



**ЎЗБЕКИСТОН АЛОҚА, АХБОРОТЛАШТИРИШ ВА
ТЕЛЕКОММУНИКАЦИЯ ТЕХНОЛОГИЯЛАРИ ДАВЛАТ ҚЎМИТАСИ
ТОШКЕНТ АХБОРОТ ТЕХНОЛОГИЯЛАРИ УНИВЕРСИТЕТИ**

**«Дастурий инжиниринг» факультети
«АТДТ» кафедраси
«ОЙДТ» фанидан**

РЕФЕРАТ

**Мавзу: Синфларда ворислик ва
объектлар**

**Бажарди: 216-11 гуруҳ талабаси
Сирожиддонов Бобур
Қабул қилди: Раҳмонова М**

Тошкент 2013-2014

Кириш

Сўнги йилларда компьютернинг дастурий таъминоти ривожланиши асосий йўналишларидан бири бу объектга йўналтирилган дастурлаш соҳаси бўлди. Объектга йўналтирилган операцион тизимлар (Масалан, Windows), амалий дастурлар ва объектга йўналтирилган дастурлаш (ОЙД) тизимлари ҳам аммвийлашди.

Биринчи ОЙД элементи Симула-67 (1967 й., Норвегия) тили бўлди. Turbo PASCAL да 5,5 версиясидан бошлаб ОЙД воситалари пайдо бўлди. Turbo PASCAL нинг ривожчи якуни якуни сифатида BORLAND филмаси томонидан DELPHI дастурлаш тизими яратилиши бўлди. Ушбу система ёрдамида тез ва осон мураккаб бўлган график интерфейсни дастурлаш имконияти мавжуд. 1991 йилда Visual BASIC нинг I версиясидан бошлаб бу тил тўлалигича объектга йўналтирилди (1997 йил).

ОЙД тилларидан яна бири 1995 йилда Жеймс Гослинг бошчилигида Sun Microsystems компаниясида яратилган JAVA тилидир. Уни ишлаб чиқишида махсус ўрганиши талаб қилмайдиган, содда тилни мақсад қилинган.

JAVA тили максимал даражада C++ тилига ўхшаши бўлиши учун яратилган яратилган JAVA Интернет учун дастурлар тайёрлашнинг идеал воситасидир. Сўнги йилларда Microsoft компанияси томонидан C++давомчиси сифатида C# (Си шарп) тили яратилди.

1985 йилда Bell Labs (АҚШ) лаюораторияси C++ дастурлаш тили яратилганлигини хабарини берди. Бугунги кунда бу тил ОЙД тиллари орасида машхурдир. Бу тил ёрдамида исталган машина учун – шахсийдан то суперкомпьютерларгача дастурлар ёзиши мумкин. Бу тилнинг асосчиси Бьорн Страуструпдир.

Масалани қўйилиши

ФУТБОЛ МУСОБАҚАСИ(номи, КОМАНДАЛАР, МАЙДОНЛАР, УЧРАШУВЛАР) Учрашувга тегишли ҳамма маълумотлар.синфи берилган.Фойдаланувчи структура массивини яратинг. Оқимда(файлда) уларни сақлаш функциясини яратинг. Дастурдан фойдаланинг. Кўрсатилган кўрсатилган структура массивини файлдан ўқиш ва чиқариш функцияларини яратинг Берилган критерий асосида файлдаги маълумотларни коррективровка (алмашиш) қилувчи функция яратинг. Дастурни яратинг ва мавжуд бўлган файлни кўриб чиқинг.

Ишдан мақсад: Стандарт библиотекадаги оқимли синфлар объектларидан фойдаланган холда киритиш ва чиқариш дастурлашни ўрганиш.

Назарий қисм

Синфлар ва объектлар

Синтаксис бўйича, C++ ва JAVA да синф – бу мавжуд бўлган типлар асосида янги яратилган структурланган тип.

Синф таърифи содда шакли:

<синф_типи> <синф_номи>{<синф_компонентлари_рўйхати>;}

бу ерда:

синф_типи – **class, struct, union** хизматчи сўзларидан бири; Java да фақат **class** ишлатилади.

синф_номи – идентификатор;

синф_компонентлари_рўйхати – синфга тегишли маълумотлар ва функциялар таърифи.

Функция – бу объектлар устида бажариладиган операцияларни аниқловчи синф усули.

Маълумотлар – бу объект структурасини хосил қилувчи майдон.

Усуллар синфдан ташқарида аниқланганда уларнинг номларини квалификация қилиш (ихтисослаштириш) керак. Усулнинг кўримлилик соҳасини аниқлайдиган унинг бундай квалификация синтаксиси қуйидаги кўринишга эга:

<синф номи>::<усул номи>

Синф ичида аниқланган усуллар кўзда тутилган бўйича жойлаштирилувчи (**inline**) функция ҳисобланади. Синф ташқарисида аниқланган усулларни ошкор равишда жойлаштирилувчи деб таърифланиши лозим.

Синф объекти (синф нусхаси) ни таърифлаш учун қуйидаги конструкциядан фойдаланилади:

<синф_номи> <объект_номи>;

Объект орқали майдонларга ва усулларга қуйидагича мурожат қилиш мумкин:

< объект_номи > . <майдон_номи>

< объект_номи >. <усул_номи>

Компоненталарга мурожат ҳуқуқлари

Компоненталарга мурожаат ҳуқуқи мурожаат спецификаторлари ёрдамида бошқарилади: **public, private, protected**.

Умумий (public) компоненталар дастурни ихтиёрий қисмида мурожаат ҳуқуқига эга. Улардан, ихтиёрий функция ушбу синф ичида ва синф ташқарида фойдаланса ҳам бўлади.

Хусусий (private) компоненталар синф ичида мурожаат ҳуқуқига эга, лекин синф ташқарисидан эса мурожаат қилиш мумкин эмас. Компоненталардан ушбу улар тавсифланган синфдаги функция - аъзолари ёки “дўстона”- функциялар орқали фойдаланиш мумкин.

Химояланган (protected) компоненталар синф ичида ва хосила синфларда мурожаат ҳуқуқига эга.

Агар синф таърифида **class** сўзи ишлатилган бўлса ҳамма компоненталари хусусий ҳисобланади, агар **struct** сўзи ишлатилган бўлса ҳамма компоненталар умумий ҳисобланади.

Конструктор

Конструктор - бу синф объектларини автоматик инициализация қилиш учун ишлатиладиган махсус компонентали функция.

Конструкторлар қуриниши қуйидагича бўлиши мумкин:

```
<Синф номи> (<формал параметрлар руйхати>)  
{<конструктор танаси>}
```

Бу компонента функция номи синф номи билан бир хил бўлиши лозим.

Дастурчи томонидан кўрсатилмаган ҳолда ҳам new оператор ёрдамида синф объекти яратилганда ёки хотирада жойлашганда конструктор автоматик равишда чақирилади.

Конструктор объект учун хотирада жой ажратади ва маълумотлар – синф аъзоларини инициализациялайди.

Конструктор бир нечта хусусиятларга эга:

- Конструкторлар учун қайтарилувчи типлар, ҳатто void типи ҳам кўрсатилмайди
- Конструктор адресини ҳисоблаш мумкин эмас. Конструктор параметри сифатида уз синфининг номини ишлатиш мумкин эмас, лекин бу номга кўрсаткичдан фойдаланиш мумкин.
- Конструкторлар ворисликга эга эмас.

Конструкторлар ихтиёрий синфлар учун доимо мавжуд, лекин агарда у кўрсатилган ҳолда тавсифланмаган бўлса, у автоматик равишда яратилади. Кўрсатилмаган ҳолда параметрсиз конструктор ва нусха конструктори яратилади. Агарда конструктор очиқ ҳолда тавсифланган бўлса, унда кўрсатилмаган ҳолда конструктор яратилмайди. Кўрсатилмаган ҳолда умумий (public) конструкторлар яратилади.

Объектлар массиви

Объектлар массиви таърифлаш учун синф кўзда тутилган (параметрсиз) конструкторга эга бўлиши керак.

Объектлар массиви кўзда тутилган конструктор томонидан, ёки ҳар бир элемент учун конструктор чақириш йўли билан инициализация қилиниши мумкин.

```
class complex a[20]; //кўзда тутилган параметрсиз конструкторни чақириш
```

```
class complex b[2]={complex (10),complex (100)};//ошкор чақириш
```

finalize

Java тилида **finalize** номли усуллар киритиш имконияти мавжуд. Бу номли усуллар C++ тилидаги (калит белги ~) ва Delphi тилидаги (калит сўз **destructor**) деструкторларга мосдир. Java бажариш мухити ҳар гал объектни ўчиришда шу услни чақиради.

This кўрсаткичи

Агарда конкрет объектга ишлов бериш учун синф аъзоси – функция чақирилса, унда шу функцияга объектга белгиланган кўрсаткич автоматик ва кўрсатилмаган ҳолда узатилади. Бу кўрсаткич **this** исмига.

Синфни қуйидаги тасвирлаш мумкин:

```
class Point { int x, y;  
Point(int x, int y) {  
this.x = x;  
this.y = y;  
} }
```

Синфнинг статик компонентлари

Синф компонентаси ягона булиб ва ҳамма яратилган объектлар учун умумий булиши учун уни статик элемент сифатида таърифлаш яъни **static** атрибути орқали таърифлаш лозимдир. Объектларни яратишда синф статик маълумотлари такрорланмайди, яъни ҳар бир статик компонентлар бирдан-бир кўринишда мавжуд.

Статик усуллар фақат бошқа статик усулларга тўғридан тўғри мурожаат қилиши мумкин ва уларда **this** иловасидан фойдаланиш мумкин эмас. Ўзгарувчилар ҳам **static** типга эга бўлиши мумкин, бу ҳолда уларга глобал ўзгарувчиларга ўхшаб дастур ихтиёрий қисмидан мурожаат қилиш мумкин. Статик усуллар ичида ностатик ўзгарувчиларга мурожаат қилиш мумкин эмас.

Синф статик маълумотларга фақатгина объект исми орқали мурожаат этиш мумкин. **<объект_номи>.<компонента_номи>**

Масалан:

```
complex a; a.count=5;
```

Лекин, статик компонентларга синф объекти аниқланмаган ҳолда ҳам мурожаат этиш мумкин. Статистик компонентларга нафақат объект исми, балки синф исми орқали ҳам мурожаат этиш мумкин.

<синф_номи>. <компонента_номи>

Лекин шундай мурожаат фақатгина *public* компонентларга тегишли. *private* статик компонентларга ташқаридан мурожаат этишда **функция – статик компонентлардан** фойдаланилади.

Бу функцияларни синф исми орқали чақириш мумкин.

<синф_номи> .: <статик_функция_номи>

Синфлар ва объектлар билан ишлаш хусусиятлари

Runtime

Runtime синфи Java интерпретаторини инкапсуляция қилади. Бу синф объектини яратиш мумкин эмас, лекин статик усулидан фойдаланиб ишлаб турган объектга мурожаат қилиш мумкин. Одатда апплетлар ва бошқа дастурлар синф усулини чақирганда SecurityException истисноси хосил бўлади. Runtime объектини тўхтатиш учун `exit(int code)` усулини чақириш етарли.

Хотирани бошқариш

Java тилида хотира автоматик тозаланса ҳам дастур эффективлигини текшириш учун “уюм” хажмини аниқлаш ва эркин хотира хажмини хисоблаш мумкин. Бу маълумотни аниқлаш учун `totalMemory` ва `freeMemory` усулларида фойдаланиш мумкин.

Зарур бўлганда хотира тозаловчини `gc` усулини чақириб автоматик ишга тушириш мумкин. Дастурга зарур хотирани хисоблаш учун аввал `gc`, сўнгра `free-Memory` усуллари чақириш керак. Шундан сўнг дастурни ишга тушириб `freeMemory` усули чақирилса, дастур қанча хотира ишлатишини аниқлаш мумкин.

Бошқа дастурларни бажариш

Хавфсиз мухитларда Java тилидан бошқа жараёнларни ишга тушириш учун фойдаланиш мумкин. Бунинг учун `exec` усулининг бир неча шаклларида фойдаланиш мумкин. Усулга дастур номи ва бир неча параметрлар узатилади.

System

`System` синфида турли глобал функциялар ва ўзгарувчилар мавжуд. Масалан `System.out.println()` усули.

Бундан ташқари `currentTimeMillis` усули тизимли вақтни 1970 йил 1 январидан бўлиб ўтган миллисекундларда қайтаради.

Массивдан нусха олиш учун `arraycopy` усулидан фойдаланилади. Қуйида бир массивдан иккинчисига нусха олишга мисол келтирилган.

```
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));} }

```

Мухит хоссалари

Java бажариш мухити Properties синфи объекти орқали мухит ўзгарувчиларига мурожаат қилишга имкон беради. Хоссалар тўлиқ рўйхатини олиш учун System.getProperties() усулини чақириш лозим.

Жадвал

Система стандарт хоссалари

Исм	Қиймат	Апплет учун рухсат
Java.version	Java интерпретатора версияси	ха
Java.vendor	Фойдаланувчи киритган идентификатор қатори	ха
java.vendor.url	Ишдаб чиқарувчи URLи	ха
java.class.version	Java API версияси	ха
java.class.path	CLASSPATH ўзгарувчиси қиймати	йўқ
java.home	Java мухити инсталляция қилинган каталог	йўқ
java.compiler	Компилятор JIT	йўқ
os.name	Операцион тизим номи	ха
os.arch	Дастур бажарилаётган компьютер архитектураси	ха
os.version	Web-тугун операцион тизими версияси	ха
file.separator	Платформага боғлиқ файл ажратувчилари (/ ёки \)	ха
path.separator	Платформага боғлиқ йўл ажратувчилари (: ёки ;)	ха
line.separator	Платформага боғлиқ сатр ажратувчилари (\n ёки \r\n)	ха
user.name	Жорий фойдаланувчи исми	йўқ
user.home	Фойдаланувчи каталог	йўқ
user.dir	Жорий ишчи каталог	йўқ

user.language	2-символли махаллий тил коди	йўқ
user.region	2-символли мамлакат тил коди	йўқ
user.timezone	Кўзда тутилган вақт зонаси	йўқ
user.encoding	Кўзда тутилган бўйича белгилар коди	йўқ
user.encoding.pkg	Махаллий символларни Unicode кодига ўтказиш учун конверторлар пакети	йўқ

Date

Сана ва вақт билан ишлаш учун Date синфидан фойдаланилади. У орқали сана, йил, ой, hafta кунига, соат, минут, секундга мурожаат қилиш мумкин. Бу синф турли конструкторлари мавжуд. Энг соддаси — Date() — объектни жорий сана ва вақт билан инициализация қилади. Қолган уч конструктор кўшимча имкониятларга эга.

- Date(year, month, date) — кўрсатилган санани ўрнатади, вақт 00:00:00 (тунги) қийматга эга бўлади.
- Date(year, month, date, hours, minutes) — кўрсатилган сана ва вақтни ўрнатади, секунд 0 қийматни олади.
- Date(year, month, date, hours, minutes, seconds) — энг тўлиқ кўриниши, сана ва вақт ҳамда секундлар ҳам ўрнатилади.

get ва set

Класс Date синфи объект атрибутларини ўрнатиш учун усулларга эга. Бу оилага кирувчи усуллар — getYear, getMonth, getDate, getDay, getHours, getMinutes и getSeconds — бутун қиймат қайтаради.

Date синфи қийматини getTime усули long типдаги сон сифатида қайтаради. Бу сон 1970 йил 1 январидан ўтган миллисекундларга тенг.

Солиштириш

Date типдаги икки объектни солиштириш учун, санани миллисекундларга айлантириш лозим. Date синфи тўғридан тўғри солиштириш учун уч усулга эга: — before, after ва equals. Масалан

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

true қайтаради, чунки ойнинг 12-куни 18-кунидан олдин келади.

Сатр ва сана

Объекты Date объектини турли форматдаги матнга кон иможно конвертация қилиш мумкин. Авваламбор toString усули Date объектини қуйидагича сатрга алмаштиради “Thu Feb 15 22:42:04 1996”. Кейинги toLocaleString усули санани қисқароқ сатрга алмаштиради, масалан: “02/15/96 22:42:04”. Ва ниҳоят toGMTString усули санани Гринвич бўйича ўртача вақт форматига ўтказди: “16 Feb 1996 06:42:04 GMT”.

Math

Math синфи геометрия ва тригонометрияда ишлатиладиган сузувчи вергулли функцияларга эга. Бундан ташқари хисоблашларда ишлатиладиган икки константа мавжуд: — E (тахминан 2.72) ва PI (тахмина 3.14159).

Тригонометрик функциялар.

Қуйида келтирилган уч функция радианларда бурчакни ифодаловчи double типдаги параметрга эга бўлиб, мос тригонометрик функция параметрини қайтаради.

- $\sin(\text{double } a)$ радианда берилган a бурчак синусини қайтаради.
- $\cos(\text{double } a)$ радианда берилган a бурчак косинусини қайтаради.
- $\tan(\text{double } a)$ радианда берилган a бурчак тангенсини қайтаради.

Кейинги тўрт функция узатилган параметр қийматига мос бурчакни радианларда қайтаради.

- $\text{asin}(\text{double } r)$ синуси r га тенг бурчакни қайтаради.
- $\text{acos}(\text{double } r)$ косинуси r га тенг бурчакни қайтаради.
- $\text{atan}(\text{double } r)$ тангенси r га тенг бурчакни қайтаради.
- $\text{atan2}(\text{double } a, \text{double } b)$ тангенси a/b га тенг бурчакни қайтаради.

Даражага кўтариш, экспонента ва логарифм функциялари

- $\text{pow}(\text{double } y, \text{double } x)$ x даражага кўтарилган y қайтаради. Масалан, $\text{pow}(2.0, 3.0)$ тенг 8.0.
- $\text{exp}(\text{double } x)$ e даражаси x қайтаради.
- $\text{log}(\text{double } x)$ x натурал логарифминини қайтаради.
- $\text{sqrt}(\text{double } x)$ x квадрат илдизини қайтаради.

Яхлитлаш

- $\text{ceil}(\text{double } a)$ қиймати a дан катта ёки a га тенг бўлган энг кичик бутун сон қайтаради.
- $\text{floor}(\text{double } a)$ қиймати a дан кичик ёки a га тенг бўлган энг катта бутун сон қайтаради.
- $\text{rint}(\text{double } a)$ каср қисми олиб ташланган double типда a қийматинини қайтаради.
- $\text{round}(\text{float } a)$ энг яқин бутун сонга яхлитланган a қийматинини қайтаради.
- $\text{round}(\text{double } a)$ энг яқин узун бутун сонга яхлитланган a қийматинини қайтаради.

Бундан ташқари Math синфида int, long, float ва double типлари билан ишловчи модул олиш, минимал ва максимал қийматни топиш усуллари полиморф версиялари мавжуд:

- $\text{abs}(a)$ a модули (абсолют қиймати) ни қайтаради.
- $\text{max}(a, b)$ ўз аргументлари энг каттасини қайтаради.
- $\text{min}(a, b)$ ўз аргументлари энг кичигини қайтаради.

Random

Random — псевдотасодифий сонлар генератори бўлиб, унда ишлатилган алгоритм Дональд Кнут “Дастурлаш санъати” китобининг 3.2.1 бўлимида келтирилган. Одатда бошланғич қиймат сифатида жорий вақт олинади, бу эса қайтарилувчи тасодифий сонлар олиниши эхтимоллигини камайтиради.

Random синфи объектдан 5 турдаги тасодифий сонларни олиш мумкин. Бу тип диапазони бўйича бир текисда тақсимланган бутун сонни олиш учун nextInt усулидан фойдаланилади. Шунга ўхшаш nextLong усули long типдаги тасодифий сонни қайтаради. Бундан ташқари nextFloat ва nextDouble усуллари мос равишда float ва double типдаги, 0.0..1.0 интервалда текис тақсимланган сонларни қайтаради.

СИНФЛАРДА ВОРИСЛИК

Ворисликда мурожаат ҳуқуқларини бошқариш

Ворислик ўзининг барча аждодларининг хусусиятлари, маълумотлари, методлари ва воқеаларини мерос қилиб оладиган ҳосила синфини эълон қилиш имкониятини беради, шунингдек янги тавсифларни эълон қилиши ҳамда мерос сифатида олинаётган айрим функцияларни ортиқча юклаши мумкин. Базавий синфнинг кўрсатиб ўтилган тавсифларини мерос қилиб олиб, янги туғилган синфни ушбу тавсифларни кенгайтириш, торайтириш, ўзгартириш, йўқ қилиш ёки ўзгаришсиз қолдиришга мажбурлаш мумкин.

В JAVA тилида тўғридан тўғри аждод суперсинф деб аталади.

Ҳосила синфни эълон қилишнинг умумлашган синтаксиси:

```
class <синф номи>: [<кириш ҳуқуқини берувчи сецификатор >]  
<аждод синф номи> {...}
```

Конструктор ва деструкторларда ворислик

Конструкторлар мерос бўлмагани учун, ҳосила синфни яратишда ундан мерос бўлган маълумот – аъзолари асосий(базавий) синф конструктори орқали инициализацияланиши лозим. Асосий синф конструктори автоматик равишда чақирилади ва ҳосила синфни конструкторидан олдин бажарилади. Асосий (базавий) синфни конструкторининг параметрлари ҳосила синфни конструкторни аниқлашда кўрсатилади. Шундай қилиб аргументларни ҳосила синфни конструкторидан асосий (базавий) синфни конструкторига узатиш вазифаси бажарилади.

Асос синф конструктори параметри ҳосила синф конструктори таърифида **super** калит сўзи ёрдамида кўрсатилади.

Синф объектлари пасдан тепага қараб конструкторланади: аввало асосий(базавий), кейин эса компонент – объектлар (агарда улар мавжуд бўлса), ундан кейин эса ҳосила синфнинг ўзи. Шундай қилиб, ҳосила синфнинг объекти қуйи объект сифатида асосий (базавий) синф объектини ўз ичига олади.

Объектлар тескари тартибда ўчирилади: аввало ҳосила, кейин унинг компонент – объектлари, ундан кейин эса асосий(базавий) объект.

Шундай қилиб объектни ўчириш тартиби унинг конструкторлаш тартибига нисбатан тескари бўлади.

Виртуал функциялар

Виртуал функциялар механизмига бирор компонент функциянинг ҳар бир ҳосилавий синфда алоҳида варианты мавжуд бўлиши лозим бўлганда

мурожаат қилинади. Бундай функцияларга эга синфлар **полиморф** синфлар, деб аталади ва объектли дастурлашда ахлохида ўринга эга.

Виртуал функциялар кечки ёки динамик боғланиш механизми асослангандир. Асос синф ҳар қандай ностатик компонент функцияси **virtual** калит сўзи ёрдамида виртуал деб эълон қилиниши мумкин.

Кечки боғланишда эрта боғланишга ўхшаб адреслар статик равишда компиляция жарёнида эмас, балким динамик дастур бажарилиши жараёнида аниқланади. Боғлаш жараёни виртуал функцияларни адреслар билан алмаштиришдан иборат. Виртуал функциялар адреслар ҳақида маълумот сақланувчи жадвалдан фойдаланади.

Виртуаллик ворисликка ўтади. Функция виртуал деб эълон қилингандан сўнг, ҳосила синфда қайта таърифи (шу прототип билан) бу синфда янги виртуал функцияни яратади, бу ҳолда `virtual` спецификатори талаб қилинмайди.

Эътибор беринг `main` усули ичида А синфи ўзгарувчиси таърифланиб, В синфи объекти ёрдамида инициализация қилинган. Кейинги қаторда `callme` усули чақирилган. Транслятор А синфида `callme` усули мавжудлигини текширди, бажарувчи тизим ўзгарувчида В объекти сақланганлиги учун, А синфи эмас, В синфи `callme` усулини чақиради.

Усулларни қайта таърифлаш

`Point` синфининг янги остки синфи `Point3D` ўз суперсинфининг `distance` усулини мерос қилиб олади (мисол `PointDist.java`). Лекин `Point` синфида текисликда нуқталар орасида масофа қайтарувчи `distance(mt x, int y)` усули берилган. Биз бу усулни уч ўлчовли фазога мос келадиган қилиб, қайта таърифлашимиз (**override**) лозим.

final

Хамма усуллар ва ўзгарувчилар кўзда тутилган бўйича қайта таърифланиши мумкин. Агар ворис синфда бирор ўзгарувчи ёки усулни қайта таърифлашга ҳаққи йўқлини кўрсатиш учун уларни `final` (`Delphi` / `C++` тилида **virtual** сўзини ёзмаслик керак) деб таърифлаш лозим.

final int FILE_NEW = 1;

Қабул қилинган қоида бўйича По общепринятому соглашению при выборе имен переменных типа `final` типдаги ўзгарувчиларни номлашда фақат юқори регистрдаги символлардан фойдаланилади. (`C++` тилида препроцессор константалар). Баъзида `final`- усуллардан фойдаланиш код бажарилишини тезлаштиради — чунки, транслятор уларни жойлаштирилувчи (`inline`) код деб эълон қилади (байт-код тўғридан тўғри кодга жойлаштирилади).

Абстракт синфлар

Жуда бўлмаса битта абстракт усулга эга синф абстракт синф деб аталади. Абстракт усул деб қуйидаги кўринишга эга компонент функцияга айтилади:

abstract<тип> <исм> (<формал_параметрлар_рўйхати>);

Бу синф объектларини яратиш мумкин эмас. Абстракт синф фақат ҳосила синфлар учун асос синф сифатида ишлатилиши мумкин.

Хар қандай abstract усулга эга синф, abstract деб таърифланиши шарт. Бундай синфларда тўлиқ реализация мавжуд бўлмагани учун, new оператори ёрдамида вакиллари яратиш мумкин эмас. Бундан ташқари абстракт конструкторлар ва статик усуллар элон қилиш мумкин эмас. Абстракт синф хар қандай вориси ёки суперсинф абстракт усуллари тўлиқ реализация қилиши керак, ёки ўзи абстракт деб элон қилиниши керак.

Глобал Суперсинф

Object: глобал суперсинф

Object синфи ҳамма синфлар аждоди ҳисобланади. Java тилида хар бир синф, Object синфини кенгайтиради. Лекин class Employee extends Objects ёзиш шарт эмас. Агар суперсинф ошкор кўрсатилмаган бўлса Object суперсинф ҳисобланади. Java тилида хар бир синф Object синфини кенгайтиргани учун Object синфи имкониятларини билиш муҳимдир.

Object типдаги ўзгарувчини ихтиёрий типдаги объектга илова сифатида ишлатиш мумкин:

```
Object obj = new Employee("Гарри Хакер", 35000);
```

Бу типдаги ўзгарувчидан фойдаланиш учун аввал бошланғич типни аниқлаб, типларни келтиришни амалга ошириш лозим:

```
Employee e = (Employee) obj;
```

equals ва toString усуллари

Object синфининг equals усули икки объект бир хиллигини текширади. Лекин equals усули Object синфига тегишли бўлгани учун, иккаласи бир хотира қисмига илова қилганлигини текширади. Икки объект эквивалентлигини текшириш учун equals усулини қўшимча юклаш лозим. Мукамал equals усули яратиш қоидалари.

1. Ошкор otherObject параметрини чақириш — кейинчалик унинг типини other деб аталган бошқа ўзгарувчи типига келтириш лозим.

2. Текшириш, this ва otherObject иловалар бир хилми:

```
if ( this == otherObject) return true;
```

Одатда объектлар майдонини солиштиргандан кўра иловаларни солиштириш осондир.

3. Текшириш otherObject илова нульга (null) тенгми. Агар ха бўлса false қиймат қайтариш. Бу текширишни албатта амалга ошириш лозим.

```
if (otherObject == null) return false;
```

4. Текшириш this ва other объектлари битта синфга тегишлими.

Бу текшириш "симметриклик қоидасига " кўра мажбурийдир.

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

5. Талаб қилинган синф ўзгарувчисига otherObject объектини ўзгартириш:

```
ClassName other = (ClassName)otherObject;
```

6. Ҳамма майдонларни солиштириш. Асосий типдаги майдонлар учун == оператори, объектли майдонлар учун —equals усули қўлланади. Агар икки объект ҳамма майдонлари бир хил бўлса true қайтариш, акс холда false.

```
return field1 == other.field1
&& field.2. equals (other . field2)
```

Объект типини getClass усули орқали аниқланади. Объектлар ўзаро тенг бўлиши учун бир синф объектлари бўлишлари керак.

Ворис ичида аввал сперсинф equals усулини чақириш лозим. Агар бу текшириш false қиймат қайтарса, демак объектлар тенг эмас. Агар текшириш муваффақиятли бажарилса остки синф майдонларини текширишга ўтиш мумкин.

Object синфининг яна бир муҳим усули toString, бўлиб объектни сатр шаклида қайтаради. Бу усул деярли ҳамма синфларда қўшимча юкланади, ва объект ҳолатини босмага чиқаришга мўлжалланган.

Кўп (хаммаси эмас) toString усуллари синф номи дан иборат бўлиб, квадрат қавсларда майдонлари қийматлари кўрсатилади.

Албатта ворис синф яратган дастурчи ўз toString усулини яратиши ва ворис синф номини қўшиши лозим. Агар суперсинфда getClass ().getNamef() усули чақирилса, ворис синф super.ToString () усулини чақиради.

Агар объект сатр билан "+" амали ёрдамида конкатенация қилинса Java тили компилятори объект жорий ҳолатини олиш учун автоматик равишда toString усулини чақиради.

Бирорр x — ихтиёрий объект учун дастурчи System.out.println(x) усулини чақирсин; Бу ҳолда println усули x. toString () усулини чақиради ва натижа сатрини чиқаради. Object синфида аниқланган toString усули синф номи ва объект адресини чиқаради. Бунинг сабаби шуки PrintStream синфида toString усули қўшимча юкланмаган. Стандарт библиотекага тегишли кўп синфларда toString усули шундай аниқланганки, унинг ёрдамида дастурни созлаш учун керакли маълумот олиш мумкин. Баъзи созловчилар объектлар ҳолатини экранда акслантириш учун toString усулини чақиришга имкон беради. Шунинг учун дастур трассировкасида, қуйидаги ифодалардан фойдаланиш мумкин

```
System.out.println {"Жорий ҳолат = " + position);
```

Файллар билан ишлаш

Киритиш чиқариш. Киритиш манбаи деб қурилмаларнинг абстрактлашган қурилмаларнинг тушунчаси ётади. Худди шунингдек чиқариш манбаи деб турли чиқариш қурилмалари тушунилади. Абстрактлашган киритиш чиқариш шуни назарда тутадикки дастур кодида киритиш чиқариш қурилмалари конкрет тасвирланмайди. Бу абстракт тушунчалар оқим тушунчаси ёрдамида амалга оширилади. Оқимлар билан ишлаш учун дастурда оқимли синфлардан фойдаланиш зарурдир. Оқимли синфлар абстракт синфлар бўлиб дисклар, ташқи хотира ва тармоқ билан ишлашга имкон беради.

InputStream синфи киритиш учун бу синфдан фойдаланилади. Бу синф куйидаги усулларга эга:

read усули ва бу усул учта кўринишга эга

- Оқимдан бутун сон ўқийди;
- Массив максимал элементларини ўқийди;
- Берилган байтдан бошлаб берилган сондаги байтларни ўқийди ва берилган массивга ёзиб кўяди.

skip усули. Оқимдан берилган сондаги байтларни ўтказиб юборишга ҳаракат қилади.

available усули ўқиш мумкин бўлган максимал байтлар сонини қайтаради.

close усули киритиш манбаини беркитади. Беркитилган манбадан ўқиш истисно генерация қилинишига олиб келади.

OutputStream бу синф ҳам абстракт бўлиб маълумотларни чиқариш учун ишлатилади. Усуллари куйидагича:

write усули уч хил кўринишга эга, улар куйидагича:

- Берилган сонни чиқариш оқимига ёзади;
- Берилган байтлар массивини тўлалигича чиқаради;
- Байтлар массивини берилган позициядан бошлаб берилган сондаги элементларни оқимга чиқаради.

flush усули чиқариш оқимини тозалайди.

close усули чиқариш оқимини беркитади. Беркитилган оқимга чиқариш хатоликка олиб келади.

Файлли оқимлар :

1. *FileInputStream*-файлдан ўқиш учун ишлатилади, 2та конструкторга эга.
2. *FileOutputStream*-файлга ёзиш учун ишлатилади. Бу синф ҳам 2та конструкторга эга. Бу синф объекти яратилганда агар файл мавжуд бўлса қайтадан яратилади ва ундаги маълумотлар ўчиб кетади.
3. *ByteArrayInputStream*-бу синфлар берилган байтлар массивини ўқийди.
4. *ByteArrayOutputStream*- берилган байтлар массивини тўлдириш учун ишлатилади. Иккала ҳолда ҳам ички буфери яратилади. Бу буфер ҳажми ишлатилган конструкторга қараб 32 ёки 1024 байт бўлиши мумкин.
5. *StringBufferInputStream*-биринчи синфга ўхшаган бўлиб фарқи шундаки ички буфер объектлари String типига тегишли бўлади.

Файллар билан ишлаш хусусиятлари

Тўғридан тўғри дискдаги файллар билан ишлаш учун java.io пакетидан жойлашган File синфидан фойдаланилади. Java тилида каталоглар оддий

файллар сифатида чиқарилади. Оддий файллардан фарқи шундаки, каталоглар ўз ичига файллар рўйхатини олади. Бу рўйхатни File синфининг List усули ёрдамида ўқиб олиш мумкин.

Каталогда файл ва остки каталоглар номлари ва уларга олиб боровчи йўл UNIX ёки MS DOS операцион тизимида қабул қилинган қоидалар асосида ёзилади, лекин Windows мухитида дастур бажарилганда автоматик равишда Windows да қабул қилинган қоидалар асосида қайтадан ёзилади. Агар файллар рўйхати олинганда уларнинг номларига бирор шарт қўйиш лозим бўлса FileNameFilter интерфейсидан фойдаланиш мумкин. Дастурда бу интерфейсдан фойдаланиш учун битта усул accept() усулиникайта таърифлаш етарлидир. Бу синфнинг юқорида кўрилган синфлардан фарқи, шундаки ҳамма усуллари синхронизация қилингандир, яъни бир нечта жараён бир пайтнинг ўзида ишлаганда турли тасодифий ходисилар юз бериш олди олингандир.

ФУТБОЛ МУСОБАҚАСИ(НОМИ, КОМАНДАЛАР, МАЙДОНЛАР, УЧРАШУВЛАР)

```
class Soccer { //Kutubxona nomli klass hosil qilindi;
    protected String name;
    protected String Muallif;
    protected String soni;
    protected String yili;
    protected int narxi; //O'zgaruvchilar e'lon qilindi;
public Soccer(String name,String Muallif,String soni,String yili,int narxi)
//Kostruktor;
{
    this.name=name;
    this.Muallif=Muallif;
    this.soni=soni;
    this.yili=yili;
    this.narxi=narxi;
};
    public void show_soccer() //show() nomli funksiya e'lon qilindi;
    {
        System.out.println("Komandalar : " +name);
        System.out.println("O`ynash vaqti : " +Muallif);
        System.out.println("Stadion nomi : " +soni);
        System.out.println("Stadion sig`imi : " +yili);
        System.out.println("Bilet narxi : " +narxi);
    }
}
class Kitob extends Soccer {
```

```
private String person;
public Kitob(String name,String Muallif,String soni,String yili,int narxi,String
person){
    super(name,Muallif,soni,yili,narxi);
    this.person=person;
}

public void show_card () {
show_soccer();
System.out.println("Umumiy hisob : " +person);
}

}

public class Main //Main nomli asosiy klass hosil
qilindi;
{
public static void main(String[] args) {

    Kitob Main= new Kitob("Bunyodkor-Paxtakor","13.10.13
17.00","Paxtakor","18567",5000," 3-1 "); //Main class o'zgaruvchilariga
qiymatlar berildi;
    Main.show_card(); //show() funksiyasi chaqirildi;

}
}
```

The screenshot shows the NetBeans IDE interface. The main editor displays the following Java code for the `Soccer` class:

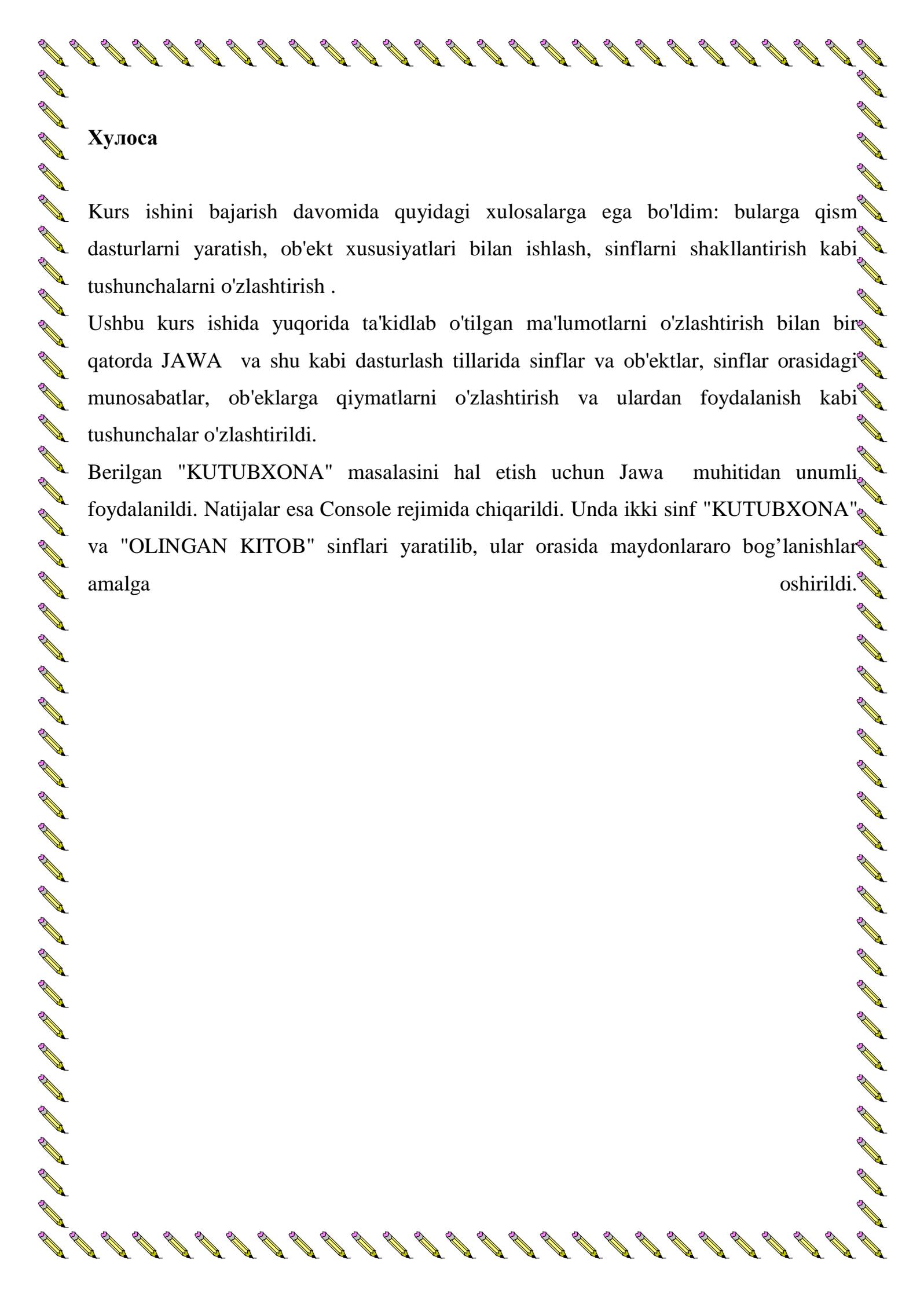
```
class Soccer { //Kutubxona nomi klass hosil qilindi;
2   protected String name;
3   protected String Muallif;
4   protected String soni;
5   protected String yili;
6   protected int narxi; //O'zgaruvchilar e'lon qilindi;
7   public Soccer(String name,String Muallif,String soni,String yili,int narxi) //Kostruktor;
8   {
9       this.name=name;
10      this.Muallif=Muallif;
11      this.soni=soni;
12      this.yili=yili;
13      this.narxi=narxi;
14  };
15  public void show_soccer() //show() nomli funksiya e'lon qilindi;
16  {
17      System.out.println("Komandalar : " +name);
18      System.out.println("O`ynash vaqti : " +Muallif);
19      System.out.println("Stadion nomi : " +soni);
20      System.out.println("Stadion sig`imi : " +yili);
21      System.out.println("Bilet narxi : " +narxi);
22  }
```

The `Output - Sirojiddinov Bobur (run)` window shows the following output:

```
run:
Komandalar : Bunyodkor-Paxtakor
O`ynash vaqti : 13.10.13 17:00
Stadion nomi : Paxtakor
Stadion sig`imi : 18567
Bilet narxi : 5000
Umumiy hisob : 3-1
BUILD SUCCESSFUL (total time: 0 seconds)
```

Output - Sirojiddinov Bobur (run)

```
run:
Komandalar : Bunyodkor-Paxtakor
O`ynash vaqti : 13.10.13 17:00
Stadion nomi : Paxtakor
Stadion sig`imi : 18567
Bilet narxi : 5000
Umumiy hisob : 3-1
BUILD SUCCESSFUL (total time: 0 seconds)
```

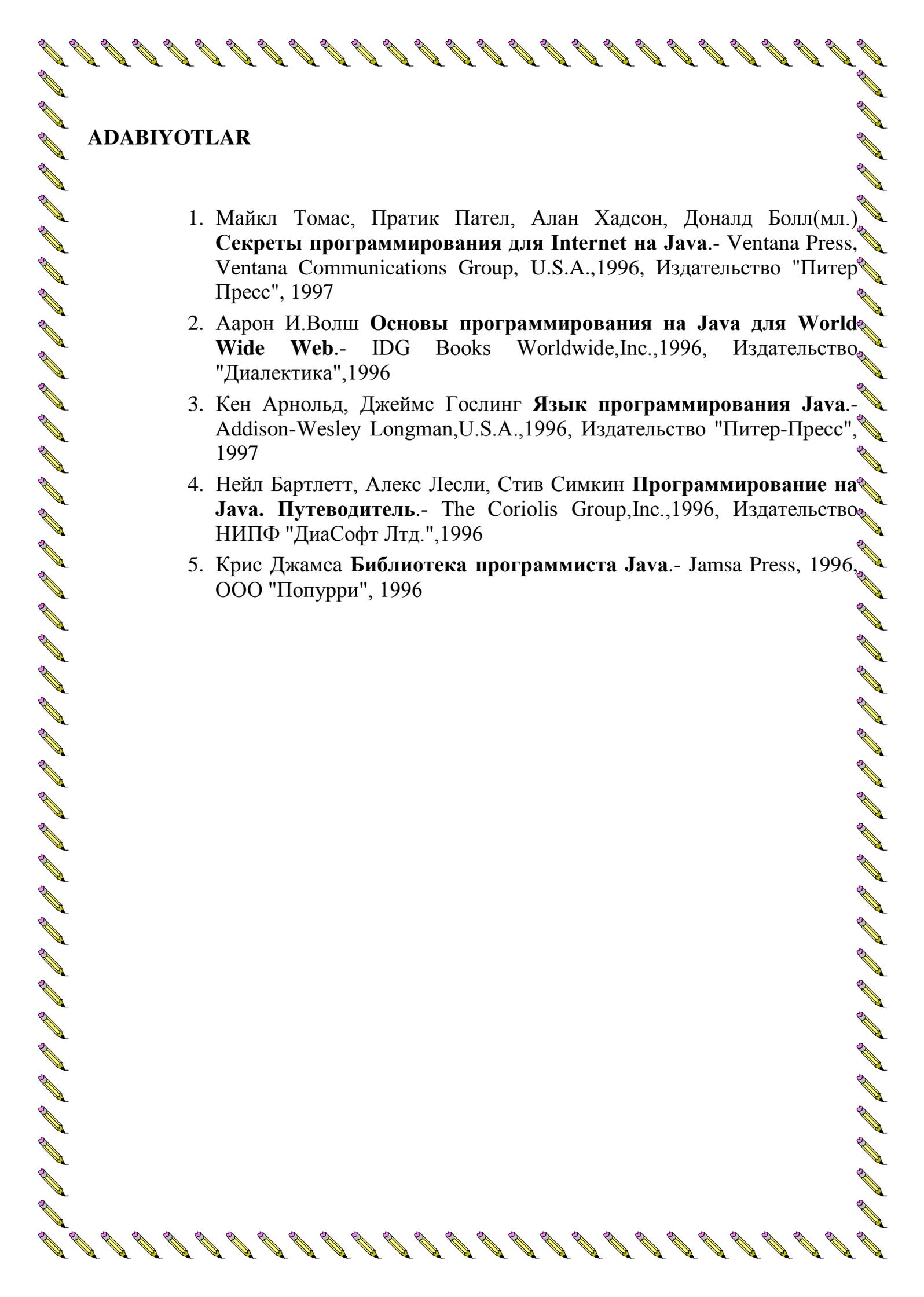


Хулоса

Kurs ishini bajarish davomida quyidagi xulosalarga ega bo'ldim: bularga qism dasturlarni yaratish, ob'ekt xususiyatlari bilan ishlash, sinflarni shakllantirish kabi tushunchalarni o'zlashtirish .

Ushbu kurs ishida yuqorida ta'kidlab o'tilgan ma'lumotlarni o'zlashtirish bilan bir qatorda JAWA va shu kabi dasturlash tillarida sinflar va ob'ektlar, sinflar orasidagi munosabatlar, ob'ektlarga qiymatlarni o'zlashtirish va ulardan foydalanish kabi tushunchalar o'zlashtirildi.

Berilgan "KUTUBXONA" masalasini hal etish uchun Jawa muhitidan unumli foydalanildi. Natijalar esa Console rejimida chiqarildi. Unda ikki sinf "KUTUBXONA" va "OLINGAN KITOB" sinflari yaratilib, ular orasida maydonlararo bog'lanishlar amalga oshirildi.



ADABIYOTLAR

1. Майкл Томас, Пратик Пател, Алан Хадсон, Доналд Болл(мл.) **Секреты программирования для Internet на Java.**- Ventana Press, Ventana Communications Group, U.S.A.,1996, Издательство "Питер-Пресс", 1997
2. Аарон И.Волш **Основы программирования на Java для World Wide Web.**- IDG Books Worldwide,Inc.,1996, Издательство "Диалектика",1996
3. Кен Арнольд, Джеймс Гослинг **Язык программирования Java.**- Addison-Wesley Longman,U.S.A.,1996, Издательство "Питер-Пресс", 1997
4. Нейл Бартлетт, Алекс Лесли, Стив Симкин **Программирование на Java. Путеводитель.**- The Coriolis Group,Inc.,1996, Издательство НИПФ "ДиаСофт Лтд.",1996
5. Крис Джамса **Библиотека программиста Java.**- Jamsa Press, 1996, ООО "Попурри", 1996