O‘ZBEKISTON RESPUBLIKASI AXBOROT TEXNOLOGIYALARI VA KOMMUNIKASIYALARINI RIVOJLANTIRISH VAZIRLIGI

MUHAMMAD AL-XOPAZMIY NOMIDAGI TOSHKENT AXBOROT TEXNOLOGIYALARI UNIVERSITETI
NUKUS FILIALI

“Himoyaga ruxsat etildi”
“Dasturiy injiniringi”
kafedrasi mudiri _____
prof. N.Uteuliev
«____» _____2019y

The searching the best android

mavzusida

BITIRUV MALAKAVIY ISHI

Bitiruvchi _____ D.Ruzmetov
Imzo

Rahbar _____ Q.Yuldashev
Imzo

Taqrizchi _____
Imzo

NUKUS – 2019

MUNDARIJA

KIRISH

XXI asr – axborot texnologiyalar asri ekanligini e‘tiborga olgan holda vatanimiz yoshlarini har tomonlama barkamol shaxslar etib shakllantirish borasidagi keng ko‘lamli chora-tadbirlar kompleksi amalga oshirish maqsadida 2019-yil respublikamizda ―Faol investitsiyalar va ijtimoiy rivojlanish deb e‘lon qilindi. Dasturda har tomonlama tadbirkorlikni va barkamol yosh avlodni shakllantirishni ta‘minlash bo‘yicha qabul qilingan davlat dasturlari hamda boshqa tadbirlarga muvofiq amalga oshiriladigan chora-tadbirlarni davom ettirish bilan bir qator ustivor yo‘nalishlarning eng muhim vazifalari belgilanib berilgan.

Hozirgi davrda boshqa texnika vositalari qatorida qo‘l telefonlari ham takomillashib, murakkablashib, mukammallashib boryapti. Bunda foydalanuvchilar soni esa kundan kunga jadallik bilan oshib boryapti. Yosh boladan tortib, keksa kishilargacha qo‘l telefon xizmatlaridan foydalanishmoqda.

Hozirda kompyuterlar bilan raqobatlasha oladigan darajada qo‘l telefonlari kuchayib ketdi. Ularda ham o‘zining maxsus operatsion tizimi, tezkor xotirasi, protsessori mavjud bo'lib, foydalanuvchining doimiy yordamchisiga aylanib qoldi.

Bularga misol qilib Blackberry, Apple iPhone, Samsung, HTC, ZTE, LG, Artel kabi yirik kompaniyalar ishlab chiqarayotgan smartfonlar va planshetlarni keltirishimiz mumkin. Ulardagi operasion tizimlar ham sun‘iy intellektga qarab takomillashib bormoqda. Shu jumladan, yaqin yillarda o‘z faoliyatini boshlagan va hozirgi kunda smartfon bozorining 79% egallashga ulgurgan Android operatsion tizimi yuqori ko'rsatkichlarni namoyish qilmoqda. Bu degani, ushbu OTdan aholining 60-70% foydalanadi. Operatsion tizim foydalanuvchilari qanchalik ko‘payib borsa, unda ishlaydigan dasturlarga bo'ladigan ehtiyoj ham shunchalik kattalashadi. Demak, O‘zbekistonda ham Android tizimi uchun milliy-zamonaviy dasturlar ishlab chiqish axborot kommunikatsiya texnologiyalari sohasidagi barcha dasturlovchilar uchun birinchi darajali masalalardan hisoblanadi.

Ishning maqsadi, mobil qurilmalar qurilmalar sotuvchi do’konlarning bazasini yig’ish orqali xaridor biror-bir qurilmani bozorlarga borib yurmasdan

to’g’ridan-to’g’ri internetdan ko’rishi va undan keyin o’zi tanlagan do’kondan ushbu xaridni amalga oshirishdan iborat.

Ishning muhimligi va dolzarbligi ushbu mobil ilova yordamida ko’pchilik xaridorlarning qimmatli vaqtini tejash hamda tadbirkorlar orasida sog’lom raqobatni shakllantirishdan iborat hisoblanadi. .

Mazkur bitiruv malakaviy ishi kirish, 2 bob, xulosa va foydalangan adabiyotlarlarda tashkil topgan. Shuningdek bitiruv malakaviy ishida 17 ta rasmdan foydalanildi.

Birinchi bobda mobil qurilmalar amaliyot tizimlari va Android amaliyot tizimi, uning afzalliklar va kamchiliklari haqida to’xtalib o’tildi.

Ikkinchi bobda hozirgi kunda mobil ilovalar yaratish uchun keng qo’llanildigan dasturlash tillari va zamonaviy muhitlar, vositlar haqida to’xtalib o’tildi.

Xulosa qilib aytgan, ushbu bitiruv malakaviy ishida ko’rib chiqilgan mobil ilova yordamida foydalanuvchilarning vaqtini tejash, tejalgan vaqtni esa boshqa zamonaviy bilimlarni egallashga sarflashga ko’maklashish hamda tadbirkorlar orasida sog’lom raqobat muhitini shakllantirish orqali ularning jahon bozorlariga chiqishlariga yo’l ochib berishdan iborat.

I BOB. MOBIL QURILMALAR AMALIYOT TIZIMLARI. ANDROID PLATFORMASI

1.1. Android amaliyot tizimi va uning versiyalari

Android - Linux yadrosiga asoslangan mobil qurilmalar uchun amaliyot tizimi bo'lib, u hozirda mobil qurilmalar va planshetlar uchun engmashhur amaliyot tizimi hisoblanadi. Bundan tashqari, ushbu amaliyot tizimi smart soatlar, elektron kitoblar, musiqa pleyerlari va netbuklarda ham foydalaniladi. Ushbu amaliyot tizimi to'liq egasi Google korporatsiyasi hisoblanadi.

Android amaliyot tizimi tarixi 2005 yildan boshlanadi. Aynan o'sha yili Google ushbu amaliyot tizimi yaratuvchisi Android inc kompaniyasini sotib oldi. 2007 yilda mobil telefonlar ishlab chiqarish standarti ishlab chiquvchi kompaniyalar ittifoqi Open Handset Alliance tashkil etildi.

2008 yilda Google rasmiy ravishda amaliyot tizimining birinchi versiyasini Android 1.0 taqdim qildi. Amaliyot tizimining har bir versiyasi o'zining takrorlanmas nomiga ega edi. Birinchi versiyaning nomi «Apple Pie» (Olmali pirog) nomini oldi. Aynan o'sha yili ushbu qurilmada ishlovchi birinchi qurilma HTC kompaniyasi tomonidan HTC Dream ishlab chiqildi. Shundan keyin amaliyot tizimining yangi versiyalari bir yilda bir necha marta chiqa boshladi. Shu yilning o'zida amaliyot tizimining ozgina o'zgarishlari bilan 1.1, 1.5 va 1.6 versiyalar chiqdi. 2009 yilda amaliyot tizimining 2.0 versiyasi «Eclair» ishlab chiqildi.

2010 yilda 2.2 versiyasi «Floyo» nomi ostida taqdim qilindi. Amaliyot tizimning 2.3 versiyasi esa bir vaqtning o'zida bir nechta kameradan foydalanish imkoniga ega edi.

2011 yil 3.0 versiyasi «Honeycomb» nomi ostida chiqdi. Ushbu versiyaning oldingi versiyalardan asosiy afzalliklari video qo'ng'iroqlarini bajarish imkoniyati, ko'p vazifali va planshetlarga ham o'rnatish imkoniyatining mavjudligi edi. Shu yilning o'zida amaliyot tizimining keyingi versiyasi «Ice Cream Sandwich» nomi ostida taqdim qilindi. Ushbu versiyada ishlab chiqaruvchilar katta o'zgarishlarga qo'l urishdi. Jumladan, barcha Android qurilmalar uchun yagona qobiq ishlab chiqildi. Bundan tashqari, internet trafigini boshqarish imkoniyati, Wi-Fi tarmog'i

orqali ma’lumot almashinish imkoniyati, jildlar yaratish, kamera bilan ishlash imkoniyati kengaytirildi va boshqa afzalliklarni ko’rsatish mumkin.

2012 yilda «Jelly Bean» nomi ostida amaliyot tizimining 4.1 versiyasi taqdim qilindi. Shu yilning o’zida ozgina qo’shimchalari bilan 4.1.2, 4.2 va 4.3 versiyalar ham taqdim qilindi.

2013 yilda «Kit-Kat» nomi ostida amaliyot tizimining 4.4 versiyasi taqdim qilindi. 2014 yilda ayrim qo’shimchalari bilan amaliyot tizimining 4.4.3 versiyasi taqdim qilindi. Shu yilning o’zida Android 5 «Lollipop» versiyasi taqdim qilindi. 2015 yilda Android 6 «Marshmallow», 2016 yilda Android 7 «Nougat», 2017 yilda Android 8 «Orea» taqdim qilindi. Ushbu amaliyot tizimining eng so’nggi versiyasi 2018 yilda Android 9 «Pie» brendi ostida taqdim qilindi.

Android amaliyot tizimining afzalliklari sifatida quyidagilarni sanab o’tish mumkin. Bular:

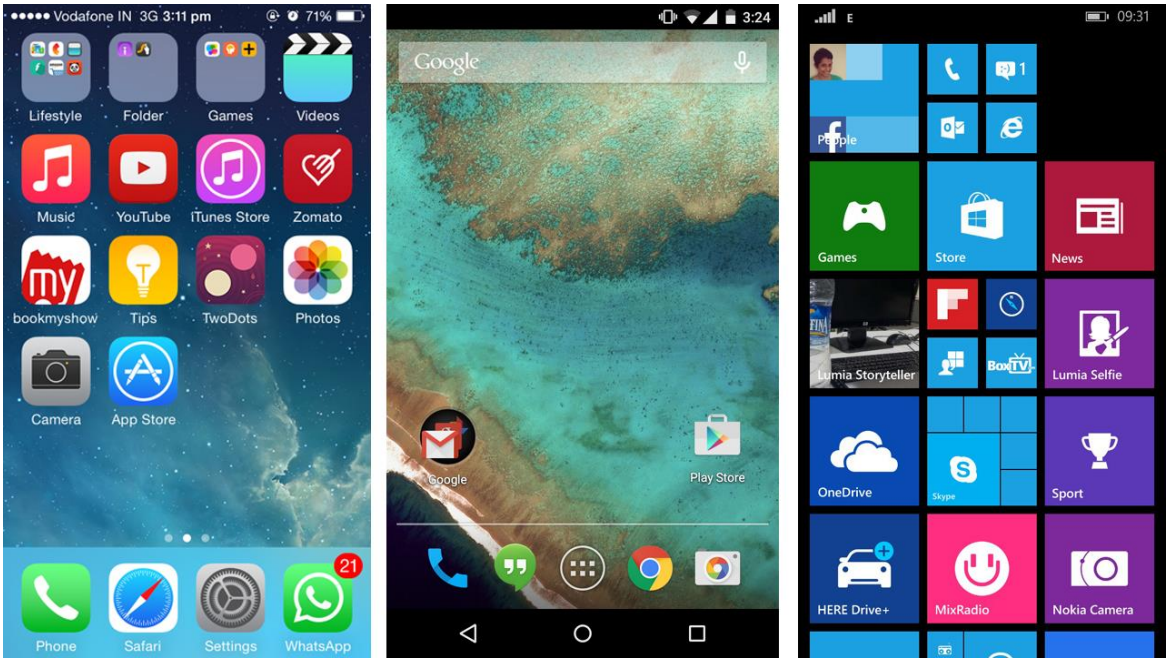
- Android tizimidagi qurilmalar bilan oson sinxronizatsiya qilish;
- Ochiqlilik;
- PlayMarketdan tashqari turli boshqa ilovalar magazinlarining mavjudligi;
- Bluetooth tizimining to’liq qo’llab-quvvatlanishi ;
- SD-kartalarini qo’llab-quvvatlanishi. Ushbu imkoniyat Apple smartfonlarida mavjud emas;
- Android amaliyot tizimida smartfonlar sonining ko’pligi. Sababi, ushbu amaliyot tizimida byudjet va premium sinfidagi qurilmalarni topish mumkin.

1.2. Mobil amaliyot tizimlarining qiyosiy tahlili

Hozirgi kunda Android amaliyot tizimi mobil qurilmalar uchun foydalaniladigan amaliyot tizimlari ichida so’nggi vaqtda yetakchilik qilib kelmoqda. Google kompaniyasi ma’lumotiga ko’ra dunyodagi barcha qurilmalarning 80% Android amaliyot tizimida foydalanib kelmoqda. Apple kompaniyasining iOS amaliyot tizimi ko’p yillar davomida o’zining qulaylilik, barqarorlik va ajoyib dizayni bilan yetakchilik qilib kelgan edi.

Android 8 Oreo o'zidan oldingi versiyalarga umuman o'xshamagan foydalanuvchi interfeysida taqdim qilindi. Yangilanish ko'pligidan Google kompaniyasi hatto interfeysni Material Designga o'zgartirishiga ham to'g'ri keldi. Yangi tizimda ikonkalar tekis va ranglari ham yangi qiyofa aks etdi.

Applening iOS 8 da o'zgarishlar unchalik katta emas. Ular o'zlarining mashhur iOS 7 versiyasiga sodiq qolishmoqda. Ushbu amaliyot tizimidagi tekis dizayn (Flat design) foydalanuvchi uchun tushunarli, ko'p funksiyali va animatsiyalari esa foydalanuvchilarga juda yoqimlidir.



Microsoft esa interfeysda katta o’zgarishlarga qo’l urmadi. Lekin, keyingi versiyalarda boshlang’ich ekranning fon rasmini o’zgartirish mumkin. Yangi jild yaratish imkoniyati ham qo’shilgan. Yangi papka yaratish uchun esa Android va iOS amaliyot tizimlaridek dasturni boshqa dasturga tekizilsa yangi papka hosil bo’ladi.

Mobil qurilmalar uchun amaliyot tizimini tanlash juda muhim sanaladi. Sababi, ko’pchilik insonlar ma’lum bir smartfonlar turini tanlashadi va o’sha qurilmadagi amaliyot tizimidan foydalanishadi. Bizning maqsad hozirda keng foydalanib kelinayotgan amaliyot tizimlarini tahlil qilish orqali ularning orasidan eng yaxshisini tanlashdan iborat.

Smartfonlar oddiy telefonlarga qaraganda amaliyot tizimining rivojlanganligi, smartfon imkoniyatlarini kengaytirish uchun boshqa dasturchilar tomonidan yaratilgan dasturlarni o’rnatish imkoniyatining mavjudligi bilan katta farq qiladi. Quyidagi jadvalda kichik dasturchilar guruhi tomonidan o’tkazilgan ijtimoiy so’rovnoma asosida tuzilgan jadval keltirilgan.

	Android	Symbian	iOS	Windows Phone
Ishlash tezligi	10	5	10	6
Xavfsizligi	8	6	10	5
Dasturchilar orasida mashhurligi	9	6	5	7
Narxi	7	9	3	8
Rivojlanishi	9	6	9	7
Jami	**43**	**32**	**37**	**33**

Hozirgi kunda smartfon va boshqa kommunikatsion qurilmalarda turli amaliyot tizimlari foydalaniladi. Lekin, eng mashhurlari aynan Android, Symbian, iOS va Windows Phonelar hisoblanadi. Yuqorida sanab o’tilgan har bir platformaning raqobatchi platformalarga qaraganda o’z afzalliklari bilan birgalikda kamchiliklari ham mavjud. Masalan, ayrim platforma uchun ilovalar juda ko’p bo’lsa, boshqa birisida foydalanuvchi interfeysi juda qulay yana boshqasi esa oddiy foydalanuvchi tushunmaydigan darajada murakkablashib yuborilgan.

Yuqorida jadvaldan kelib chiqib birinchi o’rinni Android 43 ball bilan qo’lgan kiritgan bo’lsa, ikkinchi o’ringa 37 ball bilan iOS egalladi. Eng past ballni Windows

Phone va Symbian amliyot tizimlari egalladi. Endi jadvaldagi ma’lumotlarni tahlil qilgan holda amaliyot tizimlarining eng kuchli taraflariga to’xtalsak.

Amaliyot tizimlarining eng kuchli tarafi bu uning tezligida ekan. Tezlik umumiy ballning 36%ni tashkil qildi. Bundan tashqari, smartfonlarning rivojlanishi (19%) uning xavfsizligi (17%) va mashhurligi (11%) ni tashkil qildi.

Symbian OS

Nokia kompaniyasi tomonidan qo’llab-quvvatlangan ushbu amaliyot tizimi mobil qurilmalar orasida mashhurlaridan biri hisoblanadi. Ushbu AT qurilmada katta hajm egallamaydi, tizim yadrosi va grafik interfeysi bir-biridan ajratilgan. Keyinchalik ushbu ATning Series 80, Series 90, MOAP va boshqa versiyalari ishlab chiqildi. Ushbu versiyalarning o’ziga xos xususiyatlari mavjud bo’lib, har bir versiya uchun alohida dastur yozish kerak bo’lardi. iPhoneOS, Android, Windows Mobile ATlari ishlab chiqilgandan keyin SymbianOS o’zining mashhurligini yo’qotdi. Hozirgi kunda esa faqatgina Nokia tomonidan ishlab chiqilayotgan qurilmalarda ushbu amaliyot tizimi qo’llanilmoqda.

Afzalliklari:

- Barqarorlik;
- Ilovalarning ko’pligi;
- Protsessor xotirasida kichik hajm egallaydi.

Kamchiliklari:

- Kompyuter bilan ulash uchun alohida dastur zarur bo’ladi;
- Eski va yangi versiyalari bir-birlariga mos emas.

Windows Phone

Android - o’z raqobatchilariga qaraganda ancha yosh amaliyot tizimi hisoblanadi. Ushbu amaliyo tizimi Google kompaniyasi tomonidan Open Handset Alliance tomonidan ishlab chiqilgan. ATning kodlari ochiq kodli ma’lumotlarda saqlanganligi tufayli har bir dasturchi o’zining shaxsiy ATni ishlab chiqishi mumkin. Play Marketdan pullik hamda bepul ilovalarni yuklab olish mumkin.

Afzalliklari:

- Ochiq kodli;

- Yuqori samaradorlik;
- Ko’p vazifalik;
- Ilovalarning ko’pligi;
- Google xizmatlari bilan munosabatlar.

Kamchiligi:

- Xakerlik hujumlariga moyillik;
- Ko’pchilik mobil qurilmalar uchun AT yangi versiyasi keyinroq chiqadi, shu sababli ishlab chiqaruvchilar eski amaliyot tizimlariga mos dasturlar yozishga majbur bo’ladi.

iPhoneOS

Apple kompaniyasi tomonidan ishlab chiqilgan mobil amaliyot tizimi hisoblanadi. Faqatgina ushbu kompaniya tomonidan ishlab chiqilgan qurilmalarda qo’llab-quvvatlaydi. Masalan, iPhone, iPad, iPod.

Afzalliklari:

- Yangilanish tez-tez taqdim qilinadi;
- Qulay menyu;
- Sifatli qo’llab-quvvatlash xizmati.

Kamchiliklari:

- Ko’p vazifalikning mavjud emasligi;
- O’zining o’rnatilgan matn tahrirlovchisining mavjud emasligi;
- AT bloklangan xarakteristikasi.

II BOB. MOBIL ILOVALAR YARATISH UCHUN DASTURLASH TILLARI VA MUHITLARI

2.1. JAVA dasturlash tili va uning sintaksisi

Java dasturlash tili paydo bo'lishidan oldin dasturlar aniq bir operatsion tizim boshqaruvi ostida ishlaydigan qilib yaratilgan. Ma'lum bir operatsion tizim uchun yaratilgan dastur boshqa operatsion tizimida ishlay olmagan. Java dasturlash tili ushbu tushunchani o'zgartirib operatsion tizimga bog'liq bo'lmagan dasturlashni kiritdi. Java dasturlash tili zamonaviy dasturlash tili taminlab berishi kerak bo'lgan barcha funktsiyalarni o'z ichiga olganligi uchun qisqa vaqt ichida dasturchilar orasida keng tarqaldi. Jumladan, Java dasturlash tili quyidagi xususiyatlarga ega:

- Ob'ektlarga ixtisoslashish xususiyatlari
- Ko'pvazifalilik
- Avtomatik xotira menejmenti (keraksiz ob'ektlarni yig'ishtirish)
- Tarmoq va xavfsizlik xususiyatlari
- Platformaga bog'liq emaslik

Java dasturlash tili nisbatan yangi dasturlash tili xisoblansada ayni vaqtda boshqa dasturlash tillari singari mavjud imkoniyatlari bilan bir o'rinda turadi.

Java dasturlash tili kompilyatsiya va interpretatsiya qilinuvchi til xisoblanadi. Java birlamchi kodi avval kompilyator yordamida sodda ikkilik instruktsi, bayt-kodga o'giriladi. Ushbu bayt- kod Java virtual mashinasi uchun dastur amallarini bajarish uchun ko'rsatma bo'lib xizmat qiladi. Bayt-kod universal bo'lib turli platformalar uchun maxsus yaratilgan barcha Java virtual mashinalari yordamida bajarilishi mumkin. Bu o'z navbatida Java dasturini platformaga bog'liq bo'lmasligini ta'minlaydi.

Java dasturlash tilida dastur tuzish va uni ishga tushirish jarayonini quyidagi misol yordamida ko'rib o'tamiz. Ushbu misolda keltirilgan kod sintaksisi va operatorlariga emas, balki dastur tuzish va ishga tushirish jarayoniga e'tibor qilamiz. Shartli ravishda dasturlash jarayonini 3 bosqichga bo'lib olamiz:

1-bosqich. Dasturchi dastur birlamchi kodini tuzib

Hello.java fayliga saqlaydi.

Birlamchi kodni odatda biron-bir sodda matn muharriri yordamida kiritiladi, masalan Windows operatsion tizimining Блокнот dasturi.


```
public class HelloWorld {
public static void main(String[] args){
System.out.println("Hello World!"); } }
```

2-bosqich. Hello.java faylini javac Java kompilyatori yordamida bayt-kodga o’giriladi.

3-bosqich. Hello.class bayt-kodni interpretator, Java virtual mashinasi yordamida ishga tushirish.

```
c:\>java Hello Hello World!
```

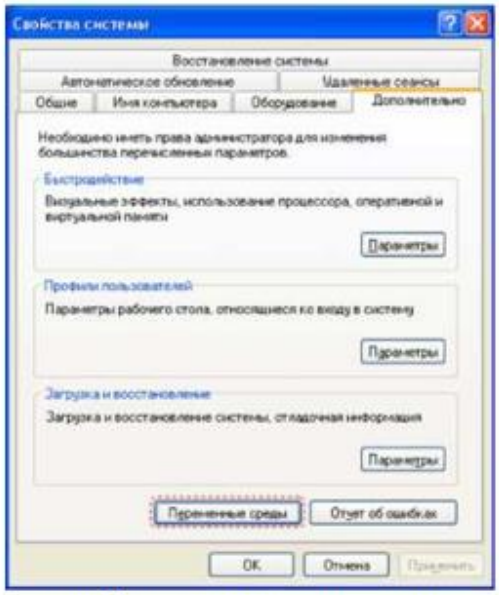
Java Dasturlash Vositasini o’rnatish

Javada dasturlash uchun avval Java Dasturlash Vositasini kompyuterga o’rnatish kerak bo’ladi. Xozirgi paytda eng to’liq va oxirgi versiyalari Solaris, Linux va Windows operatsion tizimlari uchun mavjud. Java Dasturlash Vositasini kerakli operatsion tizim uchun http://java.sun.com manzili orqali tortib olish mumkin. Ushbu qo’llanmada Windows operatsion tizimi asosiy platforma sifatida ishlatilgan va qo’llanma yozilishi paytida Javaning so’nggi versiyasi 1.6 bo’lgan.

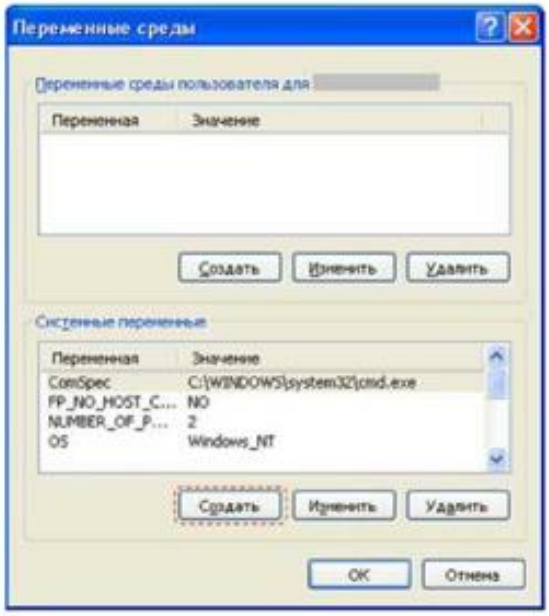
Tortib olingan Java Dasturlash Vositasini kompyuterga quyidagi papkaga o’rnatamiz:

```
C: \Program Files\Java\jdk1.6
```

Kommanda satri orqali Java kompilyatori va interpretatoridan foydalanish uchun Windowsni muhit o’zgaruvchilarini belgilash kerak bo’ladi. Buning uchun sistema xususiyatlari oynasidan shu nomli tugmani tanlash kerak bo’ladi.

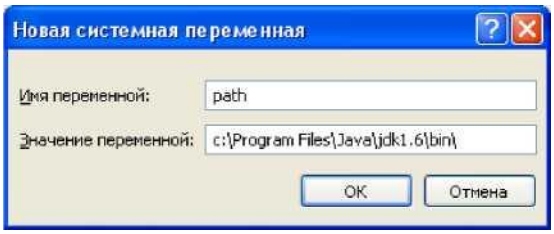


Windows operatsion tizimining xususiyatlar oynasi Natijada muhit o'zgaruvchilari oynasi ochiladi.



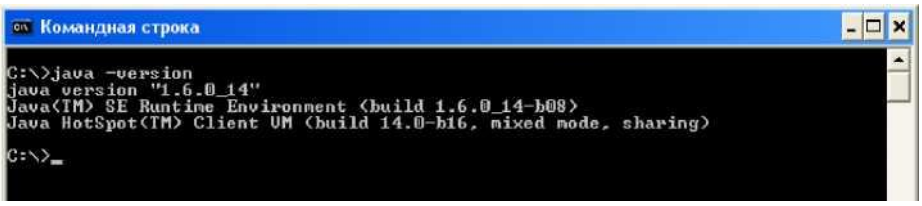
Muhit o'zgaruvchilari oynasi

Ushbu oynada Создать tugmasi orqali quyidagi yangi muhit o'zgaruvchisini kiritib olamiz: path va bizning xolimizda uning qiymatini c:\Program Files\Java\jdk1.6\bin\ deb belgilaymiz.



Yangi muhit o'zgaruvchisini yaratish dialog oynasi

Qiymatlarni to'g'ri kiritilganligini komanda satridan quyidagi buyruq yordamida tekshirib olishimiz mumkin:



Java versiyasi haqida ma'lumot olish

Dasturlash muhitini tanlash

Java dasturlash tilida dastur yozish uchun quyidagi uch xil muhitdan foydalanish mumkin:

Java dasturlash vositasi va biron-bir matn muxarriri (masalan, Notepad);

Integrallashgan Dasturlash Muhitidan foydalanish (masalan, Eclipse);

Java Dasturlash Vositasi va u bilan integrallashgan matn muxarriridan foydalanish (masalan, TextPad);

Integrallashgan Dasturlash Muhitlari asosan ko'plab birlamchi kod fayllaridan iborat dasturlar bilan ishlash uchun mo'ljallangan. Ushbu dasturlash muhitlari dasturlashni osonlashtirish uchun ko'plab uskunalarni o'z ichiga oladi, masalan xatoliklarni aniqlovchi (debugger) va versiyalarni boshqaruvchi tizimlar.

Sodda dasturlarni tuzish uchun biror-bir matn muxarriri yoki Java Dasturlash Vositasi bilan integrallashgan matn muxarriri qo'llash mumkin. Birinchi xolatda dasturning birlamchi kodi matn muxarriri yordamida kiritiladi va komanda qatori orqali kompilyatsiya qilinadi va ishga tushiriladi. Ikkinchi xolatda dasturning birlamchi kodi integrallashgan matn muxarriri yordamida kiritiladi va muxarrir ichida kompilyatsiya qilinadi va ishga tushiriladi.

Sodda Java dasturi

Java dasturlash tilida yozilgan, konsol oynasiga matnni chiqarib beruvchi quyidagi sodda dastur kodini ko'rib chiqamiz:

```
public class HelloWorld {
public static void main(String[] args){
System.out.println("Hello World!"); } }
```

Ushbu dasturni kodini matn muxarriri yordamida kiritib HelloWorld.java fayliga saqlaymiz. So'ngra javac kompilyatori yordamida kommanda satrida dasturni kompilyatsiya qilamiz:

Java kompilyatori yordamida birlamchi kodni kompilyatsiya qilish

Natijada HelloWorldjava fayli joylashgan joyda HelloWorld.class nomli fayl xosil bo'ladi. So'ngra java buyrug'i yordamida Java Virtual Mashinasini ishga tushiriladi va HelloWorld.class faylida joylashgan baytkod bajariladi.

Java Virtual Mashinasi yordamida baytkodni ishga tushirish

Korib turilganidek dastur konsol oynasiga “Hello World!" matnini chiqarib berdi. Endi ushbu dasturning birlamchi kodini taxlillab chiqamiz. Java dasturlash tilida barcha dastur elementlari class ichida joylashadi. Bizning xolatda class quyidagicha e’lon qilingan:

```
public class HelloWorld{
}
```

Bu yerda public kalit so’zi ushbu klassni murojaat o’zgartiruvchisi xisoblanib u ushbu class elementlariga murojaat darajasini belgilaydi. HelloWorld ushbu klass nomini belgilaydi. Java dasturlash tilida class nomi xarf bilan boshlanishi va undan keyin xarf va sonlar kombinatsiyasidan iborat bo’lishi mumkin. Klass nomi sifatida Java kalit so’zlarini ishlatib bo’lmaydi. Bundan tashqari klass joylashgan fayl nomi public murojaat o’zgartiruvchili klass nomi bilan bir xil bo’lishi va java kengaytmasiga ega bo’lishi kerak. Xususan, bizning sodda dasturimiz joylashgan fayl nomi HelloWorldjava bo’ladi.

Java dasturini ishga tushirish uchun Java virtual mashinasi xar doim dasturning main metodida joylashgan dastur kodini bajarishdan boshlaydi. Demak, klass ishga tushirilishi uchun unda main metodi mavjud bo’lishi kerak. Yuqoridagi misolda main metodi quyidagi ko’rinishga ega:

```
public static void main(String[] args){
     System.out.println("Hello World!");
}
```

Bu yerda, main metodida joylashgan `System.out.println("Hello World!");` qatori konsolga “`Hello World!`” matnini chiqarib beradi.

Eng birinchi oddiy dasturimizni yaratish uchun talab qilinadigan dasturlar

• JDK o’rnatiladi agar bo’lmasa, (JDK ni yuklab oling va uni o’rnating)

• dasturlash muhiti eclipse yokiy netbeans

• java dasturi yaratiladi

• dasturni compile (kompayl) qilinadi

```
class Simple{
     public static void main(String args[]){
     System.out.println("Hello Java");
     }
}
```

```
Ekranda :Hello Java
```

Eng birinchi oddiy dasturimizni yaratish uchun talab qilinadigan dasturlar

• JDK o’rnatiladi agar bo’lmasa, (JDK ni yuklab oling va uni o’rnating)

• dasturlash muhiti eclipse yokiy netbeans

• java dasturi yaratiladi

• dasturni compile (kompayl) qilinadi

```
class Simple{
public static void main(String args[]){
System.out.println("Hello Java");
}
}
Ekranda :Hello Java
```

Java birlamchi kod strukturasi va main() metodi

Java dasturlash tilida birlamchi kod birlamchi kod fayliga kiritilib bitta klassni ifodalaydi. Klass dasturning bir qismini namoyon etadi, ba’zi xollarda kichik dastur bitta klassdan iborat bo’lishi mumkin. Klass figurali qavslar ichida yozilishi kerak.

```
public class Programl {
//dastur kodi
}
```

Klass odatda bir necha metodlarni o’z ichiga olishi mumkin. Metodlar ushbu klassga doir amallarni bajarish ko’rsatmalaridan tashkil topadi. Metodlar klass ichida, ya’ni figurali qavslar orasida, e’lon qilinishi kerak.

```
public class Programl {
void go() {
//metod kodi
} }
```

Metodning figurali qavslari ichiga ushbu metod bajarilishi kerak bo’lgan ko’rsatmalar kiritiladi. Metod odatda bir-necha ifodalar to’plamidan tashkil topadi.

Umumlashtirib aytganda, klass birlamchi kod fayliga kiritiladi. Metodlar klass ichiga kiritiladi. Ifodalar metodlar ichiga kiritiladi.

Java virtual mashinasi ishga tushganda u bajarayotgan klassda joylashgan quyidagi metodni izlaydi:

```
public static void main(String[] args) {
//metod kodi
```

```
}
```

Keyin esa ushbu metodning figurali qavslari ichida joylashgan barcha instruktsiyalarni bajaradi. Xar-bir Java dasturi kamida bitta klassni va bitta main() metodini o’z ichiga olishi kerak.

class javada kalit so’z Simple klass nomi

public – murojaat huquqi: java dasturlash tilida 4 ta murojaat huquqi bor

1. public – biz yaratgan proyektimizda xohlagan paketdan murojaat qilishimiz mumkin
2. protected – faqat yaratgan paketimizdan murojaat huquqi
3. private – faqat class ichida murojaat huquqi
4. default – boshlang’ich huquq

static -bu yaratgan funksiyamiz umumiyligini bildiradi

void – bu funksiya qiymat qaytarmasligini bildiradi

dasturlashda funksiyalar ikki xil bo’ladi:

- qiymat qaytardigan va
- qiymat qaytarmaydigan void funksilar qiymat qatarmaydi

main – dastur ishlashi bilan main funksiyasiga murojaat qiladi

foydalanish mumkin bo’lgan main funksiya ko’rinishlari:

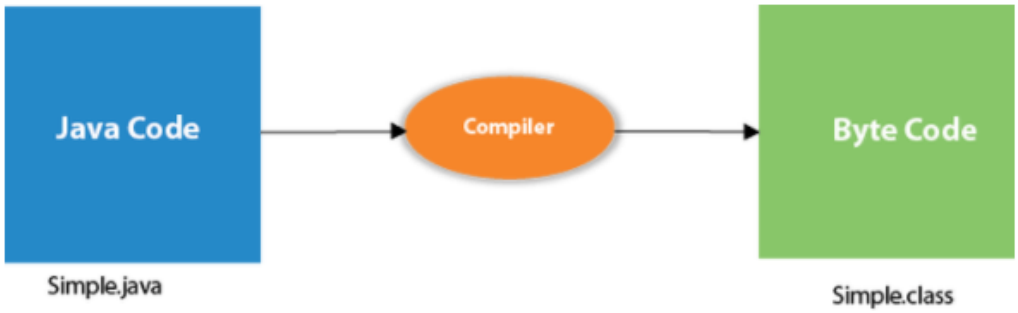
1. public static void main(String[] args)
2. public static void main(String []args)
3. public static void main(String args[])
4. public static void main(String… args)
5. static public void main(String[] args)
6. public static final void main(String[] args)
7. final public static void main(String[] args)
8. final strictfp public static void main(String[] args)

foydalanish mumkin bo’lmagan main funksiya ko’rinishlari:

1. public void main(String[] args)
2. static void main(String[] args)
3. public void static main(String[] args)

4. abstract public static void main(String[] args)

Compile (kompayl) vaqtida java fayl baytcode ga o'giriladi



Dastur bajarilish vaqtida sodir bo'ladigan jarayonlar

Runtime vaqtida nima sodir bo'ladi? runtime vaqtida quydagi qadamlar sodir bo'ladi

Class file

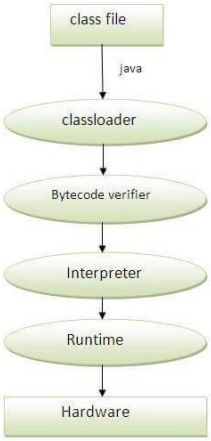
Classloader

Bytecode verifier

Interpreter

Runtime

Hardware



- **Classloader** : JVM (Java virtual mashinasi) ning quyi tizimi hisoblanadi va class fayllarni yuklash uchun ishlatiladi
- **Bytecode Verifier:** obyektga murojaat qilishdagi qonunlarni buzilmaganini kod qismlari uchun tekshiradi.
- **Interpreter:** baytkodni o'qib ko'rsatmalarni amalga oshiradi

Dasturda ishlatiladigan o'zgaruvchi, klass, metod yoki ob'yekt nomlari identifikatorlar deb ataladi. Java identifikatorlari xarflar, sonlar, "$" simvoli va "_" simvolidan tashkil topishi mumkin. Masalan, `test, testl, _test, TEST, $test` barchasi to'g'ri identifikator xisoblanadi. Identifikatorlar tanlashda quyidagi qoidalarga rioya qilish kerak:

- Identifikatorlar harf, "$" simvoli yoki "_" simvoli bilan boshlanishi kerak;

- Birinchi xarfdan keyin identifikator xarflar, “$” simvoli, “_” simvoli yoki sonlarning istalgan kombinatsiyasini o’z ichiga olishi mumkin;
- Identifikator istalgan miqdordagi simvollardan tashkil topishi mimkin;
- Java kalit so’zlarini identifikator sifatida ishlatish mumkin emas;
- Identifikatorlar xarf kattaligini farqlaydi. Masalan, test va Test ikkita xar-xil identifikatorlar.

To’g’ri tuzilgan identifikatorga misollar: _test, $test,__test_2$

juda_batavsil_berilgan_identifikator

Quyidagilar esa noto’g’ri tuzilgan identifikatorlar:

:test, -test, test#, .test, 4test

Quyidagi jadvalda identifikator sifatida qo’llab bo’lmaydigan Java kalit so’zlari berilgan.

Abstract	boolean	break	byte	case	catch
Char	class	const	continue	default	Do
Double	else	extends	final	finally	float
For	goto	if	implements	import	instanceof
Int	interface	long	native	new	package
Private	protected	public	return	short	static
Strictfp	super	switch	syncronized	this	throw
Throws	transient	try	void	volatile	while
Assert	enum				

Yuqorida keltirilgan identifikatorlarga qo’yiladigan talablar bilan birgalikda Java dasturlash tilida identifikatorlarni tuzishda quyidagi tavsiyalar ham beriladi: •

Metod va o’zgaruvchi nomlariga: birinchi xarfi kichik xarf bo’lishi va agar bir necha so’zdan iborat bo’lsa keyingi so’zlar katta xarfdan boshlanishi kerak. Masalan:

balansniTekshirish

xisobIshiniBajarish

Konstantalar nomiga: barcha xarflari katta xarflarda bo’lishi va agar bir necha so’zdan iborat bo’lsa soz’lar “_” simvoli bilan ajratilishi kerak. Masalan:

DARAJA

MIN_NARX

Ushbu tavsiyalar majburiy bo’lmasada ularga amal qilish dasturning birlamchi kodini dasturchi tomonidan o’qilishini osonlashtiriladi.

O’zgaruvchilar

Java dasturlash tilida o’zgaruvchini qo’llashdan oldin uni e’lon qilish kerak bo’ladi. E’lon qilish jarayoinida kompyuterga ushbu o’zgaruvchida qanday turdagi ma’lumot saqlanilishi aytilib o’tiladi. Masalan,

int elementlarSoni;

double elementYuzasi, elementHajmi;

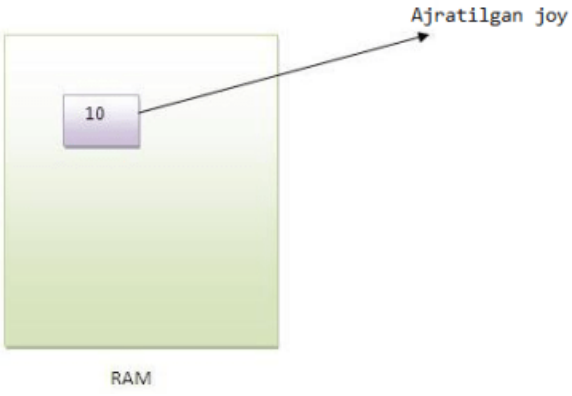
Java dasturlash tili xarflar (char), xar-xil turdagi butun (byte, short, int, long) va ratsional sonlar (float, double), mantiqiy qiymatlarni (boolean) asosiy qiymat turlariga kiritadi. Bu asosiy qiymatlar turi sodda qiymatlar deb ataladi. Quyidagi jadval Java sodda qiymatlarini ko’rsatadi.

Tur nomi	**Qiymati**	**Ishlatadigan xotira**	**Qiymat chegarasi**
boolean	true yoki false	1 byte	
Char	Bitta xarf	2 byte	Barcha Unikod xarflari
Byte	Butun son	1 byte	-128 dan 127 gacha
Short	Butun son	2 byte	-32768 dan 32767 gacha
Int	Butun son	4 byte	-2147483648 dan 2147483647 gacha
Long	Butun son	8 byte	-9223372036854775808 dan 9223372036854775807 gacha
Float	Ratsional son	4 byte	-3.40282347*10+38 dan -1.40239846*10-45 gacha
Double	Ratsional son	8 byte	±1.76769313486231570*10+308 dan ±4.94065645841246544*10-324 gacha

Birinchi misol elementlarSoni o’zgaruvchisi int turdagi qiymatlarni olishini e’lon qiladi. int butun sonlarni anglatadi. Ikkinchi misol elementYuzasi va elementHajmi o’zgaruvchilarni double turdagi qiymatlarni qabul qilishini e’lon

qiladi. double ratsional sonlarni anglatadi. Ushbu misoldan ko'rinadiki, bir turdagi o'zgaruvchilarni vergul bilan ajratib bir qatorda e'lon qilish mumkin.

O'zgaruvchi – xotiradan ajratilgan, himoyalangan maydon nomidir.



Oddiy qilib tushuntiradigan bo'lsak, o'zgaruvchi – ma'lum bir turdagi ma'lumotni o'zida saqlovchi va o'lchami chegaralangan idish. Tushunarliroq bo'lishi uchun bir ikkita hayotiy misollar keltiramiz: meva solish uchun tayyorlangan savatga suv sola olmaymiz o'zgaruvchilar ham shunday bir turdagi o'zgaruvchi uchun ajratilgan joyga boshqa turdagi o'zgaruvchini saqlay olmaymiz.

4 litrlik idishga 5 litr suv quya olmaymiz, chunki idishga 4 litr suv sig'adi. O'z o'zidan kelib chiqadiki 5 litrlik suvni saqlash uchun kattaroq idish tanlashimiz kerak. O'zgaruvchilar ham shunday ma'lumotning o'lchami xotiradan ajratilgan joydan oshib ketsa dastur xatolik yuz beradi.

5 *bayt*lik butun sonni ma'lumot turi *int* bo'lgan o'zgaruvchiga saqlay olmaymiz, chunki int = 4 *bayt*. Bu turdagi ma'lumotni saqlash uchun *long* dan foydalanamiz.

Identifikatorlar

Dasturda ishlatiladigan o'zgaruvchi, klass, metod yoki ob'yekt nomlari identifikatorlar deb ataladi. Java identifikatorlari xarflar, sonlar, "$" simvoli va "_" simvolidan tashkil topishi mumkin. Masalan, `test, testl, _test, TEST, $test` barchasi to'g'ri identifikator xisoblanadi. Identifikatorlar tanlashda quyidagi qoidalarga rioya qilish kerak:

- Identifikatorlar harf, "$" simvoli yoki "_" simvoli bilan boshlanishi kerak;
- Birinchi xarfdan keyin identifikator xarflar, "$" simvoli, "_" simvoli yoki sonlarning istalgan kombinatsiyasini o'z ichiga olishi mumkin;
- Identifikator istalgan miqdordagi simvollardan tashkil topishi mimkin;

- Java kalit so’zlarini identifikator sifatida ishlatish mumkin emas;
- Identifikatorlar xarf kattaligini farqlaydi. Masalan, test va Test ikkita xar-xil identifikatorlar.

O’zgaruvchilar

Java dasturlash tilida o’zgaruvchini qo’llashdan oldin uni e’lon qilish kerak bo’ladi. E’lon qilish jarayonida kompyuterga ushbu o’zgaruvchida qanday turdagi ma’lumot saqlanilishi aytilib o’tiladi. Masalan,

```
int elementlarSoni;
double elementYuzasi, elementHajmi;
```

Java dasturlash tili xarflar (char), xar-xil turdagi butun (byte, short, int, long) va ratsional sonlar (float, double), mantiqiy qiymatlarni (boolean) asosiy qiymat turlariga kiritadi. Bu asosiy qiymatlar turi sodda qiymatlar deb ataladi. Quyidagi jadval Java sodda qiymatlarini ko’rsatadi.

O’zgaruvchi turlari

Javada 3 ta o’zgaruvchilar turi mavjud:

1. local variable
2. instance variable
3. static variable

- local variable – funksiya ichida e’lon qilinadi va bu o’zgaruvchilar lokal (mahalliy) o’zgaruvchilar deyiladi.
- instance variable – class ichida e’lon qilinadi
- static variable – static deb e’lon qilingan o’zgaruvchi static o’zgaruvchi deyiladi. Bu local(mahalliy) bo’lishi mumkin emas.

Misol uchun:

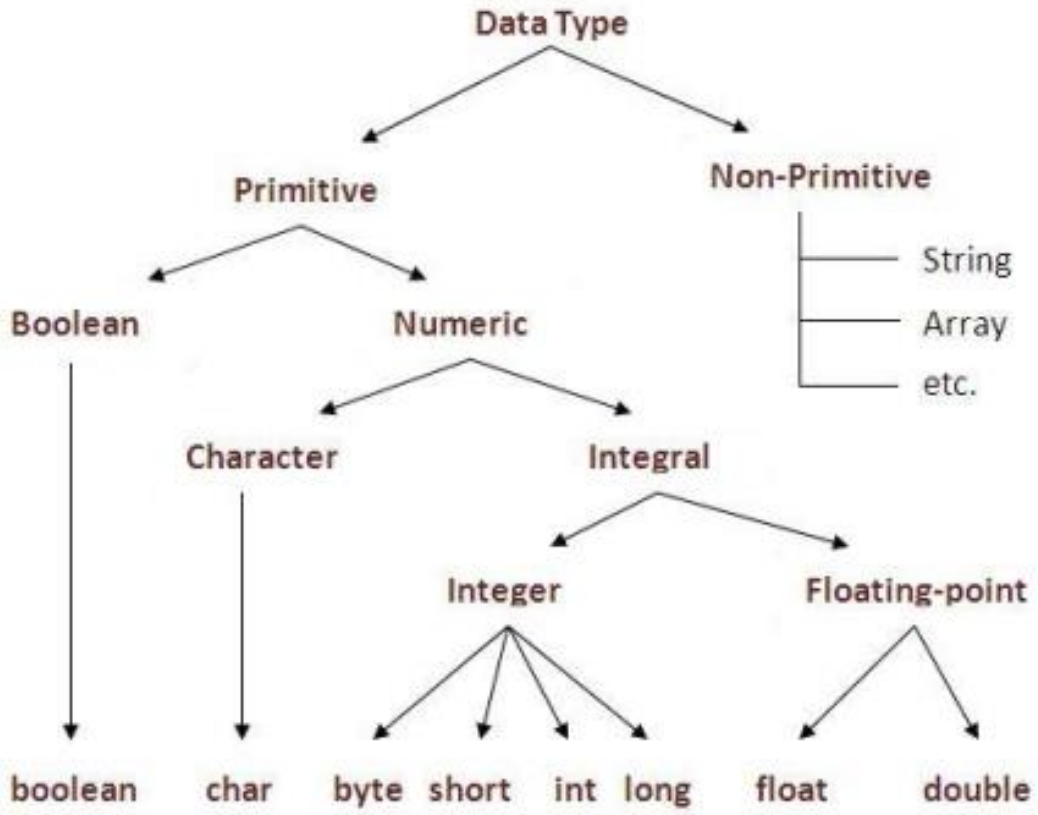
```
class A {
int data=50; //instance variable
static int m=100; //static variable
void method() {
int n=90; //local variable
 } } //endof
```

Javada ma’lumot turlari (data types)

Javada ma’lumot turlari 2 ta:

1. Sodda (*primitive*)

2. Sodda bo’lmagan (*non primitive*)



ularning oraliq qiymatlari

Ma’lumotlar turlari	Oraliq qiymati	Xotira xajmi
boolean	false	1 bit
char	'\u0000'	2 byte
byte	-128 , 127	1 byte
short	$(-2^{15}, 2^{15} - 1)$	2 byte
int	$(-2^{31}, 2^{31} - 1)$	4 byte
long	$(-2^{63}, 2^{63} - 1)$	8 byte
float	3.4e-38, 3.4e38	4 byte
double	1.7e-308, 1.73e308	8 byte

Quyidagi misolda ham o’zingiz ko’rishingiz mumkin:

```
public class Varibles {
 public static void main(String[] args) {
 System.out.println(Character.SIZE / 8 + " byte");
 System.out.println(Byte.SIZE / 8 + " byte");
 System.out.println(Short.SIZE / 8 + " byte");
 System.out.println(Integer.SIZE / 8 + " byte");
 System.out.println(Long.SIZE / 8 + " byte");
 System.out.println(Float.SIZE / 8 + " byte");
 System.out.println(Double.SIZE / 8 + " byte"); } }
```

Java dasturlash tili xarflar (char), xar-xil turdagi butun (byte, short, int, long) va ratsional sonlar (float, double), mantiqiy qiymatlarni (boolean) asosiy qiymat turlariga kiritadi. Bu asosiy qiymatlar turi sodda qiymatlar deb ataladi. Quyidagi jadval Java sodda qiymatlarini ko'rsatadi.

Qiymat berish operatori

O'zgaruvchini qiymatini o'zgartirishning to'g'ridan to'g'ri usuli bu qiymat berish operatorini qo'llashdir. Java dasturlash tilida "=" (teng) belgisi qiymat berish operatori hisoblanadi. Qiymat berish operatori ifodaning chap qismida joylashgan o'zgaruvchi, teng belgisi va ifodaning o'ng qismida joylashgan qiymat berish ifodasidan iborat. Bu yerda qiymat berish operatori boshqa o'zgaruvchilar, sonlar yoki o'zgaruvchi, sonlar, operatorlar va metodlardat tashkil topgan murakkab ifodalar bo'lishi mumkin. Quyida Java dasturlash tilining qiymat berish operatoriga misollar ko'rsatilgan:

```
xarorat = 36.6;
joriyHisob = joriyHisob + 10;
jamiOgirlik = asosiyOgirlik * qushimchaOgirlik;
```

Birinchi ifodada xarorat o'zgaruvchisiga 36.6 qiymati belgilanmoqda. Ikkinchi ifodada joriyHisob o'zgaruvchisining joriy qiymatiga 10 soni qo'shilmoqda. Uchinchi misolda jamiOgirlik o'zgaruvchisiga asosiyOgirlik va qushimchaOgirlik o'zgaruvchilarining qiymatlarini ko'paytmasidan xosil bo'lgan qiymat belgilanadi.

E'lon qilingan lekin qiymat belgilanmagan o'zgaruvchilar initsializatsiya qilinmagan deyiladi. Birlamchi kodda joyni saqlash va xatoliklarni oldini olish maqsadida o'zgaruvchini birdaniga e'lon qilish va qiymat belgilash amallarini birdaniga bajarish mumkin. Masalan,

```
int olinganSon = 0;
double tezlik = 120.5;
```

Java dasturlash tilida berilgan o'zgaruvchining qiymatini o'zgartirish uchun belgilash operatori va arifmetik operatorlarini birgalikda ishlatish ham mumkin. Masalan,

x += 2;

ifodasi

x = x + 2;

ifodasi bilan bir-xil qiymatga ega. Quyidagi jadvalda murakkab belgilash operatorlari keltirilgan.

Mukammal ifoda	Ekvivalent ifoda
x += 2;	x = x + 2;
x -= 2;	x = x - 2;
x *= 2;	x = x * 2;
x /= 2;	x = x / 2;
x %= 2;	x = x % 2;
x * 2 = y + z;	x = x * (y + z);

Arifmetik operatorlar

Java dasturlash tili qiymatlar ustida amallarni bajarish uchun quyidagi arifmetik amallarni o'z tarkibiga oladi:

+	Qo'shish
-	Ayirish
*	Ko'paytirish
/	Bo'lish
%	Modul

Yuqoridagi barcha arifmetik operatorlar butun va ratsional sonlar bilan ishlatilishi mumkin. Natijadagi qiymatning turi amalda ishlatilgan qiymatlarning turiga bog'liq. Agar amaldagi ikki operandalarning qiymatlari butun son bo'lsa natija ham butun son bo'ladi. Agar operandalarning biri yoki ikkisi ratsional son bo'lsa natija ratsional qiymat bo'ladi. Masalan,

```
count = countl + count2
```

Agar countl va count2 o'zgaruvchilar butun son bo'lsa count o'zgaruvchining qiymati ham butun son (int) bo'ladi. Agar countl yoki count2 o'zgaruvchilarining biri ratsional bo'lsa count o'zgaruvchisining qiymati ratsional qiymat bo'ladi.

Inkrement va dekriment operatorlari

Java dasturlash tilida qiymatlarni bir birlikka o'zgartiruvchi operatorlar mavjud. Qiymatni bir birlikka oshiruvchi operator inkrement operatori bo'lib

umumiy ko’rinishi n++ ko’rinishida bo’ladi. Qiymatni bir birlikka kamaytiruvchi operator dekriment operatori bo’lib n-- ko’rinishida bo’ladi. Masalan,

```
int n = 1;
int m = 4;
n++;
System.out.println("n qiymati = " + n);
m--;
System.out.println("m qiymati = " + m);
```

dastur kodi quyidagi satrlarni ekranga chiqarib beradi,

```
n qiymati = 2
m qiymati = 3
```

Increment va dekriment operatorlari o’zgaruvchi qiymatini bir birlikka o’zgartirib yangi qiymatni o’zgaruvchiga belgilab qo’yadi. Natijada qiymati o’zgargan o’zgaruvchilarni arifmetik amallarda ishlatish mumkin. Masalan,

```
2*(n++)
```

Ammo ushbu xolda inkrement operatori arifmetik operatoridan keyin bajariladi, ya’ni arifmetik amalda n o’zgaruvchining eski qiymati ishlatiladi, arifmetik amaldan keyin n qiymati bir birlikka o’zgaradi. Ushbu xolatni quyidagi misolda ko’rish mumkin,

```
int n = 3;
System.out.println(k);
System.out.println(n);
```

dastur kodi quyidagi satrlarni ekranga chiqarib beradi,

```
6
4
```

++n ham inkrement operatori xisoblanadi. Standart n++ inkrement operatoridan farqi shundaki, agar ushbu operator arifmetik amalda qatnashsa o’zgaruvchining qiymati bir birlika o’zgartiriladi va ushbu yangi qiymat arifmetik amalda ishlatiladi. Masalan,

```
int n = 3;
int k = 2*(++n);
System.out.println(k);
System.out.println(n);
```

dastur kodi quyidagi satrlarni ekranga chiqarib beradi,

```
8
4
```

n++ va ++n operatorlari inkrement operatorlari xosoblanib ikki operator ham n o’zgaruvchi qiymatini bir birlikka ko’paytiradi. Agar ++ belgisi o’zgaruvchidan keyin tursa qiymat qaytarilgandan (ishlatilgandan) so’ng u bir birlikka oshiradi. Agar ++ belgisi o’zgaruvchidan oldin tursa qiymat qaytarilishdan (ishlatilishidan) oldin u bir birlikka oshiradi.

Yuqorida inkrement operatoriga tegishli barcha xususiyatlar dekriment operatoriga ham tegishli, faqat dekriment operatorida o’zgaruvchining qiymati bir birlikka kamaytiriladi. Masalan,

```
int n = 5; int k = n--;
System.out.println(k);
System.out.println(n);
```

dastur kodi quyidagi satrlarni ekranga chiqarib beradi,

```
4
```

Aksincha,

```
int n = 5; int k = --n;
System.out.println(k);
System.out.println(n);
```

dastur kodi quyidagi satrlarni ekranga chiqarib beradi,

```
4
4
```

n-- va --n operatorlari dekriment operatorlari xisoblanib ikki operator ham n o’zgaruvchi qiymatini bir birlikka kamaytiradi. n-- operatori qiymat qaytarilgandan (ishlatilgandan) so’ng o’zgaruvchi qiymatini bir birlikka kamaytiradi. --n operatori qiymat qaytarilishdan (ishlatilishidan) oldin o’zgaruvchi qiymatini bir birlikka kamaytiradi.

Solishtirish operatori

Ikki operatorni bir-biri bilan solishtirishda ishlatiladi. Odatda, solishtirish operatorlar, shart berish operatori(**if**) va sikl(**while, for**) amallari bilan ishlatiladi.

Bu operatorlar ikki xil natija qaytarishi mumkin: **true** yoki **false**

Java dasturlash tilida quyidagi solishtirish operatorlari mavjud:

- == - teng
- != - teng emas
- > - katta
- < - kichik

- >= - katta yokiy teng
- <= - kichik yokiy teng

```
public class CompareOperators {
public static void main(String[] args) {
int a = 3;
int b = 6;
boolean c;
c = (a==b);
System.out.println(c);
}}// false
```

int tipida **a** va **b** sonlariga 3 va 6 sonlarini o'zlashtirdik, **boolean** tipida **c** ni yaratdik. **a** va **b** sonlarini solishtirish operatori yordamida (**a==b**) solishtirdik, solishtirishimiz natijasida biz ***true*** yoki ***false*** qiymatini olamiz. Demak **3** va **6** sonlari bir biriga teng emas. Natijada biz *false* qiymatini olamiz va u qiymatni **c** ga o'zlashtirib qo'ydik. Ekranda *false* qiymati chiqdi.

Mantiqiy operatorlar.

Mantiqiy operatorlar natijasi *true* yoki *false* bo'lgan operandlar ustida amalga oshiriladi. Bu operatorlar quyidagilardan iborat.

- **&&(&)** – mantiqiy VA(AND).
- **||(|)** – mantiqiy YOKI(OR).
- ^ – mantiqiy XOR(YOKI inkori)
- **!** – mantiqiy YO'Q(NOT)
- || – qisqartirilgan YOKI(OR)
- **&&** – qisqartirilgan VA(AND)

Yuqoridagi operatorlar orqali ikki operand qiymatni solishtiramiz:

a	b	a&b	a|b	a^b	~a or !a
true(1)	true(1)	true	true	false	false
true(1)	false(0)	false	true	true	false
false(0)	true(1)	false	true	true	true
false(0)	false(0)	false	false	false	true

Shu operatorlarga oid misol ko'ramiz:

```
public class CompareOperators {
public static void main(String[] args) {
boolean a = true;
```

```
        boolean b = false;
        boolean c, d, e, f;
        c = a & b;
        d = a | b;
        e = a ^ b;
        f = !a;
        System.out.println("a&b = " + c);
        System.out.println("a|b = " + d);
        System.out.println("a^b = " + e);
        System.out.println("!a = " + c);
        }}
         //a&b = false
         //a|b = true
         //a^b = true
         //!a = false
```

O‘zlashtirish operatori.

O’zlashtirish operatori tenglik(=) bilan ifodalanadi. Yuqorida ko’rgan misolimizga qaraydigan bo’lsak, **c =(a==b)** ifoda **a** va **b** ni tekshirganimizdan chiqqan qiymatni **c** ga o’zlashtiryapmiz.

Hulosa qilib shuni aytish mumkunki, o’zlashtirish bilan solishtirish operatorlari farqi

- (==) bo’lsa solishtirish
- (=) bo’lsa o’zlashtirish bo’ladi.

Java dasturlash tilida ikkita tanlash operatori bo’lib ular **if** va **switch** lardir. Masalani qo’yilishiga qarab ularning birini ishlatish mumkin. If operatori kodlashni ikkita yo’ldan biriga burib yuboradi. Hayotda shart tekshirish operatorlarini shunchalik ko’p ishlatamizki, hatto ishlatganimizni ham sezmaymiz. Tasavvur qiling siz bekatda turipsiz sizga, 11- yo’nalishdagi avtobus kerak. Uzoqdan kelayotgan avtobusga ko’zingiz tushadi va ko’zingiz orqali ko’rgan ma’lumotingiz miyyangizga uzatiladi. Shundan so’ng miyyangiz kelgan ma’lumot asosida shart tekshiradi. Agar yo’nalish raqami 11 bo’lsa, avtobusga chiq aks holda keyingisini kut degan buyruqni tanaga miyya uzatadi. Bu jarayon shunday tez bo’ladiki hatto sezmaymiz ham. Bu jarayonlar hayotimizning har daqiqasida, soniyasida yuz beradi. Endi bu shart tekshirishlarimiz javada qanday yozilishini ko’rib chiqamiz.

If tanlash operatori turlari

- if ifodasi
- if-else ifodasi
- nested if ifodasi
- if-else-if ladder (narvon)

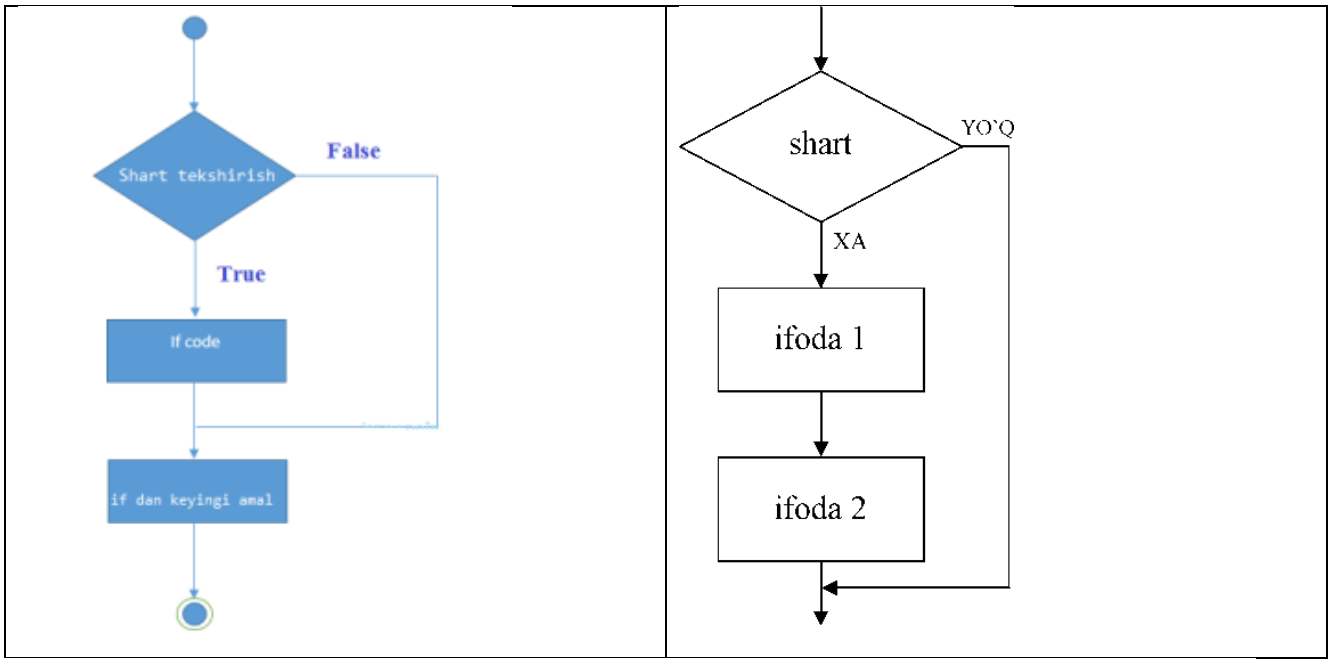
If operatori shart tekshiradi agar shart **true**(rost) bo'lsa amal bajariladi aks holda shart bajarilmaydi. If operatori, butun kod ishlashini ikkita yo'ldan biriga yo'naltirib yuboradi, ya'ni qo'yilgan shart asosida kompilyator biror yo'lni tanlab, o'z ishini davom ettiradi.

```
if(shart){ //code to be executed
}
```

Java, boshqa dasturlash tillari singari, bajarilishni boshqarish uchun shartli ifodalar va tsikllarni qo'llaydi. if/else shartli ifodasi, if shartli ifoda quyidagi umumiy ko'rinishga ega: *if (shart) ifoda*

Shartli ifodaning sharti qavs ichida bo'lishi kerak. Agar shart to'g'ri bo'lganda bir necha ifodalar bajarilishi kerak bo'lsa ular figurali qavslar ichiga olinishi kerak.

if (shart) { ifoda 1; ifoda 2; }



Misol: Tasavvur qiling, do'konga xaridor kirib tamaki maxsulotini sotib olmoqchi bo'lib, sotuvchidan tamaki mahsulotini so'radi. Sotuvchi xaridordan yoshini so'radi va xaridor sotuvchiga yoshi 17 da ekanligini aytdi. Qonunga ko'ra

18 yoshdan kichik bo’lgan xaridorlarga tamaki sotish ta’qiqlanadi. Keling endi sotuvchining miyyasida qanday jarayon bo’layotganini dasturga o’girib chiqsak:

Quloqlari orqali xaridorning yoshi nechida ekanligi miyadagi jarayon uchun kiruvchi qiymat bo’lib kirib keldi. Bu jarayonni javada dastur ko’rinishi:

```
public class IfExample {
public static void main(String[] args) {
int xaridorningYoshi=17;
if(age < 18){
 System.out.print("Tamaki mahsuloti sotilmasin");
}} }
```

Ekranda:

```
Tamaki mahsulot sotilmasin
```

Java IF-else operatori ta’rifi

Shart tekshirish jarayonida shart *true* bo’lsa, if blokdagi amal bajariladi, aks holda *else* block dagi amal bajariladi

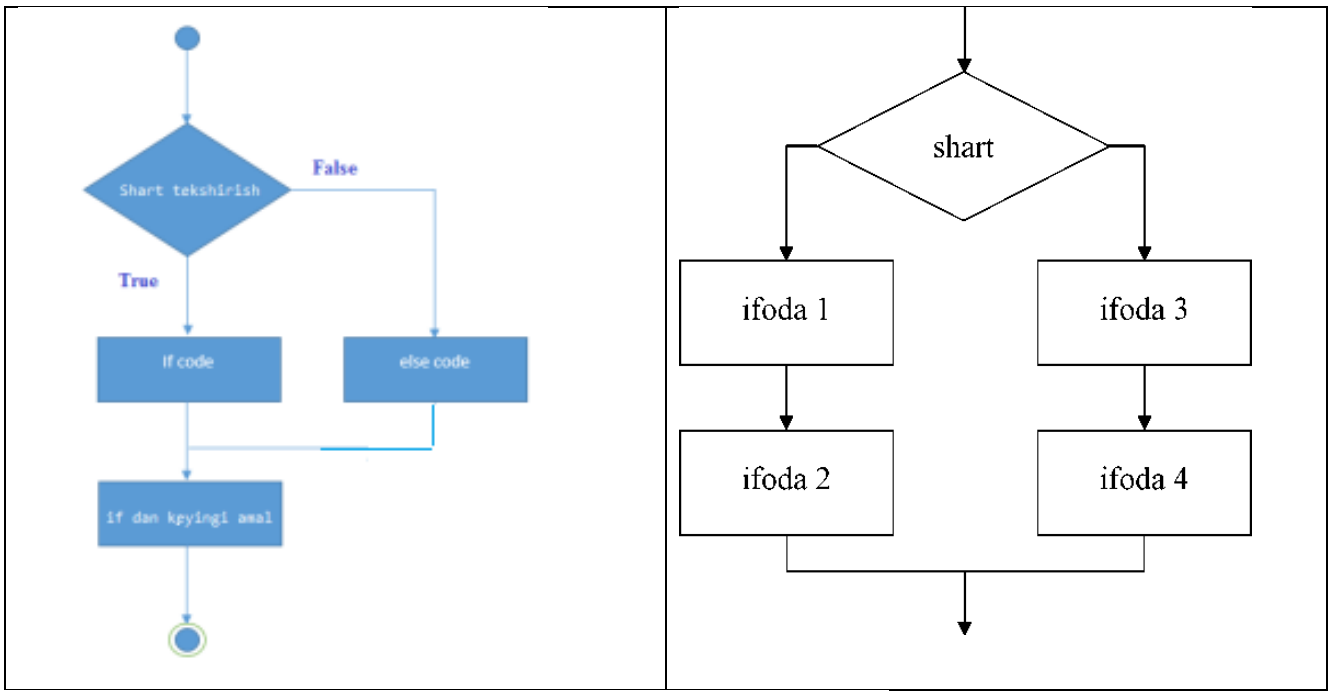
```
if(condition){ //code if condition is true
}else { //code if condition is false
}
```

if shartli ifodaning umumiyroq ko’rinishi quyidagicha bo’ladi:

```
if (shart) ifoda1 else ifoda2
```

Ushbu ko’rinishda shartli ifodaning sharti to’g’ri bo’lsa ifodal bajariladi, agar shrt noto’g’ri bo’lsa ifoda2 bajariladi. Agar shartning xar bir xolatida bir necha ifodalar bajarilishi kerak bo’lsa ifodalar figurali qavslar ichida bo’lishi kerak.

```
if (shart){ ifoda1; ifoda2; }
else { ifoda3; ifoda4; }
```



Misol: Berilgan sonni toq yoki toq emasligini bilish dasturi

```
Public class IfElseExample {
Public static void main(String[] args) {
int number=13;
if(number%2==0){
System.out.println("juft son");
}else{
System.out.println("toq son");
} } }
```

Ekranda: `Toq son`

Java Switch tanlash

Switch operatori – dastur ishlashini berilgan qiymatga mos holda biror yo‘nalishga o‘zgartirib berishni ta’minlaydi. Bu operator bir necha variantlardan kerakligini tanlab olishda, «**if**» operatori o‘rnida ishlatiladi. Albatta, barcha tanlovlarni «**If**» orqali amalga oshirish mumkin,. Farqi u shart *true* bo’lguniga qadar tekshiradi. Shart to’g’ri bo’lgandan keyin *switch*operatoridan boshqa qiymatlarni tekshirmaydi

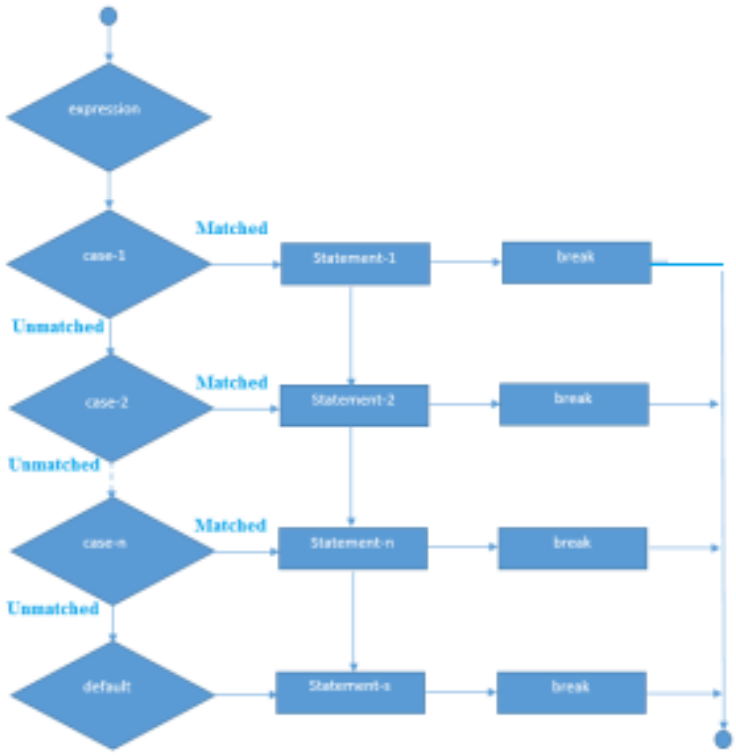
```
switch(shart){
casevalue1:
//code to be executed;
break;  //optional
casevalue2:
//code to be executed;
break;  //optional
......
default:
//hech qaysi shartni qanoantlantirmasa;
```

```
}
```

Bu operator qanday ishlaydi? «**Switch**» operatoriga berilgan qiymat, «**case**» operatoridagi qiymatlar bilan birma-bir solishtiriladi. Agar qiymat biror variant bilan mos tushsa, shu variantga tegishli blok ichidagi operator yoki operatorlar bloki ishlaydi va «**break**» orqali bu tanlov(switch) yakuniga yetadi. Agar qiymat hech qaysi variant bilan mos tushmasa, «**default**» operatori ishlaydi va uning ichidagi operatorlar dastur ishlashini davom ettiradi.

«**default**» bloki bo‘lmasligi ham mumkin, bunda agar variantlar orasida bizning qiymat topilmasa, «**switch**» hech qanday xabar bermaydi.

«**break**» operatori «**switch**» operatoridan chiqishni bildiradi, shuning uchun har bir «**case**» operatoridan keyin qo‘yilgan. Agar bu operatorni olib tashlasak, bizning qiymat variantlar orasidan topilsa ham, «**switch**» operatori ishlayveradi va hamma «**case**» operatoriga murojat qilaveradi. Bu dasturning xato ishlashiga olib keladi.




```
Public class SwitchExample {
Public static void main(String[] args) {
int number=20;
switch(number){
case 10: System.out.println("10");break;
case 20: System.out.println("20");break;
case 30: System.out.println("30");break;
```

```
     default:System.out.println("Not in 10, 20 or 30");
     } } }
Ekranda:
     20
```

JAVADA STRING

Javada String asosan char tipidagi qiymatlar ketma-ketligini ifodalovchi obyekt hisoblanadi. Belgili massiv esa javadagi String kabi ishlaydi, ya`ni:

```
      char[] ch={'j','a','v','a','t','p','o','i','n','t'};
      String s=new String(ch);
```

kodimiz

```
String s="javatpoint";
```

kabi ishlaydi.

Biz o`zgarmas(immutable) stringlarga keyinroq to`xtalamiz. Avval biz javadagi string haqida to`liq ma'lumot olamiz.

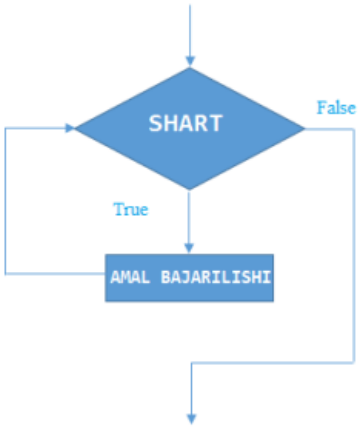
№	Funksiyalar	Ta`rifi
1	char charArt(int index)	ma'lum bir index uchun char qiymat qaytaradi
2	int length()	Stringni uzuligini qaytaradi
3	static String format(String format, Object … args)	String format qaytaradi
4	static String format(Locale I, String format, Object …args)	Berilgan local string format qaytaradi
5	String substring(int beginIndex)	boshlang'ich index uchun
6	String substring(int beginIndex, int endIndex)	Boshlang'ich va oxirgi index uchun
7	boolean contains(CharSequence s)	Ketma ket kelgan char larnig true yoki false qiymatlarini qaytaradi
8	static String join(CharSequence delimiter, CharSequence …elements)	Stringlarni qo'shadi
9	static String join(CharSequence delimiter, Iterable<? extends CharSequence> elements)	Stringlarni qo'shadi
10	boolean equals(Object another)	String va obyektlarini tekshiradi
11	boolean isEmpty()	Stringni bo'sh ekanini tekshiradi
12	String concat(String str)	Muayyan stringlarni birlashtiradi
13	String replace(char old, char new)	Belgilangan char larni almashtiradi
14	String replace(CharSequence old, CharSequence new)	Ketma ket kelgan char'lari almashtiradi
15	String trim()	Bo'sh joyni tushirib qoldiradi

16	String split(String regex)	Stringlarni teng bo'ladi
17	String split(String regex, int limit)	Stringlarni limit bilan bo'ladi
18	String intern()	String ni biror bir qiymatda ushlab turadi
19	**i**nt indexOf(int ch)	Charlarni aniq index 'ini qaytaradi
20	int indexOf(int ch, int fromIndex)	Charning boshlanish qiymati bilan aniq index qaytaradi
21	int indexOf(String substring)	Stringni qaysi index da turganini qaytaradi
22	int indexOf(String substring, int fromIndex)	Substring ko'rsatkichidan index o'rnini
23	String toLowerCase()	Harflarni kichigi bilan almashtiradi
24	String toLowerCase(Locale l)	Local berilgan harflarni kichigiga almashtiradi
25	String toUpperCase()	Berilgan harflarni kattasiga almashtiradi
26	String toUpperCase(Locale l)	Local berilgan harflarni kattasiga almashtiradi

«**While**» operatori dastur tuzishda ko'p ishlatiladigan sikl operatori hisoblanadi. Bu operator bir yoki bir necha operatorlar guruhini qo'yilgan shart yolg'on(*false*) bo'lguncha bajaradi. Qachonki shart rost bo'lsa, sikl o'z ishini boshlaydi va shartdagi qiymatlar sikl ichida o'zgartirib boriladi.

Sintaksis quyidagicha:

```
while (shart) {
// operatorlar bloki
}
```



Shart har doim mantiqiy qiymat qabul qiladi: rost(***true***) yoki yolg'on(***false***). Blok ichiga istalgancha operatorlar yozish mumkin, yoki umuman yozmaslik ham mumkin.

- «**While**» operatorida avval shart tekshirilib keyin amal bajariladi.
- «**While**» operatoridan raqamlar ketma-ketligi doimiy bo'lmagan hollarda ishlatiladi.
- «**While**» operatori takrorlanish davri har xil bo'lgan sikllar ustida ammalar bajarishda qo'l keladi.

Tasavvur qiling, siz bozordan 3kg pomidor harid qilyapsiz, sotuvchi tarozida turgan idishga pomidor solyapti bir safar 2ta bir safar 3ta qo'liga siqqanicha olib idishga solyapti. Sotuvchining idishga pomidor solishi doimiy takrorlanyapti, lekin miqdori har safar har xil, bu jarayon pomidorlar 3 kg bo'lgunicha davom etadi. Endi bu jarayonni tahlil qilib chiqsak

- Idishdagi boshlang'ich pomidor miqdori 0 kg
- Chegaraviy miqdor 3 kg
- Ortib borish miqdori har safar har xil

Sotuvchining miyyasidagi jarayonlarni dasturiy ko'rinishini qanday bo`lishini ko'rsak:

1. Sotuvchi xaridordan necha kg pomidor olishini so`rab chegaraviy miqdorni aniqlaydi
2. Idishni taroziga qo`yib boshlang`ich miqdor (massa=0) 0 deb oladi
3. Idishga pomidor solishi bilan miqdorni hissoblab boradi.
4. Agar massa 3 kg dan oshib ketsa ortiqcha pomidorlarni olib tashlaydi

Bu misolning javada dasturiy ko'rinishi:

```
public class Xarid {
 public static void main(String[] args) {
 Scanner in  = new Scanner(System.in);
 int massa = 0;
 int chegaraviyMiqdor = 3;
 while (massa < chegaraviyMiqdor) {
 System.out.println("Pomidor miqdorini kiriting");
 int m = in.nextInt();
 massa +=m;  }
 if (massa > chegaraviyMiqdor)
 System.out.println(massa  -  chegaraviyMiqdor  +"  kg  oshib
ketdi");
 else
 System.out.println("Pomidor massasi 3 kg bo'ldi");
 } }
```

Ekranda:

```
Pomidor miqdorini kiriting
2
Pomidor miqdorini kiriting
1
Pomidor massasi 3 kg bo’ldi
```

Yana bitta oddiy misol 1 dan 10 gacha bo’lgan sonlarni chiqarish dasturi

```
public class WhileExample {
 public static void main(String[] args) {
 int i=1;
 while(i<=10){
 System.out.print(i+”,);
 i++;}}}
```

Natija

```
1,2,3,4,5,6,7,8,9,10
```

Sharti keyin tekshiriladigan sikl operatori sintaksisi quyidagicha:

Ifoda:

```
1.   do{
2.   //code to be executed
3.   }while(condition);
```

«**do-while**» dan ham odatda ketma-ketlik tartibi doimiy bo’lmagan va bir marta bo’lsa ham amal bajarilishi kerak bo’lgan hollarda foydalaniladi.

```
public class MainClass {
 public static void main(String[] args) {
 int limit = 20;
 int sum = 0;
 int i = 1;
  do {
sum += i;
 i++;} while (i <= limit);
 System.out.println("sum = " + sum);}}
```

Natija:

```
sum = 210
```

Takrorlanuvchi sikl operatorlari

O’z nomi bilan ma’lum bo’lib turibdi takrorlanuvchi ya’ni qandaydir jarayonni qayta va qayta takrorlanishidir. Yerning quyosh atrofida aylanishi yil fasllarning almashinishi va h.k.larni misol qilib keltirish mumkin. Takrorlanuvchi ish harakatlar qandaydir shartlar asosida bajariladi. Ularning boshlang’ich, oxirgi nuqtalari va bajarilish davriyligi mavjud bo’ladi. Misol uchun, yerning quyosh atrofida aylanishini olsak, boshlang’ich nuqta 1-kun ya’ni 1-yanvar oxirgi nuqta 365-kun bu esa 31-dekabr takrorlanishi 1 kunga teng. Atrofimizda bunga o’xshash

takrorlanvchi ish harakatlar shunchalik ko’pki ularni sanab tugata olmaymiz. Misol tariqasida kundalik hayotimizda doimiy sodir bo’ladigan bir jarayoni tahlil qilib, uning java da dasturiy ko’rinishi qanday bo’lishini ko’rib chiqsak.

Misol: 1 dan 10 gacha bo’lgan sonlarni ekranda chiqarish

```
Public class ForExample {
publicstatic void main(String[] args) {
for(int i=1;i<=10;i++){
     out.print(i+ “, ”);}
}}
```

Ekaranda :1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Bir o‘lchamli massivlar bitta tipdagi o‘zgaruvchilar ro‘yxatidan iborat bo‘ladi. Bir o‘lchamli massivni e’lon qilish uchun, dastlab, massiv tipi, so‘ng massiv nomi ko‘rsatiladi(undan so‘ng qavslar).

```
1. int a[]=new int[5];
2. a[0]=10;
3. a[1]=20;
4. a[2]=70;
5. a[3]=40;
6. a[4]=50;
```

bu yerda uzunligi 5 ga teng bo’lgan massiv e’lon qilinyapti va ularning indexiga mos ravishda qiymatlar o’zlashtirilyapti.

```
1. class TestArray{
2. publicstatic void main(String args[]){
3. inta[]=new int[5];
4. a[0]=10;
5. a[1]=20;
6. a[2]=70;
7. a[3]=40;
8. a[4]=50;
9. for(inti=0;i<a.length;i++)
10.     System.out.println(a[i]);
11.     }}
```

Natija :

```
10
20
70
40
50
```

Massiv elementlari ichida eng kichigini aniqlash dasturi

```
1. class Testarray2{
2. public static void main(String args[]){
3. int a[]={33,3,4,5};
4. int min=a[0];
5.  for(int i=1;i<a.length;i++)
6.  if(min>a[i])
```

```
7.   min=a[i];
8. System.out.println(min); }
9. }
```

Natija:

```
3
```

Massivni e’lon qilish uchun birinchi massiv turini ko’rsatish kerak, ya’ni ushbu massiv qanday turdagi elementlarni saqlash uchun mo’ljallanganligi, keyin [] belgisini qo’yib massiv o’zgaruvchisi nomini berish kerak. Masalan, quyida butun sonlarni saqlaydigan a nomli massiv e’lon qilingan:

```
int [] a;
```

Yuqoridagi misolda a o’zgaruvchisi e’lon qilingan bo’lib ushbu o’zgaruvchiga xaqiqiy massiv xali biriktirilmagan. Quyidagi misolda yangi massivni yaratib uni a o’zgaruvchisiga biriktiramiz.

```
int [] a = new int [100];
```

Ushbu ifoda 100ta butun sonlarni saqlay oladigan massivni yaratadi.

Massiv elementlari 0 dan boshlab indekslanadi. Massiv yaratilgandan keyin uni elementlar bilan to’ldirish mumkin. Masalan,

```
int [] a = new int [4];
a [1] = 0;
a [2] = 1;
a [3] = 2;
a [4] = 3;
```

dastur kodi to’rta butun sonni saqlay oladigan massivni yaratadi va uni 0dan 3gacha bo’lgan sonlar bilan to’ldiradi.

Massivdagi elementlar sonini aniqlash uchun lengh ifodasidan foydalanish mumkin. Masalan,

```
int [] a = new int[4];
System.out.println(a.length);
```

dastur kodi massiv elementlar soni 4ligini ko’rsatadi.

Java dasturlash tilida massivni yaratish va uni elementlar bilan boyitishni qisqa usuli xam mavjud. Masalan, dastur kodi:

```
int [] a = new int [4];
a [0] = 0;
a [1] = 1;
a [2] = 2;
```

```
a [3] = 3;
```

dastur kodi bilan bir xil vazifani bajaradi, ya’ni to’rt elementni sig’dira oladigan massivni yaratib uni 0dan 3gacha bo’lgan sonlar bilan to’ldiradi va massivni a o’zgaruvchiga birikriradi.

Ko’p o’lchamli massivlar

Java dasturlash tili ko’p olchamli massivlarni qo’llash imkonini beradi. Ko’p o’lchamli massivlar massiv elementiga murojaat qilish uchun bittadan ko’p indeks ishlatiladi. Ko’p o’lchamli massivlarni massiv ichidagi massiv sifatida tasavvur qilishimiz mumkin, ya’ni massiv elementlari massiv xisoblanadi. Masalan,

```
double []    [] balans;
```

Ko’p o’lchamli massivni yaratib balans o’zgaruvchisiga biriktirish uchun quyidagidan foydalanamiz:

```
balans = new double [4]   [5];
```

Ushbu ifoda to’rta massivni o’z ishiga oluvchi massivni yaratadi. O’z navbatida xar bir ichki massiv beshta ratsional sonlardan tashkil topgan. Yaratilgan massivni elementlar bilan to’ldirish uchun quyidagi dastur kodidan foydalanish mumkin:

```
double [] b1={1.2,   1.3,1.4,1.5,1.6}
double [] b2={2.0,   2.1,2.2,2.3,2.4}
double [] b3={1.1, 1.1, 1.1, 1.1, 1.1},
double [] b4={0.1, 0.2, 0.3, 0.4, 0.5}
balans[0]=b1;
balans[1]=b2;
balans[2]=b3;
balans[3]=b4;
```

Xuddi shu massivni quyidagi qisqa usulida xam yaratib olish mumkin:

```
double []    [] balans =  {{1.2,   1.3,1.4,1.5,1.6},

                    {2.0,    2.1,2.2,2.3,2.4},
                    {1.1, 1.1, 1.1, 1.1, 1.1},
                    {0.1, 0.2, 0.3, 0.4, 0.5} };
```

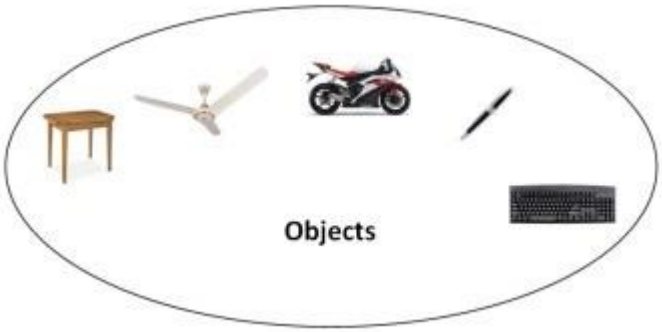
Ko’p o’lchamli massiv yaratilgandan so’ng uning elementlariga murojaat qilish uchun quyidagi ifodadan foydalanish mumkin:

```
balans [1]   [2];
```

Yuqoridagi ifoda baians massivining ikkinchi massivini uchinchi elementiga murojaat qiladi, ya’ni 2.2 qiymatiga. Massivni to’rtinchi massivining birinchi elementi, 0.1 ga murojaat qilish uchun quyidagi ifodadan foydalanish mumkin:

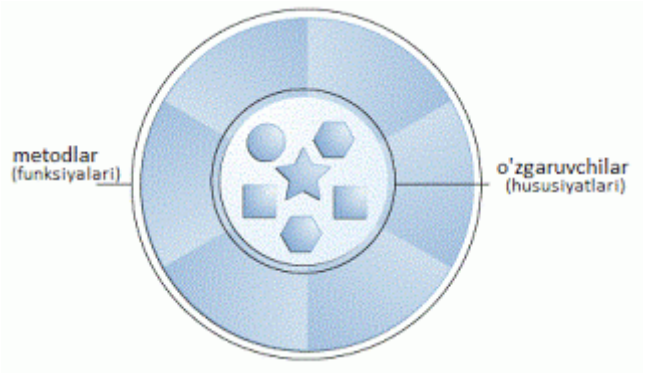
```
balans [3]   [0];
```

Obyekt – Obyektga yo’naltirilgan dasturlash(OYD) dasturlash texnologiyasining eng asosiy kalit tushunchasidir. Atrofga qarang, haqiqiy hayotdagi bir necha obyektlarni ko’rishingiz mumkin: stol, uy, qalam , motosikil , televizor va h.k



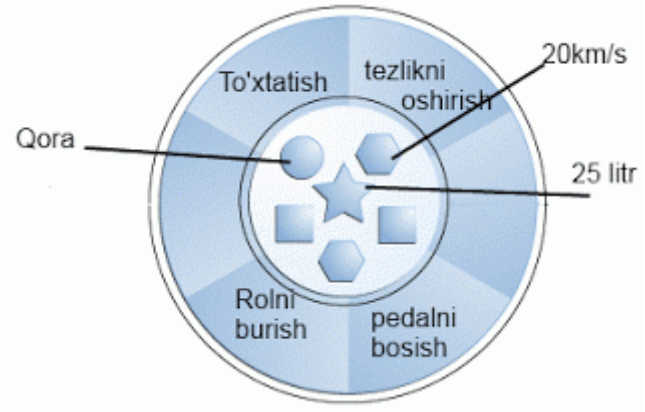
Ularning barchasining albatta hususiyatlari va bajaradigan vazifalari (funksiyalari) bor. Masalan, Mushuk hususiyatlari: rangi, qorni to’qligi, yoshi, jinsi; funksiyalari: ovqat yeyishi, miyovlashi, yurishi, sichqon tutishi. Mashina hususiyatlari: tezligi, rangi, nomi, narxi; funksiyalari: yurishi, to’xtashi, oyna artgichlarining ishlashi, eshiklarning ochilib yopilishi va h.k. Bu kabi hayotiy misollarning hususiyatlari va funksiyalarini aniqlash OYD nuqtai nazaridan fikrlashning eng zo’r ko’rinishidir.

Bir daqiqaga to’xtang va hozirda atrofingizdagi biror narsalarni tahlil qiling. Har bir obyekt uchun o’zingizdan so’rang: “Bu obyektning qanday hususiyatlari bor?”, “Qanday vazifalarni bajaradi?” kabi. Va kuzatish natijalaringizni yozib oling, sezgan bo’lsangiz tuziladigan ro’yxat obyektning murakkabligiga qarab ko’payib boradi. Kompyuter indikatorining 2 ta *hususiyati* bor o’chiq va yoniq; *funksiyalari* esa yonish va o’chish. Bu barcha kuzatishlar OYD dunyosiga o’tkazish mumkin.



Dasturlashdagi obyekt. Dasturlashdagi obyekt(bundan keyin oddiygina *obyekt* deb ketiladi) ham haqiqiy hayotdagi obyektlarga o'xshash: Ular ham qandaydir hususiyatlar va bajaradigan funksiyalardan iborat bo'ladi. Obyektning hususiyatlari har xil dasturiy *o'zgaruvchilardan* iborat bo'ladi va ularning o'zgartirish uchun qandaydir *funksiyalar* bajariladi. Bunday *funksiyalar* bilan o'zgaruvchilarning holatini berkitish mumkin ya'ni aynan o'sha o'zgaruvchini tashqaridan o'zgartirish uchun albatta maxsus funksiyadan foydalanish kerak bo'ladi. Bu jarayon "**Enkapsulatsiya**" deb atalib, OYDning eng muxim tushunchalaridan biridir.

Mashinani tasavvur qiling,



Uni dasturlash obyekti sifatida modellashtiramiz:

Uning o'zgaruvchilari(hozirgi tezligi, qolgan benzini, va h.k) va uning funksiyalari(to'xtatish, tezlikni oshirish, rolni burish va h.k.). Bu yerda uning bakidagi benzini yurishi tufayli kamayib boradi demak uning qiymatining o'zgarishi 0 dan bakning sig'imigacha bo'ladi, yoki uning tezligi ham shu kabi aynan qaysidir funksiyalarning amalga oshirilishi orqali u ham 0 dan maksimal tezligigacha

o’zgarishi mumkin. Bulardan tashqari mashinaning ba’zi hususiyatlari borki ular o’zgarmasligi mumkin, masalan rangi.

Demak, ko’rinib turipdiki mashina ham o’z navbatida bir necha mayda obyeklardan iborat bo’ladi. Va albatta ularni kodda yozganda ham alohida obyekt sifatida ifodalash kerak bu orqali nimalarga erishish mumkin:

Qismlilik: Har bir obyektga tegishli bo’lgan kodlar alohida-alohida, boshqa obyeklarga bog’liq bo’lmagan holda boshqarish imkoniyatiga ega bo’lamiz. Bu hammasi emas, tasavvur qiling mashina obyektini ifodalovchi kodni bo’lmasdan faqat bitta faylda ifodaladik; bu esa murakkabligiga qarab yuzlab hatto minglab qatorli kod bo’lishi mumkin. Undan biror narsani topib-o’zgartrish ancha mashaqqat bo’ladi.

Qayta foydalanish: Yana boshqa foydali tarafi biz bo’laklagan mashinaning detallarini boshqa obyektlarda ham ishlatishimiz mumkin. Masalan, 2 xil mashina ularning aynan bir xil qismlari bor, ana o’shalar uchun ikki marta alohida kod yozmasdan, bitta yozganimizni qayta ishlatishimiz mumkin.

Uzilib-ulanuvchanligi: buni tushunish uchun yuqoridagi misoldan foydalanamiz, aytaylik, mashinaning biror qismi ishlamayapti, xo’sh nima qilinadi? O’sha qismni ishlab turgan boshqa ehtiyot qismga almashtiramiz, yoki tuzatamiz. Mashinaning biror vinti buzilsa uni boshqa ishlab turgani bilan almashtirasiz yoki tuzatamiz lekin mashinani butunligicha yahlit almashtirmaymiz.

OYDning asosiy tushunchalari

Obyekga yo’naltirilgan dasturlash yoki *OYD* – haqiqiy hayotiylikka asoslangan dasturlash usulidir. Yana protsedurali dasturlash tillari (masalan, Paskal, Basic, Fortan) ham mavjud. OYD ning undan asosiy farqi shundaki, OYD asosan obyektlarga asoslangan holda ishlasa, *protsedurali dasturlash tillari* esa asosan funktsiyalarga asoslangan bo’ladi ya’ni bu usuldagi dasturlashda har bitta buyruqlar qadamma-qadam bajarilib boriladi masalan: faylni och, raqamni o’qi, 4 ga ko’paytir va ekranga chiqar.

OYD ni tashkil etuvchilari quydagilar:

- Object – Obyekt

- Class – Sinf
- Inheritance – Meros olish
- Polymorphism – Ko’p formalik
- Abstraction – Mavhumlik
- Encapsulation – Enkapsulyatsiya (Kapsula ichiga joylamoq)

Object – Obyekt klass turidagi o’zgaruvchi. Obyekt bu klass bilan farqli tushuncha xisoblanadi. Objekt biz yozgan klassimizdagi har xil qoidalarga bo’ysunadigan ma’lumot bo’lib, u tezkor hotirada saqlanadi, klass esa qattiq diskda saqlanadi. Har bir yasalgan Obyect tezkor hotiraning ma’lum bir honachalariga joylashadi. Hayotiy bir misol, masalan, ko’p qavatli binoni tezkor hotira deb qarasak. Unda istiqomat qiluvchi insonlar esa unda saqlanuvchi obyektlar bo’lib. Agar biz Insonning hususiyatlari, bajaradigan ishlari va hokazo hususiyatlari haqidagi bilimlarni qog’ozga tushursak bu qoralamani klass deb qaralishi mumkin garchi u texnik usulda yozilmagan bo’lsa ham. Biz ana o’sha qoralamani klassimizda kompyuter tushunadigan tilga keltiramiz.

Class – OYDning marzkazi hisoblanadi va u har xil kodlar, ma’lumotlar va shu ma’lumotlar qay tarzda o’zgarishini ifodalovchi hususiyatlar saqlanadi. Boshqacharoq qilib aytadigan bo’lsak hayotiy obyektlarning qanday faoliyat yuritishi, nimalardan iborat ekanligi, qanday hususiyatlarga ega ekanligini tavsiflovchi kichik bir hujjat sifatida qarash ham mumkin. Javada hamma narsa Klass ichida sodir bo’ladi. Klass o’z ichiga o’zgaruvchilar va metodlar(funksiyalar) va qiymati o’zgarmaydigan konstantalarni oladi. Yana shuni ham ta’kidlash kerakki, har bitta klass bitta o’zgaruvchi turi bo’lib ham hizmat qiladi. Xuddi Integer, String yoki boshqa turlar kabi har bir klass ham ma’lum bir tur sifatida qaralishi mumkin.

Javada klasslar quydagilardan tashkil topad:

1. data member – o’zgaruvchilar
2. method – funksiya
3. constructor – kontruktor (obyekt dastlabki holatida ishlaydi)
4. block – blok ({} qavslar orasi blok hisoblanadi)
5. class and interface – klass va interfeys

Javada o’zgaruvchilar(data member**)**

O’zgaruvchilar – obyektlarimizni o’zgaruvchi sifatida ham qarashimiz mumkin bunda uning qiymatlari o’zgarishi mumkin. Masalan Inson classimizda insonning yurish tezligi o’zgaradi bunda uning tezligi qandaydir sonlarda o’zgaradi masalan **yur()** metodidan **yugur()** metodiga o’tganda yoki **to’xta()** metodiga o’tganda uning tezligi o’zgaradi. Shu va shunga o’xshash obyektlar o’zgaruvchilar deb yuritiladi.

```
1.  public class Inson{
2.  int tezlik=0;
3.  public void yur(){
4.  tezlik=5;
5.  }
6.  public void yugur(){
7.  tezlik=15;
8.  }
9.  public void yur(){
10. tezlik=0;
11. }
12. }
```

“instance variable” – o’zgaruvchi namuna olish kompilyatsiya vaqtida xotiradan joy olmaydi. O’zgaruvchilarimizni obyekt sifatida qaraganimizda runtime (dastur ish payti) paytida tezkor xotiradan joy oladi.

Javada funksiyalar (Method)

Funksiyalar – obyektimiz nimalar qila olishini izohlaydi, masalan, Inson haqida yozgan qoralamamizda insonni yura olishi va boshqa harakatlarini keltirganmiz. Aynan ana o’sha qila oladigan ishlarini Metodlar orqali ifodalaymiz. Masalan quyida **yur()** metodi keltirilgan.

Funksiyalarning imkoniyatlari

1. Kodni qayta ishlash
2. Kodni optimallashtirish (muvofiqlashtirish)

Obyekt va klassga misol

Bu misolda **student** klassidan ikkita obyekt yaratildi va insertRecord funnksiyasi orqali boshlang’ich qiymatlarni

o’zlashtirildi. *displayInformation* funksiyasidan foydalanib ekranda qimatlarni chop etiladi.

```
1.  class Student{
2.   int rollno;
3.   String name;
4.   void  insertRecord(int  r,  String  n){
//method
5.     rollno=r;
6.     name=n;
7.    }
8.                                          void
displayInformation(){System.out.println(rollno+"
"+name);}//method
9.    public static void main(String args[]){
10.   Student s1=new Student();
11.   Student s2=new Student();
12.   s1.insertRecord(111,"Karan");
13.   s2.insertRecord(222,"Aryan");
14.   s1.displayInformation();
15.   s2.displayInformation();
16.  }
17. }
```

Ekranda :

```
111 Karan
222 Aryan
```

Meros olish (Inheritance). Ma’lum obyekt asosida boshqa obyektni yaratish jarayoniga aytiladi. Bunda obyektning barcha hususiyatlarini meros qilib oladi ya’ni private(shaxsiy) bo’lmagan o’zgaruvchilari funksiyalari konstantalarini bemalol foydalanish. *Meros olish*dan dastur ishchi vaqtida *Ko’p formalik*dan foydalaniladi.

Ko’p formalik (Polymorphism). OOPning uchinchi ustuni. Yuqorida berilgan ustunlar bilan doim birga yuradi. Ma’lum obyekt asosida boshqa obyektni yaratish jarayoniga aytiladi. Bir klassning boshqa klassdan meros olishi yordamida amalga oshiriladi. Meros olingan obyekt ota obyektdagi xususiyatlarni tanlovga ko’ra meros oladi. Masalan, avtoulov bu ota obyekt. Bu obyekt yordamida yengil mashina, yuk mashinasi, poyga mashinasi kabi boshqa obyektlarni yaratib olishimiz

mumkin. Ota klassda bo’lgan 4 g’ildirak farzand klasslarda ham mavjud bo’ladi. Ya’ni poyga mashinasi, avtoulovdan g’ildiraklarni meros oladi.

Mavhumlik – Abtraksiya (Abstraction). Obyekt bu real jism. Abstraksiya esa ushbu jismni reallikdan uzoq deb tassavur qilib, u haqida fikr yuritishga aytiladi. Abstraksiyaga fikrlar to’plami sifatida ham qarasa bo’ladi.. misol uchun telefon jiringlasa, telefonga qo’ng’iroq bo’layotganini bilamiz lekin uning ichida nima jarayon bo’layotganini bilmaymiz.

Enkapsulyatsiya (Encapsulation). Ma’lumotlar va funksiyalarni bir komponent ichiga yig’ishga atyiladi. Buning uchun klasslardan foydalaniladi. Enkapsulatsiya tanlov asosida classning ba’zi hususiyatlarini foydalanuvchidan yashirish imkonini beradi. Bunga misol qilib, ustidan maxsus modda bilan o’ralgan dorilarni keltirsak bo’ladi.

2.2. Mobil dasturlar ishlab chiqishning zamonaviy vositalari

Hozirgi kunda ko’pchilk foydalanuvchilar shaxsiy kompyuterlardan voz kechgan holda shaxsiy mobil qurilmalardan foydalanishni afzal ko’rishmoqda. O’z navbatida shaxsiy mobil qurilmalar foydalanuvchiga 24/7 rejimida doimiy aloqada bo’lish imkoniyatini yaratib beradi. Shaxsiy kompyuterlardan farqli ravishda, mobil telefonlar va planshetlar foydalanuvchilarning shaxsiy qurilmalari hisoblanadi. Ushbu qurilmalarda foydalanuvchilarning shaxsiy rasmlari, plastik karta raqamlari va kodlari, jamiyatlik tarmoqlardagi akkauntlari va foydalanuvchining shaxsan o’ziga tegishli bo’lgan shaxsiy ma’lumotlari saqlanadi.

So’nggi yillarda mobil qurilmalar uchun juda ko’plab dasturlar ishlab chiqilmoqda va bunday dasturlar soni kundan-kunga ko’payib bormoqda. Ushbu dasturlar xilma-xilligi bilan bir-birlaridan ajralib turadi. Misol tariqasida ayrim dasturlar foydalanuvchiga o’z vaqtini rejalashtirishga yordam berishni taklif qilsa, boshqa bir dastur esa unga sport bilan shug’ullanish vaqtini eslatib turadi. Kundan-kunga mobil qurilmalar va ulardan foydalanuvchilar soni nafaqat dunyo miqyosida, shuningdek yurtimizda ham juda tez sur’atlarda o’sib bormoqda. Hozirgi kunda dunyoda keng tarqalgan Android, iOS hamda Windows Phone amaliyot tizimlari uchun mobil dasturlar tuzish uchun zarur bo’lgan zamonaviy dasturlash muhitlari va vositalari haqida to’xtalib o’tamiz.

Mobil qurilmalar uchun zamonaviy dasturlash muhitlari va vositalari haqidagi muloxazamizni Android amaliyot tizimidan boshlaymiz. Hozirda Android amaliyot tizimi uchun dastur tuzishda keng qo’llaniladigan ikki IDE (Integrated Development Environment) muhit mavjud bo’lib, bular Eclipse hamda JetBrains kompaniyasi tomonidan ishlab chiqilgan Intellij IDEA dasturlash muhitlari hisoblanadi.

Eclipse dasturlash muhiti ochiq kodli loyiha hisoblanadi. Ushbu IDE dasturchi uchun oson sozlanishi, zarur dasturiy kompanentlari hisoblanadigan SDK (Software Development Kit) Android, NDK (Native Developer Kit) Android hamda JAVA mashinalari bilan oson integratsiya qilinishi bilan ajralib turadi. Dasturlash muhiti uchun zarur bo’lgan kompanentlar muvaffaqiyatli integratsiya qilingandan keyin dasturchi tomonidan o’zining birinchi loyihasini yaratishda hech qanday

qiyinchiliklarga duch kelmaydi. Dasturchi tomonidan Android uchun loyiha tanlangandan keyin avtomatik ravishda bo'sh "Hello world" loyihasi yaratiladi va ushbu loyihani virtual qurilmada ishlatib ko'rish mumkin. Eclipse muhitida asosiy dasturlash tili sifatida Java dasturlash tili foydalaniladi.Ushbu IDE foydalanuvchiga juda qulay va oson sozlanishi tufayli ko'pchilik dasturchilar tomonidan mobil qurilmalarga dastur yozishni ushbu muhitda boshlashni tavsiya qilishadi.

JetBrains kompaniyas tomonidan ishlab chiqilgan Intellij IDEA muhiti asosan tajribali dasturchilarga mo'ljallangan hisoblanadi. Chunki ko'pchilik dasturchilar JetBrains kompaniyasi mahsulotlarini «AqlliIDE» deb nomlashadi. Sizning e'tiboringizni tortmasligi mumkin bo'lgan, shart operatorlaridan keyin tushib qoldirilgan qavslarni avtomatik yopilishi, turli xil usullarning guruhlarga ajratilishi va (Interface, Singleton) kabi sinflar uchun avtomatik ravishda sinflarning paydo qilinishi bir ko'rinishda muhim sanalmasligi mumkin. Lekin, statistika bo'yicha bir yil davomida faqatgina 'rename' jarayonining o'ziga dasturchining 120 soat ish vaqti tejalishi aniqlangan.

Dasturchilarni qiynaydigan eng murakkab masalalardan biri dasturda o'zgaruvchilarni nomlash masalasi ushbu muhitda juda samarali hal qilingan. Masalan, agar sizda 'Item' nomli asosiy sinf bo'lsa va siz ushbu sinfdan foydalanib massiv yaratsangiz, dasturlash muhiti ushbu elementlarni 'Items' deb nomlashni taklif qiladi. Ushbu misol siz uchun juda oddiy tuyulishi mumkin, lekin ushbu amaldan amaliyotda foydalanish juda foydali va dasturchining ko'p vaqtini tejashga yordam beradi. Ushbu muhitda ham boshqa dasturlash muhitlari kabi tuzilgan dasturning qanday natija berganini muhit tomonidan tanlangan virtual qurilmalarda ko'rish imkoniyati mavjud.

Android Studio muhiti JetBrains va Google kompaniyalari bilan birgalikda ishlab chiqilgan bo'lib, asosan Android amaliyot tizimi uchun dasturlar ishlab chiqishga mo'ljallangan muhit hisoblanadi. Google kompaniyasi Android qurilmalariga dastur tuzishda aynan Android Studio muhitidan foydalanishni tavsiya qiladi.

XCode muhiti iOS amaliyot tizimida ishlovchi dasturlar yozish uchun yaratilgan muhit bo'lib, ushbu muhit iOS amaliyot tizimi uchun dasturlar yozish uchun eng qulay muhit hisoblanadi. XCode foydalanish uchun yagona workplace-window maydonidan foydalanadi hamda ushbu yagona oynada dasturchi uchun zarur bo'lgan barcha vositalar juda tushunarli ko'rinishda joylashtirilgan bo'lib, dasturchi o'zi uchun zarur bo'lgan vositani topishda hech qanday qiyinchilikga duch kelmaydi. SDK iOS doimiy ravishda XCode uchun qo'shimcha vositalar, kompilyatorlar va freymvorklarni qo'shib bormoqda va ushbu qo'shimchalar muhitning imkoniyatlarni kengaytirib bormoqda. Dasturlash muhitida asosiy dasturlash tili sifatida Objective-C dasturlash tilidan foydalaniladi.

Microsoft Visual Studio muhitiga Windows Phone SDK plagini o'rnatilgandan keyin, ushbu muhit dasturchi uchun Windows Phone amaliyot tizimida ishlovchi dasturlar yozish uchun tayyor bo'ladi.

Plaginning eng so'nggi versiyasida dasturchilar dastur tuzishi va tayyor dasturni bepul Windows Phone Marketplacega joylashtirish imkoniyati ham mavjud hisoblanadi. Muhitda dasturni tuzish va uni virtual simulyatorlarda testlash imkoniyatlari ham mavjud. Asosiy dasturlash tili sifatida C# dasturlash tilidan foydalaniladi.

Xulosa qilib shuni aytib o'tish kerakki, dunyoda keng tarqalgan mobil qurilmalarda foydalanilayotgan Android, iOS va Windows Phone amaliyot tizimlari uchun ilova yozish mumkin bo'lgan muhitlardan tashqari, Xamarin, Unity3D, Cocos2Dx, Marmelade, Phonegap kab krossplatformali dasturlash muhitlari ham mavjuddir. Krossplatformali muhitlardan asosan qisqa vaqt ichida keng jamoatchilikni jalb qilish uchun tuziladigan dasturlarda foydalaniladi. Ushbu muhitda yaratilgan dasturlar mobil qurilmaning asosiy resurslaridan foydalanish imkoniyatini bermaydi. Shu sababli, mobil qurilmalar uchun dastur tuzishda yuqorida nomlari keltirilgan IDE muhitlaridan foydalanish tavsiya qilinadi.

XULOSA

Xulosa qilib shuni aytish mumkin, Respublikamizda so’nggi yillarda innovatsiyalarga va yoshlar orasida ilmiy tadqiqot bilan shug’ullanishga bo’lgan moyillikni kuchaytirish maqsadida juda ko’plab tizimli ishlar olib borilmoqda. Respublikamizni yangi bosqichga olib chiqish va elektron hukumat tizimini joriy qilishda yurtimizda faoliyat yuritayotgan har qanday tashkilot o’z avtomatlashtirilgan tizimiga ega bo’lishi zarurati paydo bo’lmoqda.

Davlat korxonalarida avtomatlashtirilgan ish joylarini tahlil qilish va yaratishda foydalanilgan vositalar haqida quyidagi xulosaga keldim.

Ushbu tizim yordamida fuqarolarning o’zini o’zi boshqarish organlari xodimlarining malakasini oshirish markazining dasturiy ta’minoti yordamida markazda faoliyat yurituvchi xodimlar markaz hayotiga taaluqli qonun hujjatlari, yangiliklar, mediatekalarni markazda ta’lim oluvchi mahalla faollariga tez va sifatli yetkazib berish imkoniyatiga ega bo’ladi. Bu esa o’z navbatida muassasada faoliyat olib borayotgan xodimlarning ish samaradorligiga ham o’z ta’sirini o’tkazadi.

Tizim zamonaviy dasturlash tili PHP, MySQL ma’lumotlar bazasi hamda Bootstrap freymvorkidan foydalanib ishlab chiqilganligini inobatga olsak, tizim samarador va tezkor ishlash mexanizmiga ega bo’ladi. Tizimning veb platforma ko’rinishida yaratilgani esa foydalanuvchilardan qo’shimcha bilimlarni talab qilmaydi va har bir kompyuterda mavjud brauzerlar orqali ishga tushirish imkoniyati mavjud bo’ladi.

Qarab o’tilgan bitiruv malakaviy ishininng 2 bobida keltirilgan ma’lumotlar va yaratilgan tizimdan foydalangan holda quyidagi xulosalarga keldim:

- Ushbu tizim fuqarolarning o’zini o’zi boshqarish organlari xodimlarining malakasini oshirish markazida faoliyat olib borayotgan xodimlar ishlari samaradorligini oshiradi;
- Tizim veb platforma ko’rinishida yaratilganligi foydalanuvchilardan tizimdan foydalanishlarda qo’shimcha qiyinchiliklar tug’dirmaydi;
- Tizim zamonaviy dasturlash tillarida yaratilganligi sababli uning samaradorlig yuqori bo’ladi.

Umumiy holda aytganda tizimni loyihalashda foydalanilgan zamonaviy dasturlash tillari va texnologiyalaridan foydalanildi.

FOYDALANILGAN ADABIYOTLAR RO'YXATI

1. O'zbekiston Respublikasi Prezidentining «Kompyuterlashtirishni yanada rivojlantirish va axborot-kommunikatsiya texnologiyalarini joriy etish» to'g'risidagi farmoni, 30.05.2002 yil.
2. O'zbekiston Respublikasi Prezidentining PQ-1730-sonli «Zamonaviy axborot-kommunikatsiya texnologiyalarini yanada joriy etish va rivojlantirish chora-tadbirlari» to'g'risidagi qarori, 21.03.2012 yil.
3. W.Jason Gilmore. Beginning PHP and MySQL 5: From Novice to Professional 2nd edition. Apress, 2006.
4. Chuck Musciano, Bill Kennedy. HTML and XHTML. The definitive guide. O'Reilly Media, 2008.
5. Anthony Molinaro. SQL cookbook. O'Reilly Media, 2009.
6. Cristopher Schmitt. CSS cookbook. O'Reilly Media, 2011.
7. Elisabeth Robson. Head First HTML and CSS: A Learner's Guide to Creating Standards-Based Web Pages 2nd Edition. O'Reilly Media, 2012.
8. Robin Nixon. Learning PHP, MySQL, JavaScript, and CSS: A Step-by-Step Guide to Creating Dynamic Websites Second Edition. O'Reilly Media, 2012.
9. Luk Uelling. PHP and MySQL Web Development. O'Reilly Media, 2014.
10. Mett Zandstra. PHP Objects, Patterns, and Practice, Second Edition. O'Reilly Media, 2015.
11. http://www.php.su/books/ sayti
12. https://dev.mysql.com/doc/ sayti
13. https://www.sql-ex.ru sayti
14. https://sqlbooks.ru sayti
15. https://master-css.com sayti
16. https://www.eduportal.uz
17. https://www.ziyonet.uz ta'lim portalining