(class) mavjud.

• Sinfning boshlanishi *class* xizmatchi so‘zi bilan boshlanadi, undan keyin ixtiyoriy tanlanadigan sinf nomi keladi, bizning misolimizda Helloworld. Sinf ichidagi barcha kodlar katta qavslar ichiga joylashtiriladi va ular sinf tanasini(class body) tashkil qiladi.

• Barcha harakatlar ma’lumotlarni qayta ishlash metodlari orqali bajariladi, qisqacha qilib ular metodlar (method) deb ataladi. Bu nom boshqa dasturlash tillaridagi funksiya nomi o‘rniga ishlatiladi.

• Metodlar nomlari bilan bir biridan farqlanadi. Metodlarning biri albatta main nomiga ega bo‘lishi kerak. Biz yozgan ilk dasturda faqat bitta metod mavjud, demak uning nomi *main* .

• Funksiyalarga xos bo‘lganidek, metod ham ishlashi natijasida yagona natija beradi, ya’ni bitta natija qaytaradi (returns). Uning turi metod nomi oldida albatta ko‘rsatilishi kerak. Metod protseduraga o‘xshab hech qanday qiymat qaytarmasligi ham mumkin. U holda qaytarilishi kerak bo‘lgan tur o‘rniga, bizning misolimizdagi kabi *void* so‘zi yoziladi.

• Metod nomidan keyin qavs ichida argumentlar (arguments) yoki metod parametrlari sanab o‘tiladi. Har bir argument uchun tur ko‘rsatiladi va bo‘sh joy qoldirib nomi yoziladi. Argumentlar bir biridan *vergul* bilan ajratiladi. Misolimizda faqat bitta argument, uning turi simvollar satridan tashkil topgan — massiv. Simvollar satri — bu Java API ning ichki turi, u *string* turiga mansub. O‘rta qavslar — massiv belgisi. Massiv nomi ixtiyoriy bo‘lishi mumkin, misolimizda *args* nomi tanlangan.

• Metod qaytarishi kerak bo‘lgan qiymat turi nomi oldida modifikatorlar (modifiers) yozilishi mumkin. Misolimizda ular ikkita: *public* so‘zi metodga ixtiyoriy joydan murojaat qilish mumkinligini bildiradi; *static* so‘zi *main()* metodini dasturning bajarilishida birinchi bo‘lib chaqirilishini ta’minlaydi. Modifikatorlar majburiy emas, lekin *main()* metodi uchun u zarur.

Matnda metod nomidan keyin, bu nom o‘zgaruvchining nomi emas balki metod nomi ekanligini ta’kidlash uchun, qavslar qo’yiladi.

• Metodning ichidagi barcha kodlar, metod tanasi (method body), katta qavslar ichiga yoziladi.

Bizning misolimizda *main()* metodi bajaradigan yagona amal boshqa metodni chaqirishdan iborat. Chaqiriladigan metod murakkab nomga ega bo‘lgan *System.out.println* metodidir. Bu metodga argument sifatida matnli konstanta "Hello, 2lth century world!" uzatiladi. Matnli konstantalar qo‘shtirnoq ichiga yoziladi.

Murakkab nom *System.out.println* Java API tarkibiga kiruvchi *System* sinfida *out* nomli o‘zgaruvchi borligi va sinf ekzempilyari *println()* metodini o‘z ichiga olishligini bildiradi.

println() metodi bajaradigan ish o‘z argumentini chiqish oqimiga o‘zatishdan, ya’ni matnni terminal ekraniga chiqarishdan iborat. Matn ekranga chiqarilgandan keyin kursor keyingi satr boshiga o‘tadi. Bunga *println* dagi *ln* qo‘shimchasi sabab bo‘ladi.

Java dasturlash tili bosh harf va kichik harflarning farqiga boradi. Masalan, main, Main, MAIN nomlar Java kompilyator uchun har xil nomlardir. Biroq matnli konstanta ichida buning farqi yo‘q.

Java da katta va kichik harflar farqlanadi. Nomlarni ixtiyoriy ravishda yozish mumkin. Masalan, sinf nomini helloworld yoki helloworld, deb yozish mumkin. Biroq Java-dasturchilar orasida "Code Conventions for the Java Programming Language" kelishuvi mavjud. Bu kelishuv http://java.sun.com/docs/codeconv/index.html manzilda saqlanadi. Bu kelishuvga binoan:

- sinf nomlari bosh harfdan boshlanadi, agar nom bir necha so‘zdan iborat bo‘lsa, u holda har bir so‘z bosh harfdan boshlanadi;
- metod va o‘zgaruvchilar nomlari kichik harflardan boshlanadi, agar nom bir necha so‘zdan iborat bo‘lsa, u holda har bir keyingi so‘z bosh harfdan boshlanadi;
- konstantalar nomlari to‘laligicha bosh harflar bilan yoziladi, agar konstanta bir necha so‘zdan tashkil topgan bo‘lsa, u holda so‘zlar orasiga ostchiziq qo’yiladi.

Albatta, bu tartiblar JLS tarkibiga kirsada, majburiy emas, lekin kodni tushinishni yengillashtiradi va dasturga Java ga xos bo‘lgan uslub beradi.

Uslub faqat nomlarni emas, balki dastur matnini satrlar bo‘ylab joylashishini ham belgilaydi, masalan, katta qavslarning joylashishi: ochuvchi katta qavsni sinf yoki metod nomidan keyin satr oxirida qoldirish kerakmi yoki keyingi satrda alohida yozish kerakmi? Negadir bu oddiygina masala ko‘p baxslarga sabab bo‘lmoqda, ba’zi bir ishlanma vositalari, masalan, JBuilder, ochuvchi qavsni joylashtirish uslubini tanlashni taklif etadi. Ko‘pchilik firmalar o‘z ichki uslubini o‘rnatishadi. Biz "Code Conventions" uslubiga amal qilishga harakat qilamiz. Bari bir kompilyator dastur kodini o‘qiganda uni bir necha satrdan iborat yaxlit simvollar ketma ketligi deb qaraydi..

Shunday qilib, dastur biror bir matn taxrirchisida, masalan, Notepad da yoziladi. Endi uni, nomi *main()* metodini o‘z ichiga oluvchi sinf nomi bilan bir xil qilib, faylda saqlash va unga Java kengaymasini berish kerak. Bu qoida bajarilishi kerak. Java ning ijrochi tizimi fayl nomi bilan mos tushuvchi sinfdan *main()* metodini tezda topadi va keyingi jarayonni boshlaydi.

Faylni *main()* metodini o‘z ichiga oluvchi sinf nomi bilan, registrlarni saqlagan holda, bir xil nomlang.

Bizning misolimizda, dasturni joriy katalogda *HelloWorld.java* nomli faylda saqlaymiz. Keyin kompilyatorni chaqirib unga argument sifatida faylni uzatamiz:

javac HelloWorld.java

Kompilyator bayt-kodli faylni yaratadi va unga *Helloworld.class* deb nom beradi va joriy katalogga joylashtiradi.

Endi interpretatorni chaqirib, unga sinf nomini uzatish yetarli:

Java HelloWorld

Ekranda:

Hello, 21st Century World!

hosil bo‘ladi. (Interpretatorni chaqirganda kengayma class ni ko‘rsatmang).

Kommentariyalar

Dastur matnida kommentariyalarni joylashtirish mumkin. Kommentariyalarni kompilyator operator sifatida hisobga olmaydi. Kommentariyalar dastur mazmunini bayon qilishda juda foydali. Dastur sozlanayotganda bir yoki bir necha operatorlarni vaqtinchalik o‘chirib turishga to‘g‘ri keladi, bu paytda ularni kommentariyalar sifatida belgilash kerak bo‘ladi. Kommentariyalar quyidagicha kiritiladi:

- ikki ketma ket kelgan yotiq chiziq (//) dan keyingi matn satr oxirigacha kommentariya deb tushuniladi;
- yotiq chiziqdan keyin yulduzcha /* kommentariya boshlanishini, */ belgi esa kommentariya tugashini bildiradi.

Kommentariyalar kodni o‘qish va tushunish uchun juda qulay. Ular dasturni amallar tartibini bayon qiluvchi xujjatga aylantiradi. Yaxshi kommentariyali dastur o‘z o‘zidan xujjatlashtirilgan deyiladi. Shuning uchun Java da uchinchi tur kommentariya kiritilgan. JDK tarkibida maxsus dastur — *javadoc* mavjud bo‘lib, u bu kommentariyalarni ajratib oladi va HTML formatidagi alohida faylga chiqaradi. Uchinchi tur kommentariyalar /** belgidan boshlanib, */ belgi bilan tugaydi. Bunday kommentariyalarga *javadoc* ga buyruq berish uchun @ simvoldan boshlangan belgi qo‘yiladi.

JDK da xujjatlar xuddi shunday yaratilgan.

Yuqorida keltirilgan misolimizga kommentariya qo‘shamiz (1.2 listing).

1.2. listing. Kommentariyali birinchi dastur.

```
/**
* Dastur mazmuni va xususiyatlarini tushuntirish...
* @author Muallif ismi va familiyasi
* @version 1.0 (dastur versiyasi)
```

```
*/
class HelloWorld{        // HelloWorld — bu faqat nom
// Quyidagi metod dastur bajarilishini boshlaydi
public static void main(String[] args){   // args ishlatilmaydi
/* Quyidagi metod uz argumentini
* displey ekraniga chqaradi holos */
System.out.println("Hello, 21st Century World!");
// Quyidagi chaqiruv kommentariyaga aylantirilgan,
// metod bajarilmaydi
// System.out.println("Farewell, 20th Century!");
}
}
```

Satr boshidagi yulduzchalar hech qanday ahamiyatga ega emas. Ular kommentariyalarni bir biridan ajratish uchun qo‘yilgan.

Konstantalar

Java tilida har xil turdagi va har xil ko‘rinishdagi konstantalarni yozish mumkin. Ularni sanab o‘tamiz.

Butunlar

Butun konstantalarni uchta sanoq sistemasida yozish mumkin:

- o‘nlik shaklda: +5, -7, 12345678 ;
- sakkizlik shaklda, noldan boshlab: 027, -0326, 0777 ; bunday konstantalarni yozishda 8 va 9 raqamlari ishlatilmaydi;

Noldan boshlangan son o‘nlik shaklda emas, balki sakkizlik shaklda yozilgan.

- O‘n oltilik shaklda konstanta *nol* va *x* simvollaridan boshlanadi: 0xff0a, 0xFC2D, 0x45a8, 0X77FF ; Bu yerda katta va kichik harflar farqlanmaydi.

Butun konstantalar int shaklida saqlanadi.

Butun konstanta oxirida katta L yoki kichik l harflarini quyish mumkin. U holda konstanta uzun long shaklda saqlanadi. Masalan, +25L, -037l, OxffL, OXDFDFl .

Maslahat

Uzun butun konstantalar oxirida kichik l harfini ishlatmang. Uni bir soni bilan chalkashtirish mumkin.

Haqiqiylar

Haqiqiy konstantalar faqat o‘nlik sanoq sistemasida ikki xil shaklda yoziladi:

- nuqtaning joyi qaydlangan: 37.25, -128.678967, +27.035 ;
- suzuvchi nuqtali: 2.5e34, -0.345e-25, 37.2E+4 ; Katta yoki kichik lotincha E harfini yozish mumkin; probel va qavslarga yo‘l qo‘yilmaydi.

Haqiqiy konstantalar oxirida F yoki f harflarini ishlatish mumkin. U holda konstanta float shaklida saqlanadi: 3.5f, -45.67F, 4.7e-5f. D (yoki d) harfini ham yozish mumkin: 0.045D, -456.77889d , bu double turini bildiradi. Lekin bu oshiqcha amal, chunki haqiqiy sonlar o‘z o‘zidan double shaklida saqlanadi.

Simvollar

Yakka simvollarni yozish uchun quyidagi shakllar ishlatiladi.

- Bosma simvollarni apostroflarda: ' a ', ' N ', ' ? ' yozish mumkin.
- Boshqaruv simvollari apostroflarda teskari yotiq chiziq bilan yoziladi:

o ' \n ' — satrni yangi satrga o‘tkazish simvoli(newline ASCII 10 kodli);
o ' \r ' — ENTER CR 13 kodli;
o ' \f ' — sahifani o‘tkazish simvoli FF 12 kodli;
o ' \b ' — bir qadam orqaga simvoli BS 8 kodli;
o ' \t ' — gorizontal tabulyatsiya simvoli NT 9 kodli;
o ' \\ ' — teskari yotiq chiziq;
o ' \" ' — qo‘shtirnoq;
o ' \' ' — apostrof.

- O‘nlik kodirovkadagi 0 dan 255 gacha ixtiyoriy simvolni, sakkizlik sanoq sistemasida uchtadan ko‘p bo‘lmagan raqam bilan apostrof ichida teskari yotiq chiziq bilan yozish mumkin: ' \123 ' — S harfi, ' \346 ' — SR1251 kodirovkadagi J harfi. Bu yozuv shaklini bundan oldingi bandda sanab o‘tilgan bosma va boshqaruv simvollariga nisbatan ishlatish tavsiya etilmaydi, chunki kompyuter sakkizlik yozuvni yuqorida ko‘rsatilgan shakllarga aylantiradi va tushunmovchilik kelib chiqadi. Eng yuqori kod ' \377 ' bo‘lib — u o‘nlik 255 sonini ifodalaydi.
- Unicode kodirovkasidagi ixtiyoriy simvol kodi apostrof ichida teskari yotiq chiziq va lotincha u harfidan keyin to‘rtta o‘n oltilik son bilan ifodalanadi: ' \u0053 ' — S harfi, ' \u0416 ' — Jharfi.

Simvollar *char* turi shaklida saqlanadi.

Izoh

Unicode kodirovkasida ruscha harflar quyidagi diapozonda joylashadi: ' \u0410 ' — A bosh harf dan, ' \u042F ' — Я bosh harfigacha, kichik harflar ' \u0430 ' — a harfidan, ' \044F ' — я harfigacha.

Simvollar qanday ko‘rinishda yozilishidan qat’iy nazar kompilyator ularni Unicode ga o‘zgartiradi, hatto dasturning boshlang‘ich matnini ham.

Kompilyator va Java ning ijrochi tizimi faqat Unicode kodirovkasida ishlaydi.

Satrlar

Simvolli satrlar qo‘shtirnoq ichida beriladi. Boshqaruv simvollari va kodlar satrlarda xuddi oldingidagidek teskari yotiq chiziq bilan yoziladi. Ular apostroflarsiz yozilsada, lekin avval bayon qilinganidek ta’sir etadi. Satrlar faqat bir qatorda yozilishi kerak, ochuvchi qo‘shtirnoq bir satrda, yopuvchisi esa boshqa satrda bo‘lishi mumkin emas.

Misollar:

"Bu ko‘chiriladigan \ns satr"

"\"Bunyodkor\" — Chempion!"

Simvollar satrini bir satrda boshlab boshqa satrda tugatish mumkin emas.

Satr konstantalari uchun plyus belgisi bilan beriladigan ulash amali aniqlangan. " Satrlarni " + "ulash" amali natijada "Satrlarni ulash" satrini beradi.

Uzun satrlarni yagona satr ko‘rinishiga keltirish uchun birinchi va undan keyingi satrlar oxirida yopuvchi qo‘shtirnoq dan keyin + belgisini qo‘yish kerak. U holda kompilyator ikki yoki undan ko‘p satrlarni yagona satrli konstantaga jamlaydi, masalan:

"Ikki satrda yozilgan "+

"yagona satr konstantasi"

Simvollarni Unicode kodirovkasida ekranga chiqarish kerak bo‘lsa, masalan, "Rossiya" so‘zini:

System.out.println("\u0429\u043e\u0441\u0441\u0438\u044f");

deb yozish kerak.

Bu shiriftda ruscha harflar Unicode ning boshlang‘ich kodlarida joylashgan. Ular nimagadir SR866 kodirovkasida va Unicode ning boshqa segmentlari bo‘yicha sochib yuborilgan. Unicode ning barcha shiriftlari ham barcha simvollar ko‘rinishini (glyphs) o‘z ichiga olmaydi, shuning uchun satrni Unicode kodirovkasida chiqarishda ehtiyot bo‘lish kerak.

Maslahat

Unicode ni bevosita chorasiz hollarda ishlating.

Nomlar

O‘zgaruvchilar, sinflar, metodlar va boshqa ob’ektlarning nomlari (names) sodda (umumiy nomi -identifikatorlar) yoki qo‘shma (qualified names) bo‘lishi mumkin. Java da identifikatorlar Java harflar (Java letters) va arab raqamlari 0—9 dan tuziladi. Identifikatorning birinchi simvoli raqam bo‘lishi mumkin emas. Java dagi harflar tarkibiga kichik va katta lotin harflari, milliy alifbo harflari, $ belgisi va _ belgilari kiradi. Nomlarda dollar belgisini ko‘rsatmang. Java kompilyatori uni ichki sinflar nomlarini yozishda ishlatadi.

To‘g‘ri identifikatorlarga misollar:

a1 my_var var3_5 _var veryLongVarName
aName theName a2Vh36kBnMt456dX
Nomlarda kichik l harfini ishlatmagan maqul, uni 1 soni bilan chalkashtirish mumkin, xuddi shuningdek o harfini ham, buni 0 soni bilan chalkashtirish mumkin.
"Code Conventions" tavsiyalarini unutmang
Java API tarkibiga kiruvchi *Character* sinfida, berilgan simvol identifikatorda ishlatishga yaroqlimi yoki yo‘qligini tekshiruvchi ikkita metod bor. Ular: *isJavaIdentifierStart(),* simvolning Java, simvol ekanligini tekshiradi; *isJavaIdentifierPart(),* simvolning harf yoki raqam ekanligini aniqlaydi.

Java ning *class, void, static* kabi xizmatchi so‘zlari zahiralashtirilgan, ularni ob’ektlarning identifikatorlari sifatida ishlatish mumkin emas. Qo‘shma nom (qualified name) — bu o‘zaro bir biridan, bo‘sh joysiz, nuqta bilan ajratilgan bir necha identifikatorlardir, masalan, bizga ma’lum bo‘lgan qo‘shma nom *System.out.println*.

Nazorat savollari

1. Asosiy turlarni sanab o‘ting.
2. Kommentariyalar necha xil bo‘ladi?.
3. Barcha amallarni ko‘rsating.
4. Qachon oshkor turlarni keltirishga xojat yo‘q?
5. Ifodalarda turlar qanday avtomatik keltiriladi?

1.2. Operatorlar

Chiziqli dasturlash. O‘zlashtirish amallari

Oddiy o‘zlashtirish amali (simple assignment operator) tenglik belgisi = bilan yoziladi, undan chapda o‘zgaruvchi turadi, o‘ngda esa o‘zgaruvchi turiga mos ifoda: x = 3.5, u = 2 * (x - 0.567) / (x + 2), b = x < u, bb = x >= u && b, kabilar bo‘ladi. Faqat o‘zlashtirish amallaridan tashkil topgan dasturlar chiziqli dasturlar deb ataladi.

O‘zlashtirish amali quyidagicha ishlaydi: tenglik belgisidan keyin turgan ifoda hisoblanadi va tenglik belgisidan chap tomonda turgan o‘zgaruvchi turiga keltiriladi.

O‘zlashtirish amali yana bir qo‘shimcha amalni bajaradi: chap tomondagi o‘zgaruvchi o‘ng tomonning keltirilgan qiymatini oladi, uning oldingi qiymati yo‘qoladi. O‘zlashtirish amalida chap va o‘ng tomonlar teng huquqli emas, 3.5 = x deb yozish mumkin emas. x = u amalidan keyin x o‘zgaruvchi o‘zgaradi, u u ga teng bo‘ladi, u = x amalidan keyin esa u o‘zgaradi.

Oddiy o‘zlashtirish amalidan tashqari yana 11 ta qo‘shma o‘zlashtirish amallari mavjud (compound assignment operators): +=, -=, *=, /=, %=, &=, |=, ^=, <<=, >>= ; >>>=. Simvollar bo‘sh oraliqsiz yoziladi, ularning o‘rnini almashtirish mumkin emas. Barcha qo‘shma o‘zlashtirish amallari yagona sxema bo‘yicha ishlaydi:

x or = a ekvivalent x = (tur x) ga, ya’ni (x or a).

Eslatib o‘tamiz, short turidagi ind o‘zgaruvchi bizda 1 qiymat bilan aniqlangan. ind +=7.8 o‘zlashtirish natijada 8 sonini beradi, shu qiymatni ind o‘zgaruvchi ham oladi. Bu amal oddiy o‘zlashtirish amali ind = (short)(ind + 7.8) ga ekvivalent.

O‘zlashtirishdan oldin, zaruratga ko‘ra, avtomatik tarzda turni keltirish bajariladi.

Shuning uchun:

byte b = 1;
b = b + 10; // Hato!
b += 10; // To’g’ri!

b + 50 qo‘shishdan oldin b turni int turigacha ko‘tarish bajariladi, qo‘shish natijasi ham int bo‘ladi. Birinchi holda, b o‘zgaruvchi turni aniq keltirmasdan turib o‘zlashtirilishi mumkin emas. Ikkinchi holda, o‘zlashtirishdan oldin qo‘shish natijasini *byte* turigacha toraytirish bajariladi.

Qo‘shma operatorlar

Qo‘shma operatorga qo‘shma operator va blok kiradi. Ikkala holda ham bu figurali qavsga olingan operatorlar ketma ketligidir. Blok qo‘shma operatordan farqi shuki ta’riflar ham qatnashadi.

Shartli amal

Bu o‘ziga xos amal uchta operandga ega. Boshlanishida, natijada *true* yoki *false* ega bo‘ladigan, ixtiyoriy mantiqiy ifoda yoziladi, keyin so‘roq belgisi, keyin ikki nuqta bilan ajratilgan ikkita ixtiyoriy ifoda yoziladi, masalan,

x < 0 ? 0 : x

x > u ? x — u : x + u

Shartli amal quyidagicha bajariladi. Dastlab mantiqiy ifoda hisoblanadi. Agar *true* qiymat olinsa, u holda so‘roq belgisidan? keyingi ifoda hisoblanadi va uning natijasi butun amal natijasi bo‘ladi. Oxirgi ifoda bunda hisoblanmaydi. Agar dastlabki ifoda natijasi *false* bo‘lsa, faqat oxirgi ifoda hisoblanadi va uning natijasi butun amal natijasi bo‘ladi.

Bu, n == o ? ha : m / n deb, nolga bo‘lishdan havotirlanmasdan, yozish imkoniyatini beradi. Shartli amal dastlab g‘aroyib ko‘rinsada, ammo uncha katta bo‘lmagan shoxlanishlarni yozishda qulaydir.

Kiritish

Ma’lumotni "kiritish standart qurilmasidan " (ya’ni klaviaturadan) o‘qish uchun muloqot darchasi yaratish kerak. Buning uchun *JOptionPane.showInputDialog(promptString)* usulidan foydalanish lozim. Bu usul foydalanuvchi tergan satrni qaytaradi.

Masalan dastur foydalanuvchisini so‘rash:

String name = JOptionPane.showInputDialog("What is your name?");

Usul qaytargan satrni songa aylantirish uchun *Integer.parseInt* yoki *Double.parseDouble,* usulidan foydalanish lozim. Masalan:

String input = JOptionPane.showInputDialog("How old are you?");

int age = Integer.parselnt(input);

Agar klaviaturada 40 soni terilsa, satrli o‘zgaruvchi input qiymati "40" satrga teng bo‘ladi. Bu satrni *Integer .parseInt* usuli 40 soniga aylantiradi.

Dastur *JOptionPane. showsInputDialog* usulini chaqirganda, uning ishini tugatish uchun *System.exit(0)* usulini chaqirib tugatish lozim. Ekranga dialog darchasini chiqarish yangi boshqarish oqimini hosil qiladi. Bu oqim o‘z ishini *main* usuli bilan birga tugatmaydi. Hamma oqimlarni yopish uchun *System.exit* usulini chaqirish lozim. Bu usul butun sonli parametr, ya’ni dasturdan "chiqish kodini" qaytaradi. Ko‘zda tutilgan bo‘yicha, agar dastur ishi to‘g‘ri tugatilgan bo‘lsa, chiqish kodi nolga teng bo‘ladi, aks holda nolga teng bo‘lmaydi. Har xil xato holatlar haqida xabar berish uchun, har xil chiqish kodidan foydalanish mumkin.

JOptionPane sinfi *javax.swing* paketida aniqlangan. Asosiy *java.lang* paketida aniqlanmagan. Masalan:

```
import javax.swing.*;
public class InputTest
{
public static void main(String[] args)
{
// Birinchi kiritish.
String name = JOptionPane.showInputDialog("Ismingizni ayting! ");
// Ikkinchi kiritish.
String input = JOptionPane.showInputDialog ("Yoshingiz nechada?");
// Satrni songa aylantirish.
int age = Integer.parseInt(input);
```

```
// Natijani konsolga chiqarish.
System.out.println("Салом, " + name +
". Keyingi yilda sizning yoshingiz teng buladi = " + (age + 1 ) ) ;
System.exit(0) ;
}
}
```

Tanlash operatorlari

Shartli operator. Shartli tanlash operatorida avval shart tekshiriladi. Agar shart rost bo‘lsa birinchi aks holda ikkinchi operator(agar u bo‘lsa) bajariladi.

if (ifoda) 1- operator else 2- operator yoki if (ifoda) 1-operator

Misol. Simvol ikkilik ekanligini tekshiruvchi dastur:

```
class IfElse {
public static void main(String args[]) { char ch = ‘1’;
String s;
if (ch==’1’ || ch == ‘0’) {
s = "binary";
} else
s= "no binary";
System.out.println( "Simvol " + s + ".");
}
}
```

Dastur bajarish natijasi:

S: \> java IfElse

Simvol binary.

Kalit bo‘yicha tanlash opertori. Kalit bo‘yicha tanlash operatori quyidagi shaklga ega:

```
switch(<ifoda>) {
case <1-qiymat>:<1-operator>
...
default: <operator>
… }
```

Kalit bo‘yicha tanlash operatorida berilgan ifoda qiymati biror *case* qiymatiga mos kelsa, shundan keyingi hamma operatorlar bajariladi, aks holda *default*(agar u bo‘lsa) so‘zidan keyingi operator bajariladi.

Misol. Simvol ikkilik ekanligini tekshiruvchi dastur:

```
class SwitchDemo { public static void main(String args[]) {
```

```
char ch = ‘1’;
String s;
switch (month) {
case ‘0’: // FALLSTHROUGH
case ‘1’: // FALLSTHROUGH
s = "binary";
break;
default:
s = "no binary";
}
System.out.println("Simvol " + s + ".");
}
}
```

Sikl operatorlari

Oldingi shartli sikl. Quyidagi ko‘rinishga ega:
while (<shart>) < sikl tanasi> ;
Oldingi shartli siklda oldin shart tekshiriladi keyin to shart yolg‘on bo‘lguncha sikl tanasi bajariladi.

Misol. Sikl yordamida o‘n satrga «tick» so‘zini chiqarish:

```
class WhileDemo {
public static void main(String args[]) {
int n = 10;
while (n > 0) {
System.out.println("tick " + n);
n--;
}
}
}
```

Keyingi shartli sikl. Quyidagi ko‘rinishga ega:
do
<sikl tanasi>;
while (<shart>);
Oldin sikl tanasi bajarilib, keyin shart tekshiriladi. Sikl to shart yolg‘on bo‘lmaguncha davom etadi.

Misol. sikl yordamida o‘n satrga «tick» so‘zini chiqarish:

```
class DoWhile {
public static void main(String args[]) {
int n = 10;
do {
System.out.println("tick " + n);
} while (--n > 0);
}
}
```

Parametrli sikl. Quyidagi ko‘rinishga ega:
for(1-ifoda; shart; 2-ifoda)
sikl tanasi;
Avval 1 – ifoda bajariladi va to shart yolg‘on bo‘lmaguncha sikl tanasi va 2- ifoda bajariladi. Bunda ifodalar ixtiyoriy ifoda yoki bo‘sh bo‘lishi mumkin, lekin ularni ajratuvchi qavs « ; » bo‘lishi shart.
Misol. Birdan o‘ngacha sonlarni chiqarish:

```
class ForDemo {
public static void main(String args[]) {
for (int i = 1; i <= 10; i++)
System.out.println("i = " + i);
}
}
```

O‘tish operatorlari

O‘tish operatorlari boshqarishni shartsiz uzatishni amalga oshiradi. Blokdan yoki sikldan chiqish uchun *break* – operatoridan foydalaniladi. Agar siklni davom ettirish shartini sikl o‘rtasida tekshirish zarur bo‘lsa *break* operatoridan foydalanish qulaydir. Masalan, quyidagi dasturda uchta ichki joylashgan blok mavjud bo‘lib, har birining unikal belgisi mavjud. Ichki blokda joylashgan *break* operatori, b blokdan keyingi operatorga boshqarishni uzatadi. Natijada ikki *println* operatori bajarilmay qoladi.

```
class Break {
public static void main(String args[]) { boolean t = true;
a: { b: { c: {
System.out.println("Before the break"); // break operatoridan oldin
```

```
if (t)
break b;
System.out.println("This won't execute"); // Bajarilmaydi }
System.out.println("This won't execute"); // Bajarilmaydi }
System.out.println("This is after b"); // b dan so’ng
}
}
}
```

Dastur bajarish natijasi:

S:\> Java Break
Before the break
This is after b

Sikl keyingi iteratsiyasiga o‘tish uchun continue – operatoridan foydalaniladi. Xuddi break operatori kabi, continue operatorida qaysi ichki sikl itratsiyasini to‘xtatish kerakligini ko‘rsatuvchi belgidan foydalanish mumkin. Masalan ko‘paytirish jadvalini uchburchak shaklda chiqarish uchun belgili continue operatoridan foydalanuvchi dastur:

```
class ContinueLabel {
public static void main(String args[]) {
outer: for (int i=0; i < 10; i++) {
for (int j = 0; j < 10; j++) {
if (j > i) {
System.out.println("");
continue outer;
}
System.out.print(" " + (i * j));
}
}
}
}
```

Bu dasturda *continue* operatori j parametrli ichki sikl ishini to‘xtatib, i parametrli tashqi sikl keyingi iteratsiyasiga o‘tishga olib keladi:

```
C:\> Java ContinueLabel
0
0 1
```

0 2 4
0 3 6 9
0 4 8 12 16
0 5 10 15 20 25
0 6 12 18 24 30 36
0 7 14 21 28 35 42 49
0 8 16 24 32 40 48 56 64
0 9 18 27 36 45 54 63 72 81

Nazorat savollari

1. Izoh qanday ko‘rsatiladi?
2. Qo‘shma operator ta’rifini keltiring.
3. Kiritish qanday amalga oshiriladi?
4. Tanlash operatorlarini ko‘rsating.
5. Sikllar turlarini ko‘rsating.
6. Nima uchun break va continue operatorlari ishlatiladi?

1.3. MASSIVLAR

Bir o’lchovli massivlar

Java dasturlash tilida barcha boshlang‘ich ma’lumotlar ikki guruhga bo‘linadi: sodda turlar (primitive types) va ilovali turlar (reference types).

Ilovali turlar massivlar (arrays), sinflar (classes) va interfeyslar (interfaces) dan tashkil topgan. Ilovali turlardan massivlarni qaraymiz.

Massiv indeksli o‘zgaruvchidir.

Massivning sodda ta’rifi:

<tip> <o‘zgaruvchi_nomi>>[<konstanta_ifoda>] = <initsializator>;

Massiv indekslar qiymati har doim 0 dan boshlanadi.

Massivga xotira ajratish uchun maxsus new operatoridan foydalaniladi.

int a[];

int a=new int[6];

Massiv initsializatsiya qilinganda elementlar soni ko‘rsatilishi shart emas.

Masalan:

double d[] = {1, 2, 3, 4, 5};

Bu sintaksis konstruktsiya yordamida yangi o‘zgaruvchi kiritmasdan massivni qayta initsializatsiya qilish mumkin.

Masalan:

smallPrimes = new int{ 17, 19, 23, 29, 31, 37 };

Massivda elementlar sonini xisoblash uchun length usulidan foydalanish mumkin.

Masalan:

for (int i = 0; i < a.length; i++)

System.out.println(a[i]) ;

Massiv yaratilgandan so‘ng uning xajmini o‘zgartirish mumkin emas (lekin aloxida elementlarini o‘zgartirish mumkin).

Java tilida [ ] operatori ko‘zda tutilgan bo‘yicha indeks o‘zgarish diapazonini tekshiradi.

Massivni ikki usulda ta’riflash mumkin:

int [] a;

yoki

int a[];

Massivdan nusxa olish

Bitta massivdan ikkinchisiga nusxa olish mumkin, lekin bu holda ikkala o‘zgaruvchi bitta massivga ilova qiladi.

int[] luckyNumbers = smailPrimes;

luckyNuimbers[5] = 12; // Endi element smailPrimes[5] teng 12 ga.

Agar massiv hamma elementlaridan boshqa massivga nusxa olish zarur bo‘lsa java.lang.System sinfining arrausoru usulidan foydalanish lozim.

Masalan quyidagi operatorlar, ikki massiv yaratadi, so‘ngra birinchi massiv oxirgi to‘rtta elementidan ikkinchi massivga nusxa oladi. Nusxa olish birinchi massiv ikkinchi elementidan boshlanib, ikkinchi massivga uchinchi pozitsiyadan nusxa olinadi.

int[] smailPrimes = {2, 3, 5, 7, 11, 13};

int[] luckyNumbers = {1001, 1002, 1003, 1004, 1005, 1006, 1007};

System.arraycopy(smailPrimes, 2, luckyNumbers, 3, 4);

for (int i = 0; i < luckyNumbers.length; i++)

System.println(i +.": " + luckyNumbersfi]);

Bu operatorlar bajarilishi quyidagi natijaga olib keladi.

0: 1001

1: 1002

2: 1003

3: 5

4: 7

5: 11

6: 13

Massivlarni tartiblash

Agar sonlar massivini tartiblash lozim bo‘lsa java.util.Arrays sinfining sort usulidan foydalanish lozim.

Usul massivni tezkor algoritm asosida tartiblaydi.

Int []a = new int[10000];

Arrays.sort(a);

Quyidagi dasturda ishchi massivda lotereya uchun sonlar tasodifiy ravishda generatsiya qilinadi.

```
import java.util.*;
import javax.swing.*;
public class LotteryDrawing
{
public static void main(String[] args)
{
String input = JOptionPane.showInputDialog
("Qancha nomerni toppish lozim?");
int κ = Integer.parselnt(input);
input = JOptionPane.showInputDialog
("Eng kata nomer nechaga teng bo’lishi mumkin?");
int n = Integer.parselnt(input);
// massivni  1 2 3. . .n sonlar bilan to’ldiramiz.
int[] numbers = new int[n];
for (int i = 0; i < numbers.length; i++)
numbers[i] = i + 1;
// κ ta son generatsiya qilib, ikkinchi massivga joylashtiramiz.
int[] result = new int[κ];
for (int i = 0; i < result.length; i++)

{
// 0 dan to  n – 1 gacha tasodifiy index generatsiyasi.
int r = (int)(Math.random() * n);
// elementni tasodifiy yacheykaga joylashtiramiz.
result[i] = numbers[r];
// oxirgi elementni tasodifiy yacheykaga  κo’chiramiz.
numbers[r] = numbers[n - 1];
n— ;
}
// tartiblangan massivni chiqaramiz.
```

```
Arrays.sort(result);
System.out.println
("Qo'yidagi kombinatsiyaga tiking  — achinmaysiz!")
for (int i = 0; i < result.length; i++)
System.out.println(result[i]);
System.exit(0);
}
}
```

Agar " 49 tadan 6 ta", yutish lozim bo‘lsa dastur quyidagi ma’lumotni bosib chiqaradi.

Quyidagi kombinatsiyaga tiking — achinmaysiz!
4
7
8
19
30
44

Ko‘p o‘lchovli massivlar

Java tilida chin ko‘p o‘lchovli massivlar mavjud emas. Ko‘p o‘lchovli massivlar massivlar massivlari sifatida tasvirlanadi. Quyida keltirilgan dastur double tipidagi o‘n olti elementdan iborat matritsa yaratib, nolь qiymat bilan initsializatsiya qiladi. Bu massiv ichki realizatsiyasi double tipidagi — massivlar massividir.

double matrix [][] = new double [4][4];

Quyida shuncha xotira ajratiladi, lekin ikkinchi o‘lchov uchun xotira oshkor qadamma qadam ajratiladi.

```
double matrix [][] = new double [4][];
matrix [0] = new double[4];
matrix[1] = new double[4];
matrix[2] = new double[4], matrix[3] = { 0, 1, 2, 3 };
```

Boshqa hollarda agar massiv elementlari oldindan ma’lum bo‘lsa, ko‘p o‘lchovli masssivni initsializatsiya qilish uchun new operatoridan foydalanilmaydigan qisqa yozuvdan foydalanish mumkin. Masalan.

int[][] magicSquare = {{16, 3, 2, },(5, 10, 11},{9, 6, 7},{4, 15, 14}};

Massiv satrlarini osongina o‘zgartirish mumkin!

double[] temp = balance [i];
balance[i] = balance[i+1];
balance[i=l] = temp;

Bundan tashqari Java tilida "tekis bo‘lmagan" ya’ni har xil satrlari har xil uzunlikka ega bo‘lgan massivlar yaratish osondir.
Notekis massiv yaratish uchun xotiraga satrlarni saqlovchi massiv yaratiladi.
int[][] adds = new int[NMAX+1][];
So‘ngra satrlar yaratiladi.
for (n=0; n<=NMAX; n++)
adds[n] = new int[n + l] ;
Xotirada butun massiv joylashgandan so‘ng, massiv elementlariga murojaat qilish mumkin. Lekin indekslar diapazondan chiqib ketmasligi lozim.

for (n=0; n<odds.length; n++)
for (k=0; k<odds[n].length; k++)
// Imkoniyatlarni xisoblash.
adds[n][k] = lotteryOdds;
Quyidagi misolda i-chi satri va j-chi ustuni kesishmasida " j ta sondan i lotereya nomerini tanlash " imkoniyatlari soni yozilgan uchburchak massiv yaratiladi.
Massiv i-chi satrida i+1 element joylashgan

```
public class LotteryArray
{
public static void main(String[] args)
(
final int NMAX' = 10;
// Uchburchak matritsani joylashtirish.
int[][] odds = new int[NMAX + 1][];
for (int n = 0; n <= NMAX; n++)
odds[n] = new int[n + 1];
// Uchburchak matritsani to’ldirish,
for (int n = 0; n < odds.length; n++)
for (int k = 0; k < odds[n].length; k++)
{
/*
Binomial koeffitsientlarni hisoblash.
n * (n - 1) * (n - 2) * . . . * (n - k + 1)
```

```
*/

int lotteryOdds = 1;
for (int i = 1; i <= k;
lotteryOdds = lotteryOdds * (n - i + 1). / i;
odds[n][k] = lotteryOdds;
// Uchburchak matritsani chqarish.
for (int n = 0; n < odds.length; n++)
{
for (int κ = 0; κ < odds[n].length; k++)
{
// Chiqarishda bushlik belgilarni joylashtirish.
String output = " " + odds[n][k];
// Natija kengligi 4 simvolga teng maydonga chiqariladi,
output = output.substring(output.length() - 4);
System.out.print(output);
}
System.out.println();
}
}
}
```

Buyruq qatori parametrlari

Java tilidagi har bir dasturda String [ ] args parametriga ega *main* usuli mavjud. Bu parametr buyruq qatorida ko‘rsatilgan argumentlardan iborat satrlar massivini *main* usuliga uzatadi.

Masalan.

```
public class Message
public static void main(String[] args)
if (args[0].equals("-h");
System.out.print ("Salom,");
else if (args[0].equals("-g"))
System.out.print("Xayr,");
// Buyruqlar qatori qolgan argumentlarni chiqaradi,
for (int i = 1; i < args.length; i++)
System.out.print(" " + args[i]);
System.out.print("!");
}
```

}

Agar dastur quyidagicha chaqirilsa
Java Message -g bevafo dunyo
massiv args quyidagi elementlardan iborat bo‘ladi.

args[0] "-g"
args[l] "bevafo"
args[2] "dunyo"
Dastur quyidagi ma’lumotni chiqaradi
Xayr, bevafo dunyo!
Java tilidagi dasturda main usulidagi args massivi dastur nomini saqlamaydi. Masalan Message dasturni quyidagi buyruq yordamida ishga tushirilsa
Java Message -h dunyo
Buyruq qatoridagi args[0] element, "Message" yoki "Java" satriga emas "- h" satriga teng bo‘ladi.

Nazorat savollari

1. Massivni initsializatsiya qilish usullarini ko‘rsating.
2. Qanday qilib massivlar formal parametr sifatida ta’riflanishi mumkin?
3. Nima uchun new amali qo‘llanadi?
4. Massivlardan nusxa olish va massivlarni tartiblash qanday amalga oshiriladi?
5. Dinamik massivlar hosil qilish usullarini ko‘rsating.

1.4. Satrlar va satrlar ustida amallar

Ma’lumotlarni qayta ishlashda juda katta joyni matnlar bilan ishlash tashkil etadi. Boshqa ko‘pchilik elementlar qatori, Java dagi matnli satrlar ob’ektlar hisoblanadi. Ular *String* yoki *StringBuffer* sinflari ekzempilyarlari sifatida tasvirlanadi. Boshlanishida bu g‘ayritabiiy va juda noqulay hisoblanishi mumkin, ammo, o‘rganib olingandan keyin, sinflar bilan ishlash simvollar massivlari bilan ishlashga qaraganda ancha qulay ekanligiga iqror bo‘linadi. Matnni *char* turidagi simvollar massiviga, hatto, *byte* turidagi baytlar massiviga ham kiritish mumkin, faqat u holda matnli satrlar bilan ishlaydigan tayyor metodlarni ishlatib bo‘lmaydi.

Tilga nimaga, satrlarni saqlash uchun ikkita sinf kiritilgan? *string* sinfi ob’ektlarida uzunligi va mazmuni uzgarmaydigan satr-konstantalar saqlanadi. Bu, satrlarni uni ishlatadigan ob’ektlarga bo‘lgan holda, qayta ishlashni birmuncha tezlashtiradi va xotirani tejash imkonini beradi. *stringBuffer* sinfi ob’ektlarida saqlanayotgan satrlar uzunligini esa, simvollar va satrlarni kiritish, qo‘shish, ostsatrlarni uchirish yoki bir necha satrni ulab bir satr ko‘rinishiga keltirish orqali, o‘zgartirish mumkin. *string* turidagi satr uzunligini o‘zgartirish kerak bo‘ladigan

ko‘pchilik hollarda, Java kompilyatori bilvosita satrni *stringBuffer* turiga aylantiradi, uzunligini o‘zgartiradi va yana qayta string turiga aylantiradi. Masalan, quyidagi amalni

String s = "Bu" + " bitta " + "satr";

kompilyator mana bunday bajaradi:

String s = new StringBuffer().append("Bu").append(" bitta ")
.append("satr").toString();

stringBuffer sinfi ob’ekti yaratiladi, unga birin ketin "Bu", " bitta ", "satr" satrlari qo‘shiladi va hosil bo‘lgan *StringBuffer* sinfi ob’ekti *toString()* metodi yordamida String turiga aylantiriladi. Shuni eslatish zarurki, satrdagi simvollar Unicode kodirovkasida saqlanadi. Undagi har bir simvol ikki bayt joy egallaydi. Har bir simvol turi *char*.

String sinfi

Satr bilan ish boshlashdan oldin uni yaratish kerak. Bu ishni bir necha xil usul bilan amalga oshirish mumkin.

Satr yaratish

Satr yaratishning eng sodda usuli — bu satr-konstantaga *string* turidagi ilovani tashkil etishdir:

String si = "Bu satr.";

Agar konstanta uzun bo‘lsa, uni matn tahrirchisining bir necha satrida satrlarni ulash orqali yozish mumkin:

String s2 = "Bu uzun satr, " +
"boshlang‘ich matnning ikki satrida yozilishi";

Bo‘sh satr **string s = ""** hech bir simvolni o‘z ichiga olmaydi, bo‘sh ilova string **s = null** hech bir satrni ko‘rsatmaydi, ob’ekt ham bo‘lmaydi.

OYD nuqtai nazaridan ob’ekt yaratishning eng to‘g‘ri yo‘li — bu uning konstruktorini *new* operatsiyasiga chaqirishdir. *string* sinfida to‘qqizta konstruktor bor:

- string() — bo‘sh satrli ob’ekt yaratiladi;
- string (String str) — bir ob’ektdan boshqa ob’ekt yaratiladi, shuning uchun bu konstruktor kam ishlatiladi;
- string (StringBuf fer str) — BufferString sinfi ob’ektining almashtirilgan nusxasi;
- string(byte[] byteArray) — ob’ekt byteArray baytlar massividan yaratiladi;
- String (char [] charArray) — ob’ekt charArray Unicode simvollari massividan yaratiladi;

• String (byte [] byteArray, int offset, int count) — ob’ekt offset indeksdan boshlanadigan va count baytni o‘z ichiga olgan, byteArray baytlar massivi qismidan yaratiladi;
• String (char [] charArray, int offset, int count) — xuddi oldingidagidek, faqat massiv Unicode simvollaridan tuzilgan;
• String(byte[] byteArray, String encoding) — baytlar massiviga yozilgan simvollar, encoding kodirovkasini hisobga olgan holda, Unicode-satrda beriladi;
• String(byte[] byteArray, int offset, int count, String encoding) — xuddi oldingidagidek, faqat massiv qismi uchun.

offset indeksni, count yoki encoding kodirovkani noto‘g‘ri berganda, istisnoli holat yuzaga keladi.

byteArray baytlar massivini ishlatuvchi konstruktorlar, ASCII-kodirovkadagi simvollar baytlari massivlaridan Unicode-satrlarni yaratishga mo‘ljallangan. Bunday holat ASCII-fayllarni o‘qishda, ma’lumotlar bazasidan axborot chiqarayotganda yoki tarmoqda axborot uzatayotganda yuz berishi mumkin.

Eng oddiy holda kompilyator ikki baytli Unicode simvollarni olish uchun har bir bayt oldiga yuqori nol baytni qo‘shadi. Unicode kodirovkasidagi, Latin 1 kodlariga mos ' \u0000 ' — ' \u00ff ' oraliq hosil bo‘ladi. Kirilcha matnlar noto‘g‘ri chiqariladi.

Agar kompyuterda mahalliy o‘rnatish amalga oshirilgan, ya’ni “lokal o‘rnatilgan” (locale) bo‘lsa (MS Windows da bu Control Panel oynasida Regional Options utilitasi yordamida bajariladi), u holda kompilyator, bu o‘rnatishni o‘qib, mahalliy kodli sahifaga mos keluvchi Unicode simvollarini yaratadi. MS Windows ning ruschalashtirilgan variantida odatda bu SR1251 kodli sahifa.

Agar kirillik ASCII-matnli boshlang‘ich massiv SR1251 kodirovkada bo‘lsa, u holda Java satrlar to‘g‘ri yaratiladi. Kirillcha harflar Unicode kodirovkasidagi o‘z oralig‘i '\u0400'—'\u04FF' ga tushadi.

Ammo kirilchada, eng kamida yana to‘rtta kodirovka bor.

• MS-DOS da SR866 kodirovka ishlatiladi.
• UNIX da odatda KOI8-R kodirovkasi qo‘llaniladi.
• Apple Macintosh kompyuterlarida MacCyrillic kodirovkasi ishlatiladi.
• Kirilchaning xalqaro kodirovkasi ISO8859-5 ham mavjud;

Masalan, 11100011 (0xE3 o‘n oltilik shaklda) bayt SR1251 kodirovkasida kirilcha G harfni tasvirlaydi, SR866 kodirovkada esa— U harfini, KOI8-R kodirovkada — Ts harfini, ISO8859-5 da— u harfini, MacCyrillic da — g harfini.

Agar boshlang‘ich kirilcha ASCII-matn yuqoridagi kodirovkalardan birida bo‘lib, maxalliy kodirovka esa SR1251 bo‘lsa, u holda Java satrning Unicode-simvollari kirilchaga mos tushmaydi.

Bunday hollarda oxirgi ikkita konstruktor ishlatiladi. Ularning encoding parametrida satrlarni yaratishda konstruktorga qaysi kodlar jadvalini ishlatish kerakligi ko‘rsatiladi.

4.1 listing kirilcha matnni yozishning har xil holatini ko‘rsatadi. Unda uchta kodirovkadagi "Rossiya" so‘zini o‘z ichiga olgan uchta baytlar massivi yaratiladi.

- byteCP1251 massivi "Rossiya" so‘zini SR1251 kodirovkada o‘z ichiga oladi.
- byteSP866 "Rossiya" so‘zini SR866 kodirovkada o‘z ichiga oladi.
- byteKOI8R "Rossiya" so‘zini KOI8-R kodirovkada o‘z ichiga oladi.

Har bir massivdan uchta kodlar jadvalini ishlatgan holda uchtadan satr yaratiladi.

Bundan tashqari, s[] simvollar massividan s1 satr yaratiladi, SR866 kodirovkada yozilgan baytlar massividan s2 satr yaratiladi. Nihoyat, satr-konstantaga ilova yaratiladi.

1.3 listing. Kirilcha satrlarni yaratish.

```
class StringTest{
 public static void main(String[] args){
   String winLikeWin = null, winLikeDOS = null, winLikeUNIX = null;
   String dosLikeWin = null, dosLikeDOS = null, dosLikeUNIX = null;
   String unixLikeWin = null, unixLikeDOS = null, unixLikeUNIX = null;
   String msg = null;
   byte[] byteCp!251 = {
    (byte)0xD0, (byte)0xEE, (byte)0xFl,
    (byte)0xFl, (byte)0xES, (byte)0xFF
   };
   byte[] byteCp866 = {
    (byte)0x90, (byte)0xAE, (byte)0xE1,
    (byte)0xEl, (byte)0xA8, (byte)0xEF
   };
   byte[] byteKOISR = (
    (byte)0xF2, (byte)0xCF, (byte)0xD3,
    (byte)0xD3, (byte)0xC9, (byte)0xDl
   };
   char[] c = {'P', 'o', 'c', 'c', 'u', 'я'};
   String s1 = new String(c);
   String s2 = new String(byteCp866);   // MS Windows консоли учун
   String s3 = "Россия";
```

```
System.out.println();
try{
      // Cp866 да хабар, MSWindows консолига чиқ'ариш учун.
 msg = new String("\"Россия\" в ".getBytes("Cp866"), "Cpl251");
  winLikeWin = new String(byteCp1251, "Cpl251");  //To'g'ri
  winLikeDOS = new String(byteCpl251,: "Cp866");
  winLikeUNIX - new String(byteCp1251, "KOI8-R");
  dosLikeWin = new String(byteCp866, "Cpl251");  // Консол учун
  dosLikeDOS = new String(byteCp866, "Cp866");   // To'g'ri
  dosLikeUNIX = new String(byteCp866, "KOI8-R");
  unixLikeWin = new String(byteKOISR, "Cpl251");
  unixLikeDOS = new String(byteKOISR, "Cp866");
  unixLikeUNIX = new String(byteKOISR, "KOI8-R");  // To'g'ri
  System.out.print(msg + "Cpl251: ");
  System.out.write(byteCp1251);
  System.out.println();
  System.out.print(msg + "Cp866 : ");
  System, out.write (byteCp866} ;
  System.out.println();
  System.out.print(msg + "KOI8-R: ");
  System.out.write(byteKOI8R);
{catch(Exception e)(
  e.printStackTrace();
}
System.out.println();
System.out.println();
System.out.println(msg + "char array      : " + s1);
System.out.println(msg + "default encoding : " + s2);
System.out.println(msg + "string constant  : " + s3);
System.out.println();
System.out.println(msg + "Cp1251 -> Cp1251: " + winLikeWin);
System.out.println(msg + "Cp1251 -> Cp866 : " + winLikeDOS);
System.out.println(msg + "Cp1251 -> KOI8-R: " + winLikeUNIX);
System.out.println(msg + "Cp866 -> Cp1251: " + dosLikeWin);
System.out.println(msg + "Cp866 -> Cp866 : " + dosLikeDOS);
System.out.println(msg + "Cp866 -> KOI8-R: " + dosLikeUNIX);
System.out.println(msg + "KOI8-R -> Cpl251: " + unixLikeWin);
System.out.println(msg + "KOI8-R -> Cp866 : " + unixLikeDOS);
System.out.println(msg + "KOI8-R -> KOI8-R: " + unixLikeUNIX);
 }
}
```

Bu ma'lumotlarning barchasi MS Windows konsoliga chiqariladi. Bular 1.3. rasmda ko'rsatilgan.

Konsolning birinchi uchta satrida byteCP1251 , byteCP866 va byteKOI8R baytlar massivlari Unicode ga aylantirilmasdan chiqariladi. Bu java.io paketidagi FilterOutputStream sinfining write() metodi orqali bajariladi.

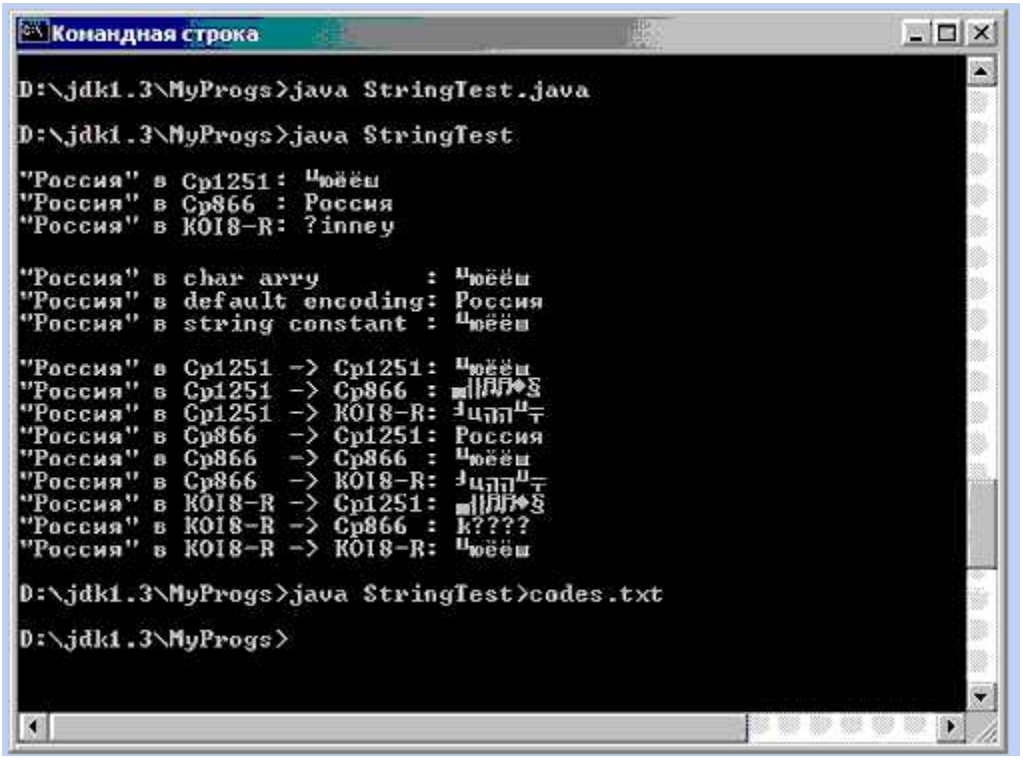
Keyingi konsoldagi uchta satrda s[] simvollar massividan olingan Java satr, byteCP866 massivi va satr-konstanta chiqarilgan.

Konsoldagi keyingi satrlar almashtirilgan massivlarni o‘z ichiga oladi.

Rasmdan ko‘rinadiki, konsolga faqat satrga SR1251 kodli jadvalni ishlatib yozilgangan SR866 kodirovkadagi massiv to‘g‘ri chiqarilgan.

Bu erda, simvollar oqimini konsolga yoki faylga chiqarishda ruschalashtirish muammosi yuzaga kelmoqda. Avval aytib o‘tilganidek, MS Windows operatsion tizimining Command Prompt konsol oynasiga matnlar SR866 kodirovkasida chiqariladi. Buni hisobga olish uchun, "\"Rossiya\" v" so‘zi SR866 kodirovkadagi simvollarni o‘z ichiga oluvchi baytlar massiviga aylantirilgan, keyin esa msg satriga o‘tkazilgan.

1.4. rasmdagi oxiridan bitta oldingi satrda dastur ishi natijalari codes.txt fayliga yo‘naltirilgan. MS Windows 2000 da matnni faylga chiqarish SR1251 kodirovkada amalga oshiriladi. 1.4. rasmda codes.txt fayli Notepad dasturi oynasida ko‘rsatilgan.



1.4. rasm. Kirilcha satrlarni faylga chiqarish

Yuqoridagilardan ko‘rinadiki, kirillcha harflar butunlay boshqacha ko‘rinish oldi. Kirilcha Unicode simvollar to‘g‘ri olinadi, agar bu yerda ham boshlang‘ich baytlar massivi yozilgan kodlar jadvali ishlatilsa.

Satr yaratishning yana bir usuli — bu ikkita *copyValueOf(chart[] charArray)* va *copyValueOf(char[] charArray, int offset, int length)* statik metodlarni ishlatishdir. Ular satrni berilgan simvollar massivi bo‘yicha yaratadi va uni ish natijasi sifatida qaytaradi. Masalan, dasturning quyidagi fragmenti bajarilganda

```
chart] c = ('C', 'u', 'м', 'в', 'о ¹ , 'л', 'л', 'u'};
String s1 = String.copyValueOf(c);
String s2 = String.copyValueOf(c, 3, 7);
```

s1 ob’ektida " Simvolli", s2 ob’ektida— " volli" satrlarini olamiz.

Satrlarni ulash

Satrlar bilan “+” belgisi bilan belgilanadigan satrlarni ulash(concatenation) amalini bajarish mumkin. Bu amal birinchi va ikkinchi satrlarning ulanishidan hosil bo‘lgan yangi satrni beradi. Uni konstantalarga ham, o‘zgaruvchilarga ham qo‘llash mumkin. Masalan:

```
StringAttention = "Diqqat: ";

String s = attention + "noma’lum simvol";
```

Ikkinchi amal — o‘zlashtirish += — chap tomondagi o‘zgaruvchiga nisbatan qo‘llaniladi:

```
attention += s;
```

+ amali sonlarni qo‘shishdan satrlarni ulash amaliga yuklangan bo‘lgani uchun, bu amalning imtiyozi masalasi paydo bo‘ladi. Satrlarni ulash amalining sonlarni qo‘shish amaliga nisbatan imtiyozi yuqori. Shuning uchun, "2" + 2 + 2 deb yozsak, " 222 "satrni olamiz. Ammo, 2 + 2 + "2" deb yozib, "42" satrni olamiz, chunki amallar chapdan o‘ngga bajariladi. Agar "2" + (2 + 2) deb yozsak, u holda "24" ni olamiz.

Satrlarni manipulyatsiya qilish

String sinfida satrlar bilan ishlashning ko‘plab metodlari bor. Ular qanday imkoniyatlar berishini ko‘rib chiqamiz.

Satr uzunligini aniqlash

Satr uzunligini bilish uchun, ya’ni undagi simvollar sonini aniqlash uchun, length() metodiga murojaat qilish kerak:

```
String s = "Write once, run anywhere.";
```

int len = s.length();

yoki yanada soddaroq

int len = "Write once, run anywhere.".length();

chunki satr-konstanta — string sinfining to‘la qonli ob’ekti. E’tibor berish kerak, satr — bu massiv emas, unda length maydoni yo‘q.

Simvolni satrdan tanlash

ind (birinchi simvol indeksi nolga teng) indeksli simvolni charAt(int ind) metodi yordamida tanlash mumkin. Agar indeks ind manfiy yoki satr uzunligidan katta bo‘lsa, istisnoli holat yuzaga keladi. Masalan, quyidagicha aniqlangandan keyin

char ch = s.charAt(3);

ch uzgaruvchi 't' qiymatga ega bo‘ladi

Satrning barcha simvollarini simvollar massivi sifatida, simvollar massivini qaytaruvchi *toCharArray()* metod yordamida olish mumkin.

Agar simvollar massivi dst ga, massivning ind indeksidan, begin indeksdan boshlanib end indeksda tugaydigan ostsatr kiritilsa, u holda void turidagi metod *getChars(int begin, int end, char[] dst, int ind)* metod ishlatiladi.

Massivda end - begin ta simvollar yoziladi. Ular massivning ind indeksidan boshlab, ind + (end - begin) – 1 gacha bulgan elementlarini tashkil etadi.

Bu metod istisnoli holatlarni quyidagi hollarda yuzaga keltiradi:

- dst = null ilova;
- begin indeks manfiy;
- begin indeks end indeksdan katta;
- end indeks satr uzunligidan katta;
- ind indeks manfiy;
- ind + (end — begin) > dst.length.

Masalan, quyidagi bajarilgandan keyin

char[] ch = {'К', 'о', 'р', 'о', 'л', 'ь', ' ', 'л', 'е', 'т', 'а'};
"Пароль легко найти".getChars(2, 8, ch, 2);

natija quyidagicha bo‘ladi:

ch = {'К', 'о', 'р', 'о', 'л', 'ь ', ' ', 'л', 'е', 'м', 'а'};

Agar ASCII baytli kodirovkadagi barcha simvollarni o‘z ichiga olgan baytlar massivini olish kerak bo‘lsa, u holda *getBytes()* metodi ishlatiladi.

Bu metod simvollarni Unicode dan ASCII ga o‘tkazayotganda lokal kodlar jadvalini ishlatadi. Agar baytlar massivini lokal kodirovkada emas, balki biror boshqa kodirovkada olish kerak bo‘lsa *getBytes(String encoding)* metodi ishlatiladi.

4.1 listingda msg ob’ektini yaratishda shunday qilingan. "\'Rossiya v\"" satri Windows 2000 operatsion tizimning Command Prompt konsol oynasida kirilchani to‘g‘ri chiqarish uchun SR866-baytlar massiviga qayta kodlashtirilgan.

Ostsatrni tanlash

substring(int begin, int end) metodi begin indeksli simvoldan end indeksli simvolgacha bo‘lgan ostsatrni ajratadi. Ostsatr uzunligi end - begin ga teng bo‘ladi.

substring (int begin) metodi ostsatrni begin indeksli simvoldan satr oxirigacha ajratadi. Agar indekslar manfiy bo‘lsa, end indeks satr uzunligidan katta yoki beginend dan katta bo‘lsa, u holda istisnoli holat yuzaga keladi.

Masalan, quyidagi bajarilgandan keyin

String s = "Write once, run anywhere.";
String sub1 = s.substring(6, 10);
String sub2 = s.substring(16);

sub1 satrda " once " qiymatni, sub2 da esa— " anywhere " qiymatni olamiz.

Satrlarni taqqoslash

Taqqoslash == amali satrga ilovani mos qo‘yadi. U ilova ikkala holda ham aynan bir satrni ko‘rsatadimi yoki yo‘qligini aniqlaydi. Masalan,

String s1 = "Qandaydir satr";

String s2 = "Boshqa satr";

bo‘lsa, s1 == s2 taqqoslash false qiymatni beradi.

true qiymat, ikkala ilova ham birdan bir satrni ko‘rsatsa olinadi. Masalan, s1 = s2 o‘zlashtirishdan keyin.

Agar biz s2 ni:

String s2 == "Qandaydir satr";

deb aniqlasak, u holda s1 == s2 taqqoslash true natijani beradi. Chunki, kompilyator "Qandaydir satr" konstantasining faqat bir ekzempilyarini yaratadi va unga ilovani yo‘naltiradi. Agar ilovani emas, balki satr ichini taqqoslash kerak bo‘lsa, buning uchun ham bir necha metodlar mavjud. equals (object obj) mantiqiy metodi object sinfidan qayta aniqlangan, agar obj argument null ga teng bo‘lmasa, string sinfi ob’ekti bo‘lsa va uning ichidaga satr berilgan satr bilan to‘la ustma ust tushsa, u holda bu metod *true* qiymat qaytaradi. Qolgan barcha hollarda u *false* qiymat qaytaradi. *equalsIgnoreCase(object obj)* mantiqiy metodi xuddi oldingidagidek ishlaydi, ammo, har xil registrdagi bir xil harflar ustma ust tushadi deb hisoblanadi. Masalan, *s2.equals("boshqa satr")* natijada false qiymat beradi false, *s2.equalsIgnoreCase("boshqa satr")* esa, true qiymat qaytaradi.

compareTo(string str) metodi int turidagi butun sonni qaytaradi. Bu butun son quyidagi tartib asosida hisoblanadi:

1. Berilgan this satr simvollari va str satrning bir xil raqamli indekslarida joylashgan simvollar taqqoslanadi. Taqqoslash bir xil indeksda har xil simvol uchraguncha, masalan, k indeksda joylashgan simvollar, yoki satrlardan birida simvollar tugaguncha davom etadi.

2. Birinchi holda *this.charAt(k) - str.charAt(k)* qiymat, ya’ni, kodirovkasidagi birinchi mos tushmaydigan simvollar farqi qaytariladi.

3. Ikkinchi holda *this.length() - str.length()* qiymat, ya’ni satrlar uzinligidagi farq qaytariladi.

4. Agar satrlar ustma ust tushsa, 0 qiymat qaytariladi.

Agar str qiymati null bo‘lsa, u holda istisnoli holat yuzaga keladi.

Nol *equals()* metodi xuddi shu holda true qaytarganda qaytariladi.

compareToignoreCase(string str) metodi taqqoslashni harflar registrlarini hisobga olmagan holda bajaradi, ya’ni quyidagi metod bajariladi:

this.toUpperCase().toLowerCase().compareTo(

str.toUpperCase().toLowerCase());

Yana bir — *compareTo (Object obj)* metodi, agar obj satr bo‘lmasa istisnoli holatni yuzaga keltiradi. Boshqa hollarda u *compareTo(String str)* metodi kabi

ishlaydi. Bu metodlar lokal kodirovkada simvollarning alfavit bo‘yicha joylashishini hisobga olmaydi. Russcha harflar Unicode da alfavit bo‘yicha joylashgan. Bitta harf bundan istisno. Katta Yo harfi barcha kirilcha harflardan oldin joylashgan, uning kodi '\ u040l ', kichik e harfi esa — barcha ruscha harflardan keyin joylashgan, uning kodi '\ u0451 '.

Agar harflarning bunday joylashishi maqul kelmasa, java.text paketidagi *RuleBasedCollator* sinfi orqali xususiy joylashishni berish mumkin.

Berilgan this satrning ostsatrini boshqa str satrning xuddi shu len uzunlikdagi ostsatri bilan taqqoslashni *regionMatches(int ind1, String str, int ind2, int len)* metodi yordamida bajarish mumkin.

Bu erda ind1 — berilgan this satrning ostsatri boshlanish indeksi, ind2 — boshqa str satr ostsatrining boshlanish indeksi. Natija false quyidagi hollarda bo‘lishi mumkin:

- ind1 yoki ind2 indekslardan birortasi manfiy;
- ind1 + len yoki ind2 + len lardan birortasi mos satr uzunligidan katta;
- hech bo‘lmaganda simvollarning biror juftligi mos tushmaydi.

Bu metod har xil registrlarda yozilgan simvollarni farqlaydi. Agar ostsatrlar registrlarni hisobga olmagan holda taqqoslansa, u holda:

regionMatches(boolean flag, int ind1, String str, int ind2, int len) mantiqiy metodni ishlatish maqsadga muvofiq bo‘ladi. Agar birinchi flag parametr true ga teng bo‘lsa, u holda ostsatrlarni taqqoslashda harflar registri hisobga olinmaydi, agar false bo‘lsa— hisobga olinadi.

Satrdagi simvolni topish

Qidiruv har doim harflar registrini hisobga olgan holda bajariladi.

Berilgan this satrda ch simvolning birinchi marotaba uchrashini *indexOf(int ch)* metodi yordamida kuzatish mumkin. Bu metod uchragan simvol indeksini qaytaradi, yoki agar this satrda ch simvol uchramasa, -1 ni qaytaradi.

Masalan, *"Moskva", indexOf('o')* 1 soni natijasini beradi.

Albatta bu metod, siklda true qiymat qabul qilmaguncha, ketma ket taqqoslash *this.charAt(k++> == ch)* amalini bajaradi.

Berilgan this satrda ch simvolning ikkinchi va undan keyingi uchrashlarini *indexOf(int ch, int ind)* metodi yordamida kuzatish mumkin.

Bu metod ch simvolni ind indeksdan boshlab qidiradi. Agar ind <0 bo‘lsa, u holda qidiruv satr boshidan boshlanadi, agar ind satr uzunligidan katta bo‘lsa, simvol qidirilmaydi, ya’ni -1 qiymat qaytariladi.

Masalan, "Molotok".indexOf('o', indexOf ('o') + 1) 3 natijani beradi. .

Berilgan this satrda ch simvolning eng oxirgi marotaba uchrashini *lastIndexOf (int ch)* metodi yordamida kuzatish mumkin. U satrni teskari tartibda qaraydi. Agar ch simvol topilmasa, -1 qaytariladi. Masalan, "Molotok".lastindexOf('o') 5 natijani beradi. Berilgan this satrda ch simvolning oxiridan bitta oldin va bundan oldingi uchrashini lastIndexOf(int ch, int ind) metodi yordamida kuzatish mumkin. Bu metod satrni ind indeksdan boshlab teskari tartibda qaraydi. Agar ind satr uzunligidan katta bo‘lsa, u holda qidiruv satr oxiridan boshlanadi, agar ind <0 bo‘lsa, u holda -1 qaytariladi.

Ostsatrni topish

Qidiruv har doim harflar registrini hisobga olgan holda bajariladi. Berilgan *this* satrga sub ostsatrning birinchi marotaba kirishini *indexOf (String sub)* metodi yordamida qidirish mumkin. U berilgan satrga sub ostsatrning birinchi marotaba kirishidagi birinchi simvol indeksini qaytaradi, yoki berilgan this satrga sub ostsatr kirmasa, -1 qaytariladi. Masalan, " Raskraska ".indexOf ("ras") 4 natijani beradi.

Agar qidiruvni satr boshidan emas, balki biror ind indeksdan boshlab bajarish kerak bo‘lsa, u holda *indexOf (String sub, int ind)* metodi ishlatiladi. Agar ind <0 bo‘lsa, qidiruv satr boshidan boshlanadi, agar ind satr uzunligidan katta bo‘lsa, u holda simvol qidirilmaydi, ya’ni -1 qaytariladi. Berilgan this satrga sub ostsatrning oxirgi marotaba kirishini *lastindexOf (string sub)* metodi yordamida aniqlash mumkin. Bu metod berilgan this satrga sub ostsatrning oxirgi kirishidagi birinchi simvol indeksini qaytaradi. Agar ostsatr berilgan satrda uchramasa, -1 qaytariladi. Berilgan this satrning barcha joylariga emas, balki uning faqat bosh qismiga sub ostsatrning ind indeksidan boshlab kirishini *lastIndexOf(String stf, int ind)* metodi yordamida qidirish mumkin. Agar ind satr uzunligidan katta bo‘lsa, qidiruv satr ortidan boshlanadi, agar ind <0 bo‘lsa, u holda -1 ni qaytaradi. Berilgan this satr sub ostsatrdan boshlanishini startsWith(string sub) mantiqiy metod yordamida aniqlash mumkin. Agar this satr sub ostsatrdan boshlansa, yoki ostsatr bilan ustma ust tushsa, yoki sub ostsatr bo‘sh satr bo‘lsa, u holda bu metod *true* qaytaradi.

Berilgan this satrda sub ostsatrning ind indeksdan boshlab paydo bo‘lishini mantiqiy *startsWith(String sub,int ind)* metod yordamida tekshirish mumkin. Agar ind indeks manfiy yoki satr uzunligidan katta bo‘lsa, u holda *false* qaytaradi. Berilgan this satr sub ostsatr bilan tugashini tekshirishda *endsWitht(String sub)* mantiqiy metodi ishlatiladi. Agar sub ostsatr butun satr bilan ustma ust tushsa, yoki sub ostsatr bo‘sh bo‘lsa, u holda metod true qaytaradi. Masalan, if (fileName.endsWith(". Java")) boshlang‘ich matni Java da bo‘lgan fayllar nomlarini kuzatadi.

Agar sub == null bo‘lsa, yuqorida sanab o‘tilgan metodlarda istisnoli holat yuzaga keladi. Agar qidiruvni harflar registrini hisobga olmasdan amalga oshirish zarurati bo‘lsa, u holda dastlab satrning barcha simvollari uchun harf registrini o‘zgartirish kerak bo‘ladi.

Harf registrini o‘zgartirish

toLowerCase() metodi *satrning* barcha harflarini quyi registrga o‘tkazgan holda yangi satrni qaytaradi, ya’ni barcha harflarni quyi registrga o‘tkazadi.

toUpperCase() metodi satrning barcha harflarini yuqori registrga o‘tkazgan holda yangi satrni qaytaradi, ya’ni barcha harflarni yuqori registrga o‘tkazadi. Bunda lokal kodli jadval jimlikka ko‘ra ishlatiladi. Agar boshqa lokal kerak bo‘lsa, *toLowerCase(Localeloc)* va *toUpperCase(Localeloc)* metodlari qo‘llaniladi.

Alohida simvolni almashtirish

replace (int old, int new) metodi berilgan satrdagi barcha uchragan old simvolni new simvolga almashtirilgan holdagi yangi satrni qaytaradi. Agar old simvol satrda uchramasa, u holda boshlang‘ich satrga ilova qaytariladi.

Masalan, " Рука в руку сует хлеб " , *replace ('u', 'e')* amal bajarilgandan keyin " Река в реке сеет хлеб " satrini ni olamiz.

Almashtirishda harf registri hisobga olinadi.

Satr boshidagi va oxiridagi bo‘sh joylarni yo‘qotish

trim() metodi satr boshidagi va satr oxiridagi kodi '\u0020 ' dan oshmagan simvollarni o‘chiradi va hosil bo‘lgan yangi satrni qaytaradi.

Ma’lumotlarning boshqa turlarini satrga aylantirish

Java tilida — har bir sinfga boshqa turlarni shu sinf turlariga aylantirish ma’suliyati yuklangan va har bir sinf bu ish uchun zarur bo‘ladigan metodlarga ega bo‘lishi kerak.

string sinfi sodda *boolean, char, int, long, float, double turlarni, char[]* massivni va *Object* turini satrga aylantiruvchi sakkizta *valueOf (type elem)* statik metodiga ega.To‘qqizinchi *valueOf(char[] ch, int offset, int len)* metodi offset indeksdan boshlanadigan va len elementga ega bo‘lgan ch massiv qismini satrga aylantiradi. Bundan tashqari, har bir sinfda Object sinfidan nasl olingan yoki qayta aniqlangan *toString()* metodi mavjud.U sinf ob’ektlarini satrga aylantiradi. Haqiqatan, *valueOf()* metodi mos sinfning toString() metodini chaqiradi. Shuning uchun, aylantirish natijasi *toString()* metodining qanday joriy qilinganiga bog‘liq.

Yana bir sodda usul — biror bir turning elem qiymatini bo‘sh satr bilan ulash: "" + elem orqali amalga oshiriladi. Bu holda bilvosita elem. *toString()* metodi chaqiriladi.

StringBuffer sinfi

StringBuffer sinfi ob’ektlari — bular o‘zgaruvchan uzunlikdagi satrlardir. Xozirgina yaratilgan ob’ekt aniq sig‘imli(capacity) buferga ega bo‘ladi, jimlik bo‘yicha u 16 ta simvolni saqlash uchun etadi. Sig‘imnm ob’ekt konstruktorida berish mumkin. Bufer to‘lishni boshlashi bilan, uning sig‘imi, yangi simvollarni sig‘dirishi uchun, avtomatik tarzda kengaya boshlaydi.

Bufer sig‘imini ixtiyoriy paytda *ensureCapacity(int minCapacity)* metodiga murojaat qilib kengaytirish mumkin. Bu metod, agar minCapacity ob’ektda saqlanayotgan satr uzunligidan katta bo‘lsa, sig‘imni o‘zgartiradi. Sig‘im quyidagi tartibda o‘zgartiriladi. Bufer sig‘imi N ga teng bo‘lsin. U holda yangi sig‘im Max(2 * N + 2, minCapacity) ga teng bo‘ladi. Shunday qilib, bufer sig‘imini ikki marotabadan kamiga kengaytirib bo‘lmaydi. *setLength (int newLength)* metodi bilan satrning ixtiyoriy uzunligini o‘rnatish mumkin.

Agar u joriy uzunlikdan katta bo‘lsa, u holda qo‘shimcha simvollar ' \uOOOO'ga teng bo‘ladi. Agar u joriy uzunlikdan kichik bo‘lsa, satr qirqiladi, oxiridagi simvollar yo‘qotiladi, aniqrog‘i, '\uOOOO' simvollar bilan almashtiriladi. Shu bilan birga sig‘im o‘zgarmaydi. Agar newLength soni manfiy son bo‘lsa, istisnoli holat yuzaga keladi. Shuning uchun ob’ektning yangi uzunligini o‘rnatayotganda ehtiyot bo‘lish kerak. Satrdagi simvollar sonini, String sinfi ob’ektlaridagidek, *length()* metodi bilan, sig‘imini esa — *capacity()* metodi bilan bilish mumkin. *StringBuffer* sinfi ob’ektlarini faqat konstruktorlar tomonidan yaratish mumkin.

Konstruktorlar

stringBuffer sinfida uchta konstruktor bor:

stringBuffer() — sig‘imi 16 simvolga teng bo‘lgan bo‘sh ob’ektni yaratadi;

stringBuffer.(int capacity) — berilgan capacity sig‘imli bo‘sh ob’ektni yaratadi;

StringBuffer(String str) — str satrni o‘z ichiga oladigan, str . length() + 16 sig‘imli ob’ektni yaratadi.

Ostsatrni qo‘shish

stringBuffer sinfida ostsatrni satr ortiga qo‘shadigan o‘nta *append()* metodlari mavjud. Ular satrning yangi ekzempilyarini yaratmaydi, balki ilovani shu satrning o‘ziga, o‘zgartirilgan ko‘rinishiga qaytaradi. *append (string str)* asosiy metod str satrni berilgan satr ortiga qo‘shadi. Agar ilova str == null bo‘lsa, u holda "null" qo‘shiladi.

Oltita *append (type elem)* metodlari satrga aylantirilgan sodda boolean, char, int, long, float, double turlarni qo‘shadi.

Ikkita metod: *append (char [] str)* va *append (char [.] , sub, int offset, int len)* satrga aylantirilgan str simvollar massivini va sub massiv qismini berilgan satrga ulaydi.

Uninchi metod *append (Object obj)* satrga ob’ektni qo‘shadi. Bundan oldin ob’ekt obj o‘z metodi *tostring()* yordamida satrga aylantirilgan bo‘lishi kerak.

Ostsatrni kiritish

O‘nta *insert()* metodlari metod parametrida ko‘rsatilgan satrni berilgan satrga kiritadi. Kiritiladigan joy metodning birinchi parametri ind orqali beriladi. Bu shunday indekski, kiritiladigan ob’ektni shu indeksli satr elementi oldiga kiritiladi. U manfiy emas va satr uzunligidan kichik bo‘lishi kerak, aks holda istisnoli holat yuzaga keladi. Satr siljiydi, bufer sig‘imi zaruratga qarab kengayyadi. Metodlar hosil qilingan yangi satrga ilovani qaytaradi. *insert (int ind, string str)* asosiy metod str satrni berilgan satrning ind indeksli elementi oldiga kiritadi. Agar str == null bo‘lsa, u holda "null" satr kiritiladi.

Masalan, quyidagi bajarilgandan keyin

String s = new StringBuffer("Bu katta satr"). insert(2, " juda").toString();

s == "Bu juda katta satr" ni olamiz.

sb.insert(sb.length(), "xxx") metodi, xuddi *sb.append("xxx")* metodi kabi ishlaydi.

Oltita *insert (int ind, type elem)* metodlari satrga aylantirilgan sodda: boolean, char, int, long, float, double turlarni kiritadi.

Ikkita: *insert(intind, chart] str)* va *insert(intind, char[] sub, intoffset, intlen)* metodlari satrga aylantirilgan str simvollar massivi va massiv qismi sub ni kiritadi.

O‘ninchi metod *insert (intind, Objectobj)* ob’ektni o‘z metodi *toString()* yordamida satrga aylantirilgan holda kiritadi.

Ostsatrni o‘chirish

delete (int begin, int end) metodi berilgan satrdan begin indeksdan boshlab end indeksgacha bo‘lgan simvollarni o‘chiradi, agar end satr uzunligidan katta bo‘lsa, u holda satr oxirigacha bo‘lgan simvollarni o‘chiradi.

Masalan, quyidagi bajarilganda

String s = new StringBuffer("Bu uncha katta bo‘lmagan satr").

delete(3, 6).toString();

s == "Bu katta bo‘lmagan satr" natijani olamiz.

Agar begin manfiy, satr uzunligidan katta yoki end dan katta bo‘lsa, u holda istisnoli holat yuzaga keladi. Agar begin == end bo‘lsa, hech nima o‘chirilmaydi.

Simvolni o‘chirish

deieteCharAt (int ind) metodi satrdan ko‘rsatilgan ind indeksli simvolni o‘chiradi. Satr uzunligi bir birlikka kamayyadi. Agar ind indeks manfiy yoki satr uzunligidan katta bo‘lsa, istisnoli holat yuzaga keladi.

Ostsatrni almashtirish

replace (int begin, int end. String str) metodi satrdan begin indeksdan boshlab end indeksgacha bo‘lgan simvollarni o‘chiradi. Agar end satr uzunligidan katta bo‘lsa, u holda satr oxirigacha bo‘lgan simvollarni o‘chiradi. Bu metod o‘chirish ishini bajarib bo‘lgach, o‘chirilgan simvollar o‘rniga str satrini qo‘yadi. Agar begin manfiy, satr uzunligidan katta yoki end dan katta bo‘lsa, u holda istisnoli holat yuzaga keladi. O‘z o‘zidan ayonki, *replace()* metodi — bu *delete()* va *insert()* metodlarining ketma ket bajarilishidir.

Satrni teskarisiga aylantirish

Reverse() metodi satrda simvollar joylashishi ketma ketligini teskarisiga aylantiradi.

Masalan, quyidagi bajarilganda

String s = new StringBuffer("Bu uncha katta bo‘lmagan satr"),

reverse().toString();

s == "rtas nagamlo‘b attak achnu uB" ni olamiz.

Satrlarni sintaksis tahlili

Kiritilgan matnni tahlil qilish masalasi — parsing (parsing) — saralash va qidirish kabi, dasturlashning azaliy masalasidir. Matnni har xil belgilariga ko‘ra tahlil qiladigan ko‘plab tahlilchi dasturlar—parserlar (parser) yozilgan. Hatto, tahlilchi dasturlarni berilgan tartibga ko‘ra generatsiya qiladigan: YACC, LEX va boshqalar kabi, maxsus dasturlar ham mavjud. Ammo masala haligacha dolzarb. Chunki tayyor dasturlarning nimasidir yangi dasturchiga mos kelmaydi va o‘z tahlilchi dasturini yaratishga harakat qiladi. *java.util* paketida satrlar tahlilini yengillashtiruvchi *StringTokenizer* sinfi joylashgan.

StringTokenizer sinfi

java.util paketidagi *StringTokenizer* sinfi uncha katta emas, unda uchta konstruktor va oltita metod mavjud.

Birinchi konstruktor *StringTokenizer (String str)* bo‘sh joylar, tabulyatsiya simvoli '\t', keyingi satrga o‘tish belgisi '\n', va karetkani qaytarish belgisi '\r' bilan satrni so‘zlarga ajratishga tayyor ob’ekt yaratadi. Ajratuvchi belgilar so‘zlar soniga kirmaydi.

Ikkinchi konstruktor *StringTokenizer(String str. String delimeters)* ajratuvchi belgilarni ikkinchi parametr delimeters da beradi, masalan:

StringTokenizer("Jazo qattiy, shikoyatga o‘rin yo‘q", " \t\n\r,:-");

Bu erda birinchi ajratuvchi belgi — bo‘sh joy. Undan keyin tabulyatsiya simvoli, keyingi satrga o‘tish belgisi, karetkani qaytarish belgisi, vergul, ikki nuqta, chiziqcha — defis kelmoqda. Ajratuvchi belgilarning delimeters satrida qanday ketma ketlikda joylashishining ahamiyati yo‘q. Ajratuvchi belgilar so‘zlar soniga kirmaydi.

Uchinchi konstruktor ajratuvchi belgilarni so‘zlar soniga kiritishga imkon beradi:

StringTokenizer(String str, String delimeters, boolean flag);

Agar flag parametr true ga teng bo‘lsa, u holda ajratuvchi belgilar so‘zlar soniga kiradi, agar false bo‘lsa — yo‘q. Masalan:

*StringTokenizer("a - (Ь + s) / Ь * s", " \t\n\r+*-/(), true);*

Satrni so‘zlarga ajratishda ikkita metod faol ishtirok etadi:

nextToken() metodi satr sifatida navbatdagi so‘zni qaytaradi;

hasMoreTokens() mantiqiy metodi, agar satrda yana so‘z bo‘lsa, true qaytaradi, va agar boshqa so‘z qolmagan bo‘lsa, false qaytaradi. Uchinchi metod *countTokens()* qolgan so‘zlar sonini qaytaradi.

To‘rtinchi metod *nextToken(string newDeiimeters)* ish jarayonida ajratuvchi belgini o‘zgartirish imkonini beradi. Navbatdagi so‘zlar yangi ajratuvchi belgi newDelimeters bilan ajratiladi. Yangi ajratuvchi belgilar konstruktorda yoki oldingi *nextToken()* metodida aniqlangan eskisining o‘rniga amal qilinadi. Qolgan ikkita metod *nextEiement()* va *hasMoreEiements()* Enumeration interfeysni joriy qiladi. Ular bevosita *nextToken()* va *hasMoreTokens()* metodlariga murojaat qiladi.

Sxema juda sodda (1.4.2 listing).

1.4.2 listing. Satrni so‘zlarga ajratish:

String s = "Biz so‘zlarga ajratmoqchi bo‘lgan satr";

StringTokenizer st = new StringTokenizer(s, " \t\n\r,.");

while(st.hasMoreTokens()){

// So‘zlarni ajratib olamiz va ular o‘stida qandaydir amallar bajaramiz, masalan, ekranga //chiqaramiz.

System.out.println(st.nextToken()) ;

}

Hosil qilingan so‘zlar odatda, matnni keyinchalik qayta ishlashga qulay bo‘lgan konteynerga, ya’ni, biror bir sinf-kollektsiyaga: Vector, Stack yoki boshqasiga kiritiladi.

1 bob buyicha xulosalar

Ushbu bobda java dasturlash tili buyicha boshlang’ich ma’lumotlar berildi. Qushimcha ma’lumotlar olish uchun boshlang’ich kodlarni tahlil qilish kerak. Foydalanilgan barcha metodlarning boshlang‘ich matnlarini ko‘rish mumkin. Ular JDK tarkibiga kiradi. Boshlang‘ich matnlarni tahlil qilib, metodning ishlashi to‘g‘risida to‘liq tasavvurga ega bo‘lish mumkin. JDK ning oxirgi versiyalarda boshlang‘ich matnlar o‘ralgan holda JDK ning bosh katalogidagi src.jar faylda saqlanadi. Uni ochish uchun quyidagi:

D:\jdkl.7 > jar -xf src.jar amalni bajarish kerak. Bu amaldan keyin jdk1.7 katalogida src ostkatalog hosil bo‘ladi, uning ichida esa, JDK ning paketlari va qismpaketlariga mos, boshlang‘ich matnlarni o‘z ichiga olgan fayllardan iborat, ostkataloglar ochiladi.

Nazorat savollari

1. Satr simvolli massivdan qanday farq qiladi?
2. Satrlarni ulash xususiyatlarini ko‘rsating.
3. Simvollarni olish qanday amalga oshiriladi?
4. Satrlarni solishtirish qanday amalga oshiriladi?
5. Simvolni yoki ostki satrni izlash usulini ko‘rsating.

II. BOB. JAVA DA OB’EKTGA YO‘NALTIRILGAN DASTURLASH ASOSLARI

2.1. Sinflar va ob’ektlar

Sinf

Sintaksis bo‘yicha, JAVA da sinf – bu mavjud bo‘lgan turlar asosida yangi yaratilgan strukturlangan tur.

Sinf ta’rifi sodda shakli:

<sinf_turi> <sinf_nomi>{<sinf_komponentlari_ro‘yxati>};

bu erda:

sinf_tipi –class xizmatchi so‘zi;

sinf_nomi – identifikator;

sinf_komponentlari_ro‘yxati – sinfga tegishli ma’lumotlar va funksiyalar ta’rifi.

Funksiya – bu ob’ektlar ustida bajariladigan operatsiyalarni aniqlovchi sinf usuli.

Ma’lumotlar – bu ob’ekt strukturasini hosil qiluvchi maydon.

Sinf ob’ekti (sinf nusxasi) ni ta’riflash uchun quyidagi konstruktsiyadan foydalaniladi:

<sinf_nomi> <ob’ekt_nomi>;

Ob’ekt orqali maydonlarga va usullarga quyidagicha murojaat qilish mumkin:

< ob’ekt_nomi >. <maydon_nomi>

< ob’ekt_nomi >. <usul_nomi>

Komponentalarga murojaat xuquqlari

Komponentalarga murojaat hukuki murojaat spetsifiqatorlari yordamida boshkariladi: public, private, protected.

Umumiy (public) komponentalar dasturni ixtiyoriy qismida murojaat xuquqiga ega. Ulardan, ixtiyoriy funksiya ushbu sinf ichida va sinf tashqarida foydalansa ham bo‘ladi.

Xususiy (private) komponentalar sinf ichida murojaat xuquqiga ega, lekin sinf tashqarisidan esa murojaat kilish mumkin emas. Komponentalardan ushbu ular

tavsiflangan sinfdagi funksiya - a’zolari orqali foydalanish mumkin. Ximoyalangan (protected) komponentalar sinf ichida va hosila sinflarda murojaat xuquqiga ega. JAVA tilida agar sinf ta’rifida class so‘zi ishlatilgan bo‘lsa hamma komponentalari umumiy hisoblanadi.

Konstruktor

Konstruktor - bu sinf ob’ektlarini avtomatik initsializatsiya kilish uchun ishlatiladigan maxsus komponentali funksiya.

Konstruktorlar ko‘rinishi quyidagicha bulishi mumkin:

<Sinf nomi> (<formal parametrlar ro‘yxati>)

{<konstruktor tanasi>}

Bu komponenta funksiya nomi sinf nomi bilan bir xil bulishi lozim.

Dasturchi tomonidan ko‘rsatilmagan holda ham new operator yordamida sinf ob’ekti yaratilganda yoki xotirada joylashganda konstruktor avtomatik ravishda chaqiriladi. Konstruktor ob’ekt uchun xotirada joy ajratadi va ma’lumotlar – sinf a’zolarini initsializatsiyalaydi. Konstruktorlar uchun kaytariluvchi tiplar, xatto void tipi ham ko‘rsatilmaydi Konstruktorlar ixtiyoriy sinflar uchun doimo mavjud, lekin agarda u ko‘rsatilgan holda tavsiflanmagan bo‘lsa, u avtomatik ravishda yaratiladi. Ko‘rsatilmagan holda parametrsiz konstruktor va nusxa konstruktori yaratiladi. Agarda konstruktor ochiq holda tavsiflangan bo‘lsa, unda ko‘rsatilmagan holda konstruktor yaratilmaydi. Ko‘rsatilmagan holda umumiy (public) konstruktorlar yaratiladi.

Misol:

```
class Point { int x, y;
Point(int x, int y) {
this.x = x;
this. y = y;
}
Point() {
x = 0;
y = 0;
}
double distance(int x, int y) {
int dx = this.x - x;
int dy = this.y - y;
return Math.sqrt(dx*dx + dy*dy);
}
```

```
double distance(Point p) {
return distance(p.x, p.y);
} }
class PointDist {
public static void main(String args[]) {
Point p1 = new Point(0, 0);
Point p2 = new Point(30, 40);
System.out.println("p1 = " + pi.x + ", " + p1.y);
System.out.println("p2 = " + p2.x + ", " + p2.y);
System.out.println("p1.distance(p2) = " + p1.distance(p2));
System.out.println("p1.distance(60, 80) = " + p1.distance(60, 80));
} }
```

Quyida dastur bajarilish natijasi keltirilgan:

```
C:\> java PointDist
p1 = 0, 0
p2 = 30, 40
p1.distance(p2) = 50.0
p1.distance(60, 80) = 100.0
```

This ko‘rsatkichi

Agarda konkret ob’ektga ishlov berish uchun sinf a’zosi – funksiya chaqirilsa, unda shu funksiyaga ob’ektga belgilangan ko‘rsatkich avtomatik va ko‘rsatilmagan holda uzatiladi. Bu ko‘rsatkich this ismiga ega.

Sinfni quyidagicha tasvirlash mumkin:

```
class Point { int x, u;

Point(int x, int u) {

this.x = x;

this.u = u;

} }
```

Usullarni qo‘shimcha yuklash

Java tilida bir xil nomli turli parametrlar ro‘yxatiga ega usullar yaratish mumkin. Bunday texnika usullarni qo‘shimcha yuklash (method overloading) deb ataladi. Qo‘shimcha yuklangan usullar konstruktorlar bo‘lishi shart emas.

Misol uchun:

```
class Point { int x, y;
Point(int x, int y) {
this.x = x;
this. y = y;
}
Point() {
x = 0;
y = 0;
}
double distance(int x, int y) {
int dx = this.x - x;
int dy = this.y - y;
return Math.sqrt(dx*dx + dy*dy);
}
double distance(Point p) {
return distance(p.x, p.y);
}
}
class PointDist {
public static void main(String args[]) {
Point p1 = new Point(0, 0);
Point p2 = new Point(30, 40);
System.out.println("p1 = " + pi.x + ", " + p1.y);
System.out.println("p2 = " + p2.x + ", " + p2.y);
System.out.println("p1.distance(p2) = " + p1.distance(p2));
System.out.println("p1.distance(60, 80) = " + p1.distance(60, 80));
}}
```

Bu dasturning ishlash natijasi quyidagicha:

```
C:\> java PointDist
p1 = 0, 0
p2 = 30, 40
p1.distance(p2) = 50.0
p1.distance(60, 80) = 100.0
```

finalize

Java tilida **finalize** nomli usullar kiritish imkoniyati mavjud. Bu nomli usullar C++ tilidagi (kalit belgi ~) va Delphi tilidagi (kalit so‘z destructor) destruktorlarga mosdir. Java bajarish muxiti har gal ob’ektni o‘chirishda shu uslni chaqiradi.

Sinfning statik komponentalari

Sinf kompanentasi yagona bo‘lib va hamma yaratilgan ob’ektlar uchun umumiy bulishi uchun uni statik element sifatida ta’riflash ya’ni static atributi orqali ta’riflash lozimdir. Ob’ektlarni yaratishda sinf statik ma’lumotlari takrorlanmaydi, ya’ni har bir statik komponentlar birdan-bir ko‘rinishda mavjud. Statik usullar faqat boshqa statik usullarga to‘g‘ridan to‘g‘ri murojaat qilishi mumkin va ularda this ilovasidan foydalanish mumkin emas. O‘zgaruvchilar ham static tipga ega bo‘lishi mumkin, bu holda ularga global o‘zgaruvchilarga o‘xshab dastur ixtiyoriy qismidan murojaat qilish mumkin. Statik usullar ichida nostatik o‘zgaruvchilarga murojaat qilish mumkin emas. Sinf statik ma’lumotlarga faqatgina ob’ekt ismi orqali murojaat etish mumkin.

<ob’ekt_nomi>.<komponenta_nomi>

Masalan:

complex a; a.count=5;

Lekin, statik komponetlarga sinf ob’ekti aniqlanmagan holda ham murojaat etish mumkin. Statistik komponentlarga nafaqat ob’ekt ismi, balki sinf ismi orqali ham murojaat etish mumkin.

<sinf_nomi>. <komponenta_nomi>

Lekin shunday murojaat faqatgina public komponentlarga tegishli.

private statik komponentlarga tashqaridan murojaat etishda funksiya – statik komponentlardan foydalaniladi.

Bu funksiyalarni sinf ismi orqali chaqirish mumkin.

<sinf_nomi> .: <statik_funksiya_nomi>

Misol.

```
class StaticClass {
static int a = 42;
static int b = 99;
static void callme() {
System.out.println("a = " + a);
} }
class StaticByName {
public static void main(String args[]) {
StaticClass.callme();
System.out.println("b = " + StaticClass.b);
```

} }
Dastur bajarilishi natijasi:

S:\> Java StaticByName

a = 42 b = 99

Nazorat savollari:

1. Sinf qanday ta’riflanadi?

2. Sinf usullarini qo‘shimcha yuklash mumkinmi?

3. Konstruktorlar vazifasini ko‘rsating.

4. Ob’ektlar massivi yaratilganda qanday konstruktorlar chaqiriladi?

5. Statik komponentalar xususiy bo‘lishi mumkinmi?

2. 2. Sinflar va ob’ektlar bilan ishlash xususiyatlari

Runtime

Runtime sinfi Java interpretatorini inkapsulyatsiya qiladi. Bu sinf ob’ektini yaratish mumkin emas, lekin statik usulidan foydalanib ishlab turgan ob’ektga murojaat qilish mumkin. Odatda appletlar va boshqa dasturlar sinf usulini chaqirganda SecurityException istisnosi hosil bo‘ladi. Runtime ob’ektini to‘xtatish uchun exit(int code) usulini chaqirish etarli.

Xotirani boshqarish

Java tilida xotira avtomatik tozalansa ham dastur effektivligini tekshirish uchun ―uyum‖ xajmini aniqlash va erkin xotira xajmini xisoblash mumkin. Bu ma’lumotni aniqlash uchun totalMemory va freeMemory usullaridan foydalanish mumkin. Zarur bo‘lganda xotira tozalovchini gc usulini chaqirib avtomatik ishga tushirish mumkin. Dasturga zarur xotirani xisoblash uchun avval gc, so‘ngra free-Memory usullarini chaqirish kerak. Shundan so‘ng dasturni ishga tushirib freeMemory usuli chaqirilsa, dastur qancha xotira ishlatishini aniqlash mumkin.

Boshqa dasturlarni bajarish

Xavfsiz muhitlarda Java tilidan boshqa jarayonlarni ishga tushirish uchun foydalanish mumkin. Buning uchun exes usulining bir necha shakllaridan foydalanish mumkin. Usulga dastur nomi va bir necha parametrlar uzatiladi.

Quyidagi misolda Windows uchun xos ravishda exec usulidan foydalanib, notepad sodda matn muharriri jarayoni ishga tushiriladi. Misol tariqasida muharrirga Java fayllaridan biri uzatiladi. E'tibor bering — exec avtomatik ravishda "/" simvollarni Windowsga xos bo'lgan — "\" simvollarga almashtiradi.

```
class ExecDemo {
public static void main(String args[]) {
Runtime r = Runtime. getRuntime();
Process p = null;
String cmd[] = { "notepad", "/java/src/java/lang/Runtime.java" };
try {
p = r.exec(cmd);
} catch (Exception e) {
System.out.println("error executing " + cmd[0]);
}
} }
```

System

System sinfida turli global funksiyalar va o'zgaruvchilar mavjud. Masalan

System.out.println() usuli.

Bundan tashqari *currentTimeMillis* usuli tizimli vaqtni 1970 yil 1 yanvaridan bo'lib o'tgan millisekundlarda qaytaradi. Massivdan nusxa olish uchun arraycopy usulidan foydalaniladi. Quyida bir massivdan ikkinchisiga nusxa olishga misol keltirilgan**.**

```
class ACDemo {
static byte a[] = { 65, 66, 67, 68, 69, 70, 71, 72, 73, 74 };
static byte b[] = { 77, 77, 77, 77, 77, 77, 77, 77, 77, 77 };
public static void main(
String args[]) {
System.out.println("a = " + new String(a, 0));
System.out.println("b = " + new String(b, 0));
System.arraycopy(a, 0, b, 0, a.length);
System.out.println("a = " + new String(a, 0));
System.out.println("b = " + new String(b, 0));
System.arraycopy(a, 0, a, 1, a.length - 1);
System.arraycopy(b, 1, b, 0, b.length - 1);
System.out.println("a = " + new String(a, 0));
System.out.println("b = " + new String(b, 0));
} }
```

Dasturdan ko'rish mumkinki massivdan o'ziga nusxa olish ham mumkin.

C:\> java ACDemo
a = ABCDEFGHIJ
b = MMMMMMMMM
a = ABCDEFGHIJ
b = ABCDEFGHIJ
a = AABCDEFGHI
b = BCDEFGHIJJ

Muxit xossalari

Java bajarish muxiti *Properties* sinfi ob'ekti orqali muxit o'zgaruvchilariga murojaat qilishga imkon beradi. Xossalar to'liq ro'yxatini olish uchun *System.getProperties()* usulini chaqirish lozim.

Jadval

Tizimning standart xossalari

Ism	Qiymat	Applet uchun ruxsat
Java.version	Java interpretatori versiyasi	ha
Java.vendor	Foydalanuvchi kiritgan identifikator qatori	ha
java.vendor.url	Ishdab chiqaruvchi URLi	ha
java.class.version	Java API versiyasi	ha
java.class.'ath	CLASSPATH o'zgaruvchisi q'iymati	yo'q
java.home	Java muxiti installyatsiya q'ilingan katalog	yo'q
java.compiler	Kompilyator JIT	yo'q'
os.name	Operatsion tizim nomi	Ha
os.arch	Dastur bajarilayotgan kompyuter arxitekturasi	Ha
os.version	Web-tugun operatsion tizimi versiyasi	Ha
file.separator	Platformaga borliq fayl ajratuvchilari (/ yoki \)	Ha
'ath.separator	Platformaga borliq yo'l ajratuvchilari (: yoki ;)	Ha
line.se'arator	Platformaga bog'liq satr ajratuvchilari (\n yoki \r\n)	Ha
user.name	Joriy foydalanuvchi ismi	yo'q
user.home	Foydalanuvchi katalogi	yo'q
user.dir	Joriy ishchi katalog	yo'q
user.language	2-simvolli maxalliy til kodi	yo'q
user.region	2-simvolli mamlakat til kodi	yo'q
user.timezone	Ko'zda tutilgan vaqt zonasi	yo'q
user.encoding	Ko'zda tutilgan bo'yicha belgilar kodi	yo'q
user.encoding.'kg	Maxalliy simvollarni Unicode kodiga o'tkazish uchun konvertorlar paketi	yo'q

Date

Sana va vaqt bilan ishlash uchun **Date** sinfidan foydalaniladi. U orqali sana, yil , oy, xafta kuniga, soat, minut, sekundga murojaat qilish mumkin. Bu sinf turli konstruktorlari mavjud. Eng soddasi — Date() —ob'ektni joriy sana va vaqt bilan intsializatsiya qiladi. Qolgan uch konstruktor qo'shimcha imkoniyatlarga ega.

- *Date(year, month, date)* — ko'rsatilgan sanani o'rnatadi, vaqt 00:00:00 (tungi) qiymatga ega bo'ladi.

- *Date(year, month, date, hours, minutes)* — ko'rsatilgan sana va vaqtni o'rnatadi, sekund 0 qiymatni oladi.

- *Date(year, month, date, hours, minutes, seconds)* — eng to'liq ko'rinishi, sana va vaqt hamda sekundlar ham o'rnatiladi.

get va set

Sinf Date sinfi ob'ekt atributlarini o'rnatish uchun usullarga ega. Bu oilaga kiruvchi usullar — getYear, getMonth, getDate, getDay, getHours, getMinutes i getSeconds — butun qiymat qaytaradi.

Date sinfi qiymatini getTime usuli long tipidagi son sifatida qaytaradi. Bu son 1970 yil 1 yanvaridan o'tgan millisekundlarga teng.

Solishtirish

Date tipidagi ikki ob'ektni solishtirish uchun, sanani millisekundlarga aylantirish lozim. Date sinfi to'g'ridan to'g'ri solishtirish uchun uch usulga ega: — before, after va equals. Masalan

new Date(96, 2, 18).before(new Date(96, 2, 12)

true qaytaradi, chunki oyning 12-kuni 18-kunidan oldin keladi.

Satr va sana

Date ob'ektini turli formatdagi matnga konvertatsiya qilish mumkin. Avvalambor toString usuli Date ob'ektini quyidagicha satrga almashtiradi —Thu Feb 15 22:42:04 1996. Keyingi toLocaleString usuli sanani qisqaroq satrga almashtiradi, masalan: —02/15/96 22:42:04. Va nixoyat toGMTString usuli sanani Grinvich bo'yicha o'rtacha vaqt formatiga o'tkazadi: —16 Feb 1996 06:42:04 GMT.

Math

Math sinfi geometriya va trigonometriyada ishlatiladigan suzuvchi vergulli funksiyalarga ega. Bundan tashqari xisoblashlarda ishlatiladigan ikki konstanta mavjud: — E (taxminan 2.72) va PI (taxmina 3.14159).

Trigonometrik funksiyalar

Quyida keltirilgan uch funksiya radianlarda burchakni ifodalovchi double tipidagi parametrga ega bo‘lib, mos trigonometrik funksiya parametrini qaytaradi.

- *sin(double a)* radianda berilgan a burchak sinusini qaytaradi.
- *cos(double a)* radianda berilgan a burchak kosinusini qaytaradi.
- *tan(double a)* radianda berilgan a burchak tangensini qaytaradi.

Keyingi to‘rt funksiya uzatilgan parametr qiymatiga mos burchakni radianlarda qaytaradi.

- *asin(double r)* sinusi g ga teng burchakni qaytaradi.
- *acos(double r)* kosinusi g ga teng burchakni qaytaradi.
- *atan(double r)* tangensi g ga teng burchakni qaytaradi.
- *atan2(double a, double b)* tangensi a/b ga teng burchakni qaytaradi.

Darajaga ko‘tarish, eksponenta va logarifm funksiyalari

- *pow(double u, double x)* x darajaga ko‘tarilgan u qaytaradi. Masalan, pow(2.0, 3.0) teng 8.0.
- *exp(double x)* e darajasi x qaytaradi.
- *log(double x)* x natural logarifmini qaytaradi.
- *sqrt(double x)* x kvadrat ildizini qaytaradi.

Yaxlitlash

- *ceil(double a)* qiymati a dan katta yoki a ga teng bo‘lgan eng kichik butun son qaytaradi.

- *floor(double a)* qiymati a dan kichik yoki a ga teng bo‘lgan eng katta butun son qaytaradi.
- *rint(double a)* kasr qismi olib tashlangan double tipida a qiymatini qaytaradi.
- *round(float a)* eng yaqin butun songa yaxlitlangan a qiymatini qaytaradi.
- *round(double a)* eng yaqin uzun butun songa yaxlitlangan a qiymatini qaytaradi.

Bundan tashqari Math sinfida int, long, float va double tiplari bilan ishlovchi modul olish, minimal va maksimal qiymatni topish usullari polimorf versiyalari mavjud:

- *abs(a)* a moduli (absolyut qiymati) ni qaytaradi.
- *max(a, b)* o‘z argumentlari eng kattasini qaytaradi.
- *min(a, b)* o‘z argumentlari eng kichigini qaytaradi.

Random

Random — psevdotasodifiy sonlar generatori bo‘lib, unda ishlatilgan algoritm Donald Knut ―Dasturlash san’ati kitobining 3.2.1 bo‘limida keltirilgan. Odatda boshlang‘ich qiymat sifatida joriy vaqt olinadi, bu esa qaytariluvchi tasodifiy sonlar olinishi extimolligini kamaytiradi.

Random sinfi ob’ektidan 5 turdagi tasodifiy sonlarni olish mumkin. Bu tip diapazoni bo‘yicha bir tekisda taqsimlangan butun sonni olish uchun nextInt usulidan foydalaniladi. Shunga o‘xshash nextLong usuli long tipidagi tasodifiy sonni qaytaradi. Bundan tashqari *nextFloat* va *nextDouble* usullari mos ravishda float va double tipidagi, 0.0..1.0 intervalda tekis taqsimlangan sonlarni qaytaradi. Va nixoyat *nextGaussian* usuli o‘rta qiymati 0.0 va dispersiyasi 1.0 bo‘lgan normal taqsimlangan tasodifiy son qaytaradi.

Nazorat savollari

1. Bo‘sh xotira xajmini qanday aniqlash mumkin?
2. Muxit o‘zgaruvchilariga qanday murojaat qilish mumkin?
3. Vaqt va sana bilan qanday sinf ishlaydi?

4. Matematik sinf usullarini ko‘rsating.

5. Qaysi sinf tasodifiy sonlar generatsiyasi uchun ishlatiladi?

2.3. Sinflarda vorislik
Voris sinf

Vorislik o‘zining barcha ajdodlarining xususiyatlari, ma’lumotlari, metodlari va voqealarini meros qilib oladigan hosila sinfini e’lon qilish imkoniyatini beradi, shuningdek yangi tavsiflarni e’lon qilishi hamda meros sifatida olinayotgan ayrim funksiyalarni ortiqcha yuklashi mumkin. Bazaviy sinfning ko‘rsatib o‘tilgan tavsiflarini meros qilib olib, yangi tug‘ilgan sinfni ushbu tavsiflarni kengaytirish, toraytirish, o‘zgartirish, yo‘q qilish yoki o‘zgarishsiz qoldirishga majburlash mumkin.

JAVA tilida to‘g‘ridan to‘g‘ri ajdod supersinf deb ataladi.

Hosila sinfni e’lon qilishning umumlashgan sintaksisi:

class <sinf nomi>: [<kirish xuquqini beruvchi setsifiqator >] <ajdod sinf nomi> {…}

Konstruktor va destruktorlarda vorislik

Konstruktorlar meros bo‘lmagani uchun, hosila sinfni yaratishda undan meros bo‘lgan ma’lumot – a’zolari asosiy(bazaviy) sinf konstruktori orqali initsializatsiyalanishi lozim. Asosiy sinf konstruktori avtomatik ravishda chaqiriladi va hosila sinfni konstruktoridan oldin bajariladi. Asosiy (bazaviy) sinfni konstruktorining parametrlari hosila sinfni konstruktorni aniqlashda ko‘rsatiladi. Shunday qilib argumentlarni hosila sinfni konstruktoridan asosiy (bazaviy) sinfni konstruktoriga uzatish vazifasi bajariladi.

Asos sinf konstruktori parametri hosila sinf konstruktori ta’rifida **super** kalit so‘zi yordamida ko‘rsatiladi.

Misol:

```
class Point3D extends Point { int z;
Point3D(int x, int y, int z) {
super(x, y); // Bu yerda supersinf κonsruktori  this.z=z chaqiriladi;
public static void main(String args[]) {
Point3D p = new Point3D(10, 20, 30);
System.out.println( " x = " + p.x + " y = " + p.y +
" z = " + p.z);
```

} }

Dastur bajarilish natijasi:

C:\> java Point3D
x = 10 y = 20 z = 30

Sinf ob’ektlari pasdan tepaga qarab konstruktorlanadi: avvalo asosiy(bazaviy), keyin esa komponent – ob’ektlar (agarda ular mavjud bo‘lsa), undan keyin esa hosila sinfning o‘zi. Shunday qilib, hosila sinfning ob’ekti quyi ob’ekt sifatida asosiy (bazaviy) sinf ob’ektini o‘z ichiga oladi. Ob’ektlar teskari tartibda o‘chiriladi: avvalo hosila, keyin uning komponent – ob’ektlari, undan keyin esa asosiy(bazaviy) ob’ekt. Shunday qilib ob’ektni o‘chirish tartibi uning konstruktorlash tartibiga nisbatan teskari bo‘ladi.

Virtual funksiyalar

Virtual funksiyalar mexanizmiga biror komponent funksiyaning har bir hosilaviy sinfda alohida varianti mavjud bo‘lish lozim bo‘lganda murojaat qilinadi. Bunday funksiyalarga ega sinflar polimorf sinflar deb ataladi va ob’ektli dasturlashda alohida o‘ringa ega. Virtual funksiyalar kechki yoki dinamik bog‘lanish mexanizmi asoslangandir. Kechki bog‘lanishda erta bog‘lanishga o‘xshab adreslar statik ravishda kompilyatsiya jaryonida emas, balkim dinamik dastur bajarilishi jarayonida aniqlanadi. Bog‘lash jarayoni virtual funksiyalarni adreslar bilan almashtirishdan iborat. Virtual funksiyalar adreslar haqida ma’lumot saqlanuvchi jadvaldan foydalanadi. Virtuallik vorislikka o‘tadi. Funksiya virtual deb e’lon qilingandan so‘ng hosila sinfda qayta ta’rifi (shu prototip bilan) bu sinfda yangi virtual funksiyani yaratadi.

JAVA tilida hamma usullar virtualdir.

Misol.

Misolda sinf / supersinf sifatida bog‘langan ikki sinf ko‘rilgan bo‘lib, supersinf yagona usuli qayta ta’riflangan.

```
class A { void callme() {
System.out.println("Inside A's callrne method");
} }
class B extends A { void callme() {
System.out.println("Inside B's callme method");
}
```

```
}
class Dispatch {
public static void main(String args[]) {
A a = new B();
a.callme();
}
}
```

E’tibor bering main usuli ichida A sinfi o‘zgaruvchisi ta’riflanib, V sinfi ob’ekti yordamida initsializatsiya qilingan. Keyingi qatorda callme usuli chaqirilgan. Translyator A sinfida callme usuli mavjudligini tekshirdi, bajaruvchi tizim o‘zgaruvchida V ob’ekti saqlanganligi uchun, A sinfi emas, V sinfi callme usulini chaqiradi. Quyida dastur bajarilishi natijasi keltirilgan:

```
S:\> Java Dispatch

Inside B's calime method
```

Misolda usullarni dinamik tayinlash (dynamic method dispatch) mexanizmidan foydalanilgan.

Usullarni qayta ta’riflash

Point sinfining yangi ostki sinfi Point3D o‘z supersinfining distance usulini meros qilib oladi (misol PointDist.java). Lekin Point sinfida tekislikda nuqtalar orasida masofa qaytaruvchi distance(mt x, int u) usuli berilgan. Biz bu usulni uch o‘lchovli fazoga mos keladigan qilib, qayta ta’riflashimiz (override) lozim. Keyingi misolda distance usulini qo‘shimcha yuklash (overloading), va qayta ta’riflash (overriding) ko‘rsatilgan.

```
class Point { int x, y;
Point(int x, int y) {
this.x = x;
this.y = y;
}
double distance(int x, int y) {
int dx = this.x - x;
int dy = this.y - y:
return Math,sqrt(dx*dx + dy*dy);
}
double distance(Point p) {
return distance(p.x, p.y);
}
}
class Point3D extends Point { int z;
Point3D(int x, int y, int z) {
```

```
super(x, y);
this.z = z;
(
double distance(int x, int y, int z) {
int dx = this.x - x;
int dy = this.y - y;
int dz = this.z - z;
return Math.sqrt(dx*dx + dy*dy + dz*dz);
}
double distance(Point3D other) {
return distance(other.x, other.y, other.z);
}
double distance(int x, int y) {
double dx = (this.x / z) - x;
double dy = (this.y / z) - y; return Math.sqrt(dx*dx + dy*dy);
}
}
class Point3DDist {
public static void main(String args[]) {
Point3D p1 = new Point3D(30, 40, 10);
Point3D p2 = new Point3D(0, 0, 0);
Point p = new Point(4, 6);
System.out.println("p1 = " + p1.x + ", " + p1.y + ", " + p1.z);
System.out.println("p2 = " + p2.x + ", " + p2.y + ", " + p2.z);
System.out.println("p = " + p.x + ", " + p.y);
System.out.println("p1.distance(p2) = " + p1.distance(p2));
System.out.println("p1.distance(4, 6) = " + p1.distance(4, 6));
System.out.println("p1.distance(p) = " + p1.distance(p));
} }
```

Dastur bajarilishi natijasi:

```
C:\> Java Point3DDist
p1 = 30, 40, 10
p2 = 0, 0, 0
p = 4, 6
p1.distance(p2) = 50.9902
p1.distance(4, 6) = 2.23607
p1.distance(p) = 2.23607
```

Biz uch o‘lchovli va ikki o‘lchovli nuqtalar orasida kerakli masofani oldik. Misolda usullarni dinamik tayinlash (dynamic method dispatch) mexanizmidan foydalanilgan.

final

Hamma usullar va o‘zgaruvchilar ko‘zda tutilgan bo‘yicha qayta ta’riflanishi mumkin. Agar voris sinfda biror o‘zgaruvchi yoki usulni qayta ta’riflashga xaqqi yo‘qligini ko‘rsatish uchun ularni final (Delphi / C++ tilida virtual so‘zini yozmaslik kerak) deb ta’riflash lozim.

final int FILE_NEW = 1;

Qabul qilingan qoida bo‘yicha final tipidagi o‘zgaruvchilarni nomlashda faqat yuqori registrdagi simvollardan foydalaniladi. (C++ tilida preprotsessor konstantalar). Ba’zida final- usullardan foydalanish kod bajarilishini tezlashtiradi — chunki, translyator ularni joylashtiriluvchi (inline) kod deb e’lon qiladi (bayt-kod to‘g‘ridan to‘g‘ri kodga joylashtiriladi).

Abstrakt sinflar

Juda bo‘lmasa bitta abstrakt usulga ega sinf abstrakt sinf deb ataladi. Abstrakt usul deb quyidagi ko‘rinishga ega komponent funksiyaga aytiladi: abstract<tip> <ism > (< formal_parametrlar_ro‘yxati>); Bu sinf ob’ektlarini yaratish mumkin emas. Abstrakt sinf faqat hosila sinflar uchun asos sinf sifatida ishlatilishi mumkin.

Har qanday abstract usulga ega sinf, abstract deb ta’riflanishi shart. Bunday sinflarda to‘liq realizatsiya mavjud bo‘lmagani uchun, new operatori yordamida vakillarini yaratish mumkin emas. Bundan tashqari abstrakt konstruktorlar va statik usullar e’lon qilish mumkin emas. Abstrakt sinf har qanday vorisi yoki supersinf abstrakt usullarini to‘liq realizatsiya qilishi kerak, yoki o‘zi abstrakt deb elon qilinishi kerak.

```
abstract class A {
abstract void callme();
void metoo() {
System.out.println("Inside A's metoo method");
} }
class B extends A {
void callme() {
System.out.println("Inside B's callme method");
} }
class Abstract {
public static void main(String args[]) {
A a = new B():
a.callme():
a.metoo():
} }
```

Bu misolda hosila sinfda realizatsiya qilingan callme usuli va supersinfda realizatsiya qilingan metoo usullarini chaqirish uchun usullarni dinamik tayinlash usulidan foydalanilgan.

S:\> Java Abstract

Nazorat savollari

1. Nima uchun avval ajdod sinf konstruktorlari chaqirilib, so‘ngra avlod sinf konstruktori chaqiriladi?

2. Usullarni dinamik bog‘lash deb nimaga aytiladi?

3. Vorislikda usullar qanday qo‘shimcha yuklanadi?

4. Sinflar bibliotekasini qurishda vorislikdan qanday foydalaniladi?

5. Abstrakt sinflar nima uchun ishlatiladi?

2.4. Vorisliklardan foydalanish xususiyatlari

Object: global supersinf

Object sinfi hamma sinflar ajdodi xisoblanadi. Java tilida har bir sinf Object sinfini kengaytiradi. Lekin class Employee extends Oblects yozish shart emas. Agar supersinf oshkor ko‘rsatilmagan bo‘lsa Object supersinf xisoblanadi. Java tilida har bir sinf Object sinfini kengaytirgani uchun Object sinfi imkoniyatlarini bilish muximdir.

Object tipidagi o‘zgaruvchini ixtiyoriy tipdagi ob’ektga ilova sifatida ishlatish mumkin:

Object obj = new Employee("Garri Xaker", 35000);

Bu tipdagi o‘zgaruvchidan foydalanish uchun avval boshlang‘ich tipni aniqlab, tiplarni keltirishni amalga oshirish lozim:

Employee e = (Employee) obj;

equals va toString usullari

Object sinfining equals usuli ikki ob’ekt bir xilligini tekshiradi. Lekin *equals* usuli Object sinfiga tegishli bo‘lgani uchun, ikkalasi bir xotira qismiga ilova qilganligini tekshiradi. Ikki ob’ekt ekvivalentligini tekshirish uchun equals usulini qo‘shimcha yuklash lozim. Mukammal *equals* usuli yaratish qoidalari.

1. Oshkor otherObject parametrini chaqirish — keyinchalik uning tipini other deb atalgan boshqa o‘zgaruvchi tipiga keltirish lozim.

2. Tekshirish, this va otherObject ilovalar bir xilmi:

i f (this == otherObject) return true;

Odatda ob’ektlar maydonini solishtirgandan ko‘ra ilovalarni solishtirish osondir.

3. Tekshirish otherObject ilova nullga (null) tengmi. Agar xa bo‘lsa false qiymat qaytarish. Bu tekshirishni albatta amalga oshirish lozim.

if (otherObject == null) return false;

4. Tekshirish this va other ob’ektlari bitta sinfga tegishlimi.

Bu tekshirish "simmetriklik qoidasiga " ko‘ra majburiydir.

if (getClass() != otherObject.getClass()) return false;

5. Talab qilingan sinf o‘zgaruvchisiga otherObject ob’ektini o‘zgartirish:

ClassName other = (ClassName)otherObject;

6. Hamma maydonlarni solishtirish. Asosiy tipdagi maydonlar uchun == operatori, ob’ektli maydonlar uchun —equals usuli qo‘llanadi. Agar ikki ob’ekt hamma maydonlari bir xil bo‘lsa true qaytarish, aks holda false.

return fieldl == other.fieldl

&& field.2. equals (other . field2)

Masalan.

class Employee{

public boolean equals(Object otherObject) {

// Ob’ektlarni tez solishtirish,

if (this == otherObject) return true;

// Agar oshkor parametr — null, false qiymat qaytaradi,

if (otherObject == null) return false;

// Agar sinflar ustma ust tushmasa, ular ekvivalent emas.

if (getClass () != otherObject.getClassO) return false;

// Ob’ekt otherObject tipi Employee va u nolga teng emas.

Employee other = (Employee) otherObject;

// Ob’ektlar maydonlarini solishtirish,

return name.equals(other.name)

&& salary = other.salary

&& hireDay.equals(other.hireDay);

}

}

Ob’ekt tipini *getClass* usuli orqali aniqlanadi. Ob’ektlar o‘zaro teng bo‘lishi uchun bir sinf ob’ektlari bo‘lishlari kerak. Voris ichida avvval spersinf equals usulini chaqirish lozim. Agar bu tekshirish false qiymat qaytarsa, demak ob’ektlar teng emas. Agar tekshirish muvaffaqiyatli bajarilsa ostki sinf maydonlarini tekshirishga o‘tish mumkin.

Masalan quyidagicha.

```
class Manager extends Employee
{
public boolean.equals(Object otherObject)
{
if (!super.equals(otherObject)) return false;
Manager.other = (Manager)otherObject;
// Усул super.equals текширади
// this ва otherObject obe’ktlari bitta sinfga tegishlimi.
return bonus == other.bonus;
} }
```

Object sinfining yana bir muxim usuli *toString*, bo‘lib ob’ektni satr shaklida qaytaradi. Bu usul deyarli hamma sinflarda qo‘shimcha yuklanadi, va ob’ekt holatini bosmaga chiqarishga mo‘ljallangan.

Ko‘p (hammasi emas) toString usullari sinf nomi dan iborat bo‘lib, kvadrat qavslarda maydonlari qiymatlari ko‘rsatiladi. Quyida Employee sinfining toString usuli realizatsiyasi ko‘rsatilgan.

```
public String toString()
{
return "Employee[name" + name
```

```
+ ",salary =" + salary
+ ",hireDay =" + hireDay
}
```

Bu usulni takomillashtirish mumkin. Sinf nomini toString usuliga kiritmasdan, *getClass().getName()* usulini chaqiramiz va sinf nomini o‘z ichiga olgan satrni olamiz.

```
public String toString()
{
return getClass().geName()
+ "[name=" + name
+ ",salary=" + salary
+ ",hireday=" + hireDay
}
```

Endi toString usuli voris sinflar bilan ham ishlaydi.

Albatta voris sinf yaratgan dasturchi o‘z toString usulini yaratishi va voris sinf nomini qo‘shishi lozim. Agar supersinfda *getClass().getNamef()* usuli chaqirilsa, voris sinf super.*ToString( )* usulini chaqiradi. Manager sinfida toString usuliga misol.

```
class manager extends Employee
{
public String toString()
{
return super.toString()+ "[bonus=" + bonus
```

Endi Manager sinfi ob’ekti holati quyidagi shaklda chiqariladi:

Manager[name=...,salary=...,hireDay=...][bonus=...]

Agar ob’ekt satr bilan "+" amali yordamida konkatenatsiya qilinsa Java tili kompilyatori ob’ekt joriy holatini olish uchun avtomatik ravishda toString usulini chaqiradi.

Birorr x — ixtiyoriy ob’ekt uchun dasturchi System.out.println(x) usulini chaqirsin;

Bu holda println usuli x. toString () usulini chaqiradi va natija satrini chiqaradi.

Object sinfida aniqlangan *toString* usuli sinf nomi va ob’ekt adresini chiqaradi. Masalan

System.out.println(System.out);

chaqirish natijasida quyidagi satr hosil bo‘ladi

java.io.PrintStream@2f668 4

Buning sababi shuki PrintStream sinfida toString usuli qo‘shimcha yuklanmagan. Standart bibliotekaga tegishli ko‘p sinflarda toString usuli shunday aniqlanganki, uning yordamida dasturni sozlash uchun kerakli ma’lumot olish mumkin. Ba’zi sozlovchilar ob’ektlar holatini ekranda akslantinrish uchun toString usulini chaqirishga imkon beradi. Shuning uchun dastur trassirovkasida, quyidagi ifodalardan foydalanish mumkin

System.out.println{"Joriy holat = " + position);

Umumlashgan dasturlash

Object tipidagi o‘zgaruvchilarda ixtiyoriy sinf o‘zgaruvchilari qiymati saqlanishi mumkin, masalan String sinfi:

Object obj = "Salom"; // To‘g‘ri.

Lekin sonlar, simvollar va mantiqiy o‘zgaruvchilar ob’ektlarga kirmaydi.

obj = 5 ; // Noto‘g‘ri.

obj = false; // Noto‘g‘ri.

Bundan tashqari hamma tipdagi massivlar, ularda ob’ektlar yoki asosiy tiplardagi o‘zgaruvchilar saqlanishiga qaramay Object sinfi vorisi xisoblanadi.

Employee staff [] = new Employee[10];

Object arr = staff; // To‘g‘ri.

arr = new int[10]; // To‘g‘ri.

Biror sinfga tegishli ob’ektlar massivini Object sinfi ob’ektlari massiviga aylantirish mumkin. Masalan, Employee[] sinfi massivini Object[ ] sinfi massivini kutayotgan usulga uzatish mumkin. Bu usul umumlashgan dasturlash uchun foydalidir (generic programming).

Quyida umumlashgan dasturlash konsepsiyasini ko‘rsatuvchi misol keltirilgan. Bu misolda massivda element indeksini aniqlash lozim.

static int find(Object[] a, Object key)

(int i;

For

(i =0; i < a.length; i++)

if (a[i].equals(key)) return i;

return -1; // Indeks topilmagan.

}

Masalan,

Employee staff[] = new Employee[10];

Employee harry;

int n = find(staff, harry);

Shuni ta’kidlab o‘tish lozimki Object[] tipi massivini faqat biror sinf ob’ektlari massiviga aylantirish mumkin. Object[] tipi massiviga int[] tipi massivini o‘zgartirish mumkin emas. Agar biror sinf ob’ektlaridan massiv, Object[] tipidagi massivga aylantirilsa, umumlashgan massiv boshlang‘ich tip haqidagi ma’lumotni o‘zida saqlab qoladi. Bu massivga boshqa tipdagi elementni joylashtirish mumkin emas.

Nazorat savollari

1. Hamma sinflar qaysi sinfning vorislaridir?

2. Global supersinf usullarini ko‘rsating.

3. Foydalanuvchi solishtirish usuli qanday yaratiladi?

4. Nima uchun toString usuli ishlatiladi?

5. Umumlashgan dasturlash mohiyatini tushuntiring.

2 bob buyicha hulosalar

Ushbu bobda javada ob’ektga yunaltirilgan dasturlash asoslari kurib chiqildi. Javada ishlatiladigan ma’lumot turlari soda va ilovali turlardan iborat buladi. Soda turlarning ishlatilishi asosan 1 bobda berilgan bulsa, Ilovali turlar ob’ektga yunaltirilgan dasturlashning asosini tashkil etadi. Ob’ektga yunaltirilgan dasturlash murakkab ob’ektlarni qullash orqali amalga oshiriladi. Shunday murakkab ob’ektlar jumlasiga interfyeslar, fayllar, istisnolar va boshqalar kiradi. Ularni

boshqarish va dasturlash jarayonida qullashni uzlashtirish ushbu bobda kuzda tutilgan maqsadga erishishga yordam beradi.

III. BOB. JAVA DA MURAKKAB OB’EKTLAR BILAN ISHLASH

3.1. Interfeyslar

Interfeys ta’rifi

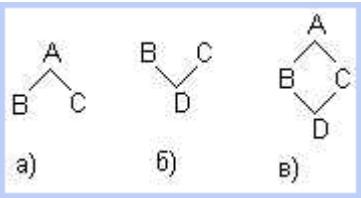
Interfeys — bu usullar jamlanmasi oshkor spetsifikatsiyasi bo‘lib, shu spetsifikatsiyani realizatsiya qilayotgan sinfda bu usullar ta’rifi albatta berilishi lozim. Interfeysda bu usullar realizatsiyasi berilmaydi. Abstrakt sinflar kabi interfeyslar ko‘plik vorislikda foydalanishi mumkin. Konkret sinf faqat bitta supersinf vorisi bo‘lishi mumkin lekin cheklanmagan sondagi interfeyslar realizatsiya qilinishi mumkin.

Interfeyslar

Shu paytgacha ma’lum bo‘ldiki, barcha kengaytmalar faqat bitta sinfdan olinishi mumkin. Chunki barcha sinflar shajarasining boshida Object sinfi yotibdi.

Lekin daturlash jarayonida ba’zi bir sinflarni ikki yoki undan ortiq sinflardan nasl olish hisobiga yaratish zarur bo‘ladi. Masalan, A sinfni B va C sinflarning nasli sifatida (9.1-rasm).

Bunday holat ko‘pnasllik holati deyiladi(multiple intertance). Ko‘pnasllikning yomon joyi yo‘q. Muammo B va C sinflarning o‘zlari A ning nasllari bo‘lganda paydo bo‘ladi. Nasl olishning turli holatlari 9.1-rasmda ko‘rsatilgan.



3.1-rasm. Nasl olishning turli variantlari.

Quyidagi holatni qaraylik. A sinfida f() metodi aniqlangan bo‘lsin, unga biror O sinfning metodi orqali murojaat qilinsin. Bunday holda f() metodi xuddi A da bayon qilingan vazifani bajaryapti deb ishonch bildirish mumkinmi? Balki u B va C sinflarida qayta aniqlangan bo‘lishi mumkin. Agar shunday bo‘lsa, biz qaysi variantdan B.f() yoki C.f() variantdan foydalanganimiz ma’qul bo‘ladi?

Albatta bunday holda sinf ekzemplyarini yaratish va shu ekzemplyarga murojaat qilish mumkin. Lekin bu butunlay boshqa gap.

Har xil dasturlash tillarida bu masala har xil echiladi. Lekin hamma holda ham KISS tamoyili buziladi.

Java tilini yaratuvchilari ko‘pnasllik holatini butunlay bekor qilishdi. Sinfni kengaytirayotganda *extends* so‘zidan keyin faqat bitta *super* sinfning nomi qo‘yiladigan bo‘ldi. Super aniqlovchisi yordamida *super sinf* a’zolariga murojaat qilish mumkin.

Lekin sinfni yaratayotganda bir necha sinfdan nasl olish kerak bo‘lib qolsa nima qilish kerak? Masalan, bizda barcha avtomobillar uchun umumiy bo‘lgan Automobile sinfi mavjud. Undan yuk mashinalari *Truck* va engil avtomobillarning *Car* sinflarini yaratish mumkin. Aytaylik, pikaplarning *Pickup* sinfini bayon qilish zarurati paydo bo‘ldi. Bu sinf yuk mashinalarining ham va engil mashinalarning ham xossalarini o‘zida mujassamlashtirishi kerak.

Bunday hollarda Java tilining yangi bir konstruktsiyasi - *interfeyslar* ishlatiladi.

Interfeyslar sinflardan farqli ravishda o‘z ichida faqat konstantalar va metodlar nomlarini ularning bayonisiz saqlaydi.

Interfeyslar sinflar joylashgan paketlar va qismpaketlarda joylashadi. Ular ham class-fayllarga kompilyatsiya qilinadi.

Interfeys bayoni *interface* so‘zidan boshlanadi. Uning oldida sinflardagi singari *public* modifikatori bo‘lishi mumkin. Bu modifikatorning mavjudligi interfeysga ixtiyoriy joydan murojaat qilish mumkinligini ko‘rsatadi. Agar public modifikatori bo‘lmasa, interfeys faqat o‘zi joylashgan paket a’zolarigagina ko‘rinishi mumkin.

interface so‘zidan keyin *interfeys* nomi ko‘rsatiladi. Undan keyin extends so‘zi va undan keyin esa ajdod interfeyslar ro‘yxati bo‘lishi mumkin. Ajdod interfeyslar nomlari bir-biridan vergul bilan ajratiladi. Shunday qilib interfeys interfeysdan yaratilishi mumkin. Interfeyslar sinflarga bog‘liqsiz ravishda o‘z shajarasini tashkil etadi. Bunda interfeyslarning ko‘pnasllik holatiga yo‘l qo‘yiladi. Bu shajarada hamma interfeyslar uchun umumiy bo‘lgan o‘zak yo‘q.

Interfeys ajdodlari ro‘yxatidan keyin katta qavs ichida interfeys tana qismi yoziladi. Tana qismi ixtiyoriy tartibda yozilgan konstantalar va metodlar nomlarinigina o‘z ichiga oladi. Aytish mumkinki, interfeysda barcha metodlar abstrakt, lekin abstract yozilmaydi. Konstantalar har doim statistik, lekin statistic yoki final so‘zlari yozilmaydi. Interfeysdagi barcha konstantalar va metodlar ochiq, public so‘zini yozish shart emas.

Avtomobillar shajarasi uchun quyidagi sxemani taklif etish mumkin:

interface Automobile{ . . . }

interface Car extends Automobile{ . . . }

interface Truck extends Automobile{ . . . }

interface Pickup extends Car, Truck{ . . . }

Shunday qilib, interfeys bu chizgi-eskiz hisoblanadi. Unda nima qilish kerakligi ko‘rsatlgan, lekin qanday qilib bajarish kerakligi ko‘rsatilmagan.

Agar interfeys to‘liq abstrakt bo‘lsa hamda unda birorta to‘liq metod bo‘lmasa, uni qanday ishlatish mumkin.

Interfeysning o‘zidan emas, balki uning joriy etilishidan (realizatsiyasidan) foydalaniladi. Interfeysning joriy etilishi - bu sinf bo‘lib, unda bir yoki bir nechta interfeyslarning metodlari bayon etiladi.

Sinf sarlavhasida uning nomidan yoki uning super sinfi nomidan keyin (agar u mavjud bo‘lsa) implements so‘zi yoziladi va undan keyin interfeyslar nomlari bir-biridan vergul bilan ajratilgan holda yoziladi.

Avtomobillar shajarasini quyidagicha joriy qilish mumkin:

interface Automobile{ . . . }

interface Car extends Automobile! . . . }

class Truck implements Automobile! . . . }

class Pickup extends Truck implements Car{ . . . }

yoki:

interface Automobile{ . . . }

interface Car extends Automobile{ . . . }

interface Truck extends Automobile{ . . . }

class Pickup implements Car, Truck{ . . . }

Interfeysning joriy qilinishi to‘liq bo‘lmasligi, ya’ni interfeysning ba’zi bir metodlari bayon etilgan, boshqalari esa bayon etilmagan bo‘lishi mumkin.

Bunday joriy etish abstrakt sinfdir. Bunday holda albatta sinfni abstract modifikatori bilan belgilash kerak. Car va Truck interfeyslarida bayon qilingan, bir xil signaturali f() metodini Pickup sinfida qanday joriy etish mumkin. Javob oddiy: Hech qanday! Bunday holatni Pickup sinfida joriy qilib bo‘lmaydi. Dasturni boshqachasiga loyihalashtirish zarur.

Shunday qilib, interfeyslar loyihalarni joriy etish muammolariga e’tibor bermasdan, Java da sof obektga yo‘naltirilgan loyihalashtirish ishlarini bajaradi.

Biz dastur loyihasi ishlanmasini yaratishda, uning joriy etilishiga e’tibor bermasdan, interfeyslar shajarasi sifatida tasvirlashimiz mumkin. Keyin bu loyihaga qarab sinflar ierarxiyasi (shajarasi) yaratiladi. Bunda birga bir nasl olish va sinf a’zolarining ko‘rinish chegaralariga e’tibor beriladi.

Bu yerda qiziqarli tomoni shundan iboratki, interfeyslarga ilovalar hosil qilinadi. Albatta, bunday ilova interfeysning biror bir joriy etilishiga ko‘rsatma beradi. Shunday qilib, biz interfeyslar orqali ob’ektga yo‘naltirilgan dasturlash tamoyillaridan bir polimorfizmning yana bir yaratish usulini hosil qildik.

3.1-listingda uy hayvonlarini qanday qilib bir joyga jamlash mumkinligi ko‘rsatilgan.

3.1-listing. Polimorfizm tashkil etish uchun interfeyslardan foydalanish.

```
interface Voice{
void voice();
}
class Dog implements Voice{
  public void voice (){
    System.out.println("Gav-gav!");
  }
}
class Cat implements Voice{
  public void voice (){
    System.out.println("Miaou!");
  }
```

```
}
class Cow implements Voice{
  public void voice(){
    System.out.println("Mu-u-u!");
  }
}
public class Chorus{
  public static void main(String[] args){
    Voiced singer = new Voice[3];
    singer[0] = new Dog();
    singer[1] = new Cat();
    singer[2] = new Cow();
    for(int i = 0; i < singer.length; i++)
      singer[i].voice();
  }
}
```

Bu erda Lights interfeysida butun loyiha uchun umumiy bo‘lgan konstantalar aniqlangan.

Timer sinfi bu interfeysni joriy etadi va konstantani bevosita o‘z xususiy konstantasi kabi ishlatadi. Bu sinfning *shift()* metodi svetafor rangiga ko‘ra harxil vaqtli signallarni almashtirish uchun signal beradi. Signallarning ushlanish vaqtini Thread sinfidagi *sleep()* metodi aniqlaydi. Bu metodga standart kutubxonadagi millisekundlardagi ushlanish vaqti uzatiladi. Bu metod istisnoli holatni try{} cath{} konstruksiyasi orqali joriy etishi zarur.

TraficRecuator sinfi Light interfeysini joriy etmaydi va to‘liq Light.RED nomdan foydalanadi va h.k. Bunday holat konstantalar RED, YELLOW va GREEN lar jimlikda statik bo‘lganlari uchun mumkindir.

Design patterns

Matematikada namunaviy masalalarni echishning umumiy metodlari ishlab chiqilgan. Teoremalarni isbotlash “Isbotlashni teskarisidan boshlab amalga oshiramiz” yoki “Buni matematik induksiya metodi bilan isbotlaymiz” kabi so‘zlardan boshlanadi va bizning tasavvurimizda teoremani isbotlash sxemasi paydo bo‘ladi hamda isbotlash yo‘li tushunarli holatga keladi. Dasturlashda ham shunday umumiy metodlar bormi? Ha, bor.

Faraz qilaylik, sizga metrologik stantsiyani avtomatlashtirish topshirildi. Harorat, bosim, namlik, shamol tezligi va boshqalarni o‘lchaydigan datchiklardan axborotlar, boshqacha qilib aytganda kontrollerlardan raqamli ko‘rinishda kompyuterga kelib tushadi.

Kompyuterda bu axborotlar qayta ishlanadi va hududlar bo‘yicha o‘rtacha qiymatlar hisoblanadi. Ko‘p kunlik kuzatuvlar asosida ertangi kunga prognoz qilinadi. Ya’ni hududning metrologik holati modeli yaratiladi. Bundan keyin prognoz har xil kanallar orqali monitor ekraniga, yozuvchi qurilmaga, printerga va hakozolarga chiqariladi va tarmoqdan uzatiladi. Natijalar har xil ko‘rinishda: sonlar ustuni, grafiklar, diagrammalar va boshqa ko‘rinishlarda tasvirlanadi.

Bunday avtomatlashtirilgan tizimni tabiiy ravishda uch qismdan iborat qilib loyihalashtirish mumkin:

- Birinchi qism - uni kontroller(controller) deb ataymiz, datchiklardan axborotlar oladi va ularni keyinchalik qayta ishlashga yaroqli holdagi umumiy shaklga keltiradi. Bunda har bir datchik uchun o‘z modulini yozish kerak. Unga aniq qurilmalardan signallar kiradi va unifikatsiya qilingan axborotlar chiqadi.

- Ikkinchi qism - uni Model deb ataymiz - unifikatsiya qilingan axborotlarni kontrollerdan qabul qilib oladi. Modulning axborotlarni qabul qilishida qaysi datchikdan kelganligi, datchikning o‘zi to‘g‘risida qanday axborotlar mavjudligi kabilar ahamiyatli emas. Modul olingan axborotlarni o‘z ichki algoritmiga ko‘ra biror turdagi ma’lumotlarga, masalan, sonlar ketma-ketligiga aylantiriladi.

- Tizimning uchinchi qismi - ko‘rinishi bevosita chiqarish qurilmalari bilan bog‘liq bo‘lib, Modeldan olinayotgan axborotlarni grafik, matn, diagramma va boshqa ko‘rinishlarga aylantiradi yoki tarmoqda uzatish uchun paket tayyorlaydi. Har bir qurilma uchun axborotni aks ettiruvchi o‘z modulini yaratishi kerak. Bu modul qurilma xususiyatlarini e’tiborga olishi kerak bo‘ladi.

Bu uch bo‘g‘inli tizimning afzallik tomonlari nimadan iborat?

U juda moslashuvchan. Kontrollerda bitta datchikni almashtirish faqat bitta modulni o‘zgartirishga olib keladi. Model ham, ko‘rinish (Vid) ham buni sezmaydi. Aytaylik, ob-havo haqidagi ma’lumotni biror yangi ko‘rinishda, masalan, televideniya uchun tasvirlash kerak bo‘lsin. Marhamat, Ko‘rinishda bitta qo‘shimcha modul yozish bilan masalani hal qilish mumkin. Agar qayta ishlash algoritmida o‘zgarish bo‘lsa, modelni o‘zgartiramiz. Boshqa bo‘g‘inlarga tegilmaydi.

Bu sxema o‘tgan asrning 80-yillaridayoq ishlab chiqilgan. Bunda Smaltalk tili qo‘llanilgan. U MVC – (Model View Controller) deb nomlangan. Bu usul metrologiyadan ancha uzoq bo‘lgan sohalarda ham qo‘llanilishi mumkin ekan. Bunda asosan, qayta ishlashni kiritish va chiqarishdan ajratish qulay bo‘lsa bo‘ldi.

Ko‘pchilik hollarda axborot yig‘ish quyidagicha tashkil etiladi. Ekran displeyida kiritish maydonlari ochiladi. Biz u erga ma’lumotlar kiritamiz. Masalan, ixtiyoriy tartibda Familiyalar. Qo‘shni maydonda qayta ishlangan axborot aks ettiriladi, masalan, familiyalar ro‘yxati alfavit bo‘yicha tartiblangan. Ishonch hosil qilish mumkinki, bu sxema MVC ko‘rinishida tashkil etilgan. Kontroller sifatida kiritish maydoni, ko‘rinish-chiqarish maydoni, model sifatida esa familiyalarni alfavit tartibida saralash usuli ishlatilmoqda. 90-yillar o‘rtasiga kelib bunday tizimlar ko‘plab paydo bo‘ldi. Ularda minglab dasturchilarning ko‘p yillik tajribalari umumlashtirilgan.

Interfeyslarda o‘zgaruvchilar

Interfeyslardan turli sinflarga birgalikda foydalaniluvchi konstantalarni import qilish uchun foydalanish mumkin. Bu holda biror sinfda interfeys realizatsiya qilinsa interfeys o‘zgaruvchilari nomlari bu sinfda konstanta sifatida ko‘rinadi. Bu S va C++ tillarida konstantlarni #define direktivasi yoki Pascal / Delphi tillarida const kalit so‘zi yordamida berishga mosdir.

Agar interfeys o‘z ichiga usullarni olmasa, interfeys realizatsiyasi deb e’lon qilingan sinf xech narsa realizatsiya qilmaydi. Konstantalarni sinf nomlar fazosiga import qilish uchun final modifikatorli o‘zgaruvchilardan foydalanish qulaydir.

```
import java.util.Random;
interface SharedConstants { int NO = 0;
int YES = 1;
int MAYBE = 2;
int LATER = 3;
int SOON = 4;
int NEVER = 5; }
class Question implements SharedConstants { Random rand = new Random();
```

```
int ask() {
int prob = (int) (100 * rand.nextDouble());
if (prob < 30) return NO; // 30%
else if (prob < 60) return YES; // 30%
else if (prob < 75) return LATER; // 15%
else if (prob < 98) return SOON; // 13%
else return NEVER; // 2% } }
class AskMe implements SharedConstants {
static void answer(int result) {
switch(result) {
case NO:
System.out.println("No");
break;
case YES:
System.out.println("Yes");
break;
case MAYBE:
System.out.println("Maybe");
break;
case LATER:
System.out.println("Later");
break;
case SOON:
System.out.priniln("Soon");
break;
case NEVER:
System.out.println("Never");
break;
} }
public static void main(String args[]) {
Question q = new Question();
answer(q.ask());
answer(q.ask());
answer(q.askO);
answer(q.ask());
} }
```

E’tibor beringki dastur har gal ishlatilganda har xil natija beradi, chunki unda java.util paketiga tegishli Random tasodifiy sonlar generator ishlatilgan.

```
S:\> Java AskMe

Later

Scon

No
```

Yes

Nazorat savollari

1. Interfeys ta’rifini keltiring.

2. Interfeys sinfdan qanday farq qiladi?

3. Nima uchun operator implements ishlatiladi?

4. Interfeys nima uchun ishlatiladi.

5. Birgalikda ishlatiladigan konstantalarni turli sinflarga import qilish uchun interfeysdan qanday foydalaniladi?

6.Interfeys ob’ektga yo‘naltirilgan dasturlashning qanday tamoyilini joriy etadi?

7.Ko‘pnasillik nimaga olib keladi va uning ahamiyati.

8.Design patterns texnologiyasining qanday afzalliklari bor?

3.2. Fayllar bilan ishlash

Kiritish/Chiqarish

Kiritish manbai umumlashgan tushunchasi bir necha ma’lumot olish usullariga tegishli: diskli fayldan o‘qish, klaviaturadan kiritish, yoki tarmoq orqali axborot uzatish qabul qilish. Xuddi shunday chiqarish umumlashgan manbai sifatida diskli fayllar, tarmoq orqali bog‘lanish va xokazolar tushunilishi mumkin. Bu abstraktsiyalar kiritish chiqarish (I/O) bilan ishlash uchun qulay imkoniyat yaratadi, chunki dasturdan, klaviatura va tarmoqni farqlashni talab etmaydi. Java tilida bu abstraktsiyalar oqim (stream) deb ataladi va java.io paketi bir necha sinflarida realizatsiya qilingan. Kiritish InputStream sinfida inkapsulyatsiya etilgan, chiqarish —OutputStream sinfida. V Java tilida bu abstrakt sinflarni bir necha spetsializatsiyalari mavjud bo‘lib, diskli fayllar, tarmoq bilan bog‘lanish va xotiradagi buferlar bilan ishlash xususiyatlarini aks ettiradi.

InputStream

InputStream —Java tilida kirish oqimlari modelini beruvchi abstrakt sinfdir. Bu sinf hamma usullari xato yuz berganda IOException istisno yaratadi. Quyida InputStream sinfi usullari keltirilgan.

• *read()* kirish oqimidagi navbatdagi simvolni butun son koʻrinishida qaytaradi.

• *read(byte b[])* maksimum b.length baytni kirish oqimidan b massivga oʻqishga harakat qiladi. Oqimdan chindan oʻqilgan baytlar sonini qaytaradi.

• *read(byte b[], int off, int len)* maksimum len baytni, raspolojiv b massivga, off elementdan boshlab oʻqishga harakat qiladi. Xaqiqatan oʻqilgan baytlar sonini qaytaradi.

• *skip(long n)* kirish oqimida n baytni oʻtkazishga harakat qiladi. Xaqiqatan oʻtkazilgan baytlar sonini qaytaradi.

• *available()* oʻqish mumkin boʻlgan baytlar sonini qaytaradi.

• *close()* kirish manbaini berkitadi. Bu oqimdan keyingi oʻqishga harakat qilish IOException istisnosini yaratadi.

• *mark(int readlimit)* kirish oqimi joriy pozitsiyasiga belgi qoʻyadi. Bu belgidan oqimdan readlimit bayt oʻqilmaguncha foydalanish mumkin boʻladi.

• *reset()* oqim koʻrsatkichini oldin qoʻyilgan belgiga qaytaradi.

• *markSupported()* agar oqim mark/reset amallarini qoʻllasa true qaytaradi.

OutputStream

InputStream kabi OutputStream — abstrakt sinf. U Java chiqish oqimlari modelini beradi. Bu sinf hamma usullari void tipiga ega va xatolik yuz berganda IOException istisno yaratadi. Quyida shu sinf usullari roʻyxati berilgan:

• *write(int b)* chiqish oqimiga bir bayt yozadi. Bu usul argumenti int tipiga tegishli, shuning uchun write usulini ifoda uzatib chaqirish mumkin va ifodani byte tipiga keltirish shart emas.

• *write(byte b[])* chiqish oqimiga koʻrsatilgan baytlar massivi hammasini yozadi.

• *write(byte b[], int off, int len)* oqimga massiv len baytini b[off] elementdan boshlab yozadi.

• *flush()* chiqarish amalini tugatib, ixtiyoriy chiqish buferini tozalaydi.

• *close()* chiqish oqimini berkitadi. Bu oqimga yozishga keyingi har qanday urinish IOException yaratadi.

Faylli oqimlar

FileInputStream

Sinf FileInputStream sinfi ma’lumotlarni fayllardan kiritish uchun ishlatiladi. Quyida keltirilgan misolda bu sinfning bitta diskli fayldan foydalanuvchi ikki ob’ekti yaratiladi.

InputStream f0 = new FileInputStream("/autoexec.bat");

File f = new File("/autoexec.bat"):

InputStream f1 = new FileInputStream(f);

FileInputStream sinfi ob’ekti yaratilganda, u o‘qish uchun ochiladi. FileInputStream sinfi InputStream abstrakt sinf olti usulini qo‘shimcha yuklaydi. Bu sinf ob’ektiga mark va reset usullarini qo‘llashga urinish IOException istisno yaratilishiga olib keladi. Quyida keltirilgan misolda qanday qilib, aloxida bayt, baytlar massivi va baytlar massivi qismini o‘qish ko‘rsatilgan. Bu misolda yana available usuli yordamida qancha o‘qilmagan baytlar qolganligi va skip usuli yordamida o‘qish kerak bo‘lmagan baytlarni o‘tkazib yuborish ko‘rsatilgan.

```
import java.io.*;
import java.util.*;
class FileInputStreamS {
public static void main(String args[]) throws Exception {
int size;
InputStream f1 = new FileInputStream("/wwwroot/default.htm");
size = f1.available();
System.out.println("Total Available Bytes: " + size);
System.out.println("First 1/4 of the file: read()");
for (int i=0; i < size/4; i++) {
System.out.print((char) f1.read());
}
System.out.println("Total Still Available: " + f1.available());
System.out.println("Reading the next 1/8: read(b[])");
byte b[] = new byte[size/8];
if (f1.read(b) != b.length) {
System.err.println("Something bad happened");
}
String tmpstr = new String(b, 0, 0, b.length);
System.out.println(tmpstr);
System.out.println("Still Available: " + f1.available());
System.out.println("Skipping another 1/4: skip()");
f1.skip(size/4);
```

```
System.out.println( "Still Available: " + f1.available());
System.out.println("Reading 1/16 into the end of array");
if (f1.read(b, b.length-size/16, size/16) != size/16) {
System.err.println("Something bad happened");
}
System.out.println("Still Available: " + f1.available());
f1.close();
}
}
```

FileOutputStream

FileOutputStream sinfi FileInputStream sinfi kabi ikki konstruktorga ega. Lekin bu sinf ob'ektlarini yaratish uchun fayl mavjud bo'lishi shart emas. FileOutputStream sinfi ob'ekti yaratilganda chiqarish uchun fayl ochishdan oldin yaratiladi.

Navbatdagi misolda klaviaturadan kiritilgan simvollar System.in oqimidan 12-baytli bufer to'lmaguncha bittadan o'qiladi. Shundan so'ng uchta fayl yaratiladi. Birinchi file1.txt faylga, buferdan simvollar yoziladi, lekin hammasi emas bitta tashlab, ya'ni oldin nolinchi keyin ikkinchi va xokazo. Ikkinchi file2.txt faylga buferga tushgan hamma ma'lumot yoziladi. Va nixoyat uchinchi faylga bufer o'rtasida joylashgan yarmi yoziladi, birinchi va oxirgi choraklar tashlab yuboriladi.

```
import java.io.*;
class FileOutputStreamS {
public static byte getlnput()[] throws Exception {
byte buffer[] = new byte[12];
for (int i=0; i<12; i++) {
buffer[i] = (byte) System.in.read();
}
return buffer;
}
public static void main(String args[]) throws Exception {
byte buf[] = getlnput();
OutputStream f0 = new FileOutputStream("file1.txt");
OutputStream f1 = new FileOutputStream("file2.txt");
OutputStream f2 = new FileOutputStream("file3.txt");
for (int i=0; i < 12; i += 2) {
f0.write(buf[i]);
}
f0.close();
f1.write(buf);
f1.close();
```

```
f2.write(buf, 12/4, 12/2);
f2.close();
} }
```

Xozirgi paytda FileOutputStream oqimni fayl oxiriga qo‘shish uchun ochish mumkin emas. Agar fayl FileOutputStream konstruktori yordamida ochilsa mavjud ma’lumot yo‘qoladi. Bu Java realizatsiyasi kamchiligi.

ByteArraylnputStream

ByteArrayInputStream - byte tipidagi massiv ishlatiluvchi kirish oqimi realizatsiyasidir. Bu sinfda ikki konstruktor bo‘lib, har biri birinchi parametr sifatida baytli massiv talab qiladi. Quyida keltirilgan misolda shu tipdagi ikki ob’ekt yaratiladi. Bu ob’ektlar lotin alfaviti simvollari bilan initsializatsiya qilinadi.

```
String tmp = "abcdefghijklmnopqrstuvwxyz";
byte b[] = new byte [tmp.length()];
tmp. getBytes(0, tmp.length(), b, 0);
ByteArrayInputStream input1 = new ByteArrayInputStream(b);
ByteArrayInputStream input2 = new ByteArreyInputStream(b,0,3);
```

ByteArrayOutputStream

ByteArrayOutputStream sinfida — ikki konstruktor. Birinchi konstruktor xajmi 32 baytga teng bufer yaratadi. Ikkinchi konstruktordan foydalanilganda konstruktor parametrida berilgan xajmli bufer yaratadia (quyida keltirilgan misolda — 1024 bayta):

```
OutputStream out0 = new ByteArrayOutputStream();

OutputStream out1 = new ByteArrayOutputStream(1024);
```

Navbatdagi misolda ByteArrayOutputStream ob’ekti klaviaturadan kiritilgan simvollar bilan to‘ldiriladi, shundan so‘ng turli manipulyatsiyalar bajariladi.

```
import java.io.*;
import java.util.*;
class ByteArrayOutputStreamS {

public static void main(String args[]) throws Exception {
int i;
ByteArrayOutputStream f0 = new ByteArrayOutputStream(12);
System.out.println("Enter 10 characters and a return");
while (f0.size() != 10) {
f0.write( System.in.read());
}
```

```
System.out.println("Buffer as a string");
System.out.println(f0.toString());
System.out.println ("Into array");
byte b[] = f0.toByteArray();
for (i=0; i < b.length; i++) {
System.out.print((char) b[i]);
}
System.out.println();
System.out. println("To an OutputStream()");
OutputStream f2 = new File0utput8tream("test.txt");
f0.writeTo(f2);
System.out.println("Doing a reset");
f0. reset();
System.out.println("Enter 10 characters and a return");
while (f0.size() != 10) {
f0.write (System.in.read());
}
System.out.println("Done.");
} }
```

Ushbu misoldagi test.txt faylini kuzatib kutgan natijaga erishganligimiz ayon bo‘lamiz:

S:\> type test.txt

0123456789

StringBufferlnputStream

StringBufferInputStream sinfi ByteArrayInputStream sinfi bilan deyarli bir xil. Farqi shundaki bu sinf ob’ekti ichki buferi baytli massiv emas String sinfi nusxasidir. Bundan tashqari Java tilida unga mos StringBufferedOutputStream sinfi yo‘q. Bu sinfda yagona konstruktor:

StringBufferInputStream(String s)

Nazorat savollari

1. Oqim deb nimaga aytiladi?

2. Kirish uchun abstrakt sinflarni ko‘rsating.

3. Chiqish uchun abstrakt sinflarni ko‘rsating.

4. Fayldan o‘qish uchun sinflarni ko‘rsating.

5. Faylga yozish uchun sinflarni ko‘rsating.

3.3. Fayllar bilan ishlash hususiyatlari

File

File —*java.io* da to‘g‘ridan to‘g‘ri diskli fayllar bilan ishlovchi yagona ob’ektdir. Appletlarda fayllardan foydalanishga chegaralar qo‘yilgan bo‘lsa ham, fayllar doimiy saqlash va birgalikda foydalanish uchun yagona resursdir. Katalog v Java tilida oddiy fayl sifatida qaraladi, faqat qo‘shimcha xossali - fayllar nomlari ro‘yxatiga ega bo‘lib, bu ro‘yxatni list usuli yordamida qarab chiqish mumkin.

Izoh

Java UNIX va DOS tizimlarida ishlatiladigan yo‘lda kataloglar nomlarini ajratuvchi belgini to‘g‘ri qayta ishlaydi. Agar UNIX stili — simvol '/' ishlatilsa, Windows sistemasida ishlanganda Java avtomatik ravishda ularni '\' belgiga almashtiradi. Unutmang agar siz DOS da qabul qilingan ajratgichlarga ya’ni '\' belgisiga o‘rgangan bo‘lsangiz, ularni yo‘l satriga ulash uchun ikkilash kerak, masalan \\java\\COPYRIGHT satrdagi kabi.

Ob’ekt standart xossalarini aniqlash uchun File sinfida turli usullar mavjud. Lekin File sinfi nosimmetrik. Ob’ekt xossalarini aniqlash uchun xossalar ko‘p, lekin bu xossalarni o‘zgartirish usullari mavjud emas. Quyidagi misolda fayl turli harakteristikalarini olish uchun turli usullardan foydalaniladi:

```
import java.io.File;
class FileTest {
static void p(String s) {
System.out.println(s);
}
public static void main(String args[]) {
File f1 = new File("/java/COPYRIGHT");
p("File Name:" + f1 .getName());
p("Path:" + f1.getPath());
p("Abs Path:" + f1.getAbsolutePath());
p("Parent:" + f1.getParent());
p(f1.exists() ? "exists" : "does not exist");
p(f1.canWrite() ? "is writeable" : "is not writeable");
p(f1.canRead() ? "is readable" : "is not readable");
p("is " + (f1.isDirectory() ? " " : "not") + " a directory");
p(f1.isFile() ? "is normal file" : "might be a named pipe");
p(f1.isAbsolute() ? "is absolute" : "is not absolute");
p("File last modified:" + f1. lastModified());
p("File size:" + f1.length() + " Bytes");
} }
```

Dastur bajarilganda quyidagi ma’lumot olish mumkin:

File Name:COPYRIGHT (fayl nomi)

Path:/java/COPYRIGHT (yo‘l)

Abs Path:/Java/COPYRIGHT (ildiz katalogdan yo‘l)

Parent:/java (ota katalog)

exists (fayl mavjud)

is writeable (yozishga ruxsat berilgan)

is readable (o‘qishga ruxsat berilgan)

is not a directory (katalog emas)

is normal file (oddiy fayl)

is absolute

File last modified:812465204000 (fayl oxirgi modifikatsiyasi)

File size:695 Bytes (fayl xajmi)

Faqat fayllarga qo‘llash mumkin bo‘lgan usullar mavjud (ularni kataloglarga qo‘llash mumkin emas). Fayl nomini o‘zgartirish uchun renameTo(File dest) usulidan foydalaniladi (faylni boshqa katalogka ko‘chirish mumkin emas). Diskdagi faylni delete usuli o‘chiradi. Bu usul faqat faylni o‘chiradi, xatto bo‘sh katalogni bu usul bilan o‘chirib bo‘lmaydi.

Kataloglar

Kataloglar File sinfi ob’ektlari bo‘lib, ularda boshqa fayllar va kataloglar ro‘yxati joylashgan. Agar File katalogga ilova qilsa, uning *isDirectory* usuli true qiymat qaytaradi. Bu holda list usulini chaqirib ob’ektda joylashgan fayllar va kataloglar nomlarini chiqarish mumkin. Navbatdagi misolda list usuli yordamida katalogni ko‘rib chiqish ko‘rsatilgan.

```
import java.io.File;

class DirList {

public static void main(String args[]) {

String dirname = "/java"; // katalog nomi
```

```
File f1 = new File(dirname);

if (f1.isDirectory()) { // f1 katalogmi

System.out.println("Directory of ' + dirname);

String s[] = f1.list();

for ( int i=0; i < s.length; i++) {

File f = new File(dirname + "/" + s[i]);

if (f.isDirectory()) { // f katalogmi System.out.println(s[i] + " is a directory"):

} else {

System.out.println(s[i] + " is a file");

} } } else {

System.out.println(dirname + " is not a directory");

} }

}
```

Ishlash jarayonida bu dastur /java katalogidagi ma’lumotlarni quyidagicha chiqarishi mumkin:

```
C:\> java DirList

Directory of /java

bin is a directory

COPYRIGHT is a file

README is a file

FilenameFilter
```

Ko‘pincha list usuli qaytarayotgan nomlar sonini cheklab, ma’lum shablonga mos nomlarni chiqarish talab etiladi. Buning uchun Dlya etogo v paket java.io paketiga FilenameFilter interfeysi kiritilgan. Ob’ekt, bu interfeysni realizatsiya qilishi uchun, har yangi fayl nomi bilan chaqiriluvchi *accept()* usulini ta’riflash lozim. Bu accept usuli ro‘yxatga kiritilishi lozim bo‘lgan nomlar uchun true va chiqarish lozim bo‘lgan nomlar uchun false qaytarishi lozim.

File sinfda kataloglar bilan ishlashga mo‘ljallangan ikki usul mavjud. Ostki katalog yaratish uchun mkdir usuli ishlatiladi. Xali yo‘li ko‘rsatilmagan katalog yaratish uchun mkdirs usulidan foydalanish lozim. U faqat ko‘rsatilgan katalog emas, hamma mavjud bo‘lmagan ajdod kataloglarni yaratadi.

Filtrlovchi oqimlar

Chiqarish tizimi parallel jarayonlarga ega muxitda ishlaganda agar sinxronizatsiya mavjud bo‘lmasa kutilmagan natijalar kelib chiqishi mumkin. Buning sababi bir necha quyi jarayonlarning bitta oqimga murojaat qilishidir. Bu sinfda mavjud hamma konstruktorlar va usullar InputStream va OutputStream sinflaridagi konstruktorlar va usullar bilan bir xil. Filtrlovchi oqimlarning farqi shundaki ularning usullari sinxronizatsiya qilingandir.

Buferizatsiyalangan oqimlar

Buferizatsiya qilingan oqimlar filtrlanuvchi oqimlar kengaytmasi bo‘lib, ularda kiritish- chiqarish oqimlariga xotirada bufer ulanadi. Bu bufer ikki funksiya bajaradi:

- U java bajaruvchi muxitiga bir necha baytni kiritish- chiqarishga imkon beradi. Buning natijasida muxit unumdorligi oshadi.

- Oqimni buferi mavjud bo‘lgani uchun ma’lumotlarni o‘tkazish, belgilarni o‘rnatish va buferni tozalash tamallarini bajarish mumkin bo‘ladi.

BufferedInputStream

Kiritish- chiqarishni buferizatsiyalash — bunday amallarni optimallash keng tarqalgan usulidir. BufferedlnputStream sinfi Java tilida InputStream ixtiyoriy ob’ektini buferizatsiya qilingan oqim bilan ―o‘rab olishga imkon beradi va shu bilan unumdorlikni oshiradi. Bu sinfda ikki konstruktor mavjud bo‘lib, birinchisi

BufferedInputStream(InputStream in)

xajmi 32 baytli buferdan foydalanib, buferizatsiyalangan oqim yaratadi. Ikkinchisida

BufferedInputStream(InputStream in, int size)

oqim buferi xajmi konstruktor ikkinchi parametri orqali beriladi. Umumiy holda bufer optimal xajmi operatsion tizimga, operativ xotira hajmiga va kompyuter konfiguratsiyasiga bog‘liq.

BufferedOutputStream

BufferedOutputStream ob’ektiga chiqarish ixtiyoriy OutputStream ob’ektiga chiqarish bilan deyarli bir xil, shu farq bilanki yangi ostki sinf qo‘shimcha flush usuliga ega. Bu usul buferni majburan tozalash va undagi ma’lumotni tashqi qurilmaga fizik chiqarish uchun ishlatiladi. Bu sinf konstruktori birinchi shakli:

BufferedOutputStream(OutputStream out)

xajmi 32 baytli buferga ega oqim yaratadi. Ikkinchi shakli:

BufferedOutputStream(OutputStream out, int size)

Kerakli bufer xajmini kiritishga imkon beradi.

PushbacklnputStream

Bufferlashtirishning noananaviy qo‘llanishlaridan biri — pushback (orqaga qaytarish) amalini joriy qilishdir. Pushback simvolni o‘qigandan so‘ng oqimga qaytarish uchun InputStream ga qo‘llanadi. Lekin PushbacklnputStream imkoniyatlari cheklangan – oqimga bittadan ortiq simvolni qaytarishga urinish IOException istisno yaratilishiga olib keladi. Bu sinfda — yagona konstruktor

PushbackInputStream(InputStream in)

InputStream usullaridan tashqari, PushbacklnputStream argumentida berilgan ch simvolni kiritish oqimiga qaytaruvchi unread(int ch) usuliga ega.

SequencelnputStream

SequencelnputStream sinfi bir necha kiruvchi oqimlarni bitta oqimga qo‘shish yangi imkoniyatiga ega. SequenceInputStream sinfi konstruktorida parametr sifatida InputStream ikki ob’ekti, yoki InputStream ob’ektlari kollektsiyasini o‘z ichiga olgan sanovchi ishlatiladi:

SequenceInputStream(Enumeratione) SequenceInputStream(InputStream s0, InputStream s1)

Ish jarayonida sinf kelib tushgan so‘rovlarni bajarib, birinchi oqimdan to tugamaguncha ma’lumotlarni o‘qiydi, so‘ngra ikkinchisiga o‘tadi va xokazo.

PrintStream

PrintStream sinfi System paketi chiqarishda faylli deskriptorlari orqali foydalanib kelingan hamma formatlash utilitalaridan foydalanishga imkon beradi. Shu paytgacha ―System.out.println, yozilganda chiqarilayotgan ma’lumotlarni

formatlovchi sinflarga e’tibor berilmayotgan edi. U sinfa PrintStream sinfida ikki konstruktor: PrintStream(OutputStream out) i PrintStream(OutputStream out, boolean autoflush). Ikkinchisining autoflush parametri Java bajaruvchi muxiti chiqarish oqim ustida buferni tozalash amali avtomatik bajarish kerak yoki kerakmasligini ko‘rsatadi.

Java tilida - PrintStream ob’ektlarida print va println usullari mavjud bo‘lib, ular ixtiyoriy ob’ektlar xatto Object ob’ektlari bilan ishlay oladi. Agar bu usullar argumetlari sifatida primitiv tiplardan biri ishlatilmasa, Object sinfining toString usulini chaqiradi va shundan so‘ng ma’lumotni chaqiradi.

Nazorat savollari

1. Diskli fayllar bilan ishlash uchun qaysi ob’ektdan foydalaniladi?

2. Katalog bu nima?

3. Katalogni ko‘rib chiqish usulini ko‘rsating.

4. Oqimni buferizatsiyalash qanday amalga oshiriladi?

5. Buferizatsiyalangan oqim deb nimaga aytiladi?

3.4. Istisnolarni boshqarish

Java tilida istisno — dastur kodi biror qismida hosil bo‘lgan istisno holatni tasvirlovchi ob’ektdir. Istisno holat yuzaga kelganda Exception sinfi ob’ekti yaratiladi. Bu ob’ekt shu tipdagi istisno holatni qayta ishlovchi usulga uzatiladi. Istisnolar g‘ayri oddiy holatlar haqida ma’lumot berish uchun «qo‘lda» yaratilishi mumkin.

Asoslar

Java tilida istisnolar qayta ishlash mexanizmi beshta kalit so‘zlardan foydalanadi: — try, catch, throw, throws i finally. Bu mexanizm ishlash sxemasi quyidagicha. Siz kod blokini bajarishga (try) urinasiz, va xatolik yuzaga kelsa, tizim istisno (throw) generatsiya qiladi. Bu istisno tipiga qarab ushlab olish (catch) yoki ko‘zda tutilgan qayta ishlovchiga (finally) uzatishingiz mumkin.

Quyida istisnolar qayta ishlash bloki umumiy ko‘rinishi keltirilgan.

```
try {
// kod blogi }
```

catch (TipIstisno1 e) {

// TipIstisno1 turidagi istisnoni qayta ishlovchi}

catch (TipIsklyucheniya2 e) {

// TipIstisno2 turidagi istisnoni qayta ishlovchi

throw(e) // istisnoni qayta yaratish }

finally {

}

Istisnolar turlari

Istisnolar ierarxiyasi yuqorisida Throwable sinfi joylashgan. Har bir tipdagi istisno Throwable sinf ostki sinfidir. Throwable sinfi ikki vorisi istisnolar ostki sinflar ierarxiyasini ikki shoxga ajratadi. Birinchisi —Exception sinfi— foydalanuvchi dastur kodi tomonidan ushlab olinishi lozim bo‘lgan istisnolarni ta’riflash uchun ishlatiladi. Ikkinchichi —Error - sinfi foydalanuvchi dastur kodi tomonidan ushlab olinmasligi lozim bo‘lgan istisnolarni ta’riflash uchun ishlatiladi.

Ushlab olinmagan istisnolar

Ob’ekt-istisnolar ma’lum istisno holatlar yuz kelishi natijasida Java bajaruvchi muhiti tomonidan avtomatik yaratiladi. Masalan navbatdagi

dastur bajarilishi natijasida nolьga bo‘lish yuzaga keladigan ifodani o‘z ichiga oladi.

class Exc0 {

public static void main(string args[]) {

int d = 0;

int a = 42 / d;

} }

Misol bajarilishi natijasi.

S:\> java Exc0

java.lang.ArithmeticException: / by zero

at Exc0.main(Exc0.java:4)

E’tibor bering yaratilgan istisnolar turi Exception ham Throwable ham emas. Bu Exception vorisi, yani: dastur bajarilish natijasida qanday xato yuz bergani haqida ma’lumot beruvchi ArithmeticException. Quyidagi shu sinf versiyasida xuddi shu istisno yaratiladi, lekin main usuli kodida emas.

```
class Exc1 {
static void subroutine() {
int d = 0;
int a = 10 / d;
}
public static void main(String args[]) {
Exc1.subroutine();
} }
```

Bu dasturda Java bajarish muxiti chaqiriqlar stekidagi hamma ma’lumotni chiqaradi.

```
S:\> java Exc1
java.lang.ArithmeticException: / by zero
at Exc1.subroutine(Exc1.java:4)
at Exc1.main(Exc1.java:7)
```

try va catch

Istisnolardan ximoya qilish lozim bo‘lgan dastur kodini berish uchun try kalit so‘zi ishlatiladi. Darhol try-blokdan so‘ng qayta ishlash lozim bo‘lgan istisno tipini beruvchi catch blok joylashadi.

```
class Exc2 {
public static void main(String args[]) {
try {
int d = 0;
int a = 42 / d;
```

```
}
catch (ArithmeticException e) {
System.out.println("division by zero");
}
} }
```

Yaxshi loyixalashtirilgan ko‘pgina catch-bo‘limlar maqsadi yaratilgan istisnoni qayta ishlash va dastur o‘zgaruvchilarini ma’lum holatga keltirish, toki dasturni xech qanday xatolik yuz bermagandek davom ettirish mumkin bo‘lsin (misolda quyidagi ogoxlantirish chiqariladi – division by zero).

Bir necha catch bo‘limlari

Ba’zi hollarda bitta dastur kodi turli tipdagi istisnolar yaratishi mumkin. Bunday holatlarni qayta ishlash uchun, Java try-blok uchun ixtiyoriy catch- bloklar kiritishga imkon beradi. Eng maxsus istisnolar sinflari birinchi kelishi kerak, chunki biror voris sinf ishlatilmaydi agar supersinfdan keyin kelsa. Quyidagi dasturda ikki tipdagi istisno ushlanadi, bu ikki maxsus qayta ishlovchilardan keyin Throwable sinfi hamma vorislarini ushlab oluvchi catch bo‘limi keladi.

```
class MultiCatch {
public static void main(String args[]) {
try {
int a = args.length;
System.out.println("a = " + a);
int b = 42 / a;
int c[] = { 1 };
c[42] = 99;
}
catch (ArithmeticException e) {
System.out.println("div by 0: " + e);
}
```

catch(ArrayIndexOutOfBoundsException e) {

System.out.println("array index oob: " + e);

}

} }

Bu misol parametrsiz ishga tushirilsa nolьga bo‘lish istisnosini keltirib chiqaradi. Agar buyruq qatorida bir yoki bir necha parametr berib a qiymati nolьdan katta o‘rnatilsa, bo‘lish operatori bajariladi lekin keyingi operatorda indeks massiv chegarasidan chiqish *ArrayIndexOutOf* Bounds istisnosi yaratiladi. Quyida ikkala usulda ishga tushirilgan dastur natijalari keltirilgan.

S:\> java MultiCatch

a = 0

div by 0: java.lang.ArithmeticException: / by zero

C:\> java MultiCatch 1

a = 1

array index oob: java.lang.ArrayIndexOutOfBoundsException:

Joylashtirilgan try operatorlari

Xuddi o‘zgaruvchilar ko‘rinish soxalari kabi try operatorlarini bir biriga joylashtirish mumkin. Agar quyi darajadagi try operatorida yaratilgan istisnoga mos catch bo‘limi mavjud bo‘lmasa, tashqi try operatori catch bo‘limlari tekshiriladi. Masalan ikki try operatori usulni chaqirish yordamida o‘zaro joylashtirilgan misol.

class MultiNest {

static void procedure() {

try {

int c[] = { 1 };

c[42] = 99;

}

```
catch(ArrayIndexOutOfBoundsException e) {
System.out.println("array index oob: " + e);
} }
public static void main(String args[]) {
try {
int a = args.length();
System.out.println("a = " + a);
int b = 42 / a;
procedure();
}
catch (ArithmeticException e) {
System.out.println("div by 0: " + e);
}
} }
```

Nazorat savollari

1. Istisno ta’rifini keltiring.
2. Istisnolarni qayta ishlovchi sinflar ierarxiyasini ko‘rsating.
3. Istisnoni qayta ishlash sintaksisini keltiring.
4. Istisnolar bilan qanday operatorlar bog‘liq?
5. Istisnoni generatsiya qiluvchi funksiya sintaksisini keltiring.

3 bob buyicha hulosalar

Ushbu bobda murakkab ob’ektlar bilan ishlash, y’ani javada tulaqonli amaliy dasturlar yaratish mexanizmlai kurib chqildi. Bunda ob’ektga yunaltirilgan dasturlashning nasl olish, inkapsulyatsiyava polimorfizm kabi tamoyillardan foydalanish va ushbu tamoyillar asosida dasturiy tuzilma shablonini yaratish

uslublari uzlashtirildi. Ushbu uslublarni uzlashtirish va dasturlash jarayonida mohirona qullay bilish har bir dasturchining asosiy vazifasi hisoblanadi.

IV.BOB JAVA DASTURLASH TILINING TURLI SOHALARGA QO‘LLANILISHI

4.1. Foydalanuvchi interfeysini yaratish

Biz oldingi mavzularda matnli terminal bilan bog‘liq va buyruq satridan ishga tushiriladigan dasturlar yozdik. Bunday dasturlar konsolli ilovalar deb ataladi. Ular foydalanuvchi bilan interaktiv aloqa talab etilmaydigan serverlarda bajarilishi uchun ishlab chiqiladi. Foydalanuvchi bilan jips muloqatda bo‘ladigan, klaviatura va sichqoncha signallarini qabul qiluvchi dasturlar grafik muhitda ishlaydi. Grafik muhitda ishlashga muljallangan har bir ilova hech bo‘lmaganda, ilovaning ishi amalga oshiriladigan va operatsion tizimning grafik qobig‘ida qayd qilinadigan, bitta oyna hosil qilishi kerak. Bu oyna operatsion tizim va boshqa oynalar bilan muloqat qilishi: qoplanishi, siljishi, ulchamlarini o‘zgartirishi, belgiga jamlanishi va boshqalarni amalga oshirishi kerak.

Ko‘plab grafik tizimlar mavjud: MS Windows, X Windows System, Macintosh va boshqalar. Ularning har birida oyna va uning komponentalari: menyu, axborot kiritish maydoni, tugmalar, ruyhatlar, aylantirish tasmasi kabilarni qurishning o‘z tartib qoidalari mavjud. Grafik API yuzlab funksiyalarni o‘z ichiga oladi.

Oyna va uning komponentalarini yaratishni osonlashtirish uchun MFC, Motif, OpenLook, Qt,Tk, Xview, OpenWindows va boshqalar kabi ko‘plab sinflar bibliotekalari yaratilgan. Bunday bibliotekaning har bir sinfi bir vaqtning o‘zida shu sinfning o‘zidagi yoki boshqa sinflardagi metodlar yordamida boshqariladigan butun bir grafik komponentni bayon qiladi.

Java texnologiyada Java ilovaning ixtiyoriy yoki ko‘plab grafik muhitlarda ishlashi kerakligi tufayli vaziyat murakkablashadi. Aniq bir grafik muhitga bog‘liq bo‘lmagan sinflar bibliotekasi zarur. JDK ning boshlang‘ich versiyalarida muammo quyidagicha hal qilingan. Grafik ob’ektlar bilan ishlash metodlarini o‘z ichiga oladigan interfeyslar ishlab chiqilgan. AWT bibleotekasi sinflari bu interfeyslarni ilova yaratish uchun joriy qilishadi. Java ilova bu metodlarni grafik ob’ektlarni joylashtirish va joyini o‘zgartirish, ularning ulchamlarini o‘zgartirish va boshqa ob’ektlar bilan bog‘lanishlarini amalga oshirish uchun ishlatadi.

Boshqa tomondan, aniq grafik muhitdagi ekran bilan ishlash uchun bu interfeyslar har bir muhit uchun alohida joriy qilinadi. Har bir grafik qobiqda ushbu grafik qobiq vositalari va operatsion tizimning grafik bibliotekasi yordamida ular o‘ziga xos bajariladi. Bunday interfeys peer-interfeys deb atalgan.

Reer-interfeyslarga asoslangan Javadagi sinflar bibliotekalari AWT (Abstract Window Toolkit) nomini oldi. Java ilovada yaratilgan va reer-interfeyslarga asoslangan ob'ektni ekranga chiqarayotganda unga juft bo‘lgan operatsion tizim grafik tizimining ekranda aks ettiriluvchi ob'ekti hosil qilinadi. Ilova ishlayotganda bu ob'ektlar o‘zaro chambarchas aloqada bo‘lishadi. Shuning uchun AWT grafik ob'ektlari har bir grafik muhitda shu muhitga xos bo‘lgan ko‘rinishga ega bo‘ladi. Masalan, MS Windows, Motif, OpenLook, OpenWindows va boshqalarda AWT da yaratilgan oynalar ular uchun o‘z oynalari kabidir.

C++ tilida yozilgan reer-interfeyslar va ular metodlarining shunday joriy qilinishi tufayli har bir platforma uchun alohida alohida JDK variantini ishlab chiqishga to‘g‘ri keladi.

JDK ning 1.1 versiyasida AWT kutubxonasi qayta ishlangan. Unga to‘laligicha Java da yozilgan va reer-interfeyslarga bog‘liq bo‘lmagan komponentalarni yaratish imkoniyati qo‘shilgan. Bunday komponentalarni reer-interfeyslar orqali joriy qilingan “og‘ir” (heavy) komponentalarga nisbatan “engil” (lightweight) komponentalar deb atash boshlashdi.

“Engil” komponentalar hamma joyda bir hil ko‘rinishga ega. Ular yaratilgandagi ko‘rinishini (look and feel) saqlaydi. Bundan tashqari ilovani shunday ishlab chiqarish mumkinki, uni ishga tushirgandan keyin biror bir aniqlangan ko‘rinish: Motif, Metal, Windows 95 yoki birortasini tanlash yoki bu ko‘rinishni ish jarayoning ixtiyoriy paytida o‘zgartirish mumkin.

“Engil” komponentalarning bu qiziqarli xususiyati PL&F (Pluggable Look and Feel) yoki “plaf” degan nomni oldi.

Java da Swing deb nomlangan “engil” komponentalarning kengaytirilgan kutubxonasi yaratigan. Unda AWT kutubxonasining barcha komponentalari qayta yozilgan. Shuning uchun Swing kutubxonasi mustaqil ishlatilishi mumkin.

Swing sinflari kutubxonasi JDK 1.7 ga qo‘shimcha sifatida qo‘yiladi. Java 2 SDK tarkibiga u asosiy grafik sinflar kutubxonasi sifatida kiritilgan. U “100% Pure Java”g‘oyasini joriy qiladi.

Java 2 da AWT kutubxonasi rasm solish, matn va tasvirni chiqarish kabi vositalarni qo‘shish, matnlarni DnD (Drag and Drop) metodi yordamida o‘zaro siljishi orqali ancha kengaytirilgan va u Java 2D deb nomlangan.

Bundan tashqari Java 2 ga kiritish/chiqarishning Input Method Framework deb ataladigan yangi metodlari va nurli pero, Accessibility deb atalgan Broyl klaviaturasi kabi qo‘shimcha qurilmalar bilan aloqa vositalari kiritilgan.

Java 2 ning barcha bu vositalari: AWT, Swing, Java 2D, DnD, Input Method Framework va Accessibility Java ning JFC (Java Foundation Classes) deb atalgan grafik vositalari kutubxonasini tashkil etdi. Bu vositalarning har birining bayoni butun boshli bir kitobni tashkil etadi, shuning uchun AWT kutubxonasining asosiy vositalarini tanishtirish bilan cheklanamiz.

Komponent va konteyner

Foydalanuvchi grafik interfeysi (FGI) ning asosiy tushunchasi grafik tizim komponenti(component) dir. Rus tilida bu so‘z biror bir tarkibiy qism, nimanidir elementi degan ma’noni bildiradi, biroq grafik interfeysda bu tushuncha ancha aniqlashtirilgan. U grafik interfeysda boshqa elementlarga bog‘liqsiz ravishda ishlatish mumkin bo‘lgan, alohida to‘liq aniqlangan elementni bildiradi. Masalan, bu ma’lumot kiritish maydoni, tugma, menyu satri, aylantirish tasmasi, radiotugma bo‘lishi mumkin. Ilova oynasining o‘zi ham uning komponenti. Komponentlar ko‘rinmaydigan bo‘lishi ham mumkin, masalan, komponentlarni birlashtiruvchi panelь ham komponent hisoblanadi.

Component sinfining ob’ekti yoki Component sinfini kengaytiruvchi ixtiyoriy sinf ob’ekti AWT komponent hisoblanadi. Component sinfida foydalanuvchi grafik interfeysining ixtiyoriy komponent bilan ishlashning umumiy metodlari jamlangan. Bu sinf —AWT kutubxonaning markazidir.

Har bir komponent ekranga chiqarilishidan oldin konteyner (container) ga joylashtiriladi. Konteyner komponentlarni ekranga qanday chiqarish kerakligini biladi. Java tilida konteyner —Container sinfining yoki uning ixtiyoriy kengaymasining ob’ektidir. Bu sinfning bevosita vorisi —jcomponent sinfi — Swing kutubxonasida ko‘plab sinflar ierarxiyasi chuqqisidir.

Component sinfi yoki uning ixtiyoriy kengaymasi ob’ekti — komponentni yaratib, uni oldindan hosil qilingan container sinfi yoki uning ixtiyoriy kengaymasiga *add()* metodlarining biri yordamida qo‘shish kerak.

Sinf Container sinfining o‘zi ham ko‘rinmaydigan komponent hisoblanadi, u Component sinfini kengaytiradi. Shunday qilib, konteynerga boshqa komponentlarni o‘z ichiga oluvchi konteynerlarni ham joylashtirish mumkin. Bunda komponentlarni joylashtirishning katta egiluvchanligiga erishiladi.

Ilovaning asosiy oynasi operatsion tizim bilan foal muloqatda bo‘ladi, uni grafik tizim qoidalariga asosan qurish zarur. U ekran bo‘ylab joyini o‘zgartirishi, ulchamlarini o‘zgartirishi, sichqoncha va klaviatura harakatlariga tasirchan bo‘lishi kerak. Oynada, kamida quyidagi standart komponentlar bo‘lishi kerak:

• Sarlavha satri(title bar), uning chap tomonida kontekst menyusi tugmalari, ung tomonida esa, jamlash, yoyish va ilovani yopish tugmalari joylashtirilishi kerak.

• Majburiy bo‘lmagan menyu satri(menu bar) menyuning ochiladigan qismlari bilan.

• Gorizontal va vertikal aylantirish tasmalari(scrollbars).

• Oyna sichqoncha harakatiga tasirchan rom(border) bilan o‘ralgan bo‘lishi kerak.

Shunday komponentalarga ega tayyor oyna Frame sinfida bayon qilingan. O‘z oynamizni hosil qilish uchun, yaratilayotgan sinfni Frame sinfning kengaymasi qilish etarli. Bu 8.1 listingda ko‘rsatilgan. Dastur jami 8 ta satrdan tashkil topgan.

4.1 listing. Ilovaning o‘ta sodda oynasi

```
import java.awt.*;
class TooSimpleFrame extends Frame{
  public static void main(String[] args){
    Frame fr = new TooSimpleFrame();
  fr.setSize(400, 150);
  fr.setVisible(true);
  }
}
```

TooSimpleFrame sinfi Frame sinfining kengaymasi bo‘lib, uning barcha xossalariga ega. Unda oyna ekzempilyari fr yaratiladi, va oyna ulchamlari — 400x150 piksel qilib— *setSize()* metodi yordamida o‘rnatiladi. Agar oyna o‘lchami berilmasa, u holda ekranda minimal ulchamli oyna hosil bo‘ladi — faqat sarlavha

satri. Albatta uni keyinchalik sichqoncha yordamida ixtiyoriy ulchamgacha kengaytirish mumkin.

Keyin oynani *setVisibie(true)* metodi yordamida ekranga chiqariladi. AWT kutubxonasi nuqtai nazaridan, oynani yaratish, operativ hotirada kerakli piksellar bilan to‘ldirilgan joy ajratilishidir, bu joy qiymatini ekranga chiqarish boshqa masala, uni *setVisible(true)* metodi yordamida xal qilinadi. Albatta, bunday oynada ishlab bo‘lmaydi. Unda sarlavha yo‘q, shuning uchun bunday oynani yopib bo‘lmaydi. Biroq uni ekranda siljitish, ulchamlarini o‘zgartirish, masalalar paneliga jamlash va yoyish mumkin. Ammo, ilovani tugatish buyrug‘ini biz dasturlashtirmadik. Oynani hech qanday vosita yordamida yopib bo‘lmaydi, uni faqat operatsion tizim vositalari yordamida yopish mumkin.

4.2 listingda 4.1 listingdagi dasturga oyna sarlavhasi, va ilovani tugatish metodiga murojaat qilish imkoniyatlari qo‘shilgan.

4.2 listing. Ilova sodda oynasi

```
import java.awt.*;
import j ava.awt.event.*;
class SimpleFrame extends Frame{
  SimpleFrame(String s){
    super (s);
    setSize(400, 150);
    setvisible(true);
    addWindowListener(new WindowAdapter(){
      public void windowClosing(WindowEvent ev){
        System.exit (0);
      }
    });
  }
  public static void main(String[] args){
    new SimpleFrame(" Moyaprogramma");
```

```
  }
}
```

Dasturga o‘zining supersinfi *Frame* konstruktoriga murojjat qiluvchi *SimpleFrame* sinfi konstruktori qo‘shilgan. U o‘z argumenti s ni sarlavha satriga yozadi.

Konstruktorga oyna o‘lchamlarini o‘rnatish, uni ekranga chiqarish va oyna bilan bog‘liq harakatlarga tasirchan addWindowListener() metodiga murojaat qo‘shilgan. Bu metodga argument sifatida WindowAdapter sinfini kengaytiruvchi nomsiz ichki sinf ekzempilyari uzatiladi. Bu nomsiz ichki sinf oynani yopishga urinishni qayta ishlaydigan *windowciosing ()* metodini joriy qiladi. Bu joriy qilish juda sodda, ilova system sinfining *exit()* statik metodi bilan tugaydi. Oyna avtomatik tarzda yopiladi.

Bularning barchasi to‘laligicha 12 bobda qaraladi, xozircha ushbu satrlarni o‘z dasturingizga qo‘shib turing. U oynani yopish va ilovani tugatish imkonini beradi.

Shunday qilib oyna tayyor. Biroq u xozircha bo‘sh. Biz unda ananaga ko‘ra. "Hello, World!" so‘zini biroz o‘zgartirilgan ko‘rinishda chiqaramiz, 3.3 listingda bu chiqarish to‘laligicha keltirilgan, 8.1 listing esa oynani namoyish etadi.

4.1 rasm. Salomlashish dasturi oynasi

4.3 listing. Salomlashuvchi grafik dastur

```
import java.awt.*;
import java.awt.event.*;
class HelloWorldFrame extends Frame{
  HelloWorldFrame(String s){
    super(s);
  }
```

```
  public void paint(Graphics g){
    g.setFont(new Font("Serif", Font.ITALIC | Font.BOLD, 30));
    g.drawstring("Hello, XXI century World!", 20, 100);
  }
  public static void main(String[] args){
    Frame f = new HelloWorldFrame("Salom, XXI asr olami!");
    f.setSize(400, 150);
    f.setvisible(true);
    f.addWindowListener(new WindowAdapter(){
      public void windowciosing(WindowEvent ev)(
        System.exit(0);
      }
    });
  }
}
```

Matnni chiqarish uchun biz component sinfining paint () metodini qayta

paint() metodi argument sifatida matnni *drawstring()* metodi yordamida chiqaruvchi Graphics sinfining g ekzempilyarini oladi.Bu metodda matndan tashqari biz oynada satr brshlanadigan nuqtani — 20 piksel chap chetidan va 100 piksel tepadan deb ko‘rsatamiz. Bu nuqta — matndagi birinchi harfning chap qo‘yi nuqtasi hisoblanadi. Bundan tashqari, yangi katta ulchamdagi — 30 punktli, yarim qora kursiv "Serif" shirifti o‘rnatildi. Barcha shiriftlar — Font, sinfi ob’ekti bo‘lib, Graphics sinfining *setFont()* metodi yordamida beriladi.

Shiriftlar bilan ishlash keyingi bobga qoldiriladi.

4.3 listingda, turlilik uchun, oyna o‘lchamlarini o‘rnatish, uni ekranga chiqarish va dasturni tugatishni biz main() metodiga o‘tkazdik. Ushbu sodda misoldan ko‘rishimiz mumkinki AWT kutubxonasi katta va shoxlangan, unda bir biri bilan muloqatda bo‘luvchi ko‘plab sinflar mavjud. Tez tez ishlatilib turadigan AWT sinflari shajarasini ko‘raylik.

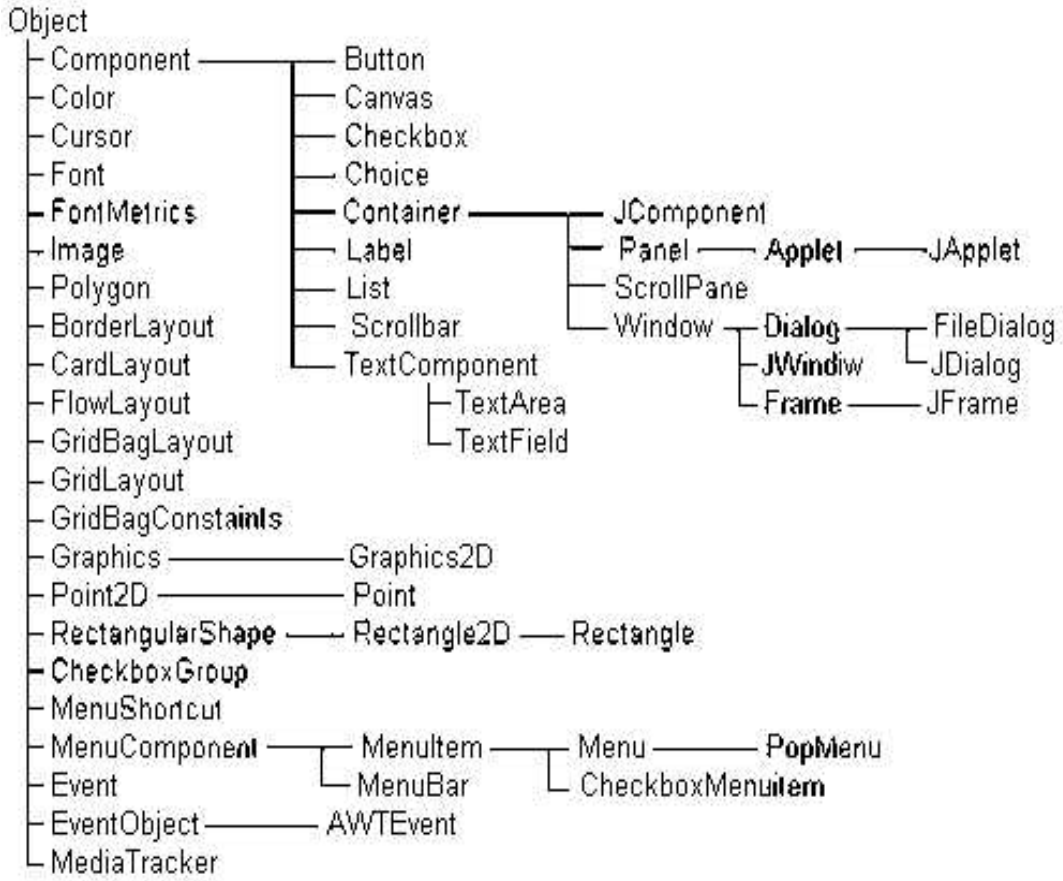
AWT sinflari shajarasi

4.2 rasmda AWT asosiy sinflari shajarasi ko‘rsatilgan. Uning asosini tayyor komponentalar: Button, Canvas, Checkbox, Choice, Container, Label, List, Scrollbar, TextArea, TextField, Menubar, Menu, PopupMenu, Menultem, CheckboxMenuItem lar tashkil etadi. Agar bular ham etishmasa, Canvas sinfidan xususiy “og‘ir” komponentalarni, Component sinfidan esa “engil” komponentlarni hosil qilish mumkin.

Asosiy konteynerlar — Panel, ScrollPane, Window, Frame, Dialog, FileDialog sinflaridir. O‘z “og‘ir” komponentalarini Panel, sinfidan, "engil" larini esa — container sinfidan hosil qilish mumkin.

Sinflarning bir butun majmuasi, komponentalarni joylashtirish, rasm va tasvirlarga rang, shirift, berish, sichqoncha va klaviatura signallarini sezishga yordam beradi.

4.2 rasmda Swing kutubxonasi shajarasining boshlang‘ich sinflari — JComponent, JWindow, JFrame, JDialo, Japplet sinflari ham ko‘rsatilgan.



4.2 rasm. AWT asosiy sinflari shajarasi

Xulosa

Ko‘rib o‘tganimizdek, grafik sinflar kutubxonasi AWT juda katta va etarlicha ishlab chiqilgan. Bu sinflar turli tumanligi grafik interfeys qurish masalasining turli tumanligini aks ettiradi. Grafik interfeysni yaxshilashga intilish cheksiz. U sinflarning yangi kutubxonalarini yaratish va borlarini takomillashtirishga olib keladi. Erkin ishlab chiqaruvchilar tomonidan ko‘plab

grafik kutubxonalar: KL Group, JBCL kabilar ishlab chiqarilgan va yangi yangilari yaratilmoqda.

Keyingi boblarda biz AWT kutubxonasini, foydalanuvchining tasvirli, animatsiya va tovushli xususiy grafik interfeysi ilovasini yaratishda qanday ishlatish mumkinligini qaraymiz.

Nazorat savollari

1. Foydalanuvchi interfeysi tushunchasi.
2. Komponent nima?
3. Konteyner nima?
4. AWT kutubxonasini tasvirlab bering.

4.2. Java Swing da xodisalar bilan ishlash

Swing kutubxonasiga kirish

Kutubxonani ochish

Zamonaviy dasturlarga grafik foydalanuvchi interfeysi (GUI) kerak. Foydalanuvchilar konsol orqali ishlash odatidan mahrum bo'ldilar: dasturni boshqarish va kirish elementlarini tugmalar, matn qutilarini, ochiladigan ro'yxatlarni va hokazolarni o'z ichiga olgan nazorat elementlari (dasturiy tarkibida ingl. Tarkibiy qismlar deb ataladi) orqali kiritadi.

Zamonaviy dasturlash tillarining har biri standart boshqaruv vositalaridan foydalanish uchun ko'plab kutubxonalarga ega. Eslatib o'tamiz, dasturlash kutubxonasi ostida ma'lum bir qator vazifalarni hal qilish uchun mo'ljallangan tayyor sinflar va interfeyslar majmui.

Java-da, grafik foydalanuvchi interfeysi yaratish uchun uchta ingl. Komponentlar kutubxonasi mavjud. Ularning eng qadrli qismi AWT deb ataladi. Uni loyihalashda bir qator kamchiliklar e'tirof etilgan, bu ish bilan ishlashni qiyinlashtiradi. Swing kutubxonasi AWT-ga asoslangan va komponentlarini aksariyat qismini o'zi bilan almashtiradi, undan ehtiyotkorlik va qulaylik bilan yaratilgan. Uchinchidan, eng yangi kutubxona SWT deb ataladi.

Har bir kutubxona tugmalar, ro'yxatlar, derazalar, menyular va boshqalar bilan ishlash uchun bir qator sinflarni taqdim etadi, lekin bu sinflar boshqacha tarzda

yaratilgan: ular turli xil parametrlarga ega bo'lgan turli uslublar to'plamiga ega, shuning uchun dasturni bir kutubxonadan boshqasiga (masalan, ishlashni oshirish uchun) oson emas. Bu deyarli bir dasturlash tilidan ikkinchisiga o'tish kabi deyarli barcha tillar xuddi shu narsani bajarishi mumkin, lekin har birining o'z sintaksisi, o'z dasturiy tuzilishi va ko'pgina fokuslari bor.

Shu sababli, uchta kutubxonani ko'rib chiqish o'rniga biz ulardan birini - Swing kutubxonasini yaxshiroq tushunishga harakat qilamiz. Uning yordami bilan to'liq grafik interfeysi ishlab chiqilishi mumkin.

JFrame oynasi

Har bir GUI dasturi derazada ishlaydi va ish vaqtida bir nechta qo'shimcha derazalar ochilishi mumkin.

Swing kutubxonasi JFrame sinfini tavsiflaydi, bu ramka va nom satriga ega bo'lgan oyna («ixcham», «To'liq ekran» va «Yopish» tugmalari bilan). Ekran o'lchamini o'zgartirishi va harakatlanishi mumkin.

Bo'sh oyna haqida

Parametrlarsiz *JFrame()* konstruktori bo'sh oyna hosil qiladi. JFrame (satrlari sarlavhasi) konstruktori nom sarlavhasi bilan bo'sh oynani yaratadi.

Bo'sh deraza ko'rsatadigan oddiy dasturni yozish uchun bizga yana uchta usul kerak:

setSize(int width, int height) - oynaning o'lchamini belgilaydi. Agar o'lchovlar aniqlanmasa, oynada nolinchi balandligi bo'ladi, unda nima bo'lishidan qat'iy nazar, foydalanuvchi ishga tushgandan so'ng oynani qo'l bilan uzata oladi. Oynaning o'lchamlari nafaqat "ishlaydigan" maydonni, balki chegaralarni va nom satrini ham o'z ichiga oladi.

setDefaultCloseOperation (int operatsiyalari) - foydalanuvchi xochni bosish orqali oynani yopganda bajarilishi kerak bo'lgan amalni belgilash imkonini beradi. Odatda, dasturning ishga tushirilishini to'xtatganda, dastur bir yoki bir nechta oynaga ega. Ushbu xatti-harakatni dasturlash uchun, JFrame sinfida tasvirlangan doimiy EXIT_ON_CLOSE operatsiya parametri sifatida o'tishingiz kerak.

setVisible(boolean visible) - deraza yaratilganda u sukut ko'rinmas. Ekranda oynani ko'rsatish uchun ushbu uslub parametr rostla chaqiriladi. Agar parametrni noto'g'ri deb chaqirsangiz, oyna yana ko'rinmas bo'ladi.

Endi biz deraza yaratadigan, uni ekranda ko'rsatadigan va foydalanuvchi oynani yopganidan so'ng chiqadigan dasturni yozishimiz mumkin.

Misol:

```
import javax.swing.*;
public class MyClass {
public static void main (String [] args) {
JFrame myWindow = new JFrame("Sinov oynasi");
myWindow.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
myWindow.setSize(400, 300);
myWindow.setVisible(true);
}
}
```

Swing kutubxonasining ko'plab sinflari bilan ishlash uchun java.swing paketini import qilish kerak.

Odatdagidek, derazani ko'rsatishdan oldin, siz oddiy dasturga qaraganda ko'proq harakat qilishingiz kerak. Ko'plab boshqaruvlarni yaratib, ularning tashqi ko'rinishini sozlashingiz kerak, ularni oynaning o'ng joylariga joylashtirishingiz kerak. Bunga qo'shimcha ravishda, dastur juda ko'p derazalarga ega bo'lishi mumkin va ularni *main()* usulida konfiguratsiya qilish noqulay va noto'g'ri, chunki u inkapsulatsiya tamoyilini buzadi: ma'lumotlarni va ularni ishlaydigan buyumlarni bir joyga to'plang. Har bir oyna uchun uning o'lchamlari va mazmunini mustaqil ravishda hal qilish mantiqan to'g'ri bo'ladi. Shuning uchun derazalar bilan dasturning sinfik tuzilishi quyidagicha ko'rinadi:

SimpleWindow.java faylida:

```
public class SimpleWindow extends JFrame {
SimpleWindow(){
super("Oyna");
setDefaultCloseOperation(EXIT_ON_CLOSE);
setSize(250, 100);
}
}
```

Faylda Program.java:

```
public class Program {
public static void main (String [] args) {
JFrame myWindow = new SimpleWindow();
myWindow.setVisible(true);
}
}
```

Misol, deraza JFrame ning izdoshi bo'lgan alohida sinfda tasvirlanganligini va uning ko'rinishini va xatti-konstruktorida (birinchi buyruq supersinf ijosorini chaqiradi) tuzilishini ko'rsatadi. Asosiy usuli dasturni boshqarishda mas'ul bo'lgan boshqa sinfda mavjud. Bu sinflarning har biri juda sodda, har bir kishi o'z biznesida ishtirok etmoqda, shuning uchun ularni tushunish va osonroq amal qilish oson (masalan, kerak bo'lsa, yaxshilash).

SetVisible() usuli oddiygina "SimpleWindow" da chaqirilmaydi, bu juda mantiqiy: bu tugma qayerda joylashganligi va qanday o'lchovlar bo'lishi kerakligi uchun oynaning o'zi kuzatiladi, ammo qaysi vaqtda qaysi oyna paydo bo'ladi? Displey - dasturning boshqarish sinfi imtiyozidir.

To'g'ridan-to'g'ri deraza nazorati ostida joylashtirilmaydi. Buni amalga oshirish uchun, deraza butun maydonini egallagan kontent panelidan * foydalaning. JFrame sinfining *getContentPane()* usuli yordamida ushbu panelga kirishingiz mumkin. Qo'shish (topgan komponentining) usuli yordamida siz unga biron-bir boshqaruvni qo'shishingiz mumkin.

Ushbu darsning misollarida biz faqat bitta nazorat elementidan foydalanamiz - tugma (qurilmaning tafsilotlariga kirmasdan). Ushbu tugma JButton sinfi tomonidan tavsiflanadi va konstruktiv tomonidan String turi parametri bilan yaratiladi - bu yozuv.

Buyruqlar bilan oynamizning tarkibiy qismiga bir tugma qo'shing:

JButton new Button = yangi JButton ();
getContentPane() funktsiyasini qo'shish (new Button);

Natijada tugmachali oyna. Button barcha mavjud oyna maydonini oladi. Ushbu effekt barcha dasturlarda foydali emas, shuning uchun paneldagi elementlarni tartibga solishning turli usullarini o'rganishingiz kerak.

Konteyner sinfi (konteyner)

Boshqa elementlarni o'z ichiga olgan elementlarga konteynırlar deyiladi. Ularning barchasi Konteyner sinfining avlodlari bo'lib, ulardan bir nechta foydali usullarni egallaydi:

qo'shing (komponentli komponent) - konteynerga komponentni qo'shadi;

olib tashlash (komponentli komponent) - komponentni idishdan olib tashlaydi;

removeAll() - konteynerning barcha elementlarini o'chiradi;

getComponentCount() - Konteynerdagi elementlarning sonini qaytaradi.

Konteyner sinfida ko'rsatilganlardan tashqari konteynerda mavjud komponentlarni boshqarish uchun taxminan ikki o'nlab usul mavjud. Ko'rib turganingizdek, ular sinf to'plash usullariga o'xshash. Buning ajablanarli joyi yo'q, chunki aslida konteyner to'plamdir, biroq alohida turdagi - ingl. Elementlarni saqlashga qo'shimcha ravishda, konteyner ularning kengaytirilishi va chizilganligi bilan shug'ullanadi. Ayniqsa, berilgan koordinatalar nuqtasi (komponentning yuqori chap burchagidan sanaladi) va boshqa qatorlarni topgan komponentni qaytaradigan *GetComponentAt (int x, int y)* usuli mavjud. Biz mavhum konteynerga batafsil ma'lumot kiritmaymiz, lekin tezda uning eng ko'p ishlatiladigan avlodiga, ya'ni JPanel sinfiga o'tamiz.

JPanel (panel) sinfi

JPanel bu boshqa elementlarning joylashtirilishi mumkin bo'lgan to'rtburchaklar bo'shlig'i bo'lgan nazoratdir. Elementlar konteyner sinfidan meros qilingan usullar bo'yicha qo'shiladi va o'chiriladi.

Button bilan namunada kontent paneliga qo'shilgan tugma hamma bo'sh joyni egallab olgani kabi biz ham tomosha qildik. Bu har doim ham sodir bo'lmaydi. Darhaqiqat, har bir panel panelga qo'shilgan elementlarning nisbatan joylashishni aniqlash strategiyasini aniqlaydigan "taqsimlash menejeri" ga ega. SetLayout metodini (LayoutManager menedjeri) ishlatish mumkin. Biroq, bu usulga kerakli parametrni kiritish uchun menejerlarning nima ekanligini bilish kerak.

FlowLayout davriy oqim boshqaruvchisi

Eng oson joylashtirish menejeri - FlowLayout. Panelning kattaligiga qarab, bu panelga qo'shilgan qismlarni o'z navbatida navbatga qarab tartibga soladi. Keyingi element joriy satrga mos kelmasa, u keyingi liniyaga o'tkaziladi. Buni misol bilan kuzatib borish yaxshidir. SimpleWindow konstruktorini quyidagicha o'zgartiraylik:

```
SimpleWindow(){
super("Sinov oynasi");
setDefaultCloseOperation(EXIT_ON_CLOSE);
JPanel panel = new JPanel();
panel.setLayout(new FlowLayout());
panel.add(new JButton("Tugma"));
panel.add(new JButton("+"));
panel.add(new JButton("-"));
panel.add(new JButton("Uzun yozuvli tugma"));
```

setContentPane(panel);
setSize(250, 100);
}

Joylashuv menejerlari java.awt paketida tavsiflanadi. Kerakli sinflarni import qilishni unutmang.

Dastur boshlanganidan keyin paydo bo'ladigan oynaning xatti-harakatiga e'tibor bering. Undagi to'rtta tugmacha matn muharriri (markazlashgan paytda) sifatida ajratilgan. Dastur ishga tushirilayotganda oynani o'lchamoqchi bo'lsangiz, effekt sezilarli bo'ladi.

Keling, misolning matnini tahlil qilaylik. Yangi FlowLayout layout menejeri parametrsiz konstruktiv tomonidan yaratiladi. Dasturda qidiruv o'zgaruvchidan foydalanilmaydi. Ya'ni, ikkita jamoaning o'rniga:

FlowLayout new Layout = yangi FlowLayout();
panel.setLayout (new Layout);

Biz ulardan foydalanamiz:

panel.setLayout (yangi FlowLayout ());

Kelajakda biz yaratilgan ob'ektga (bu misol uchun to'g'ri keladigan) kirishga hojat bo'lmagan hollarda bu juda maqbuldir. Biz tartib-menejeri yaratamiz, keyin uni darhol panelga ulang - va shu. Endi panel va boshqaruvchi bir-birlari bilan umumiy til topadi.

Panel va uning menejeri o'rtasidagi munosabatlar haqida

Xuddi shunday, panelga yangi tugmalar qo'shamiz. Biz bu erda boshqa tugmalarga murojaat qilmoqchi emasmiz, shuning uchun ular uchun o'zgaruvchan parametrlar yo'q.

SetContentPane(JPanel panel) usuli oynaning mazmun panosini almashtirish imkonini beradi.

BorderLayout Chegar administratori

BorderLayout layout menejeri panelni beshta sohaga ajratadi: markaziy, yuqorigi, pastki, o'ng va chap. Ushbu hududlarning har birida siz aniq bir komponentni qo'shishingiz mumkin va komponent unga ajratilgan maydonni egallaydi. Yuqori va pastki joylarga qo'shilgan qismlar kengligi bo'yicha o'ngga va chapga qo'shiladi

va markazga qo'shilgan komponentlar panelning qolgan qismini to'liq to'ldirish uchun cho'ziladi.

Turar joy menejeri BorderLayout bilan yig'ilishda bir elementi qo'shib bo'lsa, usul (), qaysi maydoni mo'ljallangan qo'shish ko'rsatib qo'shimcha lozim. "Shimoliy", "Janubiy", "Sharq", "G'arb" va "markazi": bu uchun asosiy nuqtalari nomlari bilan chiziqlari bor. Shimoliy, Janubiy, sharq, g'arb va MARKAZI (chiziq xato qilish mumkin va uni payqamadi, chunki, lekin siz noto'g'ri doimiy nomini yozish bo'lganingizda tuzuvchi bir ogohlantirish beradi): Lekin buning o'rniga u sinf BorderLayout belgilangan Sobit foydalanish tavsiya. Agar bitta parametr bilan, odatdagidek *Add()* usulini foydalanayotgan bo'lsangiz, element markazi qo'shiladi.

Mundarija bo'lmasiga, nima tugmasini bosing va (u markaziy mintaqadagi qo'shilgan edi) bir butun sifatida butun oynasini etgan, aynan bunday shartnoma hisoblanadi. BorderLayout effektini kuzatish uchun biz beshta joyga tugmalar qo'shamiz:

```
SimpleWindow(){
super("Sinov oynasi");
setDefaultCloseOperation(EXIT_ON_CLOSE);
getContentPane().add(new JButton("Tugma"), BorderLayout.NORTH);
getContentPane().add(new JButton("+"), BorderLayout.EAST);
getContentPane().add(new JButton("-"), BorderLayout.WEST);
getContentPane().add(new JButton("uzun yozuvli tugma"),
BorderLayout.SOUTH);
getContentPane().add(new JButton("MARKAZGA!"));
setSize(250, 100);
}
```

Agar siz oynani qayta o'lchamoqchi bo'lsangiz, ushbu effekt yaxshi kuzatiladi.

Ushbu joylashuv tasodifiy ko'rsatuv tarkib panelida ishlatilmaydi. Ko'p dasturda asboblar paneli, vaziyat paneli va boshqalarni joylashtirish uchun oynaning chekkalari atrofida joylar ishlatiladi. Markazdagi bitta komponentga nisbatan cheklash mutlaqo muhim emas, chunki bu komponent tarkibiy elementlar bilan boshqa paneli va har qanday boshqaruvchi bilan bo'lishi mumkin.

Jadvalni joylashtirish menejeri GridLayout

GridLayout paneli bir xil kenglik va balandlikdagi kameralarga bo'linadi (shuning uchun deraza jadvalga o'xshash bo'ladi). Ushbu tartib bilan panelga qo'shilgan har bir element bitta kamerani butunlay egallaydi. Hujayralar yuqoridagi chapdan boshlab o'z navbatida elementlarga to'ldiriladi.

Bu menejer, ilgari ko'rib chiqilganlardan farqli o'laroq, parametrlarga ega bo'lgan konstruktor tomonidan (to'rtta tamsayı) yaratiladi. Siz ustunlar sonini, qatorlarni va kameralar orasidagi masofani gorizontal va vertikal ravishda ko'rsatishingiz kerak. Quyidagilarni bajaring va natijani kuzating.

```
SimpleWindow(){
super("Tajriba oynasi");
setDefaultCloseOperation(EXIT_ON_CLOSE);
JPanel panel = new JPanel();
panel.setLayout(new GridLayout(2,3,5,10));
panel.add(new JButton("Tugma"));
panel.add(new JButton("+"));
panel.add(new JButton("-"));
panel.add(new JButton("Uzun yozuvli tugma"));
panel.add(new JButton("yana tugma"));
setContentPane(panel);
setSize(250, 100);
}
```

BoxLayout Blok qutisi menejeri va Box sinfi

BoxLayout menejeri elementlarni satrda yoki ustunda joylashtiradi.

Odatda, ushbu menejer bilan ishlash uchun, bloklash joylashuvi allaqachon konfigüre qilingan bir panel bo'lgan yordamchi sinf Box-dan foydalaning. Ushbu panel tuzuvchi tomonidan emas, balki Box sinfida belgilangan ikkita statik usullardan biri tomonidan yaratiladi: *createHorizontalBox() and createVerticalBox()*.

Bloklarga joylashtirilgan panelga qo'shilgan elementlar birma-bir tartibda joylashtiriladi. Elementlar orasidagi masofa sukut bo'yicha nol bo'ladi. Shu bilan birga, komponent o'rniga siz ko'rinmas "spacer" ni qo'shishingiz mumkin, faqat bitta vazifa qo'shni elementlarni uzaytirish va ular orasidagi masofani belgilashdir. Landshaft spacer, *createHorizontalStrut(int width)* statik usuli va verticalni *createVerticalStrut(int height)* usuli yordamida yaratiladi. Ikkala usul ham Box sinfida aniqlanadi va ularning har biridagi tamsayı parametri spacer o'lchamini aniqlaydi.

Bundan tashqari, ushbu paneldagi yana bir maxsus element - "bahor" turini qo'shishingiz mumkin. Panelning kattaligi barcha elementlarning maqbul joylashishi uchun zarur bo'lganidan ortiq bo'lsa, qisish qobiliyatiga ega bo'lganlar qo'shimcha joyni o'zlari bilan to'ldirishga harakat qiladi. Elementlar orasida bir

yoki bir necha "buloq" joylashtirsangiz, elementlar orasidagi intervalda qo'shimcha bo'sh joy taqsimlanadi. Landshaft va vertikal yaylar *createHorizontalGlue()* va *createVerticalGlue()* usullari yordamida yaratilgan.

Ushbu menejer ishining xususiyatlarini tushunish uchun aniq misolda yaxshiroqdir. Biz to'rtta tugmachani vertikal ravishda joylashtirib, ikkita markaziy "buloq" va qolgan 10 pixel orasida joylashamiz.

```
SimpleWindow(){
super("Пробное окно");
setDefaultCloseOperation(EXIT_ON_CLOSE);
Box box = Box.createVerticalBox();
box.add(new JButton("Tugma"));
box.add(Box.createVerticalStrut(10));
box.add(new JButton("+"));
box.add(Box.createVerticalGlue());
box.add(new JButton("-"));
box.add(Box.createVerticalStrut(10));
box.add(new JButton("Uzun yozuvli tugma"));
setContentPane(box);
setSize(250, 100);
}
```

Назорат саволлари

1. Swing kutubxonasiga kirishni tasvirlang.

2. JFrame oynasi nima?

3. Bo’sh oyna haqida nima bilasiz?

4. Konteyner sinfi (konteyner) nima?

5. JPanel (panel) qanday sinf?.

6. FlowLayout davriy oqim boshqaruvchisi nima vazifa bajaradi?

7. BorderLayout Chegar administratori funksiyalari.

8. Jadvalni joylashtirish menejeri GridLayout.

9. BoxLayout Blok qutisi menejeri va Box sinfi.

4.3. Java dasturlash tilida tarmoq komponentlarini boshqarish

Java texnologiyasi yordamida havfsiz, platformalar aro va ko‘chiriladigan kodlar yaratish mumkin. Biroq Java tarmoqni daturlash uchun ajoyib til

hisobdanadi. Bu texnologiya ishlab chiqaruvchilar tarmoq resurslariga osongina murojaat qiladigan, ishlatishga oson bo‘lgan vosita hisoblanadi. Shuning uchun informatsion va telekommunikatsion texnologiyalari sohasidagi yosh mutaxassislar, OTM muassasalari bitiruvchilari Java ni tarmoq ilovalarini yaratishga qo‘llashni o‘rganishlari va amaliyotda ishlata bilishlari muhim ahamiyatga ega.

Tarmoq bilan ishlash asoslari. Java ning tarmoqni qo‘llab quvvatlash asosini soket (socket) kontsepsiyasi tashkil etadi. Soket tarmoqning nihoyaviy nuqtasini identifikatsiyalaydi. Soketlar zamonaviy tarmoqlar asosidir, chunki soket alohida kompyuterga bir vaqtning o‘zida ko‘plab turli mijozlarga xizmat ko‘rsatishi va ko‘plab turli tuman axborotlarni qayta ishlashi mumkin. Bu alohida mashinada raqamlangan soket portini (port) ishlatish orqali amalga oshiriladi. Xozirgi vaqtda, server jarayoni portni mijoz ulanguncha tinmay “eshitadi” deb hisoblashadi. Server har bir seansi unikal hisoblanadigan portga ulangan ko‘plab mijozlarni qabul qilish imkoniyatiga ega. Ko‘plab mijozlar ulanishlariga xizmat ko‘rsatish uchun server jarayoni ko‘poqimli bo‘lishi yoki bir vaqtning uzida kiritish chiqarishni qayta ishlaydigan boshqa vositalarga ega bo‘lishi kerak bo‘ladi. Soketli kommunikatsiya IP (Internet Protocol IP) protokoli bo‘yicha amalga oshiriladi. Bu ma’lumotlarni uncha katta bo‘lmagan paketlarga ajratuvchi va ularni tarmoq orqali ma’lum manzilga yuboruvchi quyi darajadagi marshrutlovchi protokol. U ushbu manzilga barcha paketlarni etkazib berishni kafolatlamaydi. Uzatishni boshqarish protokoli (Transmission Control Protocol TCP) yuqoriroq darajadagi protokol hisoblanib, ma’lumotlarni ishonchli etkazib berish uchun zarur bo‘lgan paketlarni ishonchli yig‘ish, saralash va takroriy uzatishni ta’minlaydi. Uchinchi protokol foydalanuvchilar deytgrami protokoli (User Datagram Protocol UDP), bevosita TCP dan keyin turib, doimiy ulanishni talab etmaydigan va paketlarni ishonchsiz transportirovkasini tezkor qo‘llab quvvatlash uchun ishlatilishi mumkin. Ulanish amalga oshirilgan zaxotiyoq, ishlatilayotgan portga bog‘liq bo‘lgan yuqori darajadagi protokol ishlatiladi. TCP/IP dastlabki 1024 portni maxsus protokollar uchun rezervlashtiradi. 21 port FTP uchun, 23 port Telnet uchun, 25 port elektron pochta uchun, 80 port HTTP uchun, 119 port netnews uchun. Masalan, HTTP bu Web-brauzerlar va serverlar tomonidan gipermatnlar va grafik tasvirlarni uzatish uchun ishlatiladigan protokol. U quyidagicha ishlaydi. Mijoz HTTP serverdan faylni so‘raganda, bu harakat tushgan (hit) nomi bilan ma’lum va ma’lum formatda fayl nomini oldindan ma’lum bo‘lgan portga keyinchalik fayl ichini o‘qib ko‘rish orqali oddiy uzatishdan iborat. Server mijozga so‘rov qayta ishlangan yoki ishlanmaganligi haqida va qaysi sabab bilan amalga oshirilganganligi holat kodini habarlaydi.

Internetning kalit komponenti manzil hisoblanadi. Internetdagi har bir kompyuter xususiy manzilga ega. Internet manzil internetdagi har bir kompyuterni unikal ravishda identifikatsiya qiladigan 32-bit qiymatli sondir. Barcha Internet-manzillar 32-bit qiymatli, to‘rtta 8-bitli qiymatlardan tashkil topgan. Shunday turdagi manzillar IPv4 (Internet-protokol 4 versiya) da aniqdlangan. Biroq keyingi paytlarda manzillashning yangi sxemasi ko‘proq ishlatilmoqda. U IPv6 deb ataladi va IPv4 ga qaraganda ko‘proq manzillar fazosini qo‘llab quvvatlaydi. IPv4 bilan teskari aloqani ta’minlash uchun IPv6 ning kichik 32 bit manzillari IPv4 ning korrektmanzilini uz ichiga olishi mumkin. Shunday qilib, IPv4 manzillash pastdan yuqoriga Ipv6 bilan mos tushadi. Java IPv4 yoki IPv6 ni ishlatish bo‘yicha barcha masalalarni avtomatik tarzda xal qiladi. Xuddi shuningdek, IP-manzillash tarmoq ierarxiyasini, domen nomi deb ataladigan internetni, oxirgi paytda nomlar fazosida mashina qaerda turganligini bayon qiladi. Masalan, ru (Rossiya Federatsiyasi uchun rezervlashtirilgan) domenga tegishli, ulstu (korxona nomiga ko‘ra) nomiga ega, www esa Web-so‘rovlarni qayta ishlovchi serverni identifikatsiya qiladi. Internet domen nomi IP-manzilda domen nomlari xizmati (Domain Name Service DNS) yordamida aks etadi. Bu Internet IP-manzillar bilan ishlayotgan bir paytda foydalanuvchiga domen nomlari bilan ishlash imkoniyatini beradi. Java TCP/IP ni mavjud kiritish chiqarish interfeyslarini kengaytirish orqali, shuningdek tarmoqda kiritish chiqarish ob’ektlarini yaratish uchun zarur bo‘lgan vositalarni qo‘shish yordamida qo‘llab quvvatlaydi. Java ham TCP ham UDP protokollari oilasini qo‘llab quvvatlaydi. TCP tarmoq bo‘yicha ishonchli kiritish chiqarish oqimi uchun qo‘llaniladi. UDP nuqtadan nuqtaga deytgrammlarni uzatishning nisbatan sodda, keyin esa tezkor modelini qo‘llab quvvatlaydi (InetAddress sinfi). InetAddress sinfi raqamli IP-manzilni, shuningdek ushbu manzil uchun domen nomini inkapsulyatsiya qilish uchun ishlatiladi. Muloqat IP-xost nomini ishlatgan sinf bilan amalga oshiriladi. InetAddress sinfi IPv4 manzillar bilan ham, shuningdek IPv6 manzillari bilan ham ishlay oladi. InetAddress sinfi konstruktorlarga ega emas. InetAddress ob’ektini yaratish uchun biror mumkin bo‘lgan fabrik-metodlarini ishlatish kerak bo‘ladi. Fabrik-metod (factory method) bu sinfning statik metodlari ushbu sinf ekzempilyarini faytaradigan oddiy kelishuv. Bu turli parametrlar ro‘yhatlariga ega konstruktorlarni oshiqcha yuklanishi o‘rniga bajariladi. Bunda metodlarning unikal nomlari natijalarni aniqroq ko‘rsatadi. Quyida eng ko‘p ishlatiladigan uchta fabrik-metodlar ko‘rsatilgan InetAddress: static InetAddress getlocalhost(), static InetAddress getbyname(string hostname) va static InetAddress[] getallbyname(string hostname) . getlocalhost() metodi lokal xostni tasvirlaydigan InetAddress ob’ektini qaytaradi. getbyname() metodi unga nomi yuborilgan InetAddress xostni qaytaradi. Agar ushbu metodlar xost nomini olish holatida bo‘lmasalar, u holda UnknownHostException istisnoli holat yuzaga

keladi. Fabrik-metod getallbyname() nom almashtiriladigan barcha manzillarni tasvirlovchi InetAddress massivini qaytaradi. Ular ham hech bir manzilni uzgartira olmagan holida UnknownHostException istisnoli holatini yuzaga keltiradi. InetAddress shuningdek fabrik metod getbyaddress() ga ham ega. U IP-manzilni qabul qilib, InetAddress ob'ektini qaytaradi. Bunda ham IPv4 yoki IPv6 ham ishlatilishi mumkin. Quyidagi misolda lokal mashina manzillari va nomlari, shuningdek ikkita mashxur saytlar nomlari bosmaga chiqariladi.

// InetAddress ni qullashga misol .

```
package iad;
import java.net.*;
class iad {
public static void main(string args[])
  throws UnknownHostException {
  InetAddress Address = InetAddress.getLocalHost();
  System.out.println(Address);
  Address = InetAddress.getByName("ulstu.ru");
   System.out.println(Address);
   InetAddress SW[] = InetAddress.getAllByName("www.microsoft.com");
   for (int i=0; i<sw.length; i++)
      System.out.println(SW[i]);
```

Quyida ushbu dastur tomonidan chiqarilgan nomlar keltirilgan.

home/ ulstu.ru

Назорат саволлари

1. Tarmoq bilan ishlash asoslarini nima tashkil etadi?
2. Soketli kommunikatsiya nima?
3. TCP/IP protokollari oilasihaqida.
4. UDP protokoli qanday protocol?

4.4. Java dasturlash tilida ma’lumotlar bazasi bilan ishlash

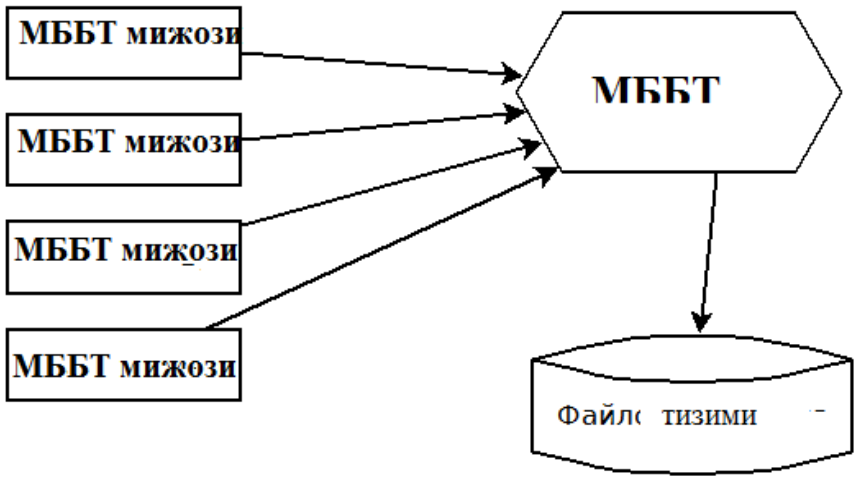
Ma’lumotlar bazasi serveri

Ma’lumotlar bazasi (MB) serveri va ma’lumotlar bazasini boshqarish tizimi (MBBT) alohida mutlaqo aniq sabablarga ko‘ra paydo bo‘ldi. MB ko‘pchilik foydalanadigan bo‘ldi. Ma’lumotlar barchaga kerak va ularga bir vaqtning o‘zida kirish isbot talab qilinmaydigan hol. Yagona fayl ko‘rinishida MB muammosi shundan iboratki, ushbu faylga ko‘plab dasturlar murojaat qiladi, ularning har biri ma’lumotlarni uzgartirishni yoki ma’lumot olishni xoxlaydi. Fayl darajasida bunday kirishni tashkil etish bajarilmaydigan ish hisoblanadi. Birinchidan fayl barcha foydalanuvchilarga kirishli bo‘lishi, bu ma’lumotlarni tarmoq bo‘yicha siljitish va ushbu faylni biror tarmoq diskida saqlashni talab qiladi. Tarmoqda uzatiladigan ma’lumotlarning ko‘payishi yaxshi holat emas.

Ikkichidan — faylga bir necha dasturlarning bir vaqtning o‘zida yozishiga urinish inqirozli holat. Shunday kirishni oddiy fayl tizimida tashkil etish xal qilib bo‘lmaydigan masala. Uchinchidan — biror bir ma’lumotlarga kirish xuquqini tashkil etish ham murakkab masala. To‘rtinchidan — biror ma’lumotga bir vaqtning o‘zida kirish natijasida yuzaga keladigan kelishmovchiliklarni boshqarish kerak bo‘ladi. Oddiy kichik tahlil yordamida ko‘pfoydalanuvchilik tartibida yana ko‘plab muammolarni ko‘rish mumkin. Natijada maxsus dastur yozishga qaror qilingan. Bu dastur ma’lumotlar bazasini boshqarish tizimi, ma’lumotlar bazasi serveri va h.k. nomlar bilan ataladi. Bu dasturning mohiyati va maqsadi — ma’lumotlarga markazlashtirilgan kirishni tashkil etishdir. Ya’ni mijoz ilovalaridan ma’lumot olish yoki ma’lumot o‘zgartirishga bo‘lgan so‘rov aynan ushbu dasturga jo‘natiladi. Ushbu dastur yuqoridagi barchcha muamolar bilan shug‘ullanadi:

1. MBBT buyruqlar majmuiga ega bo‘lib, ular ma’lumotlarni olish yoki yozishga yordam beradi.
2. MBBT fayl tizimi bilan o‘zi ishlaydi.
3. MBBT turli ma’lumotlarga kirishni cheklash mexanizmiga ega.
4. MBBT ma’lumotlarga bir vaqtning o‘zida kirish muammosini xal qiladi.

Natijada etarli darajada aniq bo‘lgan arxitekturaga ega bo‘lamiz. MBBT ma’lumotlar bilan ishlashga diqqatini jalb qiladi va MBBTga ma’lumotlar junatib turuvchi mijozlar mavjud.



MBBT mijozlar bilan ishlaganda aniq masalalarni echishi zarur:

1. Mijoz MBBT bilan ulanishi kerak. Bunda ko‘pincha tarmoq protokoli TCP/IP ishlatiladi. Mijoz ulanish paytida MBBT uni identifikatsiya qila olishi va keyinchalik unga ma’lumotlar ustida u yoki bu ishlarni amalga oshira olishga ruhsat berishi uchun uz login va parolini uzatadi.

2. Mijoz MBBT dan ma’lumot olishi yoki uzgartirishi uchun buyruqlar junatishi mumkin.

3. MBBT ma’lumotlar ma’lum tuzilmalarko‘rinishida saqlanadi va bu tuzilmalarga buyruqlar orqali murojaat qilishi mumkin.

SQL ma’lumotlar bazasi

SQL-ma’lumotlar bazasida ma’lumotlarni saqlash tuzilmasi sifatida jadvallar ishlatiladi. Jadvallarni bir biri bilan bog‘lashni munosabatlar shaklida amalga oshirib, etarli darajada murakkablikda tashkil etilgan ma’lumotlarni saqlash mumkin. Bunday ma’lumotlar bilan ishlash uchun maxus til — SQL (Structured Query Language — so‘rovlarning tuzilmalashtirilgan tili) ishlab chiqilgan. Bugungi kunda SQL-ma’lumotlar bazasi eng keng tarqalgan ma’lumotlar bazasi hisoblanadi. Bunda Java dan turib SQL ma’lumotlar bazasi bilan ishlash muhim ahamiyat kasb etadi. Java bilan ma’lumotlar bazasini uyg‘unlashtirishdan asosiy maqsad Internet uchun ilovalar yaratishdan iboratdir. Internet uchun murakkab va ishonchli dasturlarga ehtiyoj oshgan sari ma’lumotlar bazasiga ham ehtiyoj oshib boradi. Server ma’lumotlar bazasi Internetda ko‘p funksiyalarni ko‘llashi mumkin. Har kanday veb- saxifa ma’lumotlar bazasi tomonidan boshqarilishi mumkin.

Misol tarikasida o‘z katalogini WWW da e’lon qilmoqchi bo‘lgan va Internet orkali buyurtmalar kabul qilmoqchi bo‘lgan katalog bo‘yicha sotuvchini ko‘raylik.

Agar katalogni HTML-fayllar shaklida e’lon kilinsa Yangi tovar ko‘shilganda yoki narx o‘zgarganda kimdir katalogni taxrirlashi lozim bo‘ladi. Agar buning o‘rniga katalog ma’lumotlarini relyatsion ma’lumotlar bazasida saqlansa katalogdagi o‘zgarishlarni ma’lumotlar bazasidagi tovar yoki narx xakidagi ma’lumotlarni o‘zgartirish real vakt masshtabida e’lon kilish imkoniyati tug‘iladi.

Bundan tashqari katalogni mavjud buyurtmalarni qayta ishlash elektron tizimlari bilan integratsiya qilish imkoniyati tug‘iladi. Shunday kilib bunday veb-saytni boshkarish uchun ma’lumotlar bazasidan foydalanish sotuvchiga ham oluvchiga ham kulayliklar tug‘diradi.

Shu tarzda veb-saxifa ma’lumotlar bazasi bilan bog‘lanadi. Ma’lumotlar bazasi sizni veb-serveringizda yoki sizni serveringiz ma’lumot almashishi mumkin bo‘lgan boshka mashinada joylashgan bo‘lishi mumkin. (yaxshi MBBT bunday vazifalarni taksimlashni oson tashkil kila oladi). Siz o‘zingizning veb-saxifangizga forma joylashtirasiz va foydalanuvchi uzatish kerak bo‘lgan so‘rov yoki ma’lumotni shu formaga kiritadi. Formani serverga uzatgandan so‘ng, server siz yozgan dasturni ishga tushiradi va bu dastur foydalanuvchi uzatgan ma’lumotlarni ajratib oladi. Bu dasturlar ko‘pincha JSP-stsenariylar yoki Java da server dasturlari shaklida yaratiladi, lekin dasturni HTML-sahifaga to‘g‘ridan to‘g‘ri joylashtirish ham mumkin.

Endi sizni dasturingiz foydalanuvchiga qanday ma’lumotlar kerak va u ma’lumotlar bazasiga nima kiritmokchiligini biladi. Dastur ma’lumotlarni tanlash yoki o‘zgartirish uchun SQL buyruq yaratadi, ma’lumotlar bazasi bo‘lsa qolganini bajaradi. Ma’lumotlar bazasidan olingan natijalarni sizni dasturingiz yangi HTML - saxifa shakliga keltirib kaytadan foydalanuvchiga yuboradi.

Ma’lumotlar bazasidagi ma’lumotlar turlari

Ma’lumotlar bazasidagi eng ko‘p ishlatiladigan ma’lumotlar turlari quyidagilar bo‘lishi mumkin. Masalan, MuSQL shularni ishlatadi:

INTEGER - butun son (odatda 10 tagacha qiymatli rakam va ishora).

SMALLINT- "qiska butun" (odatda 5 tagacha qiymatli rakam va ishora).

DECIMAL (p,q) - o‘nli son, P rakam va ishoradan iborat (0 < P < 16). O‘nli nuqtadan so‘ng raqamlar soni k orqali beriladi (Q < P, agar k = 0 bo‘lsa, tashlab yuborilishi mumkin).

FLOAT - xaqiqiy son 15 ta qiymatli raqam va butun darajadan iborat. Daraja MBBT tipi bilan aniqlanadi (masalan, 75 yoki 307).

CHAR(n) - uzunligi o‘zgarmas, n ga teng bo‘lgan simvolli qator (0 < n < 256).

VARCHAR(n) - uzunligi o‘zgaruvchi, n simvoldan oshmagan simvolli qator (N > 0 va har xil MBBT larda har xil lekin 4096 dan kam emas).

DATE - maxsus buyruq orkali aniqlanuvchi formatdagi sana (SyBase da ko‘zda tutilgan bo‘yicha yy/mm/dd); sana maydonlari bizning eramizdan oldin bir necha ming yilliklardan boshlanuvchi va bizning eramiz beshinchi- o‘ninchi ming yilligi bilan cheklangan xakikiy sanalarni o‘z ichiga olishi mumkin.

TIME-maxsus buyruq orkali aniqlanuvchi formatdagi vakt (ko‘zda tutilgan bo‘yicha hh.mm.ss).

DATETIME - sana va vakt kombinatsiyasi. (Sybase da TIMESTAMP).

MONEY -maxsus buyruq orqali aniqlanuvchi formatdagi pul. Format o‘z ichiga pul birligi simvoli ($, rub, ...) va uning joylashuvi (suffiks yoki prefiks), kasr qism aniqligi va pul qiymatini ko‘rsatish shartlarini oladi.

Jadvallar bilan ishlash

Jadvallarni yaratish.

Jadvallar CREATE TABLE buyruqsi bilan yaratiladi. Bu buyruq qatorlarsiz bo‘sh jadval yaratadi. CREATE TABLE buyruqsi jadval nomini va jadval o‘zini ma’lum tartibda ko‘rsatilgan ustunlar nomlari ketma – ketligi ta’rifi k o‘rinishida aniqlaydi. U ma’lumotlar tiplari va ustunlar o‘lchovini aniqlaydi. HAR bir jadval juda bo‘lmaganda bitta ustunga ega bo‘lishi kerak.

CREATE TABLE buyruqsi sintaksisi:

CREATE TABLE <table-name >

(<column name> <data type>[(<size>)],

<column name> <data type>[(<size>)], ...);

Argument qiymati kattaligi ma’lumot turiga bog‘liqdir. Agar siz maxsus ko‘rsatmasangiz, tizim avtomatik qiymatni o‘rnatadi.

Bundan buyon quyida keltirilgan 3 ta jadvaldan iborat ma’lumotlar bazasini ko‘ramiz.

Sotuvchilar jadvali (Salepeople):

SNum	SName	City	Comm
11	Peel	London	0.12
12	Serres	San Jose	0.13
14	Motika	London	0.11

SNum – HAR bir sotuvchi unikal raqami,

SName – sotuvchi nomi,

SNAdress – sotuvchi adresi (shahar),

Comm – sotuvchilarning o‘nli shakldagi komission foydasi.

Buyurtmachilar jadvali (Customers):

SNum	SName	City	Rating	SNumSale
21	Hoffman	London	100	11
22	Giovanni	Rome	200	13
23	Liu	SanJose	200	12

SNum – HAR bir buyurtmachi unikal nomeri,

SName – buyurtmachi nomi,

City – buyurtmachi adresi (shaHAR),

Rating – buyurtmachining boshqalardan ustunlik darajasini ko‘rsatuvchi kod (reyting),

SNumSale – shu buyurtmachiga tayinlangan sotuvchi nomeri.

Buyurtmalar jadvali (Orders):

SNum	SSum	SDate	SKlNum	SNumSale
38	4723.00	1990/10/05	26	11
310	1309.95	1990/10/06	24	12

SNum – HAR bir sotib olishning unikal nomeri,

SSum– sotib olish summasi qiymati,

SDate – sotib olish sanasi,

SKlNum – sotib oluvchi buyurtmachi nomeri,

SNumSale – sotuvchining nomeri.

Misol uchun sotuvchilar jadvalini yaratishni ko‘rib chikamiz:

```
CREATE TABLE Salepeople
( SNum integer,
SName char (10),
City char (10),
Comm decimal( );
```

Jadvallarni o‘chirish.

Jadvalni o‘chirish imkoniga ega bo‘lish uchun, jadval egasi (Ya’ni yaratuvchisi) bo‘lishingiz kerak. Fakat bo‘sh jadvalni o‘chirish mumkin. satrlarga ega bo‘lgan, to‘ldirilgan jadvalni o‘chirish mumkin emas, Ya’ni jadval o‘chirishdan oldin tozalangan bo‘lishi kerak. Jadvalni o‘chirish buyrug‘i quyidagi ko‘rinishga ega:

DROP TABLE < < table name >;

Masalan: *DROP TABLE Salepeople;*

Jadvalni yaratilgandan so‘ng o‘zgartirish.

Jadvalni o‘zgartirish uchun ALTER TABLE buyrug‘idan foydalaniladi. Bu buyruq jadvalga yangi ustunlar qo‘shish, ustunlarni o‘chirish, ustunlar kattaligini o‘zgartirish, hamda cheklanishlarni qo‘shish va olib tashlash imkoniyatlariga ega.

Bu buyruq ANSI standarti qismi emas, shuning uchun har xil tizimlarda har xil imkoniyatlarga ega.

Jadvalga ustun qo‘shish uchun buyruqning namunaviy sintaksisi:

ALTER TABLE <table_name> ADD <column name>

<data type> <size>;

Masalan:

ALTER TABLE Salepeople ADD Pkone Char(7);1));

Nazorat savollari

1. Ma’lumotlar bazasi serveri nima?

2. SQL ma’lumotlar bazassi qanday baza?

3. Ma’lumotlar bazasida ma’lumotlar turlari.

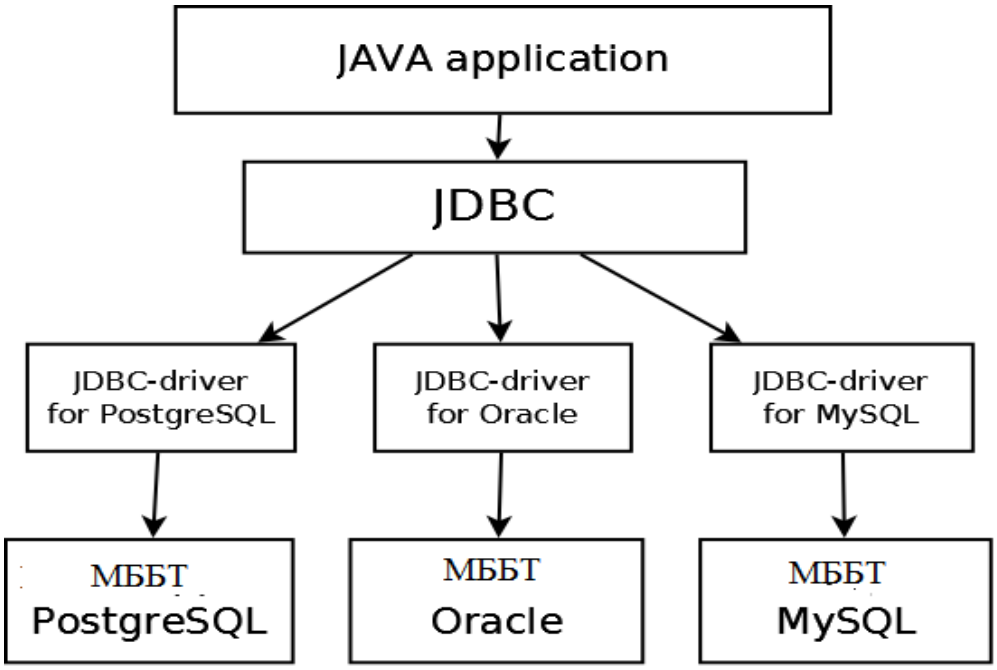
4. Relyatsion jadvallar bilan ishlash.

4.5. Java dasturlash tilida JDBC komponentlari bilan ishlash

JDBC — Java Database Connectivity — arxitekturasi

Agar JDBC ni oddiy so‘zlar yordamida aniqlamoqchi bo‘lsak, u holda JDBC Java dan turib ma’lumotlar bazasi bilan ishlash imkoniyatini beruvchi interfeyslar va ba’zi bir sinflar bayonidir. Yanada soddaroq qilib aytsak: JDBC — ma’lumotlar bazasi bilan ishlash imkoniyatini beruvchi interfeyslar (va sinflar) majmuasidir. Ushbu arxitekturaning asosiy tamoyili turli ko‘rinishdagi ma’lumotlar bazalari bilan yagona unifitsiallashtirilgan (universal, standart) uslubda muloqotni ta’minlashdan iboratdir. Ya’ni, Java ilova nuqtai nazaridan Oracle yoki PostgreSQL bilan muloqot uslubi bir biridan deyarli farq qilmasligi kerak. Ammo, SQL-so‘rovlar sanalar, satrlar va boshqa ob’ektlar funksiyalarining turli tumanligiga ko‘ra bir biridan farq qilishi mumkin. Bular Java kodlardagi satrlardir. Ammo, SQL-serverga yuboriladigan so‘rovlar, SQL-serverdan olinadigan ma’lumotlar algoritmlari va buyruqlar majmualari o‘zgarmasligi kerak. Java ilova qanday server bilan ishlayotganligi to‘g‘risida o‘ylamasligi, barcha turdagi ma’lumotlar bazasi serverlari bilan ishlash bir xil tartibda amalga oshishi kerak. Shunga qaramasdan har bir MBBT ichki ma’lumotlar uzatish mexanizmlari turlicha. Masalan, Oracle da baytlarni uzatish qoidasi MySQL va PostgreSQL

lardan farq qiladi. Natijada — bir tomondan barcha muloqotlar bir xil ko‘rinishda, ammo boshqa tomondan joriy qilish turlicha bo‘ladi. Ya’ni — turlicha joriy qilish, lekin bir xil funktsional majmua. Bu Java da ob’ektga yo‘naltirilgan dasturlashning oddiy polimorfizm tamoyilini interfeyslar orqali joriy qilishdir. Aynan mana shu uslub asosida JDBC arxitekturasi yaratiladi. Quyidagi rasm ushbu jarayonni kurgazmali aks ettiradi.



Ma’lumotlar bazasiga murojaat qiluvchi servletga misol

Quyida Java-servletni dasturlash bo‘yicha misol keltirilgan. Bu misolda brouzerda SCOTT sxemasi bo‘yicha xodimlar ro’yxati chiqariladi.

```
import java.io.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class StaffByServletTransactional extends HttpServlet
{
public void init(ServletConfig config) throws
ServletException
{
super.init(config);
try {
Class.forName("oracle.jdbc.driver.OracleDriver").new
Instance();
}
catch (Exception e) { }
```

```
}
public void doGet(
HttpServletRequest request, HttpServletResponse response)
throws IOException, ServletException {
response.setContentType("text/html");
PrintWriter out = response.getWriter();
out.println("<html>");
out.println("<head>");
out.println("<title>Servlet Per Transaction
Connection</title>");
out.println("</head>");
out.println("<body>");
out.println("<pre>");
Connection cn = null;
try {
cn = DriverManager.getConnection(
"jdbc:oracle:thin:@localhost:1521:teacher", "scott", "tiger");
}
catch (Exception e) { }
Statement st = null;
ResultSet rs = null;
try {
st = cn.createStatement();
rs = st.executeQuery("SELECT empno, ename FROM emp");
while (rs.next()) {
out.println("Number=" + rs.getString(1) + " " +
"Name=" + rs.getString(2));
}
st.close();
cn.close();
}
catch (Exception e) { }
out.println("</pre>");
out.println("<hr>");
out.println("</body>");
out.println("</html>");
}
public void doPost(
HttpServletRequest request, HttpServletResponse response)
throws IOException, ServletException {
doGet(request, response);
}
}
```

Servlet translyatsiyasi:

SET CLASSPATH=%ORACLE_HOME%\lib\servlet.jar;.

javac StaffByServletTransactional.java

Servletga murojaat qilishga misol

Servletga murojaatni bajarish uchun servlet translyatsiyasi natijasi bo‘lgan MyServletAgent.class faylni %JSERV_HOME%\servlets katalogiga o‘tkazish kerak. Bundan keyin brouzer uchun berilayotgan xujjat satrida http://localhost/servlet/StaffByServletTransactional (9 versiyada -- http://localhost:7778/servlet/StaffByServletTransactional) ko‘rsatish etarli. Ushbu servletga birinchi murojaatda servletni initsiallashtiradi va web-server xotirasiga joylashtiradi. Ushbu amaldan keyin servletga murojaat ancha tez bajariladi.

JavaServer Pages bilan ishlash

JavaServer Pages (JSP) – dinamik web-sahifalarni tashkil qilishning boshqacha texnikasi. Java Servlets lardan farqli ravishda, JSP larda dasturdan HTML-kodlarni generatsiya qilish emas, HTML-kodlarni bevosita JSP-ko‘rsatmalar ichiga joylashtirish amalga oshiriladi. JSP-ko‘rsatmalar ham serverda bajariladi. JSP- sahifaga birinchi marotaba murojaat qilinayotganda JServ uni avtomatik tarzda Java-servlet ga aylantiradi va keyinchalik barcha so‘rovlarni unga yo‘naltiradi.

(JavaServer Pages texnikasi to‘g‘risidagi to‘liq ma’lumotlarni http://java.sun.com/products/jsp/) saytidan olish mumkin.

JSP sahifani yaratishga misol

JSP-sahifa ishini ko‘rsatish uchun quyidagi faylni (MyJSPDemo.jsp) hosil qilamiz:

<html>

<head><title>MyJSPDemo JSP demo</title></head>

<body>

< h2>Hello, JSP World</h2>

whith date <%= new java.util.Date().toString() %> now

and User-Agent <%= request.getHeader("User-Agent") %>

< /body>

</html>

JSP sahifalarga murojaat qilishga misol

Hosil bo‘lgan faylni %APACHE_HOME%\htdocs\demo katalogiga o‘tkazamiz. Brouzerdan http://localhost/demo/MyJSPDemo.jsp (9 versiyada http://localhost:7778/demo/MyJSPDemo.jsp) adres bo‘yicha murojaat qilamiz.

Quyidagiga e’tibor berish talab etiladi, yuqoridagi katalogda _pages_demo ostkatalogi paydo bo‘ldi va u quyidagi fayllarni o‘z ichiga oladi:

_MyJSPDemo$__jsp_StaticText.class

_MyJSPDemo.class

_MyJSPDemo.java

JSP sahifasidan ma’lumotlar bazasiga murojaat qilish

JSP holida sahifaning ma’lumotlar bazasiga murojaat qilishining yagona usuli yo‘q. Buning o‘rniga bir qancha mumkin bo‘lgan variantlar mavjud:

- bevosita murojaat;
- sahifada user actions (foydalanuvchi harakati) ni ishlatgan holda murojaat;
- MB ga Bean-komponentlardan murojaat;
- MB ga servletni chaqirish orqali murojaat.

Quyida yuqoridagi variantlardan ikkitasi qaraladi.

Ma’lumotlar bazasiga bevosita murojaat qiluvchi JSP sahifasiga misol

Quyida JSP-sahifani dasturlovchi fayl StaffByJSPDirect.jsp misol sifatida keltirilgan. Ushbu misol bazaga bevosita murojaat orqali SCOTT sxemasi xodimlari ro‘yhatini chiqaradi:

```
<%@ page import="java.sql.*" %>
<html>
< head><title>Direct JDBC Query JSP</title></head>
< body>
< h3>JSP StaffByJSPDirect result:</h3>
< pre>
```

```
< %= StaffByJSPDirect() %>
< /pre>
< hr>
< /body>
< /html>
<%!
private String StaffByJSPDirect() throws SQLException {
Connection cn = null;
StringBuffer sb = new StringBuffer();
try {
cn = DriverManager.getConnection(
"jdbc:oracle:thin:@localhost:1521:teacher", "scott", "tiger");
}
catch (SQLException e) { };
Statement st = null;
ResultSet rs = null;
try {
st = cn.createStatement();
rs = st.executeQuery("SELECT empno, ename FROM emp");
while (rs.next()) {
sb.append("Number=" + rs.getString(1) + " " +
"Name=" + rs.getString(2) + "\n");
};
cn.close();
}
catch (SQLException e) { };
return sb.toString();
}
%>
```

JSP xususiy teglari orqali ma’lumotlar bazasiga murojaat qilish

Quyida JSP-sahifani dasturlovchi fayl StaffByJSPTagLib.jsp misol sifatida keltirilgan. Bu sahifa brouzerga SCOTT sxemasi xodimlari ro‘yhatini Oracle shtat versiyasi kutubxonasidagi xususiy teglarni (custom tags) ishlatgan holda chiqaradi (bu kutubxona bayoni %APACHE_HOME%\htdocs\WEB-INF\sqltaglib.tld faylida).

```
<%@ taglib uri="/WEB-INF/sqltaglib.tld" prefix="jml"
%>
<html>
< head>
< title>User Tag Lib JDBC Query JSP</title>
< /head>
< body>
< h3>JSP StaffByJSPTaglib result:</h3>
```

```
<
jml:dbOpen URL="jdbc:oracle:thin:@localhost:1521:teacher"
user="scott" password="tiger">
<jml:dbQuery>
select * from emp
</jml:dbQuery>
< /jml:dbOpen>
< hr>
< /body>
< /html>
```

Nazorat savollari

1. JDBC arxitekturasi qanday tashkil etilgan?
2. Ma’lumotlar bazasiga murojjat qiluvchi servlet majburiy komponentlari nimalar?
3. Servletga qanday murojaat qilinadi?
4. JSP sahifalaridan ma’lumotlar bazasiga murojaat qilishning nechta usuli mavjud?
5. JSP dan bevosita ma’lumotlar bazasiga murojaat qilish mumkinmi?.
6. JSP xususiy teglari orqali ma’lumotlar bazasiga murojaat qilishda nimaga ahamiyat berish kerak?

4.6. Java dasturlash tilida mobil ilovalar yaratish

Androidda java qanday ishga tushiriladi?

Ko‘p hollarda androidda biror ilova yoki o‘yinni ishga tushirishga to‘g‘ri keladi. Bunda androidda java ni ishga tushirishga ehtiyoj to‘g‘iladi. Odatdagi uslub bilan bu ishni amalga oshirib bo‘lmaydi. Chunki Java standarti Androidda qo‘llanilmaydi. Java ilovadarni ishga tushirish uchun bir necha xil dasturlar qo‘llaniladi, biroq eng ishonchlilardan biri jBed dasturi hisoblanadi. Ushbu dastur yordamida o‘yinlar va ilovalar xatosiz va juda tez ishga tushiriladi. Ilovani ishga tushirish uchun ushbu dastur menyusida SD-karta bo‘limini tanlash kerak bo‘ladi. Dastur avtomatik tarzda xotira qurilmasi kartasini taniydi va topilnan ilovalarni yuklashni taklif etadi. Bu usulning kamchiligi – ilovani telefonga yuklash uchun root huquq talab etiladi. Yana bir ishlatish uchun sodda bo‘lgan dasturlardan biri bu ommaviylashgan Java J2ME Runner dasturidir. Mutaxassislar dasturni yuklab olishda ShK va mobil qurilmani ishonchli himoya qilish uchun antivirus dasturlaridan foydalanishni tavsiya etishadi.

Androidda java ilovalarni ishga tushirishning o‘ziga xos xususiyatlarri

Jar fayllarni yuklashda qo‘llaniladigan ba’zi bir emulyatorlar ma’lum bir kamchiliklarga ega. Bular instalyatsiya murakkabligi bilan, hamda kirishga root-huquq olish qiyinchiliklari bilan bog‘liq. Android operatsion tizimida ishlaydigan mobil telefonlar egalarining ko‘pchiligi Javani androidda ishga tushirish bilan bog‘liq muammolarga duch keladi. Bu hayratlanarli hol emas, chunki ushbu operatsion tizimda berilgan dasturiy ta’minotni qo‘llovchi ichki mexanizim yo‘q. jar fayllarni androidga yuklash uchun Java Me deb ataladigan maxsus emulyatorni instalyatsiya qilish kerak bo‘ladi.

Androiddagi java ilovalarning asosiy kamchiliklari.

Android uchun ko‘plab dasturlar mavjud. Ularning har biri o‘zining ijobiy va salbiy xususiyatlariga ega. jar faylni androidga o‘rnatish uchun quyidagi amallarni bajarish talab etiladi: 1. Birlamchi ochiq kodga ega bo‘lgan dasturni halqaro tarmoqdan osongina topib, tekinga yuklab olish mumkin. Talab qilingan ilovani fayl menedjer tanishi uchun utilitalarni o‘rnatish zarur bo‘ladi. Shuni esdan chiqarmaslik kerakki, emulyator mobil qurilmaning SD-kartasida bo‘lgan android uchun lozim bo‘lgan dasturni topa olmaydi. Uni qayta yuklash kerak. 2. Xujjat ishga tushirish uchun kerak bo‘ladigan dasturga aniqlik kiritilishini talab etadi. Buning uchun J2ME emulyatori qo‘llaniladi. Uning asosiy vazifasi ma’lum bir ilovani o‘rnatish va instalyatsiya qilishdan iborat. Ushbu soft sodda va ishlatishga qulay. 3. Emulyator kamchiligiga ilovani ishga tushirishda yuzaga keladigan .jar kengaytmaga ega bo‘lgan ko‘plab xujjatlar bilan bog‘liq bo‘lgan muammolarning yuzaga kelishidir.

Gadjetlar nima va ular qanday yaratiladi?

Gadjet (angl. Gadget) inson hayotini engillashtirish va takomillashtirish uchun ishlab chiqilgan uncha katta bo‘lmagan qurilma. Gadjetlar turli sohalarda keng tarqalagan: sportda — fitnes-trekerlar, smart-brasletlar, sport qurilmalari, shu jumladan «aqlli» kiyimlar; meditsinada: elektron plastirlar, trikoderlar, gidrokopterlar, ekzoskeletlar; ko‘ngil ochishda: smartfonlar, planshetlar, musiqali pleerlar, uyin qurilmalari, qo‘shimcha va virtual reallik uchun ko‘zoynaklar va ko‘plab boshqa narsalar.

Dasturiy ta’minotda gadjet (vidjet) — qo‘shimcha axborot beruvchi uncha katta bo‘lmagan ilovalar, masalan, ob havo prognozi yoki valyutalar kursi haqida. Gadjetlarning na’munaviy vakili sifatida bir qancha mini ilovalarni qarashimiz mumkin. Masalan, Google Gadgets va boshqalar.

Quyida android tizimi uchun gadjet sifatida yaratilgan kalьkulyator keltirilgan:

```
package futureprogrommers.calculator;

import android.os.Bundle;
import android.support.v7.app.ActionBarActivity;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
//private double valueOne = Double.NaN;
//private double valueTwo;
private static final char ADDITION = '+';
private static final char SUBTRACTION = '-';
private static final char MULTIPLICATION = '*';
private static final char DIVISION = '/';
private char CURRENT_ACTION;
private void computeCalculation() {
    if(!Double.isNaN(valueOne)) {
    valueTwo =
Double.parseDouble(binding.editText.getText().toString());
    binding.editText.setText(null);
    if (CURRENT_ACTION == ADDITION)
    valueOne = this.valueOne + valueTwo;
    else if(CURRENT_ACTION == SUBTRACTION)
    valueOne = this.valueOne - valueTwo;
    else if(CURRENT_ACTION == MULTIPLICATION)
    valueOne = this.valueOne * valueTwo;
    else if(CURRENT_ACTION == DIVISION)
    valueOne = this.valueOne / valueTwo;
    }
    else {
    try {
    valueOne =
Double.parseDouble(binding.editText.getText().toString());
    }
    catch (Exception e){}
    }
    }
```

```
        binding.buttonAdd.setOnClickListener(new
View.OnClickListener() {
@Override
public void onClick(View view) {
        computeCalculation();
        CURRENT_ACTION = ADDITION;
        binding.infoTextView.setText(decimalFormat.format(valueOne)
+ "+");
        binding.editText.setText(null);
        }
        });

        binding.buttonSubtract.setOnClickListener(new
View.OnClickListener() {
@Override
public void onClick(View view) {
        computeCalculation();
        CURRENT_ACTION = SUBTRACTION;
        binding.infoTextView.setText(decimalFormat.format(valueOne)
+ "-");
        binding.editText.setText(null);
        }
        });

        binding.buttonMultiply.setOnClickListener(new
View.OnClickListener() {
@Override
public void onClick(View view) {
        computeCalculation();
        CURRENT_ACTION = MULTIPLICATION;
        binding.infoTextView.setText(decimalFormat.format(valueOne)
+ "*");
        binding.editText.setText(null);
        }
        });

        binding.buttonDivide.setOnClickListener(new
View.OnClickListener() {
@Override
```

```
public void onClick(View view) {
        computeCalculation();
        CURRENT_ACTION = DIVISION;
        binding.infoTextView.setText(decimalFormat.format(valueOne)
+ "/");
        binding.editText.setText(null);
        }
        });

        binding.buttonEqual.setOnClickListener(new
View.OnClickListener() {
@Override
public void onClick(View view) {
        computeCalculation();

binding.infoTextView.setText(binding.infoTextView.getText().toString(
) +
        decimalFormat.format(valueTwo) + " = " +
decimalFormat.format(valueOne));
        valueOne = Double.NaN;
        CURRENT_ACTION = '0';
        }
        if (onClick == 0){
        }
        });
```

Nazorat savollari

1. Androidda java qanday ishga tushiriladi?
2. Androidda java ilovalarni ishga tushirishning o‘ziga xos xususiyatlarri.
3. Androiddagi java ilovalarning asosiy kamchiliklari.
4. Gadjetlar nima va ular qanday yaratiladi?

4 bob buyicha hulosalar

Java dasturlash tilining keng imkoniyatlari uni turli sohalardagi dasturlash muammolarini osonlik bilan hal qila olish qobiliyatida ekanligidadir. Javani ma’lumotlar bazasi bilan ishlashga, ma’lumotlar bazasining ishki ob’ektlarini yaratishga qullash, javada tarmoqlarni dasturlash va tarmoqlarni boshqarish, turli grafik interfyesli amaliy dasturlar yaratish, barcha rusumdagi telefonlar va planshetlar uchun mobil ilovalar yaratish imkoniyatlari uni haqiqatda innovatsion dasturlash tili ekanligini isbotlaydi. Ushbu bobda qaralgan yunalishlar javaning qullanilish sohasi buyicha chegara emas.

GLOSSARIY

100% Pure Java™

«100% Pure JavaTM» atamasi Java-platformaning maxsus xususiyatiga ega bo‘lgan Java-dastur ni bildiradi. U Java core API dan tashqari platformaga bog‘liq (native) metodlarn ishlatmaydi, hech qanday dasturiy interfeyslarga bog‘liq emas. Sertifikatsiyalash dasturi berilgan ilova yoki applaet 100% Pure Java mi yoki yo‘qligini aniqlashga xizmat qiladi.

Abstract Window Toolkit — AWT

Maxsus platforma metodlaridan foydalangan xolda joriy qilingan grafik komponentlarning standart paketi. Ushbu komponentalar barcha platformalarga tegishli bo‘lgan funktsional imkoniyatlar qism to‘plaminigina qo‘llab quvvatlaydi. Sezilarli darajada Project Swing komponentlari majmuasi tomonidan siqib chiqarilgan (Swing-komponentlarga qarang).

abstract

Java dasturlash tilining kalit suzi. U sinf aniqlanishida ishlatilib, ushbu sinf ekzempilyarlarini xosil qilish mumkin emasligini, shu bilan birgalikda boshqa sinflar tomonidan nasl olish mumkinligini bildiradi. Abstrakt sinf o‘z qism sinflarida joriy qilish mumkin bo‘lgan, lekin o‘zida joriy qilinmagan metodlarni o‘z ichiga olishi mumkin.

abstrakt sinf (abstract class)

Ushbu sinfning ekzemptlyarlarini xosil qilish mvumkin bo‘lmagan, bir yoki bir necha abstrakt metodlarni o‘z ichiga oluvchi sinf. Abstrakt sinflar shunday aniqlanganki, boshqa sinflar uning abstrakt metodlarini oriy qilish orqali kengaytirishi va aniqlashtirishi mumkin.

abstrakt metod (abstract method) Joriy qilinishi yo‘q metod.

Kirish nazorati (**контроль доступа** (access control)) Yaxlitliligini, konfedentsialligini va kirishliligini oshirish maqsadida resurslar bilan hamkorlikda munosabatda bo‘ladigan foydalanuvchilar yoki dasturlarni to‘plamini cheklashni amalga oshiruvchi texnik vositalar.

ASIN (ACID — Atomicity, Consistency, Isolation and Durability) Tranzaktsiyalar tomonidan ta’minlanadigan to‘rtta xususiyat: atomarlik, yaxlitlilik, o‘ralganlik va ishonchlilik akronimi.

aktivatsiya (activation) Ma’lumotlarni xotirada saqlovchi ikklamchi qurilmadan korporativ komponentlarni (enterprise beans) uzatish jarayoni.

Haqiqiy parametrlar ro‘yhati (**список фактических параметров** (actual parameter list)) metodni chaqirishda aniqlangan argumentlar (haqiqiy parametrlar ruyhatiga qarang).

alfa-faktor (alpha value) pikselning yorqinligi (yoki intensivligi) ni ko‘rsatuvchi qiymat.

Amaliy dastur interfeysi (**интерфейс прикладного программирования** (API — Application Programming Interface)) ob’ektlar va sinflar xossalari va xolatlariga kirish metodlarini bayon qiluvchi va foydalanuvchilarga mo‘ljallangan xususiyat.

applet (applet) odatda veb brouzerda yoki ixtiyoriy boshqa applaetlarni ko‘rish dasturlaridagi komponent.

Appletlar konteyneri (**контейнер апплетов** (applet container)) o‘z ichida appletlarni dasturlash modelini mujassamlantirgan konteyner.

Qurilma (устройства (appliances)) Java texnologiyani qullovchi printerlar, terminallar kabi tarmoq qurilmalari va Java Management API (JMAPI) ni ishlatish orqali boshqariladigan mijozlar.

Ilovalar komponovshigi (компоновщик приложения (application assembler)) ishlanmaning komponentlari va modullarini katta modullarga birlashtiruvchi inson.

Mijoz ilovasi (**клиентское приложение** (application client)) xususiy virtual Java-mashinada bajariladigan birinchi sathdagi mijoz dasturi.

Mijoz ilovasi konteyneri (konteyner klientskogo prilojeniya (application client container)) mijoz ilovasini qullab quvvatlovchi va J2EE platformada API ni integrallashgan tasvirlanishini ta’minlovchi konteyner.

Mijoz ilovasi moduli (контейнер клиентского приложения (application client module))
mijoz ilovasini bayon qiluvchi bir yoki bir necha sinflardan tashkil topgan dasturiy modul.

Dasturiy komponentlarni etkazib beruvchi (модуль клиентского приложения (Application Component Provider))
komponentlarni joriy qiluvchi metodlar, JSP-saxifalar bayoni va zaruriy deskriptorlarni taqdim etuvchi Java sinflarni etkazib beruvchi.

Ilovani dasturlash modeli поставщик программных компонент (Application Programming Model — APM))
Amaliy dasturlash modeli bo‘lib, korxona predmet sohasi bo‘yicha amaliy echimlar yaratish uchun J2EE platformasi imkoniyatlarini qanday ishlatish va birlashtirish kerakligini aniqlaydi.

argument (argument)
Metodni chaqirishda ko‘rsatilgan ma’lumot elementi. Argument konstanta, o‘zgaruvchi yoki ifoda bo‘lishi mumkin.

massiv (array)
Har bir elementning joyi butun son bilan (massiv indeksi bilan) bir qiymatli aniqlangan bir turdagi ma’lumotlar elementlari majmuasi
.
Axborot almashishning amerikancha standart kodi (Американский Стандартный Код Обмена Информацией (American standard code for information interchange — ASCII))
har bir simvolga 7 bit ajratiladigan standart. Unikodga ham qarang.

Atomar ((атомарный (atomic))
bo‘linmaydigan birlik akt sifatida bajariladigan amal.

autentifikatsiya (authentication)
bir ob’ekt boshqa ob’ektga aniqlangan identifikatsion yozuv nomidan harakat qilayotganini ko‘rsatuvchi jarayon. J2EE platformasi uchta turdagi: odatdagi (basic), forma bilan bog‘langan (form-based) va o‘zaro (mutual) autentifikatsiyalarga muxtoj.

avtorizatsiya (authorization)
kirishni boshqarishga qarang.

Avtorizatsiyani cheklash (ограничение авторизации (authorization constraint))
Web-resurslarga kirishni himoyalovchi va ruxsat berishga muljallangan rollar nomlari majmuasi.
B
bazaviy (ochiq) autentifikatsiya (basic authentication)
Web-mijozga joylashgan, autentifikatsiya mexanizmi yordamida olingan foydalanuvchi nomi va parolini Web-server tomonidan tekshirish metodi.

bean-komponent (bean)
Ko‘p marotaba foydalaniladigan dasturiy komponent. Bean-komponentlar ilova yaratish uchun birlashtirilishi mumkin.

bean-komponent bilan boshqariladigan saqlash mexanizmi (механизм сохранения, управляемый bean-компонентом (bean-managed persistence))
bean-komponent ekzempilyarlari uzgaruvchilari va resurslarning asosiy ma’muri orasida ma’lumot uzatish bean-komponent tomonidan boshqariladigan mexanizm.

bean-komponent boshqaradigan tranzaktsiya (транзакция, управляемая bean-компонентом (bean-managed transaction))
tranzaktsiya chegaralarini aniqlaydigan korporativ komponent (enterprise bean).

binar operator (binary operator)
ikki argumentga ega amal belgisi.

bit (bit)
kompyuterdagi minimal axborot birligi. 0 yoki 1 qiymatlarni qabul qilishi mumkin.

Bitli operator (побитовый оператор (bitwise operator))
operandlarga bitlar majmuasi (0 va 1) kabi ta’sir etuvchi amal belgisi. Masalan, binar mantiqiy amallar (&, |, ^), siljitishning binar amallari (<<, >>, >>>) va to‘ldirish unar amali (~).

blok (block)
JavaTM dasturlash tilida ikkita katta qavslar ichiga olingan ixtiyoriy kod. Masalan, {x = 1;}.

bul (boolean)
Faqat ikkita true («chin») va false («yolg‘on») qiymat qabo‘l qila oladigan ifodalar va uzgaruvchilarga tegishli. JavaTM dasturlash tilida boolean turi mavjud, shuningdek literal konstantalar true va false ham.

Cheklangan soha ((ограничительная область (bounding box))
ko‘rsatilgan geometrik figurani o‘z ichiga oluvchi eng kichik ulchamli to‘g‘ri to‘rt burchak. Rastrli ob’ektlar uchun barcha berilgan piksellarni o‘z ichiga oladi.

break
JavaTM dasturlash tilining kalit suzi. break operatori boshqaruvni joriy satrdan keyingi satrga uzatadi. Agar break operatoridan keyin belgi (metka) kelsa, u xolda dastur «belgilangan» operatordan boshlab bajarilishni davom ettiradi.

biznes-logika (business logic)
Ilova funktsionalligini joriy qiluvchi kod. Enterprise Java Beans modelida ushbu logika korporativ komponent (enterprise bean) metodlari yordamida joriy qilinadi.

biznes-metod (business-method)
biznes-logika yoki ilova qoidalarini joriy qiluvchi korporativ komponenta (enterprise bean) metodi.

bayt (byte)
sakkiz bitdan iborat ketma ketlik. JavaTM dasturlash tilida mos tur byte aniqlangan.

bayt-kod (bytecode)
Java-kompilyator tomonidan generatsiya qilinadigan va Java-interpretator tomonidan bajariladigan mashinaga bog‘liqsiz kod.

C

Teskari aloqa metodlari (методы обратной связи (callback methods))
Komponentni xayot tsiklidagi muxim xodisalar to‘g‘risida komponentga xabar beruvchi, konteyner tomonidan chaqiriladigan komponent metodi.

Chaqiruvchi operator (вызывающий оператор (caller))
chaqiruvchi operatori administratoriga qarang

Chaqiruvchi operator administratori (**администратор вызывающего оператора** korporativ komponent metodi (enterprise bean) chaqiradigan ob’ektni identifikatsiya qiluvchi administrator.

case
Java tilining kalit so‘zi (switch operatori). O‘lanuvchi ifoda qiymati case konstruktsiyasida ko‘rsatilgan konstanta qiymatiga teng bo‘lganda bajariladigan instruktsiyalar majmuini aniqlaydi.

Turni almashtirish (преобразование типа (casting))
bir turni boshqa turga oshkor almashtirish**.**

catch
JavaTM tilining kalit suzi . Oldindagi try blokida yuzaga kelgan istisnoli xolat yoki bajarilishdagi hatolikda amalga oshiriladigan instruktsiyalar bloki.

char
simvol turidagi uzgaruvchini e’lon qilishda ishlatiladigan JavaTM dasturlash tilining kalit suzi.

sinf (class)
Maxsus ko‘rinishdagi ob’ektni joriy qilishni aniqlovchi JavaTM dasturlash tilidagi tur. Sinf bayoni sinf ekzempilyarini, uning uzgaruvchilarini va metodlarini

aniqlaydi. Shunigdek interfeyslar va supersinflarni ham aniqlaydi. Jimlikga ko‘ra ixtiyoriy sinfning super sinfi Object sinfi hisoblanadi.

Sinf metodi (metod sinfa (tipa) (class method))
Nima bo‘lishidan qat’iy nazar ma’lum bir ob’ekt uchun chaqiriladigan metod. Sinf metodlari umuman sinfga ta’sir qiladi, uning aniq ekzempilyariga emas.
Shuningdek statik metod nomiga ham ega. Ekzempilyar metodiga qarang.

Sinflarga yo‘l (путь к классам (classpath))
JavaTM virtual mashinasiga va Java ilovalarga (masalan, JDKTM 1.1.X\bin direktoriyasida joylashgan utilitalarga) sinflar kutubxonasi, foydalanuvchi kutubxonalarini ham hisobga olgan xolda, qaerda joylashganligi haqida habar beruvchi o‘zgaruvchan muhit qatlami. O‘zgaruvchan muhit qatlami yoki boshqa usul bilan, masalan JVM optsiyalari yordamida, beriladigan virtual mashina (JVM) xususiyati.

Sinf uzgaruvchisi (переменная класса (tipa) (class variable))
alohida sinf ekzempilyariga emas, balki to‘laligicha biror sinfga tegishli bo‘lgan uzgaruvchi. Sinf uzgaruvchilari sinf elementlari aniqlanishi hisoblanadi.
Ular shuningdek sinfning statik maydoni degan nomga ham ega.

Mijoz (клиент (client))
ulanishning “mijoz-server” modelida mijoz — bu hisoblash serveri resurslariga masofadan turib murojat qiluvchi jarayon.

Asosiy kod manzili (адрес основного кода (codebase))
APPLET tegida code atributi bilan ishlashda applet bosh sinfi fayliga to‘liq yo‘l ko‘rsatiladi: code fayl nomini aniqlaydi, codebase esa — faylni uz ichiga oluvchi direktoriya URLi.

Kommentariya (комментарий (comment))
Kompilyator tomonidan qaralmydigan, dasturga izoh beruvchi matn. JavaTM ilovalarda kommentariyalar // yoki /*…*/ simvollari bilan ajratiladi.

Tranzaktsiyani qayd qilish (фиксация (транзакции) (commit))
tranzaktsiya jarayonida qilingan barcha uzgarishlar ma’lumotlar bazasida qayd qilinadigan tranzaktsiya momenti.

Kompilyatsiya birligi (единица компиляции (compilation unit))
Kompilyatsiya qilinishi mumkin bo‘lgan boshlang‘ich kodning minimal birligi. JavaTM ning odatdagi joriy qilinishida kompilyatsiya birligi — bu paketlar e’lon qilinishi va import e’lon qilinishidan oldin keladigan interfeyslar va sinflar aniqlanishlari ketma ketligi.

kompilyator (compiler)
Ilovaning boshlang‘ich kodini kompilyator tomonidan bajariladigan kodga translyatsiya qiladigan dastur. JavaTM- kompilyator Java da yozilgan boshlang‘ich kodni Java virtual mashinasi uchun mashinaga bog‘liq bo‘lmagan kodga (bayt kodlarga) translyatsiya qiladi.

komponent (component)
Konteyner tomonidan qo‘llab quvvatlanadigan dastur moduli. Komponentlar ishlab chiqarish paytida konfiguratsiyalanadi. J2EE platformasi to‘rt xil turdagi: korporativ komponentlar (enterprise beans), Web-komponentlar, appletlar va mijoz ilovalari kabi komponentlarni aniqlaydi.

Komponent kontrakti (контракт компонента (component contract))
komponent va uning konteyneri orasidagi munosabatlarni muvofiqlashtiruvchi shartlar majmuasi. Kontrakt o‘z ichiga qo‘yidagilarni: komponentni xayot tsiklini boshqarishni, konteyner to‘g‘risida axborot olish uchun komponent namunasi sifatida ishlatiladigan kontekstli interfeys yoki konteyner imkoniyatlaridan foydalanish, va berilgan komponent uchun har bir konteyner qo‘llashi zarur bo‘lgan funktsional imkoniyatlar ruyhati kabilarni oladi.

Komponent atrofi (окружение компонента (component environment))
J2EE komponentlari ishlata oladigan, Dasturiy Komponentlarni etkazib beruvchi (Application Component Provider) tomonidan aniqlanadigan talablar majmuasi. Atrof yozuvlari komponent bayonida deklarativ aniqlangan. Har bir komponent komponent konfiguratsiyasi qiymatlariga yo‘l ko‘rsatadi va kirishga ruhsat oladi. Bunda java:comp/env JNDI kontekstini ishlatadi. Bu qiymatlar komponent bog‘liq bo‘lgan ob’ektlar, JDBC DataSource kabi yoki soliq stavkasi kabi oddiy qiymatlar bo‘lishi mumkin.

komponovka (compositing)
yagona tasvirni olish uchun bir tasvirni boshqa tasvir ustiga joylashtirish jarayoni.

Ulanish (соединение (connection))
ulanish menejerga qarang.

Ulanish ustasi (мастер соединения (connection factory))
ulanish ustasi resurslari menejeriga qarang.

konnektor (connector)
Boshqarish axborot tizimlari (Executive Information Systems — EISs) bilan muloqatni amalga oshirish uchun konteynerlarni kengaytirishning standart mexanizmi. Konnektor har bir EIS uchun maxsus xususiyatga ega va EIS bilan muloqat qilish uchun ilovalar yaratish vositalari va resurslar adapteridan tashkil topgan. Konnektor arxitektursida aniqlangan resurslar adapteri konteynerga tizim sathidagi kontraktlar yordamida ulangan.

Konnektor arxitekturasi (архитектура коннектора konnektora (connector architecture))
J2EE serverlarini boshqaruv axborot tizimlari (Executive Information Systems — EISs) bilan integratsiya qilishga muljallangan arxitektura. Arxitektura ikki qismdan iborat: EIS ni ishlab chiqaruvchi resurslar adapteri va ushbu adapterni qullovchi J2EE serveri. Ushbu arxitektura J2EE-serverga ulanish uchun resurslar adapteri tomonidan qullanadigan kontraktlar majmuasini aniqlaydi. Masalan, tranzaktsiyalar, havfsizlikni ta'minlash, resurslarni boshqarish.

konstruktor (constructor)
Ob'ektni yaratuvchi va uning maydonlarini initsializatsiya qiluvchi maxsus turdagi metod. Java dasturlash tilida konstruktor nomi sinf nomi bilan mos tushadi. Konstruktorlar ob'ekt ekzempilyari yaratilayotganda (new konstruktsiyasi bajarilayotganda) tizim tomonidan chaqiriladi.

const
JavaTM ning rezervlashtirilgan kalit suzi. Biroq, Java ning joriy versiyalari tomonidan ishlatilmaydi.

konteyner (container)
Komponent bajarilishi boshqaruvi, havfsizligi, ishlab chiqarilishi va servislarini ta'minlovchi mohiyat. Bundan tashqari har bir turdagi konteyner (EJB, Web, JSP, servlet, applet yoki mijoz-ilovasi) ham uzlarining maxsus servislarini taklif etadi.

Konteyner tomonidan boshqariladigan persistentsiya (saqlanuvchanlik) (персистенция (сохраняемость), управляемая контейнером (container-managed persistence)) korporativ komponenta ekzimpilyari va quyida joylashgan resurslar menejeri orasidagi ma'lumot uzatish korporativ komponentlar konteyneri (enterprise bean) tomonidan boshqariladigan mexanizm.

Konteyner tomonidan boshqariladigan tranzaktsiya (транзакция, управляемая контейнером (container-managed transaction))
Chegaralari EJB-konteyner tomonidan aniqlanadigan tranzaktsiya. Korporativ komponenta ekzemplyari (enterprise bean) konteyner tomonidan boshqariladigan tranzaktsiyalardan foydalanishi zarur.

Kontekstli atribut (контекстный атрибут (context attribute))
Servlet tomonidan assotsiatsiya qilinadigan, kontekst ichiga joylashtirilgan ob'ekt.

FOYDALANILGAN ADABIYOTLAR RO‘YXATI

Asosiy adabiyotlar

1. Herbert Schildt. Java. A Beginner’s Guide. Sixth Edition. McGraw-Hill Education (Publisher). 2014, 699 p.
2. Bjarne Stroustrup. Programming. Principles and practice using C++. Second edition. Addison-Wesley. 2014, 1274 p.
3. Герберт Шилдт. Java8. Полное руководство. 9-е издание. "Вильяме", 2015, 1375 с.
4. Ken Arnold, James Gosling, David Holmes. The Java Programming Language, Fourth Edition. 2005, 928 p.
5. Васильев А.Н. Java. Объектно-ориентированное программирование. ПитеР. 2013, 400 с.

Qo‘shimcha adabiyotlar

1. Mirziyoyev Sh.M. Buyuk kelajagimizni mard va olijanob xalqimiz bilan birga quramiz. 2017.
2. Mirziyoyev Sh.M. Qonun ustuvorligi va inson manfaatlarini ta’minlash – yurt taraqqiyoti va xalq farovonligining garovi. 2017.
3. Mirziyoev Sh.M. Erkin va farovon, demokratik O‘zbekiston davlatini birgalikda barpo etamiz. 2017 y.
4. Robert Sedjvik, Kevin Ueyn. Алгоритмы на java. Вильямс. 2013, 848 с.
5. Rudolf Pecinovsky. OOP: Learn Object Oriented Thinking and Programming. Eva & Tomas Bruckner. 2013, 527 p.
6. Tursunov N.X. Java dasturlash tili. TIU nashriyoti. 2013. 375 b.

Internet saytlar

1. www.gov.uz. – O‘zbekiston Respublikasi xukumat portali.
2. http://www.tuit.uz
3. http://www.atdt.uz
4. http://www.ziyonet.uz
5. http://www.learnjavaonline.org/
6. https://www.tutorialspoint.com/java/
7. http://www.javatpoint.com/
8. http://study-java.ru/
9. http://java-course.ru/
10. http://dasturchilar.uz/
11. http://ziyonet.uz/
12. http://www.wikipedia.org
13. http://www.intuit.ru