Muqaddima

Bu kitob yordamida sizlar bilan birgalikda eng yaxshi dasturlash tillaridan biri bo'lib kelayotgan Java dasturlash tili bilan dasturlash olamiga kiramiz.

Bu kitob kimlar uchun tayyorlangan?

Bu kitobimiz dasturlashni o'rganmoqchi bo'lgan barcha yoshdagilar uchun moslashtirilgandir.

Bu kitob ko'magida o'rganishni boshlashimizdan oldin bilishimiz kerak bo'ladigan narsalar qaysilar?

Bu kitobdan o'rganish davomida sizdan har qanday bir narsani bilishingiz talab qilinmaydi.

Kod manbalari.

Bu kitobimizda foydalanilgan barcha kodlarga GitHub onlayn platformadan erisha olasiz.

https://github.com/21040001/Java_yordamchi_kitob_kodlarimiz.git

Kitobdagi har bir mavzuning dars videolariga onlayn shaklda QR kodlar orqali erisha olasiz.

Hayotimda qarshimga chiqqan barcha qiyinchiliklardan o'tishimga va anashu vaziyatlarda o'z maslahatlarini bergani uchun eng avvalo ota-onamga, undan so'ngra barcha yaqinlarimga bu kitobim orqali o'z tashakkurlarimni bildirib qolaman.

Davronbek Abdurazzokov.

Mundarija

Java haqida barchasi 1

1.1 Java tarixi

James Gosling va Patrik Naughton Java dasturlash tili ustidagi ishlarini 1991-yilning iyun oyida boshladilar. Java birinchi bo'lib interaktiv televizorlar uchun tayyorladi va Javaning birinchi nomi Oak bo'lgan edi, bu isimni Goslingning ofisi yonidagi bir daraxtdan olishdi. Keyinchalik esa nomi Greenga almashtirildi va eng oxirida Java nomini oldi.

Gosling, Javani C/C++larga o'xshash shaklda sintaksis bilan tayyorladi va bu sababli, dasturchilar tarafidan oson bir shaklda o'rganiladigan bir dasturlash tili bo'ldi.

2022-yilning may oyidan e'tiboran rasmiy shaklda Java SE 8, Java SE 11 va Java SE 17 versiyalari bir uzun bir muddatga(LTS) sifatida, Java SE 18 esa muddatli bir shaklda qo'llab quvvatlanmoqda. Javani versiyalari pastda berib o'tilgani kabidir.

Versiya	Chiqarilgan yili	Versiya	Chiqarilgan yili
JDK Beta	1995 yil	J2SE 1.3	2000-yil 8-may
JDK 1.0	1996-yil 23-yanvar	J2SE 1.4	2002-yil 6-fevral
JDK 1.1	1997-yil 19-fevral	J2SE 5.0	2004-yil 30-sentabr
J2SE 1.2	1998-yil 8-dekabr	Java SE 6	2006-yil 11-dekabr
Java SE 7	2011-yil 28-iyul	Java SE 14	2020-yil 17-mart
Java SE 8 (LTS)	2014-yil 18-mart	Java SE 15	2020-yil 15-sentabr
Java SE 9	2017-yil 21-sentabr	Java SE 16	2021-yil 16-mart
Java SE 10	2018-yil 20-mart	Java SE 17 (LTS)	2021-yil 14-sentabr
Java SE 11(LTS)	2018-yil 25-sentabr	Java SE 18	2022-yil 22-mart
Java SE 12	2019-yil 19-mart	Java SE 19	2022-yil 13-senatabr
Java SE 13	2019-yil 17-sentabr	Java SE 20	2022-yil 7-iyun **

LTS (Long-Term Support) – uzoq muddatli qo‘llab quvvatlash ma’nosida keladi.

Biz bu kitobdagi darslarimizni Java SE 17 versiyasi asosida tayyorladik.

1.2 Java nima va qanday shaklda ishlaydi?

Java bu obyektga yo'nalgan(OPP) bir dasturlash tilidir. Barcha kodlarimiz sinflar ichkarisida yozilishi kerak. Va sinflar ichkarisiga yozilishi tarafidan, o'zidan oldin chiqarilgan C++ dasturlash tilidan farq qiladi. C++ ham obyektga yo'naltirilgan bir dasturlash tilidir ammo, C++ obyektga yo'naltirilmagan dastur kodlarini ham yozishga ruxsat beradi. Javada esa bunga o'xshash bir imkoniyat yo'q. Java programmalari to'g'ridan operatsion tizim bilan ishlamaydi, operatsion tizimiga oldindan qurilgan Java Virtual Machine (JVM) bilan ishlaydi. Boshqa dasturlash tillarida bu shaklda bir xususiyat mavjud emas. Shu sababli Java platformalarga bog'lanib qolmagan bir dasturlash tilidir, ya'ni Java programmalari barcha JVMli operatsion tizimlarda ishlay oladi.

1.3 Java qaysi yo'nalishlarda foydalanilmoqda

Java, kunimizda 4,5 milliard elektronik jihozlarda foydalanilmoqda (Manba: Java.com).

*Uyali telefonlarda

*Kameralarda, printerlarda, tibbiyot jihozlarida va hokazo.

Java dasturlash tili asosan frameworklarni (programma tashqi ko'rinishlarini) tayyorlash uchun foydalanilmoqda.

Java dasturlash tili kirish qism 2

2.1 Java dasturlash tilini yuklab olish

Yuqoridagi mavzularda berib o'tilgan ma'lumotlar yordamida Java dasturlash tili haqida yetarli darajada tushuncha hosil qilib oldik deb o'ylayman. Bundan keyin asta sekinlik bilan Java dasturlash tilida kodlarni yozish va ishlash shakllarini o'rganishni boshlasak ham bo'ladi.

Java dasturlash tilida kodlarni yozishimiz va yozishni o'rganishimiz uchun birinchi bo'lib qiladigan ishimiz kompyuterimizga Java dasturlash tilini yuklab olish va o'rnatish bo'ladi. Bu jarayonni sizlar bilan birgalikda qadam-baqadam ko'rib chiqamiz. Tayyor bo'lsangiz boshladik.

1. **Qadam**
   Dasturlash tilimizni yuklab olish uchun Java dasturlash tilining rasmiy saytiga boramiz va u yerdan Java dasturlash tilini yuklab olish uchun tayyorlangan bo'limiga o'tamiz.

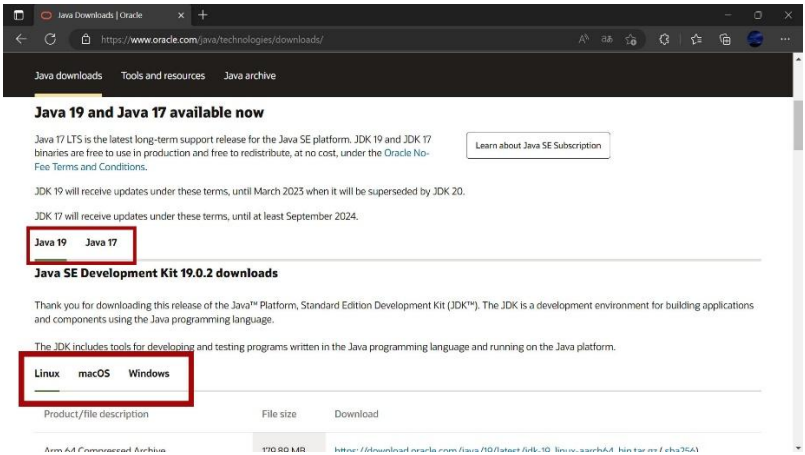
Java dasturlash tili Oracle kompaniyasi tarafidan sotib olingani sababli, Java dasturlash tilimizni yuqorida berib o'tgan Oracle saytimizdan yuklab olamiz.

Rasmiy sayti: *https://www.java.com*

Yuklab olish manzili:

https://www.oracle.com/java/technologies /downloads/#java17

Yuklab olish uchun bergan saytimizga kiramiz. Kirganimizda birinchi bo'lib , shu shaklda bir ekran keladi:

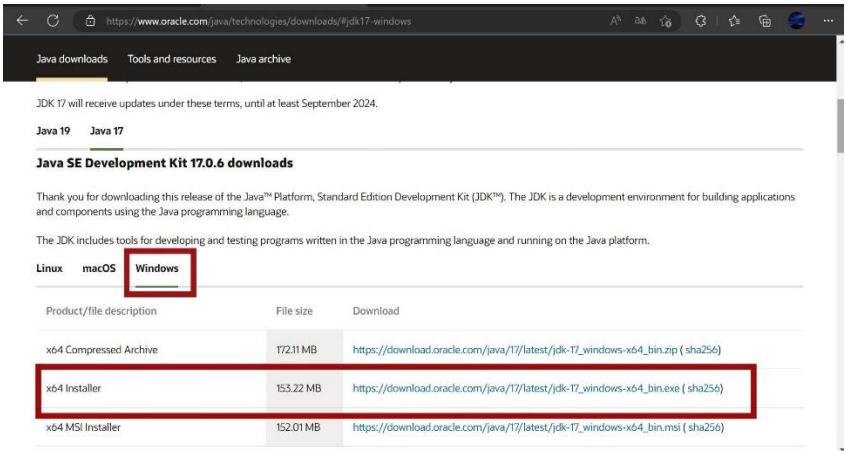


Qarshimizga kelgan bu ekranda, o'rnatadigan Java dasturlash tilimizni versiyasini va qanday operatsion tizimga o'rnatishni tanlagan holda davom etamiz.

Biz bu kitobimizdagi barcha kodlarni Java 17(LTS) versiyasi yordamida yozamiz. Shu sababli qarshimizga kelgan ekrandan Java 17ni va Windows (Linux, macOS)operatsion tizimlaridan birini(Operatsion tizimimizga ko'ra) tanlaymiz va davom etamiz.

2. Qadam

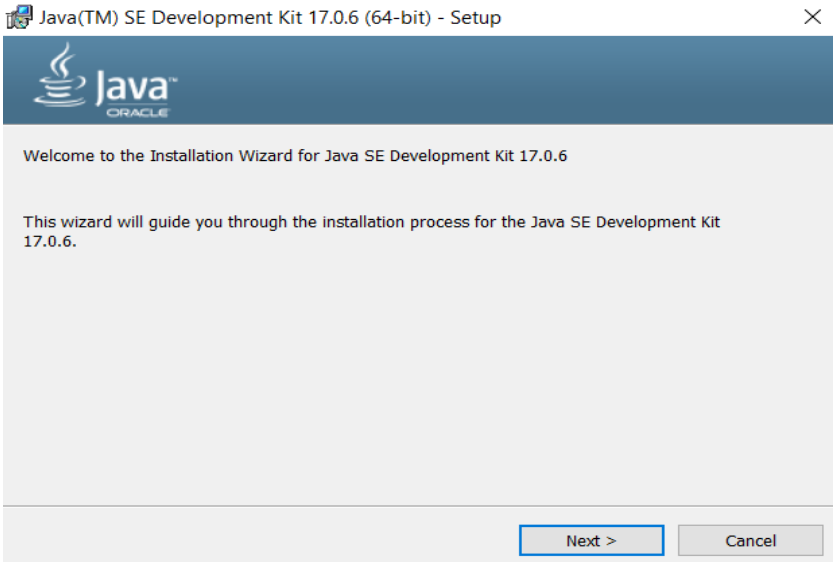
Kompyuterimizga ko’ra tanlovlarni qilganimizdan so’ngra, qarshimizga shunday bir ekran ochiladi.



Bu yerdan yuklab olishni amalga oshiramiz.

3. Qadam

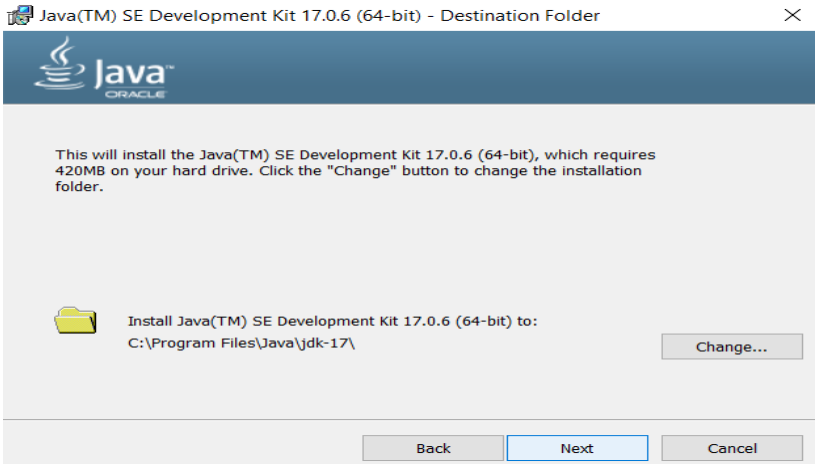
Yuklab olganimizdan keyingi qiladigan ishimiz Java dasturlash tilini kompyuterimizga o’rnatishdir.
Yuklab olgan faylimizni ochamiz. Ochganimizdan keyin qarshimizga shunday bir ekran keladi.



Bu ekrandan **<Next>** tugmamizni bosgan holda davom etamiz.

4. **Qadam**

<Next> tugmasini bosganimizdan keyin, qarshimizga bu shaklda bir ekran keladi.



Bu ekranda qila oladigan ishlarimizdan birinchisi **<Change>** tugmasi orqali o'rnatmoqchi bo'lgan Java dasturlash tilimiz uchun fayl joyini tanlashdir. O'rnatish davomida faylning joyini kompyuterimiz avtomatik shaklda programmalar uchun moslashtirilgan kompyuter faylimizni ko'rsatadi. Agar siz boshqa bir joyga o'rnatmoqchi bo'lganingizda **<Change>** tugmasi orqali almashtira olasiz.

Faylimiz uchun joy tanlaganimizdan keyin**, <Next>** tugmasini bosgan holda o'rnatishda davom etamiz.

5. Qadam

<Next> tugmasini bosganimizdan so'ngra Java dasturlash tilining fayllari tepada ko'rsatilgan joyga o'rnatilishni boshlaydi.

O'rnatish tugatilgandan keyingi keladigan ekran shu ko'rinishda bo'ladi:

Qarshimizdagi bu ekranda **<Next Steps>** va **<Close>** nomli 2 dona tugma bo'ladi. Biz ekranda **<Close>** tugmasini bosgan holda o'rnatish ishlarini tugatamiz. Shu on Java dasturlash tili kompyuterimizga o'rnatildi.

2.2 Java programmalarining tuzilish shakli

Java dasturlash tilida yozilgan barcha kodlarimizning fayl nomlari ***".java"*** bilan tugaydi. ***".java"*** yozilgan barcha fayllar bir Java dasturlash tilida yozilgan programma fayli sifatida ko'riladi. Agar programmamiz faqatgina 1 dona fayldan iborat bo'lsa, bu programmani ishlata olishimiz uchun u faylning ichkarisida mutlaqo bir **main()** metodi bo'lishi kerak. Agar programmamiz birdan ko'p fayllardan tashkil topayotgan bo'lsa, u fayllarning birining ichkarisida mutlaqo **main()** metodi bo'lishi kerak. **main()** metodi bo'lgan fayllar asosiy ona fayl

hisoblanadi ya’ni barcha yordamchi fayllarning bog’langan fayli.

Barcha dasturlash tillaridagi programmalarining tuzilish shakli shu deya olamiz.

1. Har bir programmaning bir kirishi bordir. Bular:
   a. Foydalanuvchi tarafidan kirilgan qiymatlar.
   b. Programma tarafidan tasodifi shaklda tayyorlangan qiymatlar.
   c. Ma’lumotlar omboridagi ma’lumotlar.
   d. Veb saytlaridan kelgan parametrlar.
   e. Bulardan tashqari qilinadigan ishga ko’ra kirilgan turli xil qiymatlar.
2. Programmalar tashqi tarafdan olgan qiymatlarini avval qisqa bir muddatga xotiraga saqlaydi va xotiraga saqlangan ma’lumotlar bilan algoritmlar yordamida ishlaydi. Programmaning eng ko’p vaqtini olgan qismlardan biri, bu qismdir.
3. Programma algoritmlarining ishlashi va ishlashi tugatilgan vaqtda foydalanuvchilarga bir chiqdi sifatida berilgan natijalardir. Har bir programmaning bir chiqdisi bo’lishi kerak degan narsa yo’q. Bu chiqdi berish bermaslik tamomiyla dasturchiga va programma shakliga bog’liqdir. Aytib o’tgan chiqdilarimiz shulardir deya olamiz.
   a. Ekranga chiqarilgan ma’lumotlar.
   b. Printerdan chiqarilgan narsalar.
   c. Programma ma’lumotlari saqlanadigan yoki saqlangan ma’lumotlar omborlari.
   d. Programma ishlashi natijasida ko’rsatiladigan turli xil natijalar.

2.3 Java programmalarining ishlash shakli

Java dasturlash tilida bundan so'ngra sekinlik bilan kodlar yozishni boshlaymiz va yozgan kodlarimizni, programmalarimizni ishlatib ko'ramiz va turli xil natijalar olamiz. Bu ishlarni bajarish davomida farqli-farqli savollarga duch kelishimiz mumkin. Ulardan biri shu bo'lishi mumkin "Bir programma qanday tayyorlanadi va ishga tushiriladi? " Bu mavzuda sizlar bilan bu savolning javobini o'rganamiz.

Programmalar ishlashi uchun asosiy 3 bosqichdan o'tishlari kerak.

1. Java dasturlash tili yordami bilan kodlarning yozilishi va .java fayli sifatida saqlash bosqichi.
   Java dasturlash tilidagi kodlarimizni turli xil IDElar yordamida yozamiz. Aytaylik kodimizni yozib tugatadik va uni "**Programma.java**" deb nomladik.
2. Bu bosqichda bizning yozgan kod fayllarimiz, **javac** buyrug'u bilan **bytecode**larga almashtiriladi. Almashtirilirgan **bytecode**lardan tashkil topgan yangi bir fayl ortaga keladi, u faylning nomi ***.class*** bilan tugaydi. Misol uchun "**Programma.java**" faylimizni **javac** buyrug'u bilan ishga tushuraylik.

   ```
   javac Programma.java
   ```

3. Bu bosqichda tayyorlangan ***.class*** nomi bilan tugagan faylni **java** buyrug'i bilan ishga tushiriladi. Bu bosqichda diqqat qilishimiz kerak bo'lgan narsa shudir, **java** buyrug'idan keyin fayl nomini yozishda oxiridagi ***.class'***ni yozmasdan davom etishdir.

```
java Programma.class (hato)
```

```
java Programma (to'g'ri)
```

2.4 Java dasturlash tili uchun eng yaxshi IDE

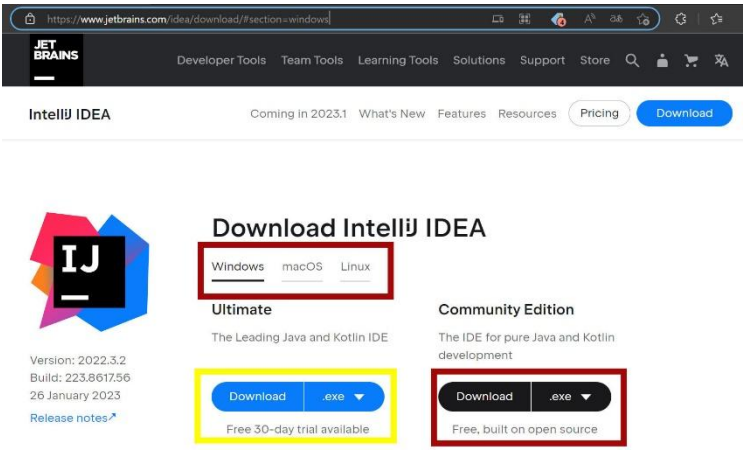
Bundan oldingi mavzularda ham aytib o'tganimiz kabi Java dasturlash tilidagi kodlarni yozish uchun IDE'larga ehtiyojimiz bor. Keling, eng avvalo IDE o'zi nima ekanligi haqida bir oz ma'lumotlarga ega bo'laylik.

IDE (Integrated Development Environment-Integratsiyalashgan rivojlanish muhiti), IDE'lar dasturchilarga programma tayyorlash davomida kerakli bo'ladigan barcha parametrlar va funksiyalar bilan to'ldirilgan bir programma turidir. IDE'lar dasturchilarga oson kod yozishlariga, va kod ichkarisidagi xatolarni osonlik bilan topishiga yordamchi bo'ladi deya olamiz.
Har bir dasturlash tili uchun moslashtirilgan turli xil IDE'lar mavjud. Java kodlarimizni yozishimiz uchun Intellij nomli IDEmizni tavsiya qila olaman. Tabiiyki, boshqa IDElarni ham foydalanishingiz mumkin, bu sizning qaroringiz. Kim uchun qaysi IDE qulay bo'lsa osha IDElardan foydalanishi mumkin. Biz kitobdagi Java kodlarimizni yozish uchun Intellijdan foydalanamiz. Bu sababli bu darsda Intellij IDEsini qanday yuklab olish haqida ma'lumot berib o'tamiz.

Intellij IDEmiz JetBrain kompaniyasi tegishli bir IDEdir. Bu sababli IDEmizni JetBrain saytidan yuklab olamiz.

Yuklab olish uchun internet manzili:
https://www.jetbrains.com/idea/download/

1. Qadam
   Yuqorida berib o'tgan internet manzilimizga kiramiz. Kirganimizda qarshimizga keladigan birinchi sahifa shu shaklda bo'ladi:



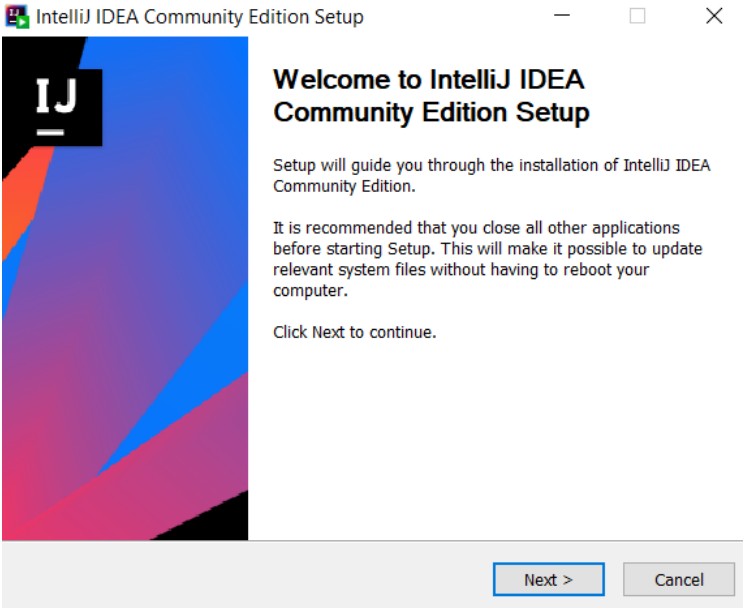
Bu ekranda ko'rib turganingiz kabi operatsion tizimingizni tanlaydigan bo'lim berilgan (yuqoridagi qizil to'rtburchak), bu yerdan operatsion tizimizni tanlaymiz. Tanlaganimizdan so'ng, 2 dona yuklab olish taklifi ko'rinadi (sariq va qizil to'rtburchaklar ichkarisidagi kabi). Biz bu yerdan Community (qizil to'rtburchak ichkarisidagi) variantini ya'ni bepul bo'lganini yuklab olgan holda davom etamiz. Siz istasangiz Ultimate (sariq to'rtburchak ichkarisidagi) variantini ham yuklay olasiz, faqat u varianti uchun har oy ma'lum bir miqdorda to'lov qilib borishingiz kerak bo'ladi.

Bu sahifadan 2 variantdan birini yuklab olgan holda davom etamiz.

2. **Qadam**

Intellij IDEsini yuklab olganimizdan keyin qiladigan ishimiz yuklab olgan Intellij programmamizni kompyuterimizga o'rnatishdir.

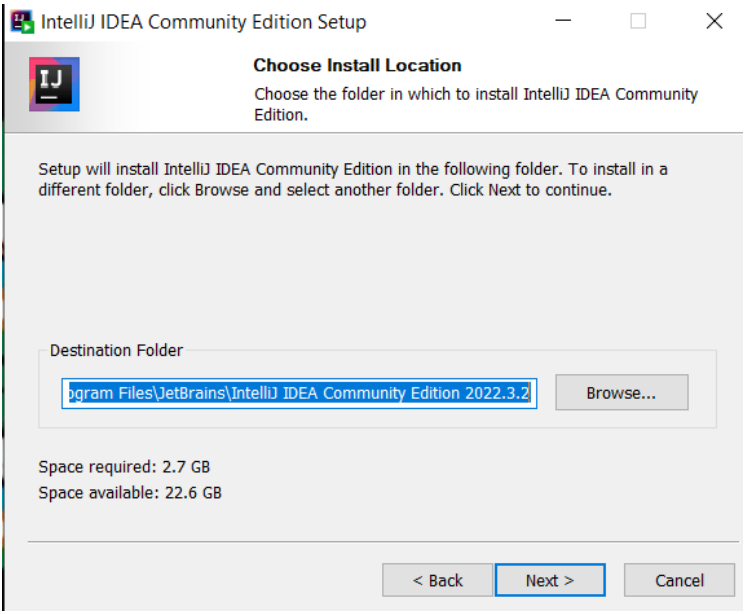
O'rnatishni boshlash uchun yuklab olgan faylimizni ochamiz. Ochganimizda qarshimizga shu shaklda bir ekran keladi:



Bu ekrandan **<Next>** tugmasini bosgan holda davom etamiz.

3. **Qadam**

<Next> tugmasini bosganimizdan so'ngra qarshimizga yangi bir ekran keladi. Shu shaklda:

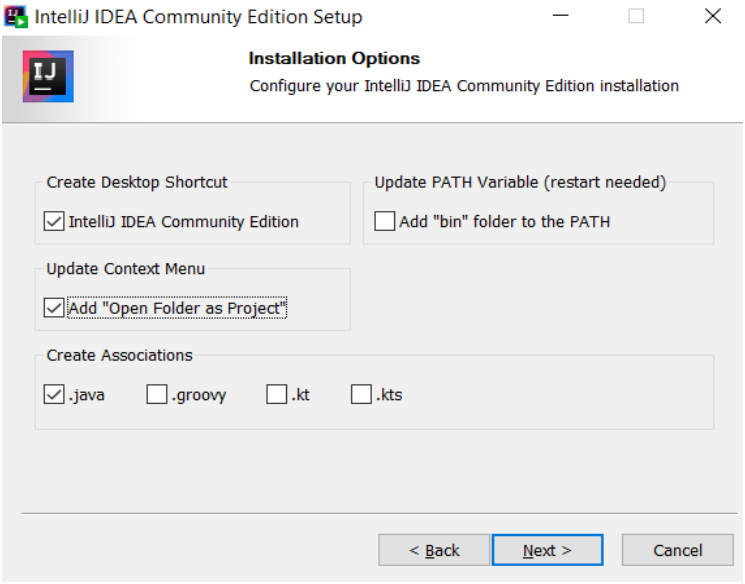


Bu ekranda ko'rib turganingiz kabi **<Browse>** nomli bir tugma ko'rinib turibdi. Bu tugma orqali biz o'rnatmoqchi bo'lgan programmamiz uchun fayl tanlay olamiz. kompyuter avtomatik bir shakl programma uchun fayl taklif qiladi, agar siz farqli bir joyga o'rnatmoqchi bo'lsangiz **<Browse>** tugmasi orqali o'zgartira olasiz.

Bu ekranda o'rnatmoqchi bo'lgan programmamiz ya'ni Intellij IDEmiz uchun joy tanlagandan so'ngra **<Next>** tugmasini bosgan holda davom etamiz.

4. **Qadam**

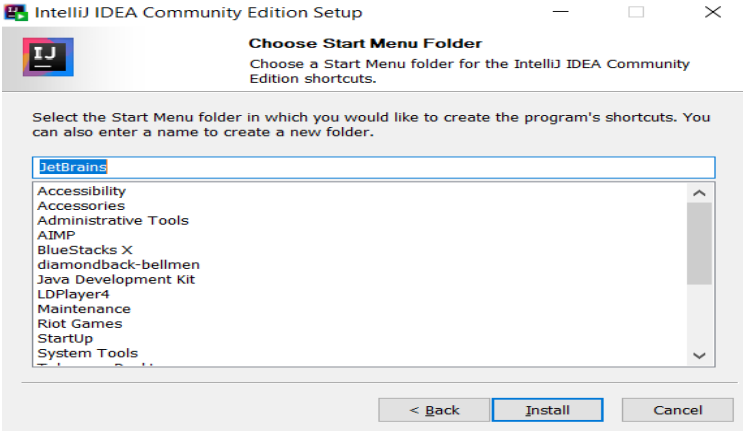
<Next> tugmasini bosganimizdan so'ngra qarshimizga yangi bir ekran keladi. Kelgan bu ekranimizda Intellij IDEmizni nima uchun o'rnatayotganimiz haqida bir oz ma'lumotlar kirishimiz kerak bo'ladi. Shu shaklda:



Bu ekranda bu shaklda ma'lumotlarni belgilaymiz va **<Next>** tugmasini bosgan holda davom etamiz.

5. **Qadam**

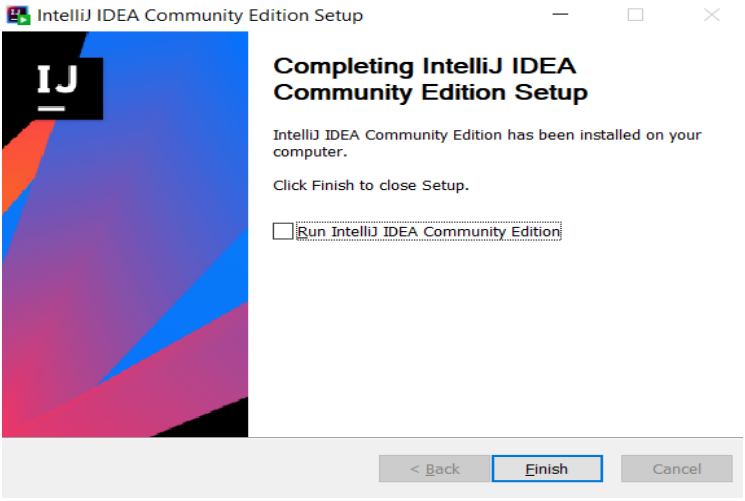
Keyingi keladigan ekranga e'tibor beradigan bo'lsak.



Bu ekranda azizlar bizdan nimalarni yuklab olish kerakligi haqida so'rayapti. Biz bu ekranda ham hech bir narsani o'zgartirmagan holda **<Install>** tugmasini bosamiz. O'zgartmaganimizning sababi azizlar shudir-bir oldingi ya'ni 4.qadamda nimalar bajarishimiz haqida ma'lumot berib o'tgan edik, bu sababdan o'zgartirmadik. Ya'ni programma 4.qadamdaki tanlovlarimizga e'tibor bergan holda yuklab olish ishlarini avtomatik shaklda bajaradi.

6. **Qadam**

Azizlar o'rnatish uchun amallarimizning oxirki qismiga yetib keldik. Eng so'ngda keladigan ekran shu ko'rinishda bo'ladi:



Bu ekranda azizlar o'rnatish tugatilgani haqida ma'lumot berilyapti. Bu sahifagacha kela olgan bo'lsangiz demakki muvaffaqiyatli bir shaklda programmani o'rnatib oldingiz deya olamiz.

Java dasturlash tilining asoslari 3

3.1 Foydalanuvchiga chiqdi berish

Har bir yangi dasturchilar, dasturlash tillarini o'rganishni boshlashganda birinchi o'rganadigan kodlari yoki yozadigan kodi ekranga "Salom dunyo" so'zlarini chiqarish bo'ladi. Bu programma kodlarini yozish davomida ekranga bir ma'lumot chiqarishni o'rganishadi aslida.

Har bir dasturlash tillarida ekranga turli xil ma'lumotlarni chiqarish uchun maxsus operatorlari bordir. Bu mavzuda sizlarga Java dasturlash tilidagi bu amalni bajaruvchi operatorlar System.out.println() System.out.print() haqida ma'lumot beramiz va qanday foydalanishimizni tushuntirib o'tamiz.

Tepada aytib o'tgan ikki operatorlarimiz (System.out.println() System.out.print()) bir xil amal bajarishadi, ya'ni ekranga bir ma'lumot chiqaradi. Bu ikki operatorlarimiz bir xil amal bajarishadi ammo bir xil natija ko'rsatmaydi. Chunki ularning ishlash shakllari bir-biridan farqli.

System.out.println() operatori ichkarisiga yozilgan qiymatni yoki ma'lumotni ekranga chiqarishga yordam beradi. Ekranga chiqarishda berilgan ma'lumotni avval chiqaradi va yangi bir satrga o'tib oladi. Programma bu operatordan keyin kodlarni yangi satrlardan foydalangan holda davom etadi *(1.0 kod*).

System.out.print() operatori ham ichkarisiga yozilgan qiymatni yoki ma'lumotni ekranga chiqarishga yordam beradi. Ammo ekranga ma'lumotni chiqarganidan keyin yangi bir satrga o'tmaydi. Programma ham undan keyin kodlarni eski satrdan davom etaveradi(*1.1 kod*).

Ikki operatorimiz orasidagi farqlarni keling kodlar orqali ham ko'rib chiqamiz.

```
public class Main {
    public static void main(String[] args) {
        System.out.println("Salom dunyo!");
    }
}
```

(1.0 kod)

Yuqoridagi kodimizni ishga tushirganimizda ekrandagi natijasi *(1.0 natija)* kabi bo'ladi.

```
Salom dunyo!


Process finished with exit code 0
```

(1.0 natija)

Yuqorida bir **System.out.println()** uchun yozilgan bir kodning ishlash shaklini ko'rdik. Endi esa bir **System.out.print()** uchun kod yozib uni ham ishlash shaklini ko'raylik.

```java
public class Main {
    public static void main(String[] args) {
        System.out.print("Salom dunyo!");
    }
}
```

1.1 kod

Kodimizni yozib oldik endi esa qanday bir javob olishimizni ko'raylik.

```
Salom dunyo!

Process finished with exit code 0
```

1.1 natija

Ikki operatorimiz uchun ham kodlarimizni yozib ishlatib ko'rdik, keling endi natijalariga qarab farqlarini ko'rib o'tamiz. *1.0 natija*ga qaraydigan bo'lsak, ishlash shakli quyidagicha :

1. **Qadam**
   "Salom dunyo!" so'zini y ozdirdi
2. **Qadam**
   Bir bosh satr ochgan
3. **Qadam**
   Va eng oxirida kodni ishlashini to'xtatgan

*1.1 natija*da esa kod bu shaklda ishladi:

1. **Qadam**

   "Salom dunyo!" so'zini yozdirdi.

2. **Qadam**
   Va kodni ishlashini to'xtatdi.

3.2 Yozgan kodlarimizga izohlar kiritish

Normal hayotda insonlar qiladigan ishlarini oldindan ro'yxatlarini va rejalarini tuzishadi va bu rejalariga oid izohlar yozishadi ya'ni buni shu uchun qilishim kerak yoki buni shuning uchun qildim kabi. Bularning barchasi unutib qo'yganimizda, bizga yordamchi bo'lishi uchundir. Dasturchilar ham xuddi shu shaklda, kodlari uchun izohlar yozishadi ya'ni bu kodni bu uchun yozdim yoki bu kod bu ishni bajaradi kabi. Bu izohlarning barchasi tabiiyki kodlarimiz yozilgan fayl ichkarisida bo'ladi. Biz yozgan bu izohlar dastur tarafidan ko'rinmaydi.

Java dasturlash tilida qanday qilib izohlar yozamiz deysizmi?

Keling qanday qilib kodlarimiz ichkarisiga dastur tarafidan ko'rinmaydigan bir shaklda izoh yozishni ko'rib chiqamiz.

Javada izohlarni yozishimizda bizga kerakli bo'ladiganlar shu ikki xarakterlardir : **/** va ***** . bu ikkisidan foydalangan holda kodlarimizning izohlarini yoza olamiz. Endi esa, qanday foydalanishimiz mumkin ekanligi haqida ko'rib chiqaylik.

1. **Usul**
   // bu ikki belgidan so'ngra yozish orqali.
   Tepada berilgan ikki dona bo'luv belgisidan so'ngra yozilgan har qanday so'zlar va kodlar bir izoh sifatida ko'riladi.
2. **Usul**
   /* va ***/** bu ikki belgining orasida yozish usuli.
   Tepada berilgan ikki belgi orasida yozilgan har qanday so'zlar va kodlar ham bir izoh sifatida ko'riladi.

Javada izohlarni tepada berib o'tilgan 2 usulda yoziladi.

3.3 O’zgaruvchilar

O’zgaruvchilar nima? deb bir savol so’ralganda har bir insonning aqliga birinchi bo’lib keladigan javob bu – “Doimiy shaklda o’zgaruvchi narsalar” keladi. Dasturlash olamida ham o’zgaruvchilar mavjud. Keling birgalikda dasturlash olamidagi “O’zgaruvchilar nima?” degan savolning javobini o’rganib chiqamiz.

O’zgaruvchilar – barcha dasturlash tillarida ma’lumotlarni yoki qiymatlarni qisqa bir muddatga xotiraga saqlab olishimizga yordam beradi. O’zgaruvchilarning ham turlari mavjud.

Java dasturlash tilidagi o‘zgaruvchi turlari mavjud. Ular:

1. **Butin sonlar**(int , byte, short, long)
2. **Matnsal** (String)
3. **Kasrli sonlar**(Double, Float)
4. **Belgi turlari**(Char)
5. **Mantiqiy True-False turi**(Boolean)

Butin sonlar

Birinchi o’zgaruvchi turlarimiz (int , byte, short, long) butin sonlarni (Integer) ko’rsatish uchun foydalaniladi. Raqam oralig’iga va xotiradan oladigan joyiga ko’ra bir-biridan farq qiladi.

1. **Byte**
   Bu o’zgaruvchi turimiz butin sonlarni saqlab turuvchi o’zgaruvchi turlari ichkarisida eng kuchugidir. Xotiradan **8 bit’**lik bir joy egallaydi. Har bir butin son o‘zgaruvchi turlarining raqam oraliqlari mavjud. Byte

o'zgaruvchi turimiz , faqatgina **-2^7** va **2^7 -1** orasidagi , ya'ni **(-128** va **127** orasidagi) butin sonlarni xotirada tuta oladi.

2. **Short**
   Bu o'zgaruvchi ham butin sonlarni xotirada tutishimizda yordam beradi. Raqam oralig'iga keladigan bo'lsak **-2^{15}**dan **2^{15} -1**gacha ya'ni **(-32768**dan 32767gacha) bo'lgan sonlardir.
3. **Int**

   Bu o'zgaruvchi turi o'rtacha uzunlikdagi butin sonlarni xotirada saqlab tura oladi. Bu o'zgaruvchi turimiz eng ko'p foydalanilgan o'zgaruvchilardan biridir va xotiradan **32 bit'**lik joy oladi. Bu turdagi o'zgaruvchilar **-2^{31}** va **2^{31}-1** (**-2147483648** va **2147483647**) orasidagi sonlarni xotirada saqlay oladi.

4. **Long**
   Bu o'zgaruvchi turimiz butin sonlarni saqlovchi o'zgaruvchilar ichinda eng katta oraliqli hisoblanadi. Ya'ni biz xotiraga oladigan sonimiz 2 milliarddan katta bo'lsa, demakki bu o'zgaruvchi turiga ehtiyojimiz bor. Xotiradan oladigan joyi **64-bit**dir. Raqam oralig'i esa **-2^{63}**dan **2^{63} -1** orasi(boshqa so'z bilan aytganda **-9223372036894775808** va **9223372036894775807**) orasidagi barcha butin sonlardir.

Matnsal o'zgaruvchi turlari

Matnsal o'zgaruvchi turlari bu bir necha turdagi o'zgaruvchilarni bir joyda matn sifatida qabul qilgan holda xotirada tutuvchi o'zgaruvchi turidir. Matnsal o'zgaruvchilarni **String** orqali ko'rsatamiz. Keyingi mavzularda chuqurroq

o’rganib chiqamiz. Hozir sizning bilishingiz kerak bo’lgan narsa matnsal qiymatlarni **String** orqali ko’rsatish kerakligidir.

Kasrli sonlar

Bu turdagi o’zgaruvchi turlarimiz Kasrli sonlarni(10.3, 34.567 kabi sonlarni) tanitishga yordam beradi. Bu o‘zgaruvchilarni dastur ichida ko’rsatish uchun **Double** va **Float** o’zgaruvchi turidan foydalanamiz. **Double** va **Float** bir-biridan xotiradan oladigan joyi va qiymat oralig’i tarafidan farq qiladi.

1. **Float**
   Float o’zgaruvchi turi **Double** o’zgaruvchi turiga qaragan kichikroq qiymatlar oladi. Xotiradan **32-bit**lik bir joy egallaydi. Oladigan qiymat ya’ni raqam oralig’i **$-3.4x10^{38}$**dan **$3.4x10^{38}$**gachadir va nuqtadan keyin 7 xonagacha son olishi mumkin. Bu o’zgaruvchimiz ko’pgina qiymatlarni tutish uchun yetarli. Shu sababli **Float** o’zgaruvchi turimiz **Double** o’zgaruvchi turiga qaraganda ko’p foydalaniladi.
2. **Double**
   Double o’zgaruvchi turimiz orqali tanitilgan barcha qiymatlar xotiradan **64-bit**lik joy ya’ni **8-bayt** joy egallaydi. Bu o’zgaruvchi turimiz bizga Float raqam oralig’idan katta sonlarni xotiraga olishimizga yordam beradi. Double o’zgaruvchi turimizning raqam oralig’i **$-1.7x10^{308}$**dan **$1.7x10^{308}$**gacha bo’lgan sonlardir, **Doubl**e o‘zgaruvchilarini ko‘rsatishda nuqtadan keyin maksimum **17 xonagacha** son bera olamiz.
   Misol uchun 27.23549678 sonini xotiraga vaqtinchalik olish uchun Float o‘zgaruvchi turimizdan foydalana olmaymiz, chunki berilgan sonda nuqtadan keyin 8

xonali bir raqam mavjud, **Float** nuqtadan so'ngra maksimum 7 xonali son oladi. Bu sababli bu raqamni xotiraga vaqtinchalik saqlash uchun **Double** o'zgaruvchi turidan foydalanishimiz kerak.

Har xil turdagi belgilar uchun o'zgaruvchi turimiz

Har xil turdagi belgilar deganda azizlar $,#,@,&,% kabi belgilarni o'z ichiga olgan qiymatlarni nazarda tutayapmiz. Aslida bu turdagi qiymatlarni ko'rsatish uchun String o'zgaruvchidan ham foydalansak bo'ladi. Lekin Java dasturlash tilida bu kabi belgilarni ichkarisiga olgan turli xil qiymatlarni ya'ni so'zlar, raqamlarni xotiraga olish uchun **Char** o'zgaruvchidan foydalaniladi. **Char** o'zgaruvchi turi xotiradan **16-bit**lik joy egallaydi. Bu o'zgaruvchimiz shuningdek String qiymatlarini ham xotiraga vaqtinchalik saqlay oladi.

Mantiqiy qiymat javoblari uchun o'zgaruvchi turi

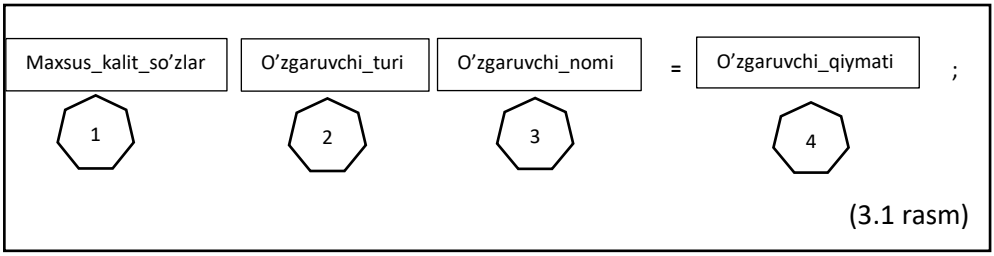
Bu safar azizlar sizlar bilan Boolean o'zgaruvchisini o'rganamiz. Boolean o'zgaruvchi turimiz bizga **True** va **False** qiymatlarini ko'rsatadi. Aytaylik programmadan biz shunday bir narsa so'radik olti beshdan katta mi? , programma bizga agar olti soni beshdan katta bo'lsa **True** javobi beradi, aks holda **False** javobini beradi. Bu yerdagi berilgan **True** va **False Boolean** o'zgaruvchisi orqali bizga berilmoqda.

Yuqorida ko'rib o'tgan barcha o'zgaruvchi turlarimizni keyingi mavzularda misollar ustida ham ko'rib chiqamiz.

O'zgaruvchilarga qiymatlar berish

Biz o'zgaruvchilarga qiymatlar bergan holda u qiymatlarni xotiraga qisqa bir muddatga saqlaymiz.

Java , **type-safe** (tur ishonchli) bo'lgan bir tildir. Ya'ni Javada har o'zgaruvchiga qiymatlar berishda mutlaqo o'zgaruvchining turini berishimiz kerak. Bu o'zgaruvchi turlari, tepada aytib o'tgan o'zgaruvchilarimizdan biri bo'la olgani kabi dasturchining o'zi tayyorlagan bir sinf ham bo'lishi mumkin yoki JDK'da entegre sifatida kelgan har qanday bir sinf bo'lishi mumkin. Bir o'zgaruvchiga shu shaklda qiymat berishimiz mumkin:



(3.1 rasm)

Tepadagi rasmni (3.1 rasm) ko'rib chiqadigan bo'lsak, uning 4 dona bo'limi mavjud:

1. **Maxsus_kalit_so'zlar**: *public, private, protected, static, final* kabi har xil Java dasturlash tilidagi maxsus kalit so'zlardir. Bu bo'lim istakka bog'liqdir va maxsus bir holat bo'lmasa bo'sh tashlab ketish mumkin. Bu kalit so'zlar qaysi ma'noda kelishini keyingi mavzularda batafsil o'rganib o'tamiz.
2. **O'zgaruvchi_turi**: Java type-safe turidagi bir dasturlash tili bo'lganligi uchun har bir o'zgaruvchilarni turini berishga majburmiz. Bu o'zgaruvchi turlari yuqorida ko'rib chiqilgan o'zgaruvchi turlari yoki sinflardir. Sinflar mavzusini o'rganish davomida, bir sinfning nomi ham o'zgaruvchi turiga to'g'ri kelishi mumkin ekanligini ko'rib chiqamiz.

3. **O‘zgaruvchi_nomi**: O‘zgaruvchi nomi bu – bir o’zgaruvchiga dasturchi tarafidan berilgan bir isimdir. Bu isim yoki nom, faqatgina bir so’zdan tashkil topadi. Tabiiyki, o‘zgaruvchilarga nom berishda ham qoidalar mavjud. Bu qoidalar quyidagilar:
   - O‘zgaruvchi nomlari orasida bosh joy bo’lmasligi kerak.
   - O‘zgaruvchi nomlari Javadagi har qanday bir kalit so’z bo’lishi mumkin emas. Javadagi kalit so’zlarni **3.2 jadval**da ko’rishingiz mumkin.
   - O‘zgaruvchi nomlari raqam bilan boshlanishi mumkin emas, ammo boshiga farqli bir harf yoki belgi yozganimizdan keyin raqamlar ham yozsak bo’ladi. Misol: b123 , _123 kabi

O’zgaruvchi turi	Avtomatik beriladigan qiymatlari
Byte	0
Short	0
Int	0
Long	0L
Float	0.0f
Double	0.0d
Char	‘\u0000’
String	Null
Boolean	False

3.1 jadval

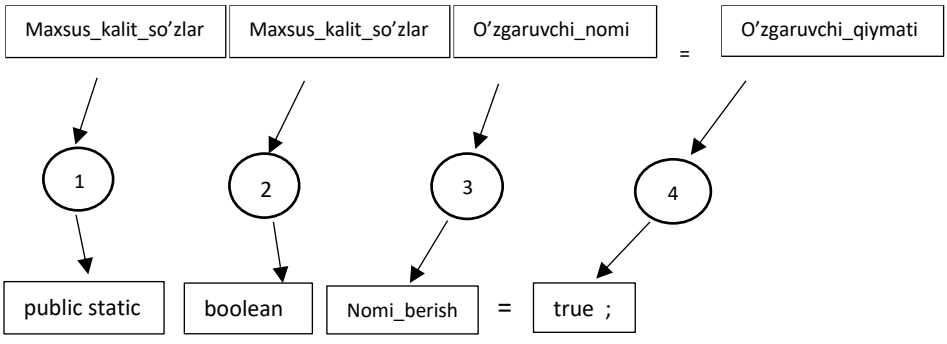
4. **O‘zgaruvchi qiymati:** o‘zgaruvchilarning qiymatlari mutlaqo berilgan o‘zgaruvchi turiga to‘g’ri kelishi

kerak. Ya'ni biz o'zgaruvchini qiymatini berishdan oldin uning turini **int** tanlagan bo'lsak, unga bir butin son berishga majburmiz. Agar o'zgaruvchi turini va nomini bergan holda unga qiymat bermasak avtomatik shaklda 3.1 jadvaldagi kabi qiymat oladi.

Abstract	*byte*	*Else*
defualt	*throws*	*Instanceof*
If	*public*	*Return*
Private	*import*	*Transient*
This	*double*	*Case*
Boolean	*break*	*Extends*
Do	*throw*	*Int*
Interface	*static*	*Void*
Char	*finally*	*Long*
Strictfp	*volatile*	*Class*
Float	*native*	*Super*
While	*for*	*Const*
New	*switch*	*Continue*
Goto	*package*	*Synchronized*
Try	*catch*	*Final*
Implements	*protected*	*Short*

3.2 jadval

Yuqorida ko’rib chiqqanlarimizga misollar beradigan bo’lsak: (3.2 rasm va 1.2 kod).



3.2 rasm

```
1:
2: int a = 5;     // a isimda  va int o‘zgaruvchi turida bir o‘zgaruvchi ber
                  //  qiymati  5 bo’lsin.
3: char c1 = ‘#$’   // c1 nomli va char turidagi bir o’zgaruvchi tayyorla va
                       // qiymati #$ bo’lsin.
4: String z = “ITIMUS Academy”  //  nomi z bo’lgan va String o‘zgaruvchi turida
//Bo’lgan o‘zgaruvchi tayyorla va uning qiymati ITIMUS Academy bo’lsin.
```

1.2 kod

3.4 Bir o‘zgaruvchi turidan boshqa bir o‘zgaruvchi turiga o’tish

Bir o’zgaruvchi turidan boshqa bir o’zgaruvchi turiga o’tishga Java dasturlash tili ruxsat beradi. Misol uchun, aytaylik bizda **Float** o’zgaruvchi turida bir qiymat mavjud va bizga bu qiymatning verguldan keyingi qismi kerak emas faqatgina butin qismi kerak. Bu vaziyatda biz berilgan qiymatimizni **int** yoki boshqa bir **integer** turiga o’tkazganimizda, Java avtomatik bir shaklda u sonning butin qismini oladi. O’tkazishning ham qoidalari mavjud. Yuqoridagi misolimizni Javada kod sifatida yozadigan bo’lsak:

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

        float y = 4.56F;
        int x= (int)y;
        System.out.println(x);

}
}
```

1.3 kod

Yuqoridagi (1.3 kod) kodimizni ishga tushirganimizda 4 javobini olamiz.

Yuqoridagi (1.3 kod) kodimizni ishlash shaklini qadam-baqadam ko‘rib chiqsak:

1. **Qadam**
   O’tgan mavzularda aytib o’tganimiz kabi Java dasturlash tilidagi barcha kodlarimiz bir sinf ichkarida

yoziladi. Bizning ham bu yozgan kodimiz bir sinf ichida yozilgan. Birinchi satrda bir **Main** nomli sinf ochdik.

2. **Qadam**
   Biz yozgan kodlarimizni ishga tushira olishimiz uchun kodlarimizning ichkarisida main() metodi bo'lishi kerak edi, o'tgan mavzularda aytib o'tdik. Shu sababli ikkinchi qatorimizda main() metodini ham yozdik.
3. **Qadam**
   To'rtinchi qatorimizda esa float o'zgaruvchi turidagi y nomli bir o'zgaruvchi tanitdik va qiymatini 4.56 ekanligini yozdik. Bu yerda sizlarga bir narsani yana aytib o'tishim kerak: *Float o'zgaruvchi turidagi o'zgaruvchiga qiymat berayotganda qiymatimizni yozganimizdan keyin f yoki F dan birini yozishimiz kerak yuqoridagi kabi.*
4. **Qadam**
   Beshinchi qatorda int turidagi x nomli bir o'zgaruvchi tanitdik va uni qiymatini esa y nomli o'zgaruvchining butin qismiga teng ekanligini aytib o'tdik va shu usulda o'zgartirish amalimizni bajardik. Oltinchi qatorga keladigan bo'lsak, bu qatorda x o'zgaruvchining qiymatini ekranga chiqarishini xohladik.

Int o'zgaruvchi turidan String o'zgaruvchi turiga o'tish.

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

      int x = 15;

      // 1- usul
      String y1 = String.valueOf(x);
```

```
        // 2- usul
        String y2 = x + "";

        // 3- usul
        String y3 = Integer.toString(x);
}
}
```

Bu kodimiz ham yuqoridagi kodimiz kabi ishlaydi va 3 xil usulda berilgan int o‘zgaruvchi turidagi o‘zgaruvchini String turiga almashtiradi.

String o‘zgaruvchi turidan Int o‘zgaruvchi turiga o’tish.

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

        String x = " 15 ";

        // 1- usul
        int y1 = Integer.valueOf(x);
        // 2- usul
        int y3 = Integer.parseInt(x);
}
}
```

Double yoki Float o‘zgaruvchi turidan String o‘zgaruvchi turiga o’tish.

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

        double x = 15.1234;
        float x2 = 15.1234f;

        // 1- usul
        String y =String.valueOf(x);
        String y2=String.valueOf(x2);
```

```
        // 2- usul
        String y3 = x+"";
        String y4 = x2+"";

        // 3-usul
        String y5 = Double.toString(x);
        String y6 = Float.toString(x2);

}
}
```

String o‘zgaruvchi turidan Double yoki Float o‘zgaruvchi turiga o’tish.

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

        String x="12.34";
        // 1-usul
          double y1=Double.valueOf(x);
          float y2=Float.valueOf(x);

          //2-usul
          double y3=Double.parseDouble(x);
          float y4=Float.parseFloat(x);
}
}
```

3.5 O‘zgaruvchilarni takrorlash

O’zgaruvchilar mavzusini ham oxiriga keldik azizlar. Bu mavzuda sizlar bilan birgalikda bugungacha o’rgangan barcha mavzularimizni bir dastur kodini yozgan holda, qaytadan o‘rganib chiqamiz.

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

        byte byte_turi = 6;
        short short_turi = 12;
        /* 2 dona byte va short turida
        o'zgaruvchi tanitdik va qiymatlarini
        berdik */
        System.out.println("byte turidagi
                o'zgaruvchi ="+ byte_turi);

        System.out.println("short turidagi
                o'zgaruvchi ="+ short_turi);

        int int_turi=123;
        long long_turi= 64765;
        System.out.println("int turidagi
                o'zgaruvchi ="+ int_turi);

        System.out.println("long turidagi
                o'zgaruvchi ="+ long_turi);

        int x = byte_turi+short_turi;
        System.out.println("Byte + short= "
                        +x);

        float float_turi= 12.4f;
        double double_turi= 12.34;
        System.out.println("float turidagi
                o'zgaruvchi="+float_turi);

        System.out.println("double turidagi
                o'zgaruvchi="+double_turi);

        double y=float_turi+double_turi;
        System.out.println("float + double = "
                +y);

        String string_turi="ITIMUS_Academy";
        char char_turi='$';
```

```
        /* String turidagi bir o'zgaruvchiga
        qiymat berishda juft qo‘shtirnoqlardan
        foydalaniladi ya’ni "ITIMUS_Academy "
        char turidagi bir o'zgaruvchiga qiymat
        berishda birlik bo'lgan
        qo‘shtirnoqlardan foydalaniladi ya’ni
        '$ '
         */
        System.out.println("String turidagi
                o'zgaruvchi = "+string_turi);

        System.out.println("char turidagi
                o'zgaruvchi = "+char_turi);

}
}
```

1.3 kod

Yuqoridagi kodimizni ishga turishganimizda, quyidagi natijalarni olamiz:

```
byte turidagi o'zgaruvchi =6

short turidagi o'zgaruvchi =12

int turidagi o'zgaruvchi =123

long turidagi o'zgaruvchi =64765

Byte + short= 18

float turidagi o'zgaruvchi=12.4

double turidagi o'zgaruvchi=12.34

float + double = 24.739999618530273

String turidagi o'zgaruvchi = ITIMUS_Academy

char turidagi o'zgaruvchi = $
```

Yuqoridagi 1.4 kodimizga GitHub manzilimizdan erisha olasiz.
Manzil:
https://github.com/21040001/Java_yordamchi_kitob_kodlarimiz/tree/main/O'zgaruvchilar

Java dasturlash tilida operatorlar 4

4.1 Operatorlar

Operatorlar barcha dasturlash tillarida mavjud bo'lib, o'zgaruvchilar ustida turli xil amallarni bajarishimizda va ularning qiymatlarini o'zgartirishimizga yordam beradi. Java dasturlash tilida operatorlarni istaganimizcha foydalana olamiz. Agar Java dasturlash tilidagi operatorlarni qiladigan ishlariga ko'ra guruhlantiradigan bo'lsak:

1. **Qiymat operatori**
2. **Arifmetik operatorlar**
3. **Arifmetik qo'shish/ayirish operatorlari**
4. **Arifmetik qiymat berish operatorlari**
5. **Taqqoslash operatorlari**
6. **Mantiqiy operatorlari**
7. **Ikkilik operatorlari**
8. **?: operatorlari**

Birgalikda bu operatorlarni ko'rib chiqaylik.

4.2 Qiymat operatori (=)

Bu operatorimiz bir o‘zgaruvchiga qiymat berishimizda yordam beradi. Agar bu operator uchun bir formula yozadigan bo’lsak, quyidagicha bo’ladi:

```
<o’zgaruvchi> = <qiymat>
```

Misol beradigan bo’lsak:

```
int x = 10;

String y = “ITIMUS Academy” ;
```

Yuqorida berib o’tgan misolimizdagi kabi bu operatorning o’ng tarafiga qiymat , chap tarafiga o‘zgaruvchi beriladi. Ya’ni bir so’z bilan aytadigan bo’lsak bir o‘zgaruvchining qiymati nimaga teng ekanligini ko’rsatib turadi.

4.3 Arifmetik operatorlar (+,-,*,/,%)

Bu operatorlardan foydalangan holda, o‘zgaruvchilar ustida turli xil matematik amallar qo‘shish, ayirish, bo’lish, ko‘paytirish, qoldig‘ini olish kabi amallar bajaramiz.

Qo‘shish operatori(+)

Bu operator yordamida biz ikki o‘zgaruvchi qiymatlarini yoki o‘zgaruvchilarni qo’sha olamiz, ya’ni matematikadagi qo‘shish amali bilan bir xil. Buni ham bir formula holatiga keltiradigan bo’lsak:

```
<O’zgaruvchi> = <qiymat_1> + <qiymat_2>;
```

Misol beradigan bo‘lsak:

```
int a  =  10,  b = 16 ;

int c = a + b;

int d = c + 2;

System.out.println(“a=” + a +“, b=”+b+ “, c=”+c+”, d  = ”+d)
```

Bu kodimiz ishga tushirilganda ekranimizga quyidagicha javob keladi:

```
a=10, b=16, c=26, d= 28
```

Ayirish operatori(-)

Bu operatorimiz ikki o‘zgaruvchi orasidagi farqlarni hisoblaydi, ya’ni chap tarafida berilgan qiymatdan, o’ng tarafidagi qiymatni ayiradi va orasidagi farqni topishimizga yordam beradi. Buni ham bir formula holatiga keltiradigan bo’lsak.

```
<o’zgaruvchi> = <qiymat_1> - <qiymat_2>;
```

Misol beradigan bo’lsak:

```
int a  =  30,  b = 15 ;

int c = a - b;

int d = c - 2;

System.out.println(“a=” + a +“, b=”+b+ “, c=”+c+”, d  = ”+d)
```

Bu kodni ishga tushirganimizda, quyidagicha javob olamiz:

```
a=30, b=15, c=15, d= 13
```

Ko‘paytirish operatori (*)

Bu operatorimiz berilgan ikki o’zgaruvchi qiymatini ko‘paytirish uchun foydalaniladi. Formula holatiga keltiradigan bo’lsak:

```
<o’zgaruvchi> = <qiymat_1> * <qiymat_2>
```

Misol beradigan bo’lsak:

```
int a= 5 ;

int b=10 ;

int c= a*b;

int d= c*2;

System.out.println( “a=”+a+ “, b=”+b+ “, c=”+c+ “, d=”+d)
```

Bu kodimizni ishga tushirganimizda, quyidagicha bir javob olamiz:

```
a=5, b=10, c=50, d= 100
```

Bo’lish operatori (/)

Bo’lish operatori berilgan matematikadagi bo’lish amalini bajarishga yordam beradi. Bu operatorning chap tarafida kelgan sonni o’ng tarafida kelgan songa bo’ladi. Formula holatiga keltiradigan bo’lsak:

```
<o’zgaruvchi>= <qiymat_1> / <qiymat_2>;
```

Misol beradigan bo’lsak:

```
int a= 50 ;

int b=10 ;

double c= a/b;

double d= c/2;

System.out.println( “a=”+a+ “, b=”+b+ “, c=”+c+ “, d=”+d)
```

Bu kodimizni ishga tushirganimizda, quyidagicha bir javob olamiz:

```
a=50, b=10, c=5, d= 2.5
```

Bu operatordan foydalanishda amalni bajargandan keyin chiqadigan o‘zgaruvchi turini Double yoki Float tanlash mantiqiy chunki bo’lish amali bajarilgandan keyin chiqqan javob kasrli son ham bo’lishi mumkin.

Qoldiq olish operatori (%)

Bu operatordan biz bir o’zgaruvchi qiymatini boshqa o’zgaruvchi qiymatiga bo’lganimizda qoladigan qoldiq sonni olish uchun foydalanamiz. Bu amalni bajarish uchun bir formula tayyorlaydigan bo’lsak :

```
<o’zgaruvchi> = <qiymat_1>  %  <qiymat_2>;
```

Bu formula orqali tushuntiradigan bo'lsak, <o'zgaruvchi> , <qiymat_1>ni <qiymat_2>ga bo'lganda qoldiqqa teng bo'ladi.

Misol beradigan bo'lsak:

```
int a  =  33,  b = 15 ;

int c = a % b;

int d = c % 2;

System.out.println("a=" + a +", b="+b+ ", c="+c+", d  = "+d)
```

Yuqoridagi kodimizni ishga tushiradigan bo'lsak, quyidagicha javob olamiz:

```
a=33, b=15, c=3, d= 1
```

Bu yerda dasturimiz a va b o'zgaruvchilarni qiymatini avval vaqtinchalik xotiraga saqladi, keyin esa a nomli o'zgaruvchini b nomli o'zgaruvchiga bo'ldi va qolgan qoldiqni olib c nomli o'zgaruvchiga qiymat sifatida berib, uni ham vaqtinchalik xotiraga old. So'ngra vaqtinchalik xotiraga olgan c o'zgaruvchini qiymatini olib uni 2ga bo'ldi va qolgan qoldiq sonni d nomli o'zgaruvchi qiymat sifatida berdi. Keyin esa barcha xotiraga olgan o'zgaruvchilarni ekranga chiqardi. Bu yuqoridagi kodimizning ishlash shakli edi.

4.4 Arifmetik qo‘shish/ayirish operatorlari (++, - -)

Bu operatorlarimiz o‘zgaruvchi qiymatining ong yoki chap tarafida yoziladi va u qiymatga avtomatik shaklda 1 qo‘shadi yoki 1 ayiradi. Keling bularni ko’rib chiqaylik.

Avtomatik shaklda qo‘shish operatori(++)

Bu operator o‘zgaruvchining o’ng tarafida yoki chap tarafida bo’ladi va u o‘zgaruvchining qiymatiga 1 qo’shadi. Bu operatorni foydalanish formulasi quyidagicha:

```
<o’zgaruvchi> = ++ <qiymat> ;

<o’zgaruvchi> = <qiymat> ++ ;
```

Misol berib o’tadigan bo’lsak:

```
int a = 12;

int b = 17;

++a ;

b++ ;

System.out.println(“a=”+a+ “, b=”+b);
```

Bu kodimizni ishga tushirganimizda, quyidagicha bir javob olamiz:

```
a=13, b=18
```

Avtomatik shaklda ayirish operatori(--)

Bu operator (2 dona ayiruv operatori) o‘zgaruvchining o’ng tarafida yoki chap tarafida bo’ladi va u o‘zgaruvchining qiymatidan 1 ayiradi. Bu operatorni foydalanish formulasi quyidagicha:

```
<o’zgaruvchi> = -- <qiymat> ;

<o’zgaruvchi> = <qiymat> -- ;
```

Misol berib o’tadigan bo’lsak:

```
int a = 12;

int b = 17;

--a ;

b -- ;

System.out.println(“a=”+a+ “, b=”+b);
```

Bu kodimizni ishga tushirganimizda, quyidagicha bir javob olamiz:

```
a=11, b=16
```

Avtomatik qo‘shish va ayirish operatorlarini qiymat oldida va orqasida kelishi orasidagi farqlar.

Ishlash shakli bir xil, ammo ham bir qo’shib ham bir o‘zgaruvchiga tenglamoqchi bo‘lganimizda bir biridan boshqacha shaklda foydalaniladi. Keling buni bir misolda ko‘raylik. *(4.1 kod)*

```
public class avto_qoshish_va_ayirish_daturi {
    public static void main(String[] args){
        int x = 12;
        int y = 77;
        //farqlarni ko'rishni boshladik

        int x_1 = ++x;
        /* yuqoridagi qatorda shunday dedik
        avval x qiymatiga 1 qo'sh keyin esa
        uni x_1 nomli o'zgaruvchiga qiymat
        sifatida ber. */
        int y_1 = --y;
        /* yuqoridagi kod qatorimizda shunday
        dedik avval y qiymatidan 1 ayir keyin
        esa uni y_1 nomli o'zgaruvchiga qiymat
        sifatida ber. */
        System.out.println("x="+x+",
                y="+y+",x_1="+x_1+
                ",y_1="+y_1);
        int x_2 = x++;
        /* yuqoridagi kod qatorimizda dasturga
        Shunday buyruq berdik: avval x nomli
        o'zgaruvchi qiymatini x_2 nomli
        o'zgaruvchi qiymati sifatida ol keyin
        esa x nomli o'zgaruvchi qiymatiga 1
        qo'shib qo'y. */

        int y_2 = y--;
        /* yuqoridagi kod qatorimizda dasturga
        shunday buyruq berdik: avval y nomli
        o'zgaruvchi qiymatini y_2 nomli
        o'zgaruvchi qiymati sifatida ol keyin
        esa y nomli o'zgaruvchi qiymatidan 1
        ayirib  qo'y. */
        System.out.println("x="+x+
                ",y="+y+",x_2="+x_2+
                ", y_2="+y_2);
    }}
```

4.1 kod

Yuqoridagi (4.1 kod) kodimizni berilgan GitHub manzilidan yuklay olasiz.

Manzil:
https://github.com/21040001/Java_yordamchi_kitob_kodlarimiz/tree/main/operatorlar

Keling endi yuqorida yozgan (4.1 kod) kodimizni ishga tushirib ko'ramiz. Ishga tushirganimizda azizlar quyidagicha bir javob olamiz:

```
x=13, y=76,x_1=13, y_1=76

x=14, y=75,x_2=13, y_2=76


Process finished with exit code 0
```

4.5 Arifmetik qiymat berish operatorlari (+= , -= , *= , /= , %=)

Java dasturlash tilida jami bo'lib 5 dona arifmetik qiymat berish operatorlari mavjud. Bu operatorlari bizga bir o'zgaruvchi ustida bir arifmetik amalni bajarib, olgan javobni qaytadan o'zgaruvchiga qiymat sifatida berishga yordam beradi. Keling yaxshisi bu operatorlarni ko'rib chiqaylik.

Qo'sh va qiymat ber operatori (+=)

Bu operatorimiz berilgan o'zgaruvchiga, berilgan qiymatni qo'shadi va chiqqan javobni qaytadan shu o'zgaruvchiga qiymat sifatida beradi. Formula holatiga keltiradigan bo'lsak:

```
<o’zgaruvchi> += <qiymat>
```

Bu shunday degani azizlar:

```
<o’zgaruvchi> = <o’zgaruvchi> + <qiymat>
```

Bu operatorning ishlatilishiga misol beradigan bo’lsak:

```
int a = 19;

int a += 1;   //  kodni ochiqlaydigan bo’lsak shu shaklda: a=a+1

System.out.println(“a=”+a)

// Bu yerda a ning qiymatini ekranga yozishini istadik.
```

Yuqorida berib o’tgan misolimizni ishga tushirganimizda shunday bir olamiz:

```
a=20
```

Ayir va qiymat ber operatori (-=)

Bu operatorimiz berilgan o‘zgaruvchi qiymatidan berilgan qiymatni ayiradi so‘ngra olgan javobini o‘zgaruvchiga qaytadan qiymat sifatida beradi. Formula holati shu shaklda bo’ladi:

```
<o’zgaruvchi> -= <qiymat>
```

Bu formulani ochiqlaydigan bo’lsak:

```
<o’zgaruvchi> = <o’zgaruvchi> - <qiymat>
```

Bu operatorimizga ham bir misol berib o’taylik:

```
int a = 16;

int a -= 1;   //  kodni ochiqlaydigan bo'lsak shu shaklda: a=a-1

System.out.println("a="+a)
```

Bu misolimizni ishga tushirganimizda , quyidagicha javob olamiz:

```
a=15
```

Ko'paytir va qiymat ber operatori (*=)

Bu operatorimiz berilgan o'zgaruvchini berilgan qiymatga ko'paytirib, chiqargan qiymatini qaytadan shu o'zgaruvchiga qiymat sifatida beradi. Formula shaklda ko'radigan bo'lsak:

```
<o'zgaruvchi> *= <qiymat>
```

Bu formulani ochiqlaydigan bo'lsak:

```
<o'zgaruvchi> = <o'zgaruvchi> * <qiymat>
```

Keling bu operatorimizni ham misol ustida ko'rib chiqaylik.

```
int a = 16;

int a *= 2;   //  kodni ochiqlaydigan bo'lsak shu shaklda: a=a*2

System.out.println("a="+a)
```

Bu misolimizni ishga tushirganimizda , quyidagicha javob olamiz:

```
a=32
```

Bo’l va qiymat ber operatori (/=)

Bu operatorimiz berilgan o‘zgaruvchini berilgan qiymatga bo’ladi va chiqargan qiymatini qaytadan shu o‘zgaruvchiga qiymat sifatida beradi. Formula shaklda ko’radigan bo’lsak:

```
<o’zgaruvchi> /= <qiymat>
```

Bu formulani ochiqlaydigan bo’lsak:

```
<o’zgaruvchi> = <o’zgaruvchi> / <qiymat>
```

Keling bu operatorimizni ham misol ustida ko‘rib chiqaylik.

```
int a = 16;

int a /= 2;  //  kodni ochiqlaydigan bo’lsak shu shaklda: a=a/2

System.out.println(“a=”+a)

// bu yerda ekranga a qiymatini yozishini istadik
```

Bu misolimizni ishga tushirganimizda , quyidagicha javob olamiz:

```
a=8
```

Bo’l va qoldiq sonni qiymat sifatida ber operatori (%=)

Bu operatorimiz berilgan o‘zgaruvchini berilgan qiymatga bo‘ladi va bo‘lganida qolgan qoldiq sonni qaytadan shu o‘zgaruvchiga qiymat sifatida beradi. Formula shaklda ko’radigan bo’lsak:

```
<o’zgaruvchi> %= <qiymat>
```

Bu formulani ochiqlaydigan bo’lsak:

```
<o'zgaruvchi> = <o'zgaruvchi> % <qiymat>
```

Keling bu operatorimizni ham misol ustida ko‘rib chiqaylik.

```
int a = 13;

int a %= 2;

 //  kodni ochiqlaydigan bo'lsak shu shaklda: a=a%2

System.out.println("a="+a)
```

Bu misolimizni ishga tushirganimizda , quyidagicha javob olamiz:

```
a=1
```

4.6 Taqqoslash operatorlari (> ,< , >=, <=, ==, !=)

Taqqoslash operatorlari dasturchilarga berilgan ikki qiymat orasidagi turli xil vaziyatlarni ko'rishlarida yordamchi bo'ladi. Misol uchun, dasturchiga ikki qiymat berildi 5 va 6 qiymatlari. Dasturchidan berilgan ikki sonni taqqoslashini istadi, bu vaziyatda dasturchi taqqoslash operatorlaridan foydalanadi. Taqqoslash operatorlari javoblarni **Boolean** o‘zgaruvchi turida ko‘rsatadi, ya'ni **true** yoki **false** shaklida. Bu operatorlarni ko'rib chiqadigan bo'lsak:

Kichikmi operatori (<)

Bu operatorimiz dasturdan shunday bir savol so'raydi: " **<** operatorining chap tarafida berilgan qiymat o'ng tarafida berilgan qiymatdan kichikmi? ". Dastur javobni shu shaklda

beradi: Agar chap tarafda berilgan qiymat rostdan ham kichik esa **true** javobini beradi, kichik bo'lmasa **false** javobini beradi.

Katta mi operatori (>)

Bu operatorimiz esa dasturdan bu shaklda bir savol so'raydi: " **>** operatorining chap tarafida berilgan qiymat o'ng tarafida berilgan qiymatdan katta mi ? ". Dastur javobni shu shaklda beradi: Agar chap tarafda berilgan qiymat rostdan ham katta bo'lsa **true** javobini beradi, katta bo'lmasa **false** javobini beradi.

Katta yoki teng operatori (>=)

Bu operatorimiz dasturdan shu shaklda bir savol so'raydi: " **>=** operatorining chap tarafida berilgan qiymat o'ng tarafida berilgan qiymatdan katta yoki teng mi? ". Dastur javobni shu shaklda beradi: Agar chap tarafda berilgan qiymat rostdan ham katta yoki teng bo'lsa **true** javobini beradi, katta yoki teng bo'lmasa **false** javobini beradi.

Kichik yoki teng operatori (<=)

Bu operatorimiz ham dasturdan shunday bir savol so'raydi azizlar: " **<=** operatorining chap tarafida berilgan qiymat o'ng tarafida berilgan qiymatdan kichik yoki teng mi? ". Dastur ham javobni **true** va **false** orqali beradi: Agar chap tarafda berilgan qiymat rostdan ham kichik yoki teng bo'lsa **true** javobini beradi, kichik yoki teng bo'lmasa **false** javobini beradi.

Teng mi operatori (==)

Bu operatorimiz esa dasturdan shunday bir savol so'raydi " **==** operatorining chap tarafida berilgan qiymat va o'ng

tarafida berilgan qiymat teng mi? “. Dastur javobni shu shaklda beradi: Agar chap tarafda berilgan qiymat va o’ng tarafda berilgan qiymat teng bo’lgan holatida, **true** javobini, teng bo’lmagan holatida esa **false** javobini beradi.

Teng emas mi operatori (!=)

Bu operatorimiz ham dasturdan shu shaklda bir savol so’raydi “ **!=** operatorining chap tarafida berilgan qiymat va o’ng tarafida berilgan qiymat teng emas mi? “. Dastur javobni shu shaklda beradi: Agar chap tarafda berilgan qiymat va o’ng tarafda berilgan qiymat teng bo’lmasa, **true** javobini, teng bo’lsa **false** javobini beradi.

Azizlar barcha taqqoslash operatorlarini qanday shaklda ishlashini o’rganib chiqdik. Keling endi ularni barchasini misol kodlar ichida ko’rib, o’rganib chiqamiz.

```
public class taqqoslash {
    public static void main(String[] args){
        int a= 13;
        int b=3;
        boolean d= a>b;
        System.out.println("a soni b sonidan
                katta mi: "+d);
        boolean c= a<b;
        System.out.println("a soni b sonidan
                kichik mi: "+c);
        boolean l= a>=b;
        System.out.println("a soni b sonidan
                katta yoki teng mi: "+l);
        boolean w= a<=b;
        System.out.println("a soni b sonidan
                kichik yoki teng mi: "+w);
        boolean u= a==b;
        System.out.println("a va b soni
                teng mi: "+u);
```

```
        boolean p= a!=b;
        System.out.println("a va b soni
                teng mi: "+p);

    }
}
```

Ushbu kodimizni GitHub manzilimiz orqali yuklay olasiz azizlar.

Manzil:
https://github.com/21040001/Java_yordamchi_kitob_kodlarimiz/blob/main/operatorlar/taqqoslash.java

Yuqorida taqqoslash operatorlari uchun yozgan kodimizni ishga tushirganimizda, quyidagicha javob olamiz:

```
a soni b sonidan kattami: true

a soni b sonidan kichikmi: false

a soni b sonidan katta yoki tengmi: true

a soni b sonidan kichik yoki tengmi: false

a va b soni tengmi: false

a va b soni tengmi: true


Process finished with exit code 0
```

Keling bu kodimizni ishlash shaklini qadam-baqadam ko’rib chiqamiz.

1. **Qadam**
   Birinchi bo'lib dasturimiz kodlarni 1- qatordan o'qishni boshlaydi. 1-qatorni o'qib dasturning **sinf** nomini o'rganadi va 2-qatorga o'tadi va u yerdagi **main()** metodini ko'rib bu kod faylni asosiy kod fayl sifatida ishga tushiradi.
2. **Qadam**
   Ikkinchi qadamda esa 3- 4- qatorlarda berib o'tilgan 2 **int** turidagi o'zgaruvchini vaqtinchalik xotiraga saqlaydi va davom etadi.
3. **Qadam**
   Bu bosqichda ya'ni 5-qatorda **boolean** turidagi d nomli bir o'zgaruvchi tayyorlaydi va dasturdan **a>b** deya so'raydi. Keyin esa olgan javobini **boolean** turidagi d nomli o'zgaruvchiga qiymat sifatida beradi.
4. **Qadam**
   6- qatorda esa **boolean** turidagi d nomli o'zgaruvchini qiymatini ekranga yozdiradi.
   Qolgan qatorlar ham uchinchi va to'rtinchi qator kabi ishga tushadi va davom etadi.

4.7 Mantiqiy operatorlari

Java dasturlash tilida asosiy 4 dona mantiqiy operatorlar mavjud, bular : **And, Or, Xor, Not** operatorlaridir. Bu operatorlarni o'rganishimizdan oldin bu operatorlarning bir - birlari bilan bo'lgan munosabatlarini jadval asosida o'rganib chiqaylik. (*4.1 jadval*)

x	y	x and y (x && y)	x or y (x || y)	x xor y (x ^ y)	Not x (! x)	Not y (! y)
false	false	false	false	false	true	true
false	true	false	True	true	true	false
true	false	false	True	true	false	true
true	true	true	True	false	false	false

(4.1 jadval)

And (va) operatori (&&)

Bu operatorimiz chap va o'ng tomonlarida berilgan qiymatlarni mantiqiy **and** ifodasiga tashlaydi. Ya'ni, agar **and**ning chap va o'ng tomonlarida berilgan qiymatlarning ikkisi ham **true** yoki **false** bo'lsa **true** javobini chiqaradi. Bu haqida 4.1 jadvalda batafsil berib o'tilgan.

Or (yoki) operatori (| |)

Bu operatorimiz chap va o'ng tomonlarida berilgan qiymatlarni mantiqiy **or** ifodasiga tashlaydi. Boshqa so'z bilan aytganda, bu operatorning ikki tarafida berilgan qiymatlarning ikkisi ham **false** bo'lganda javob **false** bo'ladi. Qolgan barcha holatlarda javob true bo'ladi.

Not (emas) operatori (!)

Bu operatorimiz o'zidan keyin berilgan qiymatni teskarisini oladi. Ya'ni bu operatordan keyin kelgan qiymat **true** esa **false** javobini oladi.

Xor operatori (^)

Bu operatorimiz ikki tarafida ya'ni chap va o'ng taraflarida berilgan qiymatlar farqli-farqli bo'lganda **true** javobini beradi. Bir xil bo'lganda esa **false** javobini beradi.

Mantiqiy operatorlarni yanada chuqurroq o'rganish uchun keling bir misol kod yozib ishlash shaklini ko'rib chiqamiz.

```
boolean x = true;

boolean y = false;

System.out.println("x && y = "+ (x&&y));

System.out.println("x & y = "+ (x&y));

System.out.println("x | | y = "+ (x||y));

System.out.println("x |  y = "+ (x|y));

System.out.println("x ^ y = "+ (x^y));

System.out.println("!x = "+ (!x));
```

Yuqorida berilgan kodimizni ishga tushiradigan bo'lsak quyidagicha javob olamiz.

```
x&&y= false

x&y= false

x||y= true

x|y= true

x^y = true

!x = false
```

Yuqori e'tibor beradigan bo'lsak, && va & ni, || va | operatorlarining javoblari bir xil shaklda. Ya'ni && ning yeriga & buni, || ning yeriga | ni foydalana olamiz.

4.8 Ikkilik operatorlari (Bitwise)

Bu operatorlar ham mantiqiy operatorlar bilan deyarli bir xil. Ammo, bu operatorlar ishlashni boshlashdan oldin barcha qiymatlarni 2lik sanoq sistemasiga (0 va 1)larga almashtirgan holda ishlashni boshlaydi va u shaklda davom etadi. Asosiy Bitwise operatorlari 5 dona:

And operatori(&)

Bu operator chap va o'ng taraflarida kelgan qiymatlarni 2 sanoq sistemasiga o'tkazib , ular ustida mantiqiy **and** amalini bajaradi. Pastda berilgan misolda ko'rishingiz mumkin:

```
byte a = 5;  // 2lik sanoq sistemasida :  101

byte b = 3;  // 2lik sanoq sistemasida :  011

byte d= a&b ;  // javob 2lik sanoq sistemasida : 001
```

Or operatori(|)

Bu operator ham chap va o'ng taraflarida kelgan qiymatlarni 2 sanoq sistemasiga o'tkazib , ular ustida mantiqiy **Or** amalini bajaradi. Misol uchun bir kod yozadigan bo'lsak:

```
byte a = 5;  // 2lik sanoq sistemasida :  101

byte b = 3;  // 2lik sanoq sistemasida :  011

byte d= a | b ;  // javob 2lik sanoq sistemasida : 111
```

Xor operatori(^)

Bu operator ham chap va o'ng taraflarida kelgan qiymatlarni 2'lik sanoq sistemasiga o'tkazib olgan holda , ular ustida mantiqiy **Xor** amalini bajaradi. Misol uchun bir kod yozadigan bo'lsak:

```
byte a = 5;  // 2lik sanoq sistemasida :  101

byte b = 3;  // 2lik sanoq sistemasida :  011

byte d= a ^ b ;  // javob 2lik sanoq sistemasida : 111
```

Complement (almashtirish) operatori(~)

Bu operator biz berilgan qiymatni 2'lik sanoq sistemasiga o'tkazadi va chiqargan qiymatidagi 1larni 0'ga , 0'larni 1ga almashtiradi. Misol beradigan bo'lsak:

```
byte a = 5;  // 2lik sanoq sistemasida :  101

byte d= ~a ;  // javob 2lik sanoq sistemasida : 010
```

Siljitish operatorlari(<<, >>)

Bu operatorlar ham avvalo berilgan qiymatni 2'lik sanoq sistemasiga olib o'tadi. Va berilgan qiymatga teng shaklda o'ng yoki chap tarafga siljitadi. Misolni ko'rganimizda yanada yaxshi tushunamiz.

```
byte a = 5;  // 2lik sanoq sistemasida :  101

byte b =a>> 2;  // 2lik sanoq sistemasida :  00101

byte d= a << 1;  // javob 2lik sanoq sistemasida : 1010
```

4.9 ?: operatori

Bu operatorimiz Java dasturlash tilidagi 3 argument oladigan operatordir. Bu kabi boshqa operator yo'q. Bu operatorda 1 o'zgaruvchiga shartli qiymat beriladi . Tushunishimiz yanada oson bo'lishi uchun, keling foydalanish shaklini va ishlash shaklini ko'rib chiqaylik:

```
int x = 11, y = 5, z=0, z1=0;

z = x > y  ?  x : y;

z1 = x < y ?  x : y ;

System.out.println("z : "+ z);

System.out.println("z1 : "+ z1);
```

Yuqoridagi misol uchun yozilgan kodimizning 2- qatorida yozilgan kodimizning ma'nosi shu shaklda: Agar x o'zgaruvchi y o'zgaruvchidan katta bo'lsa z o'zgaruvchining qiymati x bo'lsin, agar katta bo'lmasa z o'zgaruvchining qiymati y bo'lsin. Yuqoridagi kodimizni ishga tushirganimizda quyidagicha javob olamiz:

```
z : 11

z1 : 5
```

Kod bloklari 5

5.1 Kod bloklari nima?

Java dasturlash tilidagi **if, for, while** kabi blokli kod ifodalarini o‘rganishdan oldin kod bloklari nima ekanligini o’rganib o’taylik. Java dasturlash tilida kod bloklari (**{ .....}**) shunday qavslar orasida yozilgan holatda kod bloki ekanligini ko’rsatib turadi. Bu shokildaki qavslar orasida berilgan yoki yozilgan kodlar faqatgina o‘zining darajasidagi yoki bir past darajadagi kod bloklarini o‘qiy oladi. Kod bloklarini va kod bloklarini o’qish shaklini tushunish uchun pastda berib o’tilgan misolni ko’rib chiqishingiz mumkin.

```
{ Daraja-1
        … Bu yerda daraja birning kodlari yoziladi …
        { Daraja – 2
            …Bu yerda daraja ikkining kodlari yoziladi …
        Daraja-2 }
Daraja-1 }
```

Kod bloklari haqida bir oz bo'lsa ham tushuncha hosil qilib oldik deb o'ylayman. Bu mavzumizni ya'ni kod bloklarini keyingi mavzularda kodlar ichkarisida foydalanganimizda batafsil tushunib o'rganib o'tamiz.

5.2 IF shart operatori

Dasturlash davomida turli xil o'zgaruvchi qiymatlarini yoki qiymatlarni turli xil shartlar asosida ishlatish ehtiyojini his qilasiz. Bu holatda bizga If shart operatori yordam beradi. Deyarli barcha dasturlash tillarida if shart operatori mavjud. Dasturchilar qiymatlar bilan ishlashda if shart operatoridan ko'p foydalanadi. Bu sababdan, dasturlashning muhim qismlaridan bir deya olamiz. Java dasturlash tilida if shart operatorini foydalanish shakllari, past qisimda umumiy formula shaklda berib o'tildi.

```
// Usul 1
if ( Bu qavs ichkarisida shart berib o'tiladi )
        shart_bajarilganda_ishga_tushadigan_kodlar;
```

Berilgan shart bajarilganda ishga tushishi kerak bo'lgan kod bir dona bo'lganda, if shart operatorini pastda berilgan misoldagi kabi bir qatorda yoza olamiz.

```
// Usul 2
if ( shart ) shart_bajarilganda_ishga_tushadigan_kod;
```

Agar if shart operatoridagi shart bajarilganda ishga tushishi kerak bo'lgan kodlar birdan ko'p bo'lsa, if operatori kod

bloklaridan foydalanilgan holda yozilishi kerak. Pastda berilgan misol kabi:

```
// Usul 3
if ( shart ) {
        shart_bajarilganda_ishga_tushadigan_kod1;
        shart_bajarilganda_ishga_tushadigan_kod2;
        shart_bajarilganda_ishga_tushadigan_kod3;
        ……
}

// Usul 4
if ( shart )
 {
        shart_bajarilganda_ishga_tushadigan_kod1;
        shart_bajarilganda_ishga_tushadigan_kod2;
        shart_bajarilganda_ishga_tushadigan_kod3;
        ……
}
```

Java dasturlash tilida if shart operatorini yuqorida berib o’tilgan 4 usulda yoza olamiz. Dasturlash tarafidan qaraydigan

bo’lsak, if shart operatorini kodimiz ichkarisida foydalanayotganimizda 3 va 4 usullardan foydalanish ma’qul ko’rinadi. Chunki, 3 va 4 usullarda blok holatiga keltirgan holatda yozamiz, bu sababli kodlarimizni o’qilishi osonlashadi. If shart operatoridagi shart bajarilganda ishga tushishi kerak bo’lgan kod bir dona bo’lsa ham 3 va 4 usullarni foydalana olamiz. Pastki qismda foydalanishga misol berib o’tildi.

```
// Usul 3
if ( shart ) {
        shart_bajarilganda_ishga_tushadigan_kod;
}

// Usul 4
if ( shart )
 {
        shart_bajarilganda_ishga_tushadigan_kod;
}
```

Foydalanish shaklini ko’rib chiqdik azizlar. Endi esa keling birgalikda if shart operatoridan foydalangan bir misol kodni ko’rib chiqaylik. Ko’rib chiqish davomida if shart operatorini ishlash shaklini ham o’rganib chiqamiz.

```
public class if_shart {
    public  static void main(String[] args){
        int a = 20;
        int b = 15;

        if (a>b){
            System.out.println("Berilgan a
             soni b sonidan katta");
            System.out.println("Berilgan ikki
             sonning yig'indisi = " + (a+b));
        }


    }
}
```

5.1 kod

Yuqorida berib o'tilgan 5.1 raqamli kodimizni ishga tushirganimizda, quyidagicha javob olamiz:

```
Berilgan a soni b sonidan katta

Berilgan ikki sonning yig'indisi = 35
```

5.1 natija

If shart operatori ishlatilgan bir kod (5.1 kod) misol berib o'tdik va uni ishga tushirgan holda natijalarimizni oldik. Ammo if shart operatorini ishlash shaklini ko'rib chiqmadik, keling azizlar en yaxshisi yuqorida berilgan 5.1 kod orqali if'ning ishlash shaklini qadam-baqadam ko'rib chiqamiz.

1. **Qadam**
   Kodimizni ishga tushirganimizda birinchi va ikkinchi qatorlardagi kodlarni e'tiborga olgan holda kodimiz ishga tushishni boshlaydi.
2. **Qadam**

Bu qadamda esa, dasturlash tilimiz ikki dona int turidagi a va b nomli o‘zgaruvchilarni yaratadi. So’ngra esa, bu ikki o‘zgaruvchiga mos shaklda 20 va 15 qiymatlarini berdi.

3. **Qadam**

Bu qadamda esa dasturimiz if shart blogimizni ishga tushiradi. Yuqoridagi kodimizda if shart blogimiz oltinchi qatorda boshlanib, o’n birinchi qatorda tugamoqda. If blogimizning ishlashiga e’tibor beradigan bo’lsak , dasturimiz birinchi bo’lib oltinchi qatorda berilgan if blogimizning shartini tekshiradi ya’ni shu shartni (a>b) tekshiradi. If so’zimiz bu yerda “agar” ma’nosi keladi azizlar. Bu sababdan dasturimiz oltinchi qatorni shaklda o’qiydi: “ Agar berilgan a nomli o’zgaruvchi b nomli o’zgaruvchidan katta bo’lsa” deb o’qiydi.

Dasturimiz shartni tekshirib chiqargan javobiga ko’ra blok ichkarisidagi kodlarni ya’ni yettinchi va to’qqizinchi qatordagi kodlarni ishga tushiradi. Ya’ni agar oltinchi qatorda berilgan shart bajarilsa ya’ni berilgan a sonimiz b sonidan katta bo’lsa yettinchi va to‘qqizinchi qatorlardagi kodlarni ishga tushiradi. Agar a nomli o‘zgaruvchining qiymati b nomli o’zgaruvchining qiymatidan katta bo’lmasa yettinchi va to‘qqizinchi qatordagi kodlarni ishga tushirmasdan tashlab ketadi.

Yuqoridagi misolimizda 5.1 kodda a nomli o‘zgaruvchining qiymati b nomli o’zgaruvchining qiymatidan katta bo‘lgani uchun dasturimiz blok ichkarisidagi kodlarni ishga tushirdi va 5.1 natijani berdi.

If shart operatoridan foydalanish: Hisoblash.java

Hozir sizlar birgalikda if operatoridan foydalangan holatda tayyorlangan **Hisoblash.java** nomli bir programmaning kodini ko'rib chiqamiz. **Hisoblash.java** nomli kodimizni berilgan GitHub manzilimizdan yuklay olasiz.

Manzil:
https://github.com/21040001/Java_yordamchi_kitob_kodlarimiz/blob/main/Hisoblash.java

Hisoblash.java

```
import java.util.Random;
import java.util.Scanner;

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

        Random random = new Random();
        double hisobim = random.nextInt(2000);

        System.out.println("Hisobingizdagi pul
               miqdori = " + hisobim);

        System.out.print("Hisobingizdan
               yechmoqchi bo'lgan pul
               miqdoringizni kiring : ");

        Scanner scanner =
                       new Scanner(System.in);

        double yechiladigan_pul =
                       scanner.nextDouble();

        if (hisobim>yechiladigan_pul){

            hisobim -= yechiladigan_pul;
```

```
            System.out.println("Hisobingizdan
                yechib olingan pul miqdori = "
                + yechiladigan_pul);

            System.out.println("Hisobingizda
                qolgan pul miqdori = "
                + hisobim);
        }

    }
}
```

5.2 kod

Ushbu kodimizni ishga tushirganimizda quyidagicha ishlaydi va shunga o‘xshash bir javob chiqarib beradi:

```
Hisobingizdagi pul miqdori = 162.0

Hisobingizdan yechmoqchi bo'lgan pul miqdoringizni
kiring : 100

Hisobingizdan yechib olingan pul miqdori = 100.0

Hisobingizda qolgan pul miqdori = 62.0
```

5.2 natija

Yuqorida berilgan **Hisoblash.java** nomli kodimizda sizga notanish bo’lgan kutubxonalar va sinflar mavjud, bu sababli kodimizni ishlash shaklini va u yerda foydalanilgan barcha notanish kodlarni birgalikda ko’rib chiqaylik. Yuqoridagi 5.2 kodimizning 7 va 8 qatorlarida Java dasturlash tilida mavjud bo’lgan **Random** nomli sinfdan foydalangan holda, 0 va 2000 sonlari orasidan bir sonni tasodifan olayapti va olgan u sonini double turidagi bir o‘zgaruvchiga aylantirib *hisobim* nomli bir o’zgaruvchiga qiymat sifatida berdi. Programmamiz har ishga

tushirilganda *hisobim* nomli o'zgaruvchining qiymati o'zgaradi. Programmamiz ishga tushirilganda foydalanuvchi tarafidan hisobdan yechib olinadigan pul miqdori kirilishi kerak azizlar, bu vaziyatda bizga Java dasturlash tilidagi **Scanner** nomli sinfi bizga yordam beradi. Bu sinfdan foydalangan holda 17 va 21 qatorlarda foydalanuvchidan double turidagi bir qiymat oladi va olgan qiymatini *yechiladigan_pul* nomli double turidagi o'zgaruvchiga qiymat sifatida beradi va uni xotiraga oladi.

Yuqoridagi **Hisoblash.java** kodimizdagi bizga notanish bo'lgan sinflarni ko'rib chiqdik azizlar !

Yuqoridagi 5.1 kodimizdagi if shart operatori haqida qisqacha ma'lumot berib o'taman. Bu kodimizda if shart blogimiz ichkarisidagi kodlarimiz ishga tushishi uchun *hisobim* nomli o'zgaruvchi qiymati *yechiladigan_pul* nomli o'zgaruvchi qiymatidan katta bo'lishi kerak azizlar ! Agar *hisobim* nomli o'zgaruvchi qiymati *yechiladigan_pul* nomli o'zgaruvchi qiymatidan katta bo'lmasa 22 va 30 qatorlar orasidagi kodlar ya'ni if kod blogimiz ichkarisidagi kodlar ishga tushirilmasdan programma ishlashi to'xtaydi. Yuqoridagi kodimiz bu shaklda ishlaydi. Yuqorida berilgan manzil orqali kodimizni yuklab olib o'zingiz ishlatib ko'rganingizda tushunishingiz osonlashadi deb o'ylayman.

5.3 If-Else shart operatorlari

If shart operatoridan foydalanganimizda e'tibor bergan bo'lsangiz, faqatgina if shart blogi ichkarisidagi shart true ekanligida bir natija olayapmiz. Ammo dasturlash davomida bizdan if shart blogi ichkarisidagi shart true bo'lganda farqli natija , false bo'lganda farqli bir natija chiqarishini istaydigan

vaziyatlar ham bo'ladi. Bu vaziyatdan chiqishimizga if-else shart blok operatorlari yordamchi bo'ladi. Bu operatorlarning ham foydalanish shaklini ko'rib chiqadigan bo'lsak:

```
// Usul 1
if ( shart ) {
        shart_bajarilganda_ishga_tushadigan_kod;
}
else{
        shart_bajarilmaganda_ishga_tushadigan_kod;
}

// Usul 2
if ( shart )
 {
        shart_bajarilganda_ishga_tushadigan_kod;
}
else
{
        shart_bajarilmaganda_ishga_tushadigan_kod;
}
```

```
// Usul 3
if (shart)
        shart_bajarilganda_ishga_tushadigan_kodlar;
else
        shart_bajarilmaganda_ishga_tushadigan_kodlar;
// Usul 4
if ( shart ) shart_bajarilganda_ishga_tushadigan_kodlar;
else shart_bajarilmaganda_ishga_tushadigan_kodlar;
```

If- else blok operatorlarining foydalanish usullariga e'tibor bergan bo'lsangiz if shart blok operatorining foydalanish usullari bilan deyarli bir xildir. If- else shart blok operatoridan foydalanish davomida e'tiborli bo'ladigan taraflarimiz mavjud. Keling ularni birgalikda ko'rib chiqamiz:

- Har bir else blogi bir dona if blogiga to'g'ri kelishi kerak(ya'ni har elsening bir dona ifi bo'lishi kerak).
- Har bir If blogidan so'ngra else blogi kelishi kerak degan bir qoida yo'q.
- Bir if – else shart blok operatorlarida berilgan kodlardan faqatgina bir blok ichkarisidagi kod ishga tushadi. Ya'ni if blogi ichkarisidagi kod ishga tushganda, else blogi ichidagi kod ishga tushmaydi. Yoki else blogi ichkarisidagi kod ishga tushganda, if blogi ichidagi kod ishga tushmaydi (qaysi blokdagi kod

ishga tushishi shartni true yoki false ekanligi bog‘liqdir).

- Agar bir kod ichkarisida birdan ko‘p if shart blogi va bir else blogi foydalanilganda bu kod ichidagi else blogi o‘zidan oldingi if blogiga tegishli bo’ladi.

If-else haqida yetarlicha ma’lumotlarga ega bo‘ldik deb o‘ylayman. Keling endi birgalikda if else uchun kichik bir kod yozib ko’ramiz:

```
public class if_else_shart {
    public  static void main(String[] args){
        int a = 20;
        int b = 25;

        if (a>b){
            System.out.println("If blogi
               ichidagi kod ishga tushirildi
               chunki berilgan a soni b
               sonidan katta.");
        }
        else{
            System.out.println("Else blogi
               ichidagi kod ishga tushirildi
               chunki berilgan a soni b
               sonidan katta emas.");
        }


    }
}
```

5.3 kod

Yuqoridagi 5.3 kodimizni ishga tushirganimda quyidagicha javob olamiz:

```
Else blogi ichidagi kod ishga tushirildi
chunki berilgan a soni b sonidan katta emas.
```

5.3 natija

5.3 kodimizni ishlash shakli qisqacha ko‘rib chiqamiz. Foydalanuvchi tarafidan kodimiz ishga tushirilganda programmamiz birinchi bo’lib ikki int turidagi a va b nomli o‘zgaruvchilarni tanitdi va ularga 20 va 25 qiymatlarini bergan holda xotiraga oldi (5.3 kod 3 va 4- qatorlarda). Bu amalni bajarib olgandan so‘ngra, if-else bloklariga oid bo’lgan If blogi ichkarisidagi shartni, ya’ni (a>b) shartni o’qib shartni tekshirdi va shartga ko’ra false javobini chiqardi, chunki berilgan a nomli o’zgaruvchi qiymati b nomli o’zgaruvchi qiymatidan katta emasligi uchun bizga false javobini berdi. False javobini olganligi uchun programmamiz if blok kodlarini ishga tushirmasdan, else blok ichkarisidagi kodni ishga tushirdi. Yuqorida aytib o’tganimiz kabi if-else sharti bizga true javobini bersa if blogi ichkarisidagi kodni, false javobini bersa else ichkarisidagi kodni ishga tushiradi.

If-else shart operatoridan foydalanish: Hisoblash_v2.java

5.2 kodimizda if shart operatorining qanday ishlashini ko’rib chiqdik. Agar eslasangiz 5.2 kodimizda ya’ni **Hisoblash.java** nomli kodimizda foydalanuvchi hisobidan pul yechmoqchi bo‘lganda agar hisobida yetarlicha pul bo‘lsa yecha olayotgandi. Faqat hisobida yetarlicha pul yo’q bo‘lgan vaziyatlarda foydalanuvchiga hech qanday bir ma’lumot berilmayotgandi. Keling birgalikda if-elsedan foydalangan holda, hisobida yetarlicha pul bo‘lmaganda foydalanuvchiga bir ma’lumot ham beradigan bir programma kodini ko’rib chiqamiz. Kodimizni nomi **Hisoblash_v2.java** bo’lsin.

Bu kodni ushbu GitHub manzilidan yuklay olasiz: *https://github.com/21040001/Java_yordamchi_kitob_kodlarimiz/blob/main/Hisoblash_v2.java*

Hisoblash_v2.java

```
import java.util.Random;
import java.util.Scanner;

public class Hisoblash_v2 {
    public static  void  main(String[] args){
        Random random = new Random();
        double hisobim = random.nextInt(2000);
        System.out.println("Hisobingizdagi pul
                miqdori = " + hisobim);

        System.out.print("Hisobingizdan
                yechmoqchi bo'lgan pul
                miqdoringizni kiring : ");
        Scanner scanner = new
                Scanner(System.in);
        double yechiladigan_pul =
                scanner.nextDouble();

        if (hisobim>yechiladigan_pul){
            hisobim -= yechiladigan_pul;
            System.out.println("Hisobingizdan
                yechib olingan pul miqdori = "
                + yechiladigan_pul);
            System.out.println("Hisobingizda
                qolgan pul miqdori = "
                +hisobim);
        }
        else {
            System.out.println("Hisobingizda
                yetarlicha pul miqdori mavjud
                emas!!");
        }

    }
}
```

5.4 kod

Keling 5.4 kodimizni ishga tushirib ko‘raylik:

```
Hisobingizdagi pul miqdori = 1646.0

Hisobingizdan yechmoqchi bo'lgan pul miqdoringizni kiring
: 2000

Hisobingizda yetarlicha pul miqdori mavjud emas!!
```

5.4 natija

5.4 kodimizni ishga tushirdik , programmamiz bizning hisobimizda 1664 qiymatida pul bor ekanligi ko'rsatdi. Va bizdan yechib oladigan pul miqdorimizni kirishimizni istadi, biz 2000 yechib olishimizni aytdik. Hisobimizda 2000 pul yoqligi uchun ushbu natijani berdi azizlar.

If-elsening ishlash shaklini tushunib oldik deb o'ylayman, bu sababli mavzularimizda asta sekinlik bilan davom etamiz.

If Shart operatori ichkarisida aralash mantiqiy ifodalarni foydalanish

Keling azizlar, birgalikda 5.2 kodimizni ya'ni **Hisoblash.java** nomli kodimizni yanada takomillashtiraylik. Kodimiz ichkarisiga foydalanuvchining bir kunda cheka oladigan eng maksimum miqdorini ham kirtaylik. Ya'ni agar foydalanuvchi hisobidan pul yechmoqchi bo'lganda hisobidagi pul yetsa va yechadigan pul miqdori bir kunlik yechadigan pul miqdoridan kam bo'lsa, foydalanuvchi pul yecha olsin. Boshqa holatlarda esa pul yecha olmasin. Keling eng yaxshisi bularni **Hisoblash_v3.java** nomli kodimizda ko'rib chiqaylik.

Hisoblash_v3.java

```
import java.util.Random;
import java.util.Scanner;

public class Hisoblash_v3 {
    public static  void  main(String[] args){

        Random random = new Random();
        double hisobim = random.nextInt(2000);

        System.out.println("Hisobingizdagi pul
               miqdori = " + hisobim);

        double kunlik_limit = 500 ;

        System.out.print("Hisobingizdan
               yechmoqchi bo'lgan pul
               miqdoringizni kiring : ");

        Scanner scanner = new
                       Scanner(System.in);

        double yechiladigan_pul =
                       scanner.nextDouble();

        if (hisobim>yechiladigan_pul &&
            yechiladigan_pul<kunlik_limit ){

            hisobim -= yechiladigan_pul;

            System.out.println("Hisobingizdan
               yechib olingan pul miqdori = "
               + yechiladigan_pul);

            System.out.println("Hisobingizda
               qolgan pul miqdori = "
               + hisobim);
        }
        else {
```

```
            System.out.println("Hisobingizda
                yetarlicha pul miqdori mavjud
                emas yoki kunlik limitdan
                oshdingiz!!");
        }

    }
}
```

5.5 kod

Yuqorida 5.5 kodimizni ya'ni **Hisoblash_v3.java** nomli kodimizni berilgan GitHub manzilimizdan yuklay olasiz: *https://github.com/21040001/Java_yordamchi_kitob_kodlarimiz/blob/main/Hisoblash_v3.java*

Yuqoridagi 5.5 kodimizning 13-qatorida double turidagi kunlik_limit nomli bir o'zgaruvchi tanitish orqali, bir kunda maksimum yecha oladigan pul miqdorini kirdik. Va if-else ichkarisida and mantiqiy ifodasi orqali ikki shart berdik shu shaklda: yechiladigan_pul miqdori hisobdagi pul miqdoridan va kunlik limitdan ko'p bo'lmagan vaziyatlarda foydalanuvchiga pul yechishiga ruxsat ber. Shartning ko'rinishi shu shaklda (hisobim>yechiladigan_pul && yechiladigan_pul < hisobim). Kodimizni ishga tushirganimizda 5.5 natijaga o'xshash bir natija olamiz:

```
Hisobingizdagi pul miqdori = 1173.0

Hisobingizdan yechmoqchi bo'lgan pul miqdoringizni
kiring : 900

Hisobingizda yetarlicha pul miqdori mavjud emas yoki
kunlik limitdan oshdingiz!!
```

5.5 natija

5.4 If- else ichkarisida if-else operatorlaridan foydalanish

If-else shart operatorlari ichkarisida if-else operatorlarini foydalana olamiz, bunga Java dasturlash tili ruxsat beradi. Buni ko'rib chiqish davomida **Hisoblash_v3.java** nomli kodimizdagi ko'ringan bir muammoni ham qilamiz . **Hisobash_v3.java** kodimizga e'tibor bergan bo'lsangiz, shu shaklda ishlayotgandi foydalanuvchi yechmoqchi bo'lgan pul miqdori hisobidagi pul miqdoridan yoki kunlik limitidan oshganda programma shunday bir javob chiqarayotgandi: "Hisobingizda yetarlicha pul miqdori mavjud emas yoki kunlik limitdan oshdingiz!!". Bu javobda yetarlicha pul mavjud emas mi yoki kunlik limitdan oshdi mi bu noma'lum edi. Ya'ni foydalanuvchi ikkovidan qaysi biri ekanligini bilmayotgandi.

Hozir birgalikda **HIisoblash_v4.java** nomli kodda bu muammoni hal qilishga harakat qilamiz.

Hisoblash_v4.java

```
import java.util.Random;
import java.util.Scanner;

public class Hisoblash_v4 {
    public static  void  main(String[] args){

        Random random = new Random();
        double hisobim = random.nextInt(2000);

        System.out.println("Hisobingizdagi pul
                miqdori = " + hisobim);

        double kunlik_limit = 500 ;

        System.out.print("Hisobingizdan
```

```
                yechmoqchi bo'lgan pul
                miqdoringizni kiring : ");

        Scanner scanner = new
                        Scanner(System.in);

        double yechiladigan_pul =
                        scanner.nextDouble();

        if (hisobim>yechiladigan_pul &&
            yechiladigan_pul<kunlik_limit ){

            hisobim -= yechiladigan_pul;

            System.out.println("Hisobingizdan
               yechib olingan pul miqdori = "
               + yechiladigan_pul);

            System.out.println("Hisobingizda
               qolgan pul miqdori = "
               + hisobim);
        }
        else {

            if(hisobim>yechiladigan_pul){
               System.out.println("
               Hisobingizda yetarlicha pul
               miqdori mavjud emas!!");
            }else {
               System.out.println("Kunlik
               limitdan oshdingiz!!");
           }

        }

    }
}
```

5.6 kod

Yuqoridagi 5.6 kodimizda if-else blog ichkarisida if-else blogidan foydalandik. Va bu foydalanish orqali **Hisoblash_v3.java** kodimizdagi muammoni hal qila oldik. Keling 5.6 kodimizni ishlatib ko'ramiz.

```
Hisobingizdagi pul miqdori = 1408.0

Hisobingizdan yechmoqchi bo'lgan pul miqdoringizni kiring
: 600

Kunlik limitdan oshdingiz!!
```

5.6 natija 1

```
Hisobingizdagi pul miqdori = 889.0

Hisobingizdan yechmoqchi bo'lgan pul miqdoringizni kiring
: 1000

Hisobingizda yetarlicha pul miqdori mavjud emas!!
```

5.6 natija 2

Yuqorida e'tibor bergan bo'lsangiz 5.6 kodimizni ikki xil usulda ishga tushirib ko'rdik va ikki farqli bo'lgan javoblar oldik. Birinchi ishlatishimizda, yechmoqchi bo'lgan pul miqdorimizni hisobimizdagi pul miqdoridan kam , kunlik limitdan ko'p bo'lgan holda kirib ko'rdik va (5.6 natija 1)dagi kabi natija oldik. Ikkinchi ishga tushirishimizda esa, yechib oladigan pul miqdorimizni hisobimizdagi pul miqdoridan ko'p bo'lgan holatda kiritdik va (5.6 natija 1)dagi kabi natija oldik.

If- Else if bloklari

If - else if shart operatoridan qachonki birinchi if blok ichidagi shart bajarilmaganda, else ichidagi shartni tekshirishimiz kerak bo'lgan vaziyatlarda foydalanamiz. Buni tushunishimiz oson bo'lishi uchun, keling birgalikda bu bloklarimizni kodlar ichida foydalangan holda qaytadan o'rganib chiqamiz.

Misol: Yoshga ko'ra kredit miqdorini chiqaradigan dastur tuzing. Agar foydalanuvchining yoshi o'n sakkizdan kichik bo'lsa hech qanday kredit berilmaydi, agar foydalanuvchining yoshi 18 va 30 orasida bo'lsa foydalanuvchiga maksimum 50000000 so'm , agar 30 va 45 orasida bo'lsa 40000000 so'm, 45 yoshdan katta esa 20 000 000 so'm beriladi. Bu ma'lumotlar asosida tayyorlang.

Yuqorida misolimizni dastur kodlarini yozish davomida if-else if blogidan foydalanamiz va qanday foydalanishimizni ko'rib o'rganib chiqamiz. Dastur kodimizning nomi **Kredit.java** bo'lsin.

```
import java.util.Scanner;

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

        System.out.print("Yoshingizni kiring:
                          ");
        Scanner scanner = new
                Scanner(System.in);

        double yosh = scanner.nextDouble();

        if (yosh<=18 ){
```

```
            System.out.println("Voyaga
                yetmaganingiz sababli kredit
                ola olmaysiz.");
        }
        else if(18 < yosh && yosh<= 30 ){
            System.out.println("Maksimum
                50000000 so'm miqdorda kredit
                olishingiz mumkin.");
        }
        else if(30 < yosh && yosh<= 45 ){
            System.out.println("Maksimum
                40000000 so'm miqdorda kredit
                olishingiz mumkin.");
        }
        else if(45 < yosh ){
            System.out.println("Maksimum
                20000000 so'm miqdorda kredit
                olishingiz mumkin.");
        }
    }
}
```

5.7 kod

Yuqoridagi 5.7 kodimizni berilgan GitHub manzilimizdan yuklay olasiz.

Manzil:
https://github.com/21040001/Java_yordamchi_kitob_kodlarimiz/blob/main/Kredit.java

5.7 kodimizda if-else if bloklaridan shu shaklda foydalandik. Birinchi bo'lib if ichida shartimizni berdik so'ngra esa else if ichida shartimizni berdik va bu shaklda davom etgan holda barcha shartlarimizni 5.7 kodda ko'rganingiz kabi yozib chiqdik. Keling bu kodimizni ishga tushirib ko'ramiz. Ishga tushirganimizda bizdan yoshimizni kirishni so'raydi, yoshimizni 20 sifatida kiritgan holda programmani ishga tushiramiz.

```
Yoshingizni kiring : 20

Maksimum 50000000 so'm miqdorda kredit olishingiz
mumkin.

                                        5.7 natija
```

5.5 Switch - Case blok kodlari

Switch-case bir o‘zgaruvchining qiymatiga ko’ra programmani ishlatadi. Bu ishni if-else kodlari bilan ham bajara olamiz, lekin Switch-case kodlari bilan if-else kodlariga qaraganda tezroq yoza olamiz , shuning uchun dasturchilarning ko’p qismi switch-casedan foydalanadi.

Switch-case asosan byte, char, int, short, long yoki indekslangan (enumerated) turidagi o‘zgaruvchilarning qiymatlari bilan ishlay oladi. Switch-casedan foydalanishda ushbu qolipdan foydalanamiz.

```
switch(foydalaniladigan_o’zgaruvchi_nomi) {

        case qiymat_1: buqiymatdaishgatushadigankod;

        case qiymat_2: buqiymatdaishgatushadigankod;

        …..

        case qiymat_n: buqiymatdaishgatushadigankod;

        default: boshqaqiymatlardaishgatushadigankod ;

}
```

Keling Switch-case blokli kodlarini foydalanishni kodlar ichida ham ko'rib, o'rganib chiqamiz.

```
import java.util.Scanner;

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

        System.out.print("Bir son kiring : ");
        Scanner scanner = new
               Scanner(System.in);
        int son = scanner.nextInt();
        switch (son){
            case 1:
                System.out.println("Kirgan
                       soningiz : 1 ");
                break;
            case 2:
                System.out.println("Kirgan
                       soningiz : 2 ");
                break;
            case 3:
                System.out.println("Kirgan
                       soningiz : 3 ");
                break;
            case 4:
                System.out.println("Kirgan
                       soningiz : 4 ");
                break;
            default:
                System.out.println("Kirgan
                       soningiz 1,2,3,4
                       sonlaridan biri emas");
                break;

        }

    }
}
```

Switch-case kodlarini yozishda har bir case ichidagi kodlarni oxirida break funksiyasidan foydalanish kerak, agar foydalanilmasa qabul qilingan case qiymatidan keyingi barcha kodlar ishga tushadi va programmaning to'g'ri ishlanmasiga sabab bo'ladi.

Misol: Kirilgan sonlarga ko'ra hafta kunining nomini chiqaradigan programma tayyorlang.

Keling bu misolning kodlarini birgalikda yozib chiqamiz: Kodimizning nomi **Hafta_kuni.java,** GitHub manzili esa shudir: *https://github.com/21040001/Java_yordamchi_kitob_kodlarimiz/blob/main/Hafta_kuni.java*

Hafta_kuni.java

```
import java.util.Scanner;

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

        System.out.print("1 va 7 orasida
               bo'lgan bir son kiring : ");

        Scanner scanner = new
               Scanner(System.in);
        int son = scanner.nextInt();

        switch (son){
            case 1:
              System.out.println("Dushanba");
              break;
            case 2:
              System.out.println("Seyshanba");
              break;
            case 3:
             System.out.println("Chorshamba");
             break;
```

```
            case 4:
              System.out.println("Payshanba");
              break;
            case 5:
              System.out.println("Juma");
              break;
            case 6:
              System.out.println("Shanba");
              break;
            case 7:
              System.out.println("Yakshanba");
              break;
            default:
              System.out.println("Kirgan
                        soningiz 1 va 7
                        orasidagi bir son
                        emas");
              break;

        }


    }
}
```

Bu kodimizni ishga tushirib ko’radigan bo’lsak.

```
1 va 7 orasida bo'lgan bir son kiring : 3

Chorshamba
```

Ko‘rib turganingiz kabi biz 3 sonini kirdik programma esa haftaning uchinchi kuni chorshanba ekanligi aytdi.

Massivlar

5.6 Bir o’lchovli massivlar

Bu yergacha sizlar bilan birgalikda kodlarimiz ichkarisida turli xil qiymatlarni o’zgaruvchilarga qiymat sifatida bergan holda ularni xotiraga saqlagan holda ular bilan ishladik. Ammo ba’zi programmalarda bir xil turdagi bir nechta qiymatlarni bir joyga to‘plagan xolda, bir o’zgaruvchiga saqlash kerak bo’ladi. Misol uchun, bir mototsiklning har 1 soat ichida olgan tezligi ustida ishlamoqchimiz. 100 soatda erishgan barcha tezliklarini oladigan bo’lsak, jami bo’lib 100 dona double turidagi qiymalar olamiz. 100 dona qiymatni ustida ishlashimiz uchun ularni avvalo bir o‘zgaruvchiga saqlashimiz kerak, ya’ni 100 dona yangi o‘zgaruvchi yaratishimiz kerak bo’ladi. Bu dasturchining ishini tabiiyki qiyinlashtiradi.

Massivlar, dasturchilar bunday bir qiyin vaziyatda qolmasliklari uchun yaratilgan deya olaman. Ya’ni massivlar ma’lum bir miqdorda , o‘xshash turdagi o’zgaruvchilarga bir nom va turli xil indekslar yordamida bog‘lanishimizni saqlagan maxsus bir Java metodidir. Javada massivlar 2 xil usulda ko’rsatiladi.

1-usul:

```
O’zgaruvchi_turi [] massiv_nomi = new o’zgaruvchi_turi [qiymatlar_soni];
```

1-usul (Statik shaklda):

```
O’zgaruvchi_turi [] massiv_nomi = { o’zgaruvchi_turi qiymat1,
                                    o’zgaruvchi_turi qiymat2, ……..};
```

2-usul:

```
O’zgaruvchi_turi massiv_nomi[ ] = new o’zgaruvchi_turi [qiymatlar_soni];
```

2-usul (Statik shaklda):

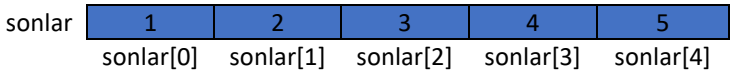
```
O'zgaruvchi_turi  massiv_nomi[ ] = { o'zgaruvchi_turi qiymat1,
                                     o'zgaruvchi_turi qiymat2, ……..};
```

Keling aziz yuqoridagi ikki usulga ham misollar beraylik.

```
// ….
int [] sonlar = new int[7];  // Usul 1
int  sonlar[] = new int[7];  // Usul 2
int [] sonlar = {1,2,3,4,5}; // Usul 1 -statik
int sonlar[] = {1,2,3,4,5}; // Usul 2 -statik
//….
```

Yuqoridagi misolimizdagi 2 va 3 – satrlardagi kodlar bir xil. Ikkalasi ham har bir qiymati int turida bo'lgan 7 dona qiymatli va sonlar nomli massiv tayyorlaydi. 4 va 5 – satrlar ham bir biriga o'xshaydi, ya'ni ikkalasiga ham sonlar nomi berilgan holda massiv tayyorlandi va massivning qiymatlari to'g'ridan to'g'ri massiv ichkarisiga yozilgan holda ko'rsatildi.

Boshqacha qilib aytilganda, bu satrlardan biri ishlaganda massiv pastda ko'ringani kabi bo'ladi.



Massivlar nechta qiymatlardan tashkil topsa topsin qiymatlarining turi bir xil bo'lishga majbur. Ya'ni birinchi qiymati int bo'lib ikkinchi qiymati String bo'la olmaydi. Massivlar faqatgina bir xil turdagi qiymatlarni qabul qiladi. Yana bir qancha misollar berib o'tadigan bo'lsak:

```
//….
String [ ] ismlar = new String[3];
char [  ] massiv1 = new char [ 10];
bayte [ ] massiv2 = new byte[12];
boolean [ ] ha_yo'q_anketa_javoblari = new boolean[12];
//….
```

Yuqoridagi misollarda massivlarimiz nechtadan qiymatlar olishini ko'rsatib o'tdik, misol uchun ikkinchi qatordagi kodni Java dasturlash tili shu shaklda o'qiydi "String turida, ismlar nomli 5 dona String turidagi qiymatlarga ega bo'lgan bir massiv beriladi." Bu yerda berilishi kerak bo'lgan qiymatlar pastda ko'rsatib o'tilgani kabi beriladi.

```
//….
String [ ] ismlar = new String[3];
ismlar[0]= "Davronbek";
ismlar[1]= "Abubakir";
ismlar[2]= "Mustafo";
//….
```

Yuqoridagi misollarimizni tushuntirib o'tadigan bo'lsak: Birinchi bo'lib String turida bo'lgan ismlar nomli 3 dona qiymatdan tashkil topgan bir massiv ekanligini dasturga aytib o'tdik. Keyin esa, aytilgan 3 qiymatning qiymatlarini berib o'tdik 3,4 va 5- qatorlardagi kabi. E'tibor bergan bo'lsangiz qiymat berishda nechanchi qiymatning qiymati ekanligi ismlar[0] shu shaklda ko'rsatib o'tdik. Ya'ni qiymat berish qisqacha shu shaklda:

```
Massiv_nomi [indeks_raqami] = qiymat;
```

Indeks raqami degani azizlar massivning nechanchi qiymati ekanligini ko'rsatadigan raqam. Indekslar 0'dan boshlangan holda sanaladi. Bir massivning biron bir qiymatini chaqirmoqchi bo'lganimizda u qiymatning indeks raqamidan foydalanamiz. Misol uchun massivdagi 5-qiymatni chaqirmoqchi bo'lganimizda shu shaklda chaqiramiz: massiv_nomi[4], ya'ni 5-qiymatning indeks raqami 4 bo'ladi chunki dastur sanashni 0'dan boshlaydi . Yuqoridagi misolimiz orqali tushuntiradigan bo'lsak:

ismlar	Davronbek	Abubakir	Mustafo
	ismlar[0]	ismlar[1]	ismlar[2]

Agar biz bu yerda biron bir qiymatning qiymatini bermasdan davom etganimizda, o'sha qiymatning qiymatini dasturimiz avtomatik bir shaklda beradi. Dasturimiz quyidagicha qiymatlarni beradi:

- Son turidagilar uchun(int, float, double, byte,short) **0**
- Boolean turidagi qiymatlar uchun **false**
- Char turidagi qiymatlar uchun **'\u0000'**
- String turidagi qiymatlar uchun **null**

Yuqorida ma’lumotini berib o’tgan barcha narsalarimizni bir joyga to’plagan holda bir programma yozaylik. Programmamizga **Massiv.java** nomini beraylik.

```
import java.util.Scanner;

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

        String [] ism = new String[3];
        int [] yosh = new int[3];
        String [] millati = new String[3];
        ism[0]= "Davronbek Abdurazzokov";
        yosh[0]=20;
        millati[0]="O'zbek";
        ism[1]= "Abubakir Jagitay";
        yosh[1]=24;
        millati[1]="Pullo";
        ism[2]= "Mert Arikan";
        yosh[2]=22;
        millati[2]="Turk";

        Scanner scanner =
                        new Scanner(System.in);

        System.out.print("Kim haqida
               ma'lumotga ega bo'lmoqchi
               bo'lsangiz, iltimos ismini
               katta harf bilan boshlagan
               holda kiring.");
        String malumot = scanner.nextLine();

        switch (malumot){
            case "Davronbek":
                System.out.println("To'liq
                        ismi: " +ism[0]);
                System.out.println("Yoshi: "
                        +yosh[0]);
```

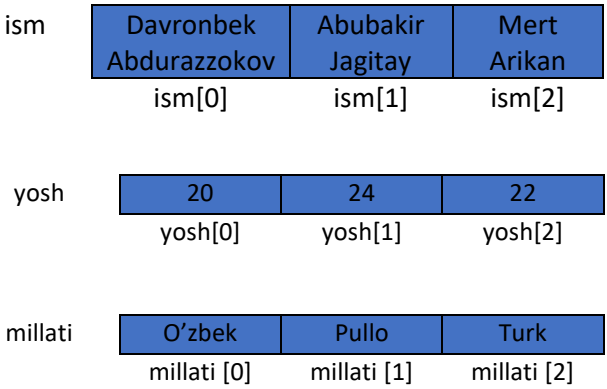
```
                System.out.println("Millati:"
                        +millati[0]);
                break;

            case "Abubakir":
                System.out.println("To'liq
                        ismi: " +ism[1]);
                System.out.println("Yoshi: "
                        +yosh[1]);
                System.out.println("Millati: "
                        +millati[1]);
                break;

            case "Mert":
                System.out.println("To'liq
                        ismi: " +ism[2]);
                System.out.println("Yoshi: "
                        +yosh[2]);
                System.out.println("Millati: "
                        +millati[2]);
                break;

            default:
                System.out.println("kiritilgan
                        shaxsga oid har qanday
                        bir malumot mavjud
                        emas. Iltimos tekshirib
                        qaytadan kiring.");
        }

    }
}
                                                5.8 kod
```

5.8- kodimizni pastda berib o’tilgan GitHub manzilidan yuklay olasiz.

Manzil:
https://github.com/21040001/Java_yordamchi_kitob_kodlarimiz/blob/main/Massiv.java

Yuqoridagi 5.8-kodimizni ishlash jarayonini qadama-baqadam ko'rib chiqaylik azizlar.

1. **Qadam.**
   Dasturimizni ishga tushirganimizda programmamiz birinchi bo'lib 3 dona massiv tayyorlaydi. Massivlarimizning nomlari: **ism, yosh, millati**. Har bir massivimiz 3 donadan qiymat oladi.
2. **Qadam.**
   Bu qadamda esa dasturimiz yuqoridagi massivlarimizning qiymatlarini kod ichkarisida berilgan qiymatlar asosida beradi, ya'ni shu shaklda:



Bu uch massivlarimiz shaxslarga oid bo'lgan ma'lumotlarni xotirada saqlamoqda. Birinchi ism nomli massivimizda shaxslarning to'liq ismi, ikkinchi yosh nomli massivimizda esa shaxslarning yoshlari, uchinchi millati nomli massivimizda esa shaxslarning qaysi millatda ekanligi saqlanmoqda. Biz bu massivlarimizdagi ma'lumotlardan foydalangan holda

foydalanuvchilarimizga shaxslar haqidagi ma’lumotlarni bera olamiz.

3. **Qadam.**
   Bu qadamda esa azizlar dasturimiz foydalanuvchidan, qaysi shaxsga oid ma’lumotlarni olmoqchi bo’lsa u shaxsning ismini **Scanner** sinfidan foydalangan holda oladi. Va ismga ko’ra Switchda berilgani kabi ma’lumotlarni keltiradi.
   Misol uchun kodimizni ishga tushirib ko’raylik:

```
Kim haqida ma'lumotga ega bo'lmoqchi bo'lsangiz, iltimos ismini katta harf bilan boshlagan holda kiring. Davronbek

To'liq ismi: Davronbek Abdurazzokov

Yoshi: 20

Millati: o'zbek
```

5.8 natija

5.8 natija e’tibor beraylik. Dasturimiz bizdan ko’rib turganingiz kabi ism kirishimizni istadi. Bizda Davronbek ismini yozdik. So’ngra Davronbekga tegishli barcha ma’lumotlarni Switchga ko’ra ekranga chiqardi, ya’ni Switichda Davronbek nomli case holati bor. Ya’ni agar foydalanuvchi Davronbek ismini kirgizganda **ism** massividagi **0-ideksda** turgan qiymatni, **yosh** massiviga joylashtirilgan **0-indeksdagi** qiymatni va **millati** nomli massivdagi **0-indeksdagi** qiymatni ekranga chiqarishi kerak edi. Shuning uchun dasturimiz bizga quyidagicha javoblarni berdi. 5.8 kodni qaytadan ishlatib ko’ring, shunda ishlash jarayonini yaxshi tushunib olasiz.

5.7 Ko’p o’lchamli massivlar

Ko’p o’lchamli massivlarning ham tayyorlanishi va tuzilishi normal bir o’lchamli massivlar kabidir. Ko’p o’lchamli massivlar qiymatlarni matritsalar kabi saqlaydi. Ko’p o’lchamli massivlarda ham qiymatlarning turlari bir xil bo’lishi kerak. Ikki o’lchamli bir massivni shu shaklda yoza olamiz.

```
qiymat_turi[][] masiv_nomi= new qiymat_turi [qator_s] [ustun_s];
```

Yuqoridagi formuladan foydalangan holda misollar beradigan bo’lsak.

```
//….

int [][] raqamlar = new int[3][4];

String[][] ismlar= new String[2][5];

 byte [][] sonlar=new byte[1][2];

//….
```

Misol uchun pastdagi kabi ikki o’lchamli bir massivning qiymatlarini saqlashini pastdagi berilgan 5.1- jadvalimizda ko’rishingiz mumkin.

```
//….

int [][] raqamlar = new int[2][3];

//….
```

Matris[0][0]	Matris[0][1]	Matris[0][2]
Matris[1][0]	Matris[1][1]	Matris[1][2]

5.1 jadval

Ko'p o'lchamli massivlarimizning qiymatlariga ham indekslar orqali bog'lanamiz. Bu massivlarimizga indeks berish shaklini 5.1- jadvalda ko'rib chiqishimiz mumkin. Misol uchun [0][0] bu indeksning ma'nosi shu shaklda: "0-qatorning, 0-ustunu". yuqoridagi 5.1 jadvalimizning 1-qatoridagi 1-ustuniga va 1-qatoridagi 2-ustuniga qiymat bermoqchi bo'lganimizda, quyidagi shaklda qiymat beramiz.

```
//…

Matris[1][1]=5;

Matris[ 1][2]=6;

//…
```

Ko'p o'lchamli massivlarimizda qiymatlarni berishni 1-yoli shu shaklda edi azizlar. Keling birgalikda 2-yolini ya'ni massivni tayyorlash davomida qiymatlarni ham berish usulini ko'rib chiqamiz. 2-usulga bir misol beraylik.

```
//…

int[][] matris = {{1,23,19},{12,45,67},{54,45,98}};

//…
```

Bu massivimizda qavslar ichida massivimizning qiymatlarini berib ketdik, 1-qavsdagi {1,23,19} qiymatlari 1-qatorga, 2-qavs ichidagilar 2-qatorga, 3-qavs ichkarisidagi qiymatlar 3-qatorga mos shaklda joylashtiradi. Pastdagi jadvalda bu qiymatlarni joylashtirish shakli ko'rsatib o'tilgan.

	Ustun 1	Ustun 2	Ustun 3
Qator 1	1	23	19
Qator 2	12	45	67
Qator 3	54	45	98

Ko’p o’lchamli massivlarimiz ham bir o’lchamli massivlarimiz bilan deyarli bir xildir.

Sikl operatorlari.

Deyarli barcha dasturlash tillarida sikl operatorlari mavjud. sikl operatorlari programmada ma’lum bir ish harakatni, ma’lum bir shartlarga ko’ra , ma’lum bir miqdorda takrorlashga yordam beradi. Sikl operatorining Java dasturlash tilida 4 dona turi mavjud, bular:

1. while takrorlash operatori
2. do while takrorlash operatori
3. for takrorlash operatori
4. moslashtirilgan for takrorlash operatori

5.8 While takrorlash operatori

Bu operatorimiz shartga ko’ra ishlaydi ya’ni while blogi ichkarisidagi kodlarimiz, while ichidagi shartimiz true ekanligida takrorlanadi. Ya’ni har bir takrorlanish amalga oshirilishidan oldin shart tekshiriladi. While takrorlash operatorimizning kod ichkarisidagi blok shakli, shu shaklda:

```
while(shart) {
        kod1;
        kod2;
        ……..
        kod_n;
}
```

E’tibor bergan bo’lsangiz while takrorlash operatorimizning foydalanish shakli if shart operatorimizning foydalanish shakli bilan bir xil. Agar while ichidagi shart true esa, while ichidagi kodlar while ichidagi shart false bo’lishiga qadar takrorlanadi. Agar while ichidagi kodimiz bir dona bo’lganida {} bu shakldagi qavslardan foydalanmasak ham bo’ladi, if shart operatoridagi kabi. While takrorlash operatoridan foydalanishning pastada ko’rsatib o’tilgani kabi 3 usuli bor.

```
// 1-usul
while (shart)
        kod;
//2-usul
while (shart){
        kod;
}
```

```
// 3- usul
while (shart)
{
        kod;
}
```

Yuqoridagi 3 usul ichkarisidan 2 va 3- usuldan foydalanishni tavsiya qilaman. Chunki {} bu shakldagi qavs ichkarisida yozilgan kodlar har doim tushunarli bo'ladi dasturchi uchun. Bu sababdan ko'pgina dasturchilar 2 va 3-usullardan foydalanadi. Boshqa tarafdan qaraydigan bo'lsak, bir blok ichkarisida ishga tushishi kerak bo'lgan kodlar soni 1 dan ko'p bo'ladigan bo'lsa 1- usuldan foydalana olmaymiz. Bu holatlar barcha sikl operatorlari uchun bir xildir. Bu sababdan boshqa sikl operatorlari do-while for va foreach sikllarini tushuntirish davomida bular haqida ma'lumot bermaymiz.

Keling yuqorida ko'rib chiqqan while sikl operatorimizga bir misol berib o'taylik.

```
import java.util.Scanner;

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

        System.out.print("Son kiriting : ");
        Scanner scanner =
                        new Scanner(System.in);
        int son = scanner.nextInt();

        int s=0;
        while(s<son) {
```

```
            System.out.println("Son: "+s+" Bu
                    sonning kvadrati:"
                    +(s*s));
            s=s+1;
        }
    }
}
```

5.9 kod

Yuqorida berib o'tilgan 5.9 kodimizni pastda berib o'tilgan GitHub manzilidan yuklay olasiz.

Manzil:*https://github.com/21040001/Java_yordamchi_kitob_kodlarimiz/blob/main/while_sikl.java*

Yuqoridagi 5.9 kodimizning ishlash shaklini ko'rib chiqadigan bo'lsak. Programmamiz birinchi bo'lib foydalanuvchidan bir son kirishini istaydi. Foydalanuvchi sonni kiritganidan keyin esa while takrorlash operatorimiz 0dan boshlab foydalanuvchi kiritgan songacha bo'lgan barcha butin sonlarni va ularning kvadratlarini ekranga chiqaradi.

Keling eng yaxshisi 5.9 kodimizni ishga tushirib ko'raylik.

```
Son kiriting : 5
son: 0 sonning kvadrati:0
son: 1 sonning kvadrati:1
son: 2 sonning kvadrati:4
son: 3 sonning kvadrati:9
son: 4 sonning kvadrati:16
```

5.9 kodimizdagi while sikl operatorini ishlash shaklini o'rganib chiqaylik azizlar. Bu kodimizda while blogimiz ishga tushishdan oldin, e'tibor bergan bo'lsangiz foydalanuvchidan son nomida bir int turiga kirgan bir raqam olindi. Undan so'ngra while operatorimiz ichkarisida berilgan shart ya'ni (s<son) shu shart tekshirildi bu yerda s=0 , son esa foydalanuvchi kirgan raqamga teng. Bu shartimiz bajarilganda programmamiz while blogi ichkarisidagi kodlarni ishga tushiradi. While blok kodimiz ichkarisidagi kodlarimizga e'tibor beraylik, bu yerda ikki kodimiz bor birinchisi kodimiz ekranga berilgan ma'lumotlarni chiqaradi, ikkinchi kodimiz esa mavjud bo'lgan s nomli o'zgaruvchiga 1 qiymatini qo'shadi buning natijasida s=1 bo'ladi. Bu yergacha while operatorimiz bir marta takrorlandi. Programmamiz while kodimizning boshiga qaytadi va qaytadan shartni tekshiradi bu safar ham shartimiz true javobini bersa while ichidagi kodlar yana ishga tushiriladi. Bu shaklda berilgan shart false bo'lishigacha blok ichkarisidagi kodlar ishga tushishi takrorlanaveradi.

5.9 Do-while takrorlash operatori

Do- while takrorlash operatorimiz, while operatori bilan deyarli bir xil hisoblanadi. Bu takrorlash operatorimiz while takrorlash operatoridan birgina farq bilan ajralib turadi. Bu farq, while operatorimizning blok ichkarisidagi kodlar ishga tushishi uchun, while blogimiz ichkarisidagi shart tekshirilganda **true** javobi chiqishi kerak edi. Do-while takrorlash operatorimizda esa birinchi blok ichidagi kodlar ishga tushadi va undan keyin shart tekshiriladi. Agar shart **true** javobini bersa yana qaytadan blok ichkarisidagi kodlar ishga tushiriladi va qaytadan shart

tekshiriladi. Bu shaklda davom etadi. Foydalanish shaklini ko'rib chiqadigan bo'lsa.

```
do
{
        kod1;
        kod2;
        .....
        kod_n;
} while (shart) ;
```

Keling do-while takrorlash operatorimizga bir misol berib o'taylik.

```
import java.util.Scanner;

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

        int s = 0;
        Scanner scanner =
                new Scanner(System.in);

        do{
            System.out.print("Kubini
               hisoblamoqchi bo'lgan
               soningizni kiriting : ");

            int son = scanner.nextInt();
```

```
            System.out.println("kiritgan
                soningiz : "+son+" kiritgan
                soningizning kubi : "
                + (Math.pow(son,3)));

            System.out.print("Dasturdan
                chiqish uchun 0 sonini
                kiriting. Boshqa bir sonning
                kubini hisoblamoqchi bo'lsangiz
                iltimos farqli bir son kiriting
                :");

            s =scanner.nextInt();

        }while (s!=0);
        System.out.println("Programmamizning
                ishga tushishi to'xtatildi.");

    }
}
```

5.10 Kod

5.10-kodimizni berilgan GitHub manzilidan yuklay olasiz.

Manzil:*https://github.com/21040001/Java_yordamchi_kitob_kodlarimiz/blob/main/Do_while.java*

5.10 kodimizda foydalanuvchi tarafidan bir son kiritiladi, so'ngra programmamiz kiritilgan sonning kubini hisoblab ekranga chiqaradi va foydalanuvchi programmani ishlashini to'xtatmagunicha programmamiz foydalanuvchidan son olgan holda, uning kubini hisoblashda davom etadi. Foydalanuvchi programmani ishlashini to'xtatmoqchi bo'lganda, 0 sonini kiritadi. Bu sonni kiritgandan so'ngra while ichidagi shart **false** bo'ladi, bu sababdan programmamiz do-while takrorlashidan chiqadi va undan keyingi kodni ishlatgan holda ekranga "Programmamizning ishga tushishi to'xtatildi." degan yozuvni

ekranga chiqaradi. Va dasturimiz ham kodlarni barchasini ishga tushirib bo’lganligi uchun ishlashni to’xtatadi.

5.10-kodimizni ishga tushirganimizda shu shaklda bir javob olamiz:

```
Kubini hisoblamoqchi bo'lgan soningizni kiriting : 4

kiritgan soningiz : 4 kiritgan soningizning kubi : 64.0

Dasturdan chiqish uchun 0 sonini kiriting. Boshqa bir sonning kubini hisoblamoqchi bo'lsangiz iltimos farqli bir son kiriting :1

Kubini hisoblamoqchi bo'lgan soningizni kiriting : 5

kiritgan soningiz : 5 kiritgan soningizning kubi : 125.0

Dasturdan chiqish uchun 0 sonini kiriting. Boshqa bir sonning kubini hisoblamoqchi bo'lsangiz iltimos 1 sonini kiriting :0

Programmamizning ishga tushishi to'xtatildi.
```

5.10 For takrorlash operatori

Java dasturlash tilidagi sikl operatorlarining uchinchisi va oxirgisi bo'lgan for takrorlash operatoridir. Keyingi mavzuda ko’radigan moslashtirilgan for takrorlash operatorimiz ham bu for operatoridan kelib chiqqandir. For takrorlash operatorimiz qolgan takrorlash operatorlariga qaraganda ko’p foydalaniladi. For takrorlash operatorimizning asosiy foydalanish usuli quyidagicha:

```
for (qiymat_ifodalari; shart ; qo'shish_ayirish_ifodalari)
{
        kod1;
        kod2;
        …
        kod_n;
}
```

qiymat_ifodalari: takrorlash ichida foydalaniladigan sanoq o'zgaruvchilari berib o'tiladigan qismdir. Normalda asosiy qiymat ifodalari berib o'tiladi.(int i = 0, long son = 15L , vh)

shart: bu yerda esa for blogimizni ishlashi uchun beriladigan shart berib o'tiladi. Ya'ni for blok ichidagi kodlarimiz berilgan shartimiz true bo'lsa ishga tushadi, false bo'lsa blokdan chiqadi.

qo'shish_ayirish_ifodalari: sanoq ichida foydalaniladigan qiymat ifodalarining qiymatini o'zgartiradigan bir qism.

For blok ichidagi qiymat_ifodalari, shart, qo'shish_ayirish_ifodalari orasida nuqta vergul bo'lishi shart, chunki bu yerda har biri farqli-farqli ish bajarmoqda. Keling for takrorlash operatoridan foydalangan holda 1dan 15gacha bo'lgan sonlarni ekranga chiqarishni ko'rib chiqaylik.

```
//…
for(int i = 0 ; i < 16 ; i++)
{
        System.out.println( i );
}
//…
// Buni agar while orqali yozadigan bo'lsak quyidagicha bo'ladi.
//…
int i =1
while(i <=15)
{
        System.out.println( i );
        i++;
}
//…
```

For takrorlash operatorimizdagi barcha qismlar to'ldirilmasa ham bo'ladi. Lekin bu qismlarni bir-biridan ayirish uchun nuqtali vergul qo'yishga majburmiz. Pastdagi misolda ko'rsatib o'tilgani kabi:

```
//…
int = 0;
for( ; i<=15;  ;)
{
        System.out.println(i);
        i++;
}
//…
```

For takrorlash operatorimiz ichkarisida nuqtali vergullar bilan ajratilgan qismlar ichkarisida 1dan ko'p ifodalar berilganda, ular vergullar bilan ajratilib yoziladi. Bu holatni pastdagi misolimizda ko'rib chiqamiz.

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

        for (int a=5 , b =6;  (a*b)<=100;
                                a++,b++){

            System.out.println("aning shu
                      onadagi qiymati : "+a);
            System.out.println("bning shu
                      onadagi qiymati : "+b);
            System.out.println("(axb)ning shu
                      onadagi qiymati : "
                      +(a*b));
        }
    }
}
```

Yuqoridagi kodimizni ishlash shaklini tushuntirib o'tadigan bo'lsak. Yuqoridagi kodimizni umumiy ishlash shakli shu shaklda: programmamiz a va b'ning ko'paytirilgandagi qiymati 100dan katta bo'lmagunicha ishlayveradi. Keling ishlatib ko'ramiz.

```
aning shu onadagi qiymati : 5
bning shu onadagi qiymati : 6
(axb)ning shu onadagi qiymati : 30
aning shu onadagi qiymati : 6
bning shu onadagi qiymati : 7
(axb)ning shu onadagi qiymati : 42
aning shu onadagi qiymati : 7
bning shu onadagi qiymati : 8
(axb)ning shu onadagi qiymati : 56
aning shu onadagi qiymati : 8
bning shu onadagi qiymati : 9
(axb)ning shu onadagi qiymati : 72
aning shu onadagi qiymati : 9
bning shu onadagi qiymati : 10
(axb)ning shu onadagi qiymati : 90
```

5.11 Moslashtirilgan for takrorlash operatori

Bu takrorlash operatorimiz Java dasturlash tilining 1.5 versiyasidan so'ngra Java dasturlash tili hayotiga kirgan. Bu sikl turimiz bir massiv ichidagi qiymatlarga yoki a'zolarga osonlik bilan yetishib olishimizni ta'minlaydi. Bu sikldan ya'ni moslashtirilgan for takrorlash operatorimizdan foydalanish usuli quyidagicha:

```
for(massiv_turi  o'zgaruvchi : massiv_nomi){

kodlar;

}
```

Misol uchun aytaylik, bizda 5 dona qiymatdan tashkil topgan bir massiv bor. Va biz bu massivning barcha qiymatlariga erishgan holda ularni ekranga chiqarmoqchi bo'lganimizda, bu ishni moslashtirilgan for takrorlash operatori bilan qisqa bir shaklda bajara olamiz. Keling eng yaxshisi, buni kodimizni yozgan holda o'rganib chiqaylik:

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

        String [] ismlar = new String[5];
        ismlar[0]= "Davronbek";
        ismlar[1]= "Abubakir";
        ismlar[2]= "Mert";
        ismlar[3]= "Jafer";
        ismlar[4]= "Murodjon";

        int b = 0;
```

```
        for (String a: ismlar){
            System.out.println("Berilgan
                ismlar nomli massivimizning"
                +b+"-indeksdagi qiymati = "+a);
            b++;
        }
    }
}
```

Bu kodimizni ishlash shaklini tushuntirib o'tadigan bo'lsak. Programmamizni ishga tushirganimizda birinchi bo'lib ismlar nomli massivimizni va b nomli o'zgaruvchi qiymatlarimizni xotiraga oladi. So'ngra, for blok ichidagi kodlarni ishga tushiradi. For ichidagi (String a: ismlar) qismini tushuntirib o'taman, bu yerda azizlar dasturga shunday demoqdamiz "String turidagi, ismlar nomli massivni a nomini olgan holdan aylanib chiq". Buni natijasida for blogimiz har takrorlanishida massivning bir qiymatini olib a nomli o'zgaruvchiga tenglaydi, blok ichkarisida esa a nomli o'zgaruvchini ekranga chiqaradi va yuqorida berilgan b nomli o'zgaruvchi qiymatini 1ga oshiradi. String.Keling buni ishlatib ham ko'ramiz.

```
Berilgan ismlar nomli massivimizning 0-indeksdagi qiymati = Davronbek

Berilgan ismlar nomli massivimizning 1-indeksdagi qiymati = Abubakir

Berilgan ismlar nomli massivimizning 2-indeksdagi qiymati = Mert

Berilgan ismlar nomli massivimizning 3-indeksdagi qiymati = Jafer

Berilgan ismlar nomli massivimizning 4-indeksdagi qiymati = Murodjon
```

5.12 Massiv va For sikl operatori

Bu safar massiv va for operatorlarimizdan foydalangan holda bir oz mukammal bo'lgan, bir misol ko'rib o'tamiz.

Misol: 3 ustunli va 3 qatorli bo'lgan 2 matritsani bir-biri bilan ko'paytirib beradigan bir dastur tayyorlang.

```
import java.util.Scanner;

public class massiv_for {

    public static void main(String[] args){
        Scanner gr=new Scanner(System.in);
        Scanner hr=new Scanner(System.in);

        int[][] matris=new int[3][3];
        int[][] matris2=new int[3][3];

        for (int i=0;i<3;i++){
            for(int k=0;k<3;k++){
                System.out.print("Birinchi
                   matritsaning "+i+" ustun "
                    +k+" a'zosini kir : ");
                int a= gr.nextInt();
                matris[i][k]=a;
            }
        }
        for (int h=0;h<3;h++){
            for(int l=0;l<3;l++){
                System.out.print ("Ikkinchi
                   matritsaning "+h+" ustun "
                    +l+" a'zosini kir : ");

                int b= hr.nextInt();
                matris2[h][l]=b;
```

```
            }
        }
        int r=(matris[0][0]*matris2[0][0])
              +(matris[0][0]*matris2[0][1])
              +(matris[0][0]*matris2[0][2]);
        int b=(matris[1][0]*matris2[1][0])
              +(matris[1][0]*matris2[1][1])
              +(matris[1][0]*matris2[1][2]);
        int c=(matris[2][0]*matris2[2][0])
              +(matris[2][0]*matris2[2][1])
              +(matris[2][0]*matris2[2][2]);
        int k=(matris[0][1]*matris2[0][0])
              +(matris[0][1]*matris2[0][1])
              +(matris[0][1]*matris2[0][2]);
        int q=(matris[1][1]*matris2[1][0])
              +(matris[1][1]*matris2[1][1])
              +(matris[1][1]*matris2[1][2]);
        int p=(matris[2][1]*matris2[2][0])
              +(matris[2][1]*matris2[2][1])
              +(matris[2][1]*matris2[2][2]);
        int j=(matris[0][2]*matris2[0][0])
              +(matris[0][2]*matris2[0][1])
              +(matris[0][2]*matris2[0][2]);
        int m=(matris[1][2]*matris2[1][0])
              +(matris[1][2]*matris2[1][1])
              +(matris[1][2]*matris2[1][2]);
        int v=(matris[2][2]*matris2[2][0])
              +(matris[2][2]*matris2[2][1])
              +(matris[2][2]*matris2[2][2]);
        System.out.println("["+r+" "+b+" "
                                        +c+"]");
        System.out.println("["+k+" "+q+" "
                                        +p+"]");
        System.out.println("["+j+" "+m+" "
                                        +v+"]");

    }
}
```

Yuqoridagi bu kodimizni berilgan GitHub manzilimizdan yuklay olasiz.

Manzil:*https://github.com/21040001/Java_yordamchi_kitob_kodlarimiz/blob/main/massiv_for.java*

Keling eng yaxshisi, bu dasturimizni ishga tushirib ko'raylik.

```
Birinchi matritsaning 0 ustun 0 a'zosini kir : 2
Birinchi matritsaning 0 ustun 1 a'zosini kir : 3
Birinchi matritsaning 0 ustun 2 a'zosini kir : 5
Birinchi matritsaning 1 ustun 0 a'zosini kir : 7
Birinchi matritsaning 1 ustun 1 a'zosini kir : 6
Birinchi matritsaning 1 ustun 2 a'zosini kir : 5
Birinchi matritsaning 2 ustun 0 a'zosini kir : 7
Birinchi matritsaning 2 ustun 1 a'zosini kir : 5
Birinchi matritsaning 2 ustun 2 a'zosini kir : 5
Ikkinchi matritsaning 0 ustun 0 a'zosini kir : 7
Ikkinchi matritsaning 0 ustun 1 a'zosini kir : 5
Ikkinchi matritsaning 0 ustun 2 a'zosini kir : 5
Ikkinchi matritsaning 1 ustun 0 a'zosini kir : 7
Ikkinchi matritsaning 1 ustun 1 a'zosini kir : 9
Ikkinchi matritsaning 1 ustun 2 a'zosini kir : 7
```

```
Ikkinchi matritsaning 2 ustun 0 a’zosini kir : 4
Ikkinchi matritsaning 2 ustun 1 a’zosini kir : 4
Ikkinchi matritsaning 2 ustun 2 a’zosini kir : 6
[34 161 98]
[51 138 70]
[85 115 70]
```

5.13 Break va Continue operator kodlari

Break operatori: bu operatorimizdan, takrorlash operatori ichida ma’lum bir shartlar o’rtaga kelganda, takrorlash operatorini to’xtatish uchun foydalaniladi. **Break** operatorini yanada yaxshi tushunishimiz uchun pastki qismda berib o’tilgan kodni, o’rganib chiqaylik.

```
public class Break_misol{
    public  static void main(String[] args){
        for(int i=0; i<10; i++){
            if(i==5){
                break;
            }
            System.out.println(i);

        }
    }
}
```

Break_misol nomli kodimizda for takrorlash operatoridan foydalangan holda 0dan 10gacha bo’lgan sonlarni ekranga chiqarishini istadik. Keyin esa for blok ichkarisida i o’zgaruvchi qiymatimiz 5ga teng bo’lganda, break operatori orqali kodni

ishlash jarayonini to'xtatishni istadik. Buning natijasida dasturimiz ishga tushadi va sonlarni ekranga chiqarishni boshlaydi, 4 soniga kelganda ishdan chiqadi kodimiz. Ya'ni quyidagicha:

```
0
1
2
3
4
```

Continue operatori: bu operatorimiz break operatorimizga deyarli o'xshab ketadi. Ikki operator orasidagi farq, break operatorimiz ma'lum bir shartga to'g'ri kelib qolganda, takrorlash operatorining ishlashini to'xtatayotgandi. Continue esa ma'lum bir shartga kelganda, osha shartga oid kodlarni ishga tushirmaydi va bir keyingi kodlarni normal shaklda ishlatishda davom etadi. Yuqorida berilgan **Break_misol.java** nomli kodimizda Continue operatorini foydalanib ko'raylik.

```
public class Continue_misol {
    public  static void main(String[] args){
        for(int i=0; i<10; i++){
            if(i==5){
                continue;
            }
            System.out.println(i);
        }
    }
}
```

Continue_misol nomli kodimizni ishga tushirganimizda, quyidagicha javob olamiz:

```
0
1
2
3
4
6
7
8
9
```

Bu kodimizning chiqargan natijalariga e’tibor bergan bo’lsangiz, 5 soniga yetib kelgan va 5 sonini ekranga chiqarmasdan davom etgan, break uchun bergan misolimizning natijasida esa 5gacha bo’lgan sonlarni ekranga chiqarib, 5ga kelganda ishdan chiqqan edi.

Obyektga yo'naltirilgan Java

6

6.1 Kirish

Java dasturlash tilining asosiy qismlari 2 dona hisoblanadi birinchisi sinflar, ikkinchisi esa obyektlar qismidir. Java dasturlash tilida tayyorlamoqchi bo'lgan yoki tayyorlangan barcha programmalarda sinflar va obyektlar mavjud. Chunki bu ikki qismlardan foydalanmagan holda bir programma kelishtirish imkonsiz. Bu sababdan Java dasturlash tiliga tarif berishda obyektga yo'naltirilgan bir dasturlash tili demoqdamiz.

Obyektga yo'naltirilgan dasturlash tillarida bilishimiz kerak bo'lgan birinchi qism bu **sinflar (class)** qismidir. Sinflar aslida programma ichidagi foydalaniladigan o'zgaruvchi gruppalari uchun shablonlar tayyorlaydi. Sinflar, ichidagi o'zgaruvchilar orqali turli xil turdagi ma'lumotlarni saqlay oladi. Ya'ni sinflar ichkarisida turli xil metodlar ko'rsata olamiz. Bu metodlar odatda o'zgaruvchilarga erishgan holda sinflarning harakatlarini o'zgartira oladi yoki sinflarning holati haqida tashqi dunyoga ma'lumotlar berish uchun foydalaniladi. Bir

sinf tayyorlaganimizdan so'ngra, bu sinfni yangi bir o'zgaruvchi turi yoki ma'lumot turi sifatida foydalana olamiz.

Obyektlar sinflarda berib o'tilgan shablonlarga moslashtirilgan xotiradagi misollardir. Misol uchun, shahar so'zi bir sinf va Farg'ona, Toshkent, New York bular esa shahar sinfining obyektlari hisoblanadi.

Bu bo'limda Javadagi sinf, obyekt va metod qismlarini ko'rib chiqamiz. Bularni ko'rib chiqish davomida turli xil misollar ham ko'rib o'rganib chiqamiz. Bu kodlarni barchasini GitHub onlayn platformasidan yuklay olasiz.

Bu bo'limdagi barcha kodlar GitHub platformasida bir fayl ichkarisida, ya'ni obyekt_java nomli fayl ichkarisida berilgan, buni bilishimiz kerak bo'ladi chunki bundan keyingi mavzularda boshqa fayl ichidagi fayllarni chaqirib olgan holda ulardan foydalanamiz. Bu fayl manzilini ham berib o'taylik. *https://github.com/21040001/Java_yordamchi_kitob_kodlarimiz/tree/main/obyekt_java*

6.2 Sinflar

Sinflar Java dasturlash tilidagi eng muhim qismlardan biri hisoblanadi. Butin kodlarimiz sinflar ichkarisida yoziladi. Sinflar aslida, ma'lumotlarni birma-bir o'zgaruvchilarga saqlamasdan, bir-biriga aloqali bo'lgan ma'lumotlarni bir joyga to'plagan holda bir ma'lumot sifatida tanitish va erishish uchun foydalaniladi.

Misol uchun, bir foydalanuvchining ma'lumotlarini saqlay olishimiz uchun, foydalanuvchining ismi, familiyasi, yoshi, jinsi kabi ma'lumotlari kerak bo'ladi. Bularni har birini bir

o‘zgaruvchiga saqlash o’rniga, foydalanuvchi nomi bilan bir sinf tayyorlab , foydalanuvchi haqidagi ma’lumotlarni bu yerda gruppalay olamiz.

Pastdagi misolda o’quvchilarning ma’lumotlarini saqlashimizga yordamchi bo’lishi uchun tayyorlangan **Foydalanuvchi.java** nomli sinf ko’rsatib o’tilgandir.

```
package Sinflar;
public class Foydalanuvchi {

    //o‘zgaruvchilar
    private String ism ;
    private String familiya;
    private int yosh;
    private String jinsi;

    //quruvchi metodlar
    public Foydalanuvchi() {
        ism = "";
        familiya = "";
        yosh =-1 ;
        jinsi="";
    }
    public Foydalanuvchi(String pism ,
                    String pfamiliya,
                    int pyosh,
                    String pjinsi){

        ism = pism;
        familiya =pfamiliya;
        yosh =pyosh;
        jinsi = pjinsi;
    }

    /* foydalanuvchi haqidagi ma’lumotlarga
    erishishimizni saqlaydigan getter/setter
```

```
    metodlari*/

    public String getIsm() {
        return ism;
    }

    public void setIsm(String ism) {
        this.ism = ism;
    }

    public String getFamiliya() {
        return familiya;
    }

    public void setFamilya(String familya) {
        this.familiya = familiya;
    }

    public int getYosh() {
        return yosh;
    }

    public void setYosh(int yosh) {
        this.yosh = yosh;
    }

    public String getJinsi() {
        return jinsi;
    }

    public void setJinsi(String jinsi) {
        this.jinsi = jinsi;
    }
    // boshqa metodlar
    public String toliq_ismi(){
        return this.ism + " " + this.familiya;
    }
}
```

Bu kodimizni quyidagi GitHub manzilidan yuklay olasiz: *https://github.com/21040001/Java_yordamchi_kitob_kodlarimiz/blob/main/obyekt_java/sinflar/Foydalanuvchi.java*

Yuqoridagi kodimizda e'tibor berishingiz kerak bo'lgan eng muhim narsa, sinfimizning turgan fayl nomi bilan sinfning nomini bir xil bo'lishi kerakligidir. Sinfimizning nomi Foydalanuvchi bo'lganligi uchun sinfimizni **Foydalanuvchi.java** nomli fayl ichiga saqlab qo'ydik. Agar sinfin kodlarini ko'rib chiqadigan bo'lsak, sinfimiz to'rt qismlardan tashkil topganini ko'ramiz.

- **O'zgaruvchilar :** Sinf ichkarisida saqlanadigan ma'lumotlarning, sinf ichidagi maxsus o'zgaruvchilari.
- **Quruvchi metodlar:** Odatda sinflar yangilanib chaqirilayotganda new operatori bilan birgalikda chaqiriladigan metodlar beriladi.
- **Foydalanuvchi haqidagi ma'lumotlarga erishishimizni saqlaydigan getter/setter metodlari:** Bu metodlar normal hollarda public sifatida ko'rsatiladi va private sifatida ko'rsatib o'tilgan ma'lumot o'zgaruvchilarga tashqi tarafdan erishishni ta'minlaydi.
- **Boshqa metodlar:** Sinf ichkarisidagi ma'lumotlardan foydalangan holda, sinf bilan bog'liq bo'lgan turli xil natijalar chiqaradigan metodlar.

Deyarli barcha sinflarimiz bu to'rt qismdan tashkil topadi. Keyingi mavzularda bu qismlar haqida batafsilroq o'rganib chiqamiz, o'rganib chiqishimizdan oldin obyektlarni ko'rib chiqaylik.

6.3 Obyektlar

Bir sinf tayyorlaganimizda, aslida bir ma’lumot turi tayyorlagan bo’lamiz. Java dasturlash tilida, tayyorlangan bu yangi ma’lumot turiga ega bo’lgan o‘zgaruvchilarga obyektlar deyiladi. Misol uchun, yuqorida tayyorlagan Foydalanuvchi nomli sinfimizdan yangi obyektlar tayyorlash uchun quyidagi kodimizni ko’rib chiqaylik.

```
package Obyekt;
import Sinflar.Foydalanuvchi;
public class obyekt_misol {

    public static void main(String[] args) {

        Foydalanuvchi foydalanuvchi_1 =
                        new Foydalanuvchi();

        foydalanuvchi_1.setIsm("Akmal");

        foydalanuvchi_1.setFamilya
                                ("Roziboyev");

        foydalanuvchi_1.setYosh(21);

        foydalanuvchi_1.setJinsi("Erkak");

        /* birinchi foydalanuvchi
        ma'lumotlarini ekranga chiqarish: */
        System.out.println
                (foydalanuvchi_1.toliq_ismi());

        System.out.println("Yoshi: "
                +foydalanuvchi_1.getYosh());

        System.out.println("Jinsi: "
                +foydalanuvchi_1.getJinsi());
```

```
        System.out.println("----------------");

        /* boshqa bir foydalanuvchi
        ma'lumotlarini qo'shish*/
        Foydalanuvchi foydalanuvchi_2 =
                new Foydalanuvchi
                ("Shukurano","Abdurazzokova",
                 20,"Ayol");

        /* ikkinchi foydalanuvchi
        ma'lumotlarini ekranga chiqarish: */
        System.out.println
                (foydalanuvchi_2.toliq_ismi());
        System.out.println("Yoshi: "
                +foydalanuvchi_2.getYosh());
        System.out.println("Jinsi: "
                +foydalanuvchi_2.getJinsi());
    }
}
```

Yuqoridagi kodimizni berilgan GitHub manzilidan yuklay olasiz:

https://github.com/21040001/Java_yordamchi_kitob_kodlarimiz/blob/main/obyekt_java/Obyekt/obyekt_misol.java

Yuqoridagi **obyekt_misol.java** nomli kodimizning ishlash jarayoni haqida qisqacha ma'lumot berib o'taman. Bu kodimizda foydalanuvchi sinfidan 2 dona obyekt tayyorlandi va foydalanuvchi_1 , foydalanuvchi_2 nomlari mos shaklda berildi. Undan so'ngra, har bir obyektga **set** metodi yordami bilan qiymatlar berildi. So'ngra, bu obyektlarga berilgan qiymatlar **get** metodi yordami bilan ekranga chiqarildi. Bu kodimizni ishga tushirganimizda quyidagicha bir javob olamiz:

```
Akmal Roziboyev

Yoshi: 21

Jinsi: Erkak

---------------------------

Shukurano Abdurazzokova

Yoshi: 20

Jinsi: Ayol
```

6.4 public, private va protected kalit so’zlari yordamida sinf ichidagi o’zgaruvchilarga va metodlar uchun erishish darajalari

Bu kalit so’zlar sinf ichkarisidagi o’zgaruvchilarga qanday shaklda qaysi fayllardan erisha olishimizni ko‘rsatib turishi uchun foydalaniladi. Hozir sizlar bilan birgalikda ushbu kalit so’zlarni o’rganib chiqaylik.

private kalit so’zi: Bu kalit so’zi orqali biron berilgan bir o’zgaruvchilarni yoki metodlarni tashqi tarafdan ko’rinmaydigan holga keltiramiz. **Private** kalit so’zi orqali ko’rsatilgan o’zgaruvchilar yoki metodlardan tashqi tarafdan foydalana olmaymiz. Bulardan faqatgina o’zgaruvchilar yoki metodlar berilgan sinfda foydalana olamiz. Misol uchun, Foydalanuvchi nomli sinfimiz ichkarisida berilgan ism ,yosh kabi o’zgaruvchilar private kalit so’zi orqali berilgan. Bu sababdan, bu o‘zgaruvchilarga **obyekt_misol.java** nomli

programmamizdan to'g'ridan-to'g'ri bog'lana olmadik. Ulardan foydalana olish uchun Foydalanuvchi nomli sinfimizda turli xil ishlar bajardik getter/setter kabi.

public kalit so'zi: Bu kalit so'zimiz private kalit so'zining vazifasiga tamomiyla teskari bo'lganini qiladi. Ya'ni bu kalit so'z bilan berilgan o'zgaruvchi yoki metodlar istalgan joydan chaqirilib foydalanilishi mumkin. Agar bir o'zgaruvchi yoki metodni barcha sinflar ichida foydalaniladigan qilmoqchi bo'lsangiz uning yoniga public kalit so'zini yozib qo'yish kifoya.

protected kalit so'zi: Bu kalit so'zi bilan birgalikda ko'rsatilgan o'zgaruvchilar yoki metodlar faqatgina ko'rsatilgan sinf ya'ni shu **protected** orqali ko'rsatilgan o'zgaruvchi yoki metodlar joylashgan sinfi bilan bir fayl ichkarisida bo'lgan sinflardan chaqirib foydalana olamiz.

Keling birgalikda bir qancha misollar ko'rib chiqamiz:

```
package Sinflar;

public class kalit_sozlar {
    public int yosh1 =0;
    private int yosh2 = 0;
    protected int yosh3 =0;
}
```

Ko'rganingiz kabi **kalit_sozlar** nomli sinfimizning kodlarini yozib oldik, endi esa Obyekt nomli fayl ichkarisida joylashgan boshqa kod fayliga bu sinfimizni chaqirgan holda foydalanish shaklarini o'rganib chiqaylik.

```
package Obyekt;

import Sinflar.kalit_sozlar;

public class kalit_sozlar_main {
```

```
public static void main(String[] args) {
    kalit_sozlar kalit_soz =
                    new kalit_sozlar();

    /*Bu faylimiz ishga tushirilganda
    pastki qatordagi kodimiz xatolarsiz
    ishga tushadi chunki Sinflar fayli
    ichkarisidagi kalit_sozlar faylimizda
    yosh1 o'zgaruvchini public kalit so'zi
    bilan berilgan edi. Ya'ni public kalit
    so'zi bilan berilgan o'zgaruvchilarni
    istalgan fayldan chaqirib, undan
    foydalana olamiz.  */

    kalit_soz.yosh1= 15;

    /*Bu faylimiz ishga tushirilganda
    pastki qatordagi kodimiz xato beradi
    chunki Sinflar fayli ichkarisidagi
    kalit_sozlar faylimizda yosh2
    o'zgaruvchi private kalit so'zi bilan
    berilgan edi. Ya'ni private kalit
    so'zi bilan berilgan o'zgaruvchilarni
    boshqa fayllardan chaqirib foydalana
    olmaymiz  */

    kalit_soz.yosh2 = 15;

    /*Bu faylimiz ishga tushirilganda
    pastki qatordagi kodimiz xato beradi
    chunki Sinflar fayli ichkarisidagi
    kalit_sozlar faylimizda yosh3
    o'zgaruvchi protected kalit so'zi
    bilan berilgan edi. Ya'ni protected
    kalit so'zi bilan berilgan
    o'zgaruvchilar, faqatgina shu fayl
    bilan bir fayl ichida bo'lgan boshqa
    bir Java faylda chaqirib foydalana
    olar edik. Misol uchun kalit_sozlar
```

```
        sinfimiz joylashgan fayl ichida ya'ni
        Sinflar fayli ichida boshqa bir kod
        fayli ochib yosh3 qiymatini chaqirib
        foydalana olamiz chunki ikkala faylimiz
        ham bir fayl joylashgan bo'ladi. */

        kalit_soz.yosh3 =15;

    }

}
```

Yuqorida berib o'tgan misollarimiz GitHub sahifamizga joylashtirilgan.

E'tibor bergan bo'lsangiz, misol sifatida berayotgan kodlarimizning birinchi qatorida **"package"** so'zi bilan yozilgan kod bor. Bu kodlar shu ma'noda keladi, bu kod faylimiz shu fayl ichkarisida joylashgan ma'nosida. Misol uchun "**package Sinflar;**", bu kodimiz Sinflar nomli bir fayl ichkarisida turganligini yoki saqlanganligini ko'rsatib turibdi

Yuqorida berilgan ikki kod fayllarimiz ham ushbu, manzili berib o'tilgan fayl ichkarisida joylashgan. Qaysi kod qaysi fayl ichida ekanligini kodlarimizning birinchi qatorida berib o'tilgan **package** kodi orqali bilib olishingiz mumkin.

Manzil:
https://github.com/21040001/Java_yordamchi_kitob_kodlarimiz/tree/main/obyekt_java

6.5 New operatori

Bu operatorimiz bir sinf nomi bilan birgalikda foydalaniladi. Berilgan sinf uchun xotiradan yangi bir obyekt tayyorlaydi. Foydalanish shakliga bir misol:

```
Talaba talaba1 = new Talaba();
```

Yuqoridagi misolimiz talaba1 obyekti uchun Talaba sinfiga oid bo'lgan xotira qismidan bir joy ajratib beradi. Agar bir obyekt *new* operatori bilan berilmagan bo'lsa, Java unga avtomatik bir shaklda **null** qiymatini beradi. Bu holatda bir sinf ichkarisidagi metodlarga yoki o'zgaruvchilarga erishmoqchi bo'lganimizda, dasturimiz **NullPointerException** nomli bir xato beradi.

6.6 Nuqta (.) operatori

Obyektlar tayyorlanib new operatori orqali xotiradan joy olgan so'ngra, obyektlarning public va protected kalit so'zlari bilan berilgan o'zgaruvchilar yoki metodlarni tashqaridan foydalanish uchun nuqta (.) operatori foydalaniladi. Bu operatordan foydalanishni ko'rib chiqaylik.

```
//….

Talaba talaba1 = new Talaba();

talaba1.setIsm("Jafar");

talaba1.setFamilya("Sultan");

//….
```

Yuqorida berilgan bu misolimizni tushuntirib o'tadigan bo'lsam, birinchi bo'lib, Talaba turida bo'lgan talaba1 nomli obyekt tayyorlangan. Bu obyektning sinfida setIsm nomli bir metod tayyorlangan, u metodni tayyorlashda esa public kalit so'zidan foydalanilgan. Yuqoridagi kodimizda talaba1 metodini chaqirish uchun avval sinfning xotiraga saqlangan ismini

yozamiz keyin esa nuqta qo'ygan holda bu sinf ichkarisida foydalanmoqchi bo'lgan metodimizni chaqirgan holda foydalana olamiz, uchinchi va to'rtinchi qatordagi kabi.

6.7 Metodlar

Metodlar obyektga yo'naltirilgan dasturlash tillarida juda ham muhim bo'limlardan biri hisoblanadi. Metodlar bo'limi obyektga yo'naltirilgan dasturlash tillari yaratilishidan oldin boshqa dasturlash tillarida juda ham ko'p foydalanilgan bir bo'lim. Metodlar o'ziga parametr sifatida berilgan ma'lumotlar bilan turli xil ishlar bajaradi. Bu ishlar bajarilgandan so'ngra istaklarga bog'liq bir shaklda turli xil natijalar chiqaradi.

Metodlar deyarli barcha dasturlash tillarida mavjud. ammo nomlanish shakli boshqacha bo'lishi mumkin, misol uchun Java dasturlash tilidagi metodlar C dasturlash tillarida funksiyalar nomi olishi mumkin. Bir metodning tuzilish shakli quyidagicha:

```
Maxsus_kalitsoz      nat_turi    metod_nomi(parametrlari)
{

        metod_kodlari;

}
```

Yuqoridagi metod shaklining har bir a'zosini detalli bir shaklda ko'rib chiqadigan bo'lsak:

Maxsus_kalitsoz: Bu qismni berish majburiy. Bu qismda metodga erishish holatlarini ko'rsatib turadigan **private, public, protected** kalit so'zlarini va metodning statik yoki statik

emas ekanligini ko'rsatib turadigan **static** kalit so'zlarini foydalana olamiz.

Nat_turi: Normal vaziyatlarda metodlarimiz bir natija chiqaradi. Metodlarning natija chiqarishi, metod ichkarisidagi return kodi bilan bog'liq. Return kodi bilan chiqarilgan natija turi bilan nat_turi qismida berilgan ma'lumot turi bir xil bo'lishi kerak. Misol uchun metodimiz ishga tushgandan keyin, int turida bo'lgan bir natija chiqaradigan bo'lsa, nat_turi qismi ham int bo'lishga majbur. Agar metodimiz har qanday bir natija chiqarmaydigan bir metod bo'lsa, nat_turi qismiga **void** kalit so'zi yoziladi.

Metod_nomi: Bu qismda metodga nom beriladi berilgan nom orqali tashqaridan chaqirib foydalana olamiz.

Parametrlar: Bu qismda metodning bajaradigan ishlari uchun kerak bo'ladigan parametrlar berib o'tiladi. Bu parametrlar qavs ichkarisida beriladi va bir biridan vergul orqali ajratiladi. Pasta berilgan **hisobla** nomli metodga e'tibor berganimizda qavs ichida **a** va **b** nomli ikki parametrlar berilganligini ko'ramiz. Agar bir metodning parametrlari mavjud bo'lmasa qavs ichkarisi bo'sh qolishi kerak.

Metod_kodlari: Bu qismda metodimiz chaqirilganda bajariladigan kodlar yoziladi. Kodlarimiz bir qator ham bo'lishi mumkin, minglab qator ham bo'lishi mumkin har qanday bir chegara berilmagan.

Keling birgalikda 2 dona qiymatni bir birga qo'shib beradigan bir metod kodini yozib ko'ramiz.

```
//….
public int hisobla (int a , int b){
int d;
            d= a+b;
return d ;
}
//…
```

Quyidagi metodimizni chaqirib foydalanmoqchi bo'lganimizda, bizdan 2 dona a va b o'zgaruvchilar uchun qiymat berishimizni istaydi. Qiymatlarimizni qavs ichiga vergul bilan ajratgan holda yozamiz. So'ngra bu metodimiz bergan qiymatlarimizning yig'indisini bizga chiqarib beradi. Bu metodimiz normal bir statik bo'lmagan bir metod hisoblanadi.

6.8 Metodlardan foydalanish shakllari

Metodlar bir marta ko'rsatib o'tilgandan so'ngra, ulardan istaganimizcha foydalana olamiz. Misol uchun yuqoridagi **hisobla** nomli metodimizni kod ichida quyidagicha foydalana olamiz.

```
int son_1 = 7;
int son_2 = 8;
int yigindi = hisobla(son_1, son_2);
System.out.println(yigindi);
```

Yuqorida kodimizda birinchi bo'lib ikki dona son_1 va son_2 nomli o'zgaruvchilar berdik va ularga mos ravishda 7 va 8 qiymatlarini berdik. Keyin esa , **yigindi** nomida bir o'zgaruvchi berdik va bu o'zgaruvchi uchun qiymat sifatida **hisobla** nomli metodimizning chiqaradigan natijasini berib o'tdik. Metodimiz bir natija berishi uchun biz ikki dona qiymat berishimiz kerak edi, shuning uchun biz metodimizga parametr qiymati sifatida **son_1** va **son_2** o'zgaruvchi qiymatlarini berib o'tdik, uchinchi qatorda ko'rib turganingiz kabi. Eng so'ngda esa **yigindi** nomli o'zgaruvchining qiymatini boshqa so'z bilan aytganda **hisobla** metodining chiqaradigan natijasini ekranga chiqarishini istadik.

Metodlar dasturchilarning eng katta yordamchilaridir, ya'ni dasturchilarga kodlarni tushunarli va tez yozishga yordam beradi deya olamiz. Bir dasturchi kodlarni metod yordami bilan yozib uni istagan joyida chaqirib foydalana oladi ya'ni dasturchilarga kodlarni takror-takror kodlamasligi uchun yordam beradi, metodlar.

6.9 Konstruktor metodlar (Constructors)

Bu metodlar, new kalit so'zi yordami bilan yangi bir obyekt tayyorlanganda avtomatik bir shaklda ishga tushiriladi. Bu metodlarning nomi , ichida joylashgan sinf nomi bilan bir xil bo'lishi kerak. Ya'ni bu metod qaysi sinf ichida yozilgan bo'lsa, metodning nomi o'sha sinfning nomi bilan bir xil bo'lishga majbur. Keling bu metodimizga bir misol ko'rib chiqamiz.

```
public class Azolar {
    private String ism;
    private String familiya;
    private int yoshi;
```

```
    // konstruktor metod - 1
    public Azolar(){
        ism ="";
        familiya="";
        yoshi=0;

    }
    // konstruktor metod - 2
    public Azolar(String pism,
                  String pfamilya,
                  int pyoshi){

        ism = pism;
        familiya = pfamilya;
        yoshi= pyoshi;

    }
}
```

Konstruktor metodlarni boshqa metodlardan ajratish uchun metodga berilgan nomga qarashingiz yetarli, agar metod nomi va metod joylashgan sinf nomi bir xil bo'lsa demakki bu metod konstruktor metodlaridan bir hisoblanadi. Yuqoridagi kodimizda ikki dona konstruktor metodlari mavjud. E'tibor bergan bo'lsangiz metodlarning nomi bilan sinfning nomi bir xil, ya'ni ikkisi ham **Azolar** nomini olgan.

Agar metod har qanday bir parametrdan foydalanmagan holda chaqirilsa, birinchi metodimiz chaqirilgan hisoblanadi. Agar uch dona parametr berilgan holda chaqirilsa ikkinchi metodimiz chaqirilgandir. Berilgan parametrlar, tabiiyki metodimiz ichidagi parametr turlariga mos bo'lishi kerak, ya'ni metodimiz int turidagi parametr istasa int turidagi parametr beriladi.

Har bir sinf ichkarisida konstruktor metodi bo'lishi kerak degan bir majburiyat yo'q. Agar sinf ichkarisida konstruktor metod

bo'lmasa, dasturlash tilimiz sinf ichkarisida bir bo'sh bo'lgan konstruktor metod mavjud deb hisoblaydi.

Konstruktor metodlar new operatori bilan birgalikda ishlaydi. Pastdagi misol kodimizda azo_1 uchun new operatoridan foydalanilgan holda **Azolar** sinfidan yangi bir obyekt tayyorlangan va parametr berilmagani uchun birinchi metod chaqirilgan deb hisoblangan. Kod ichkarisida yana bir azo_2 nomli obyekt ham tayyorlangan, tayyorlash davomida uch dona parametr berilgani uchun ikkinchi metod chaqirildi deb hisoblandi.

```
// Birinchi metod chaqiriladi

Azolar azo_1 = new Azolar();

// Ikkinchi metod chaqiriladi

Azolar azo_2 =  new Azolar("Mert", "Arikan", 22);
```

6.10 Metodlarning ortiqcha yuklanishi (Method overloading)

Metodlarni bir-biridan ajrata olishimiz uchun metodlarga farqli-farqli ismlar beramiz. Metodlarning ortiqcha yuklanishi (metod overloading) deb - bir xil nomda bo'lgan bir qancha metodlar, parametrlari soni farqli yoki parametr turi farqli bo'lgan metodlar bir sinf ichkarisida tayyorlanishiga aytiladi. Keling eng yaxshisi bir misol ko'rib chiqaylik.

```
package Obyekt;

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


        int natija1 = Hisobla(12, 23);

        int natija2 = Hisobla(34, 23, 2);

        int natija3 = Hisobla(34, 23, 2 ,7);

        int natija4 = Hisobla(36);

        System.out.println("Birinchi natija :"
                                +natija1);
        System.out.println("Ikkinchi natija :"
                                +natija2);
        System.out.println("Uchinchi natija :"
                                +natija3);
        System.out.println("To'rtinchi natija
                                :"+natija4);

    }

    public static int Hisobla(int a, int b){
        return a+b;
    }

    public static int Hisobla(int a,
                                int b,
                                int g){
        return a+b+g;
    }

    public static int Hisobla(int a,
                                int b,
                                int x,
                                int y){
        return (a+b+x+y)/4;   }
```

```
    public static int Hisobla(int a) {
        return a*2;
    }

}
```

Yuqoridagi kodimizda, Hisobla metodi to'rt marta yaratilgan. Birinchi yaratilishidan keyingi qolgan uch yaratilishiga ortiqcha metod yuklanishi deyiladi. Yuqoridagi sinfimiz ichkarisidagi main metodning ichkarisida esa, metodlarning ortiqcha yuklanishi Hisobla metodini foydalanish shaklini ko'rishingiz mumkin. Bu metod ichkarisida metodlardan foydalanish uchun bir-biridan farqli miqdorlarda parametrlar berilgan, bu parametrlar soniga qarab qaysi bir metod foydalanilganini bilib olishingiz mumkin.

6.11 Static metodlar

Bir metod yaratilishida, metodning nomidan oldin static kalit so'zi foydalanilsa, bu metod statik bo'lgan bir metod hisoblanadi. Static metodlarni, yangi bir obyekt yeri ochmasdan, to'g'ridan to'g'ri sinf nomi bilan chaqira olamiz. Misol uch dona static metodlardan tashkil topgan pastki qismda berilgan misolimizni ko'rib chiqaylik.

```
                Zamon_vaqt.java
package Sinflar;

import java.sql.Timestamp;
import java.text.SimpleDateFormat;
import java.util.Date;
```

```
public class Zamon_Vaqt {

    public static String
                sananikorsat(Timestamp ts) {

        Date date = new Date(ts.getTime());
        String korsatish = "dd/mm/yyyy";

        SimpleDateFormat sdf =
              new SimpleDateFormat(korsatish);

       return sdf.format(date);
    }

    public static String
                kunni_korsat(Timestamp ts) {

        Date date = new Date(ts.getTime());

        String korsatish = "dd";

        SimpleDateFormat sdf =
              new SimpleDateFormat(korsatish);

        return sdf.format(date);
    }
    public static String
                vaqtni_korsat(Timestamp ts) {

        Date date = new Date(ts.getTime());

        String korsatish = "hh:mm";

        SimpleDateFormat sdf =
              new SimpleDateFormat(korsatish);

        return sdf.format(date);
    }

}
```

Yuqorida misol shaklida berib o'tgan bu kodimiz keyingi mavzularda ham foydalaniladi. Hozir bu sinf ichidagi static metodlar qanday chaqirilishini ko'rib chiqaylik. Bu sinfimizni Vaqt_zamon_main.java nomli kod faylimizga chaqirgan holda, qanday chaqirishni ko'rib o'rganib chiqamiz.

```
package Obyekt;

import Sinflar.Zamon_Vaqt;
import java.sql.Timestamp;

public class Vaqt_zamon_main {

    public static void main(String[] args) {
        Timestamp ts =new Timestamp((
              newjava.util.Date()).getTime());

      /*statik metodlarni chaqirishimiz uchun
        new operatoridan foydalanishga kerak
        yo'q metodlarga sinf nomi bilan erisha
        olamiz.  */

        String natija1 =
               Zamon_Vaqt.kunni_korsat(ts);
        String natija2 =
               Zamon_Vaqt.sananikorsat(ts);
        String natija3 =
               Zamon_Vaqt.vaqtni_korsat(ts);

        System.out.println("Bugunki kun : "
                           +natija1);

        System.out.println("Bugunki sana : "
                           +natija2);
        System.out.println("Soat : "+natija3);
    }

}
```

Yuqoridagi kodimizda ko'rganingiz kabi boshqa bir sinf ichida berilgan metodni chaqirish uchun new operatoridan foydalanmagan holda metodlarimizni chaqira oldik, chunki bu metodlarimiz static metodlar edi. Chaqirishda sinfning nomini yozdik keyin esa, chaqirmoqchi bo'lgan metodimizni nomini yozgan holda osonlik bilan metodga erisha oldik.

Yuqorida berib o'tgan misollarimizning GitHub manzili quyidagicha:

Zamon_vaqt.java nomli kod faylimizning GitHub manzili: *https://github.com/21040001/Java_yordamchi_kitob_kodlarimiz/blob/main/obyekt_java/Sinflar/Zamon_Vaqt.java*

Vaqt_zamon_main.java nomli kod faylimizning GitHub manzili: *https://github.com/21040001/Java_yordamchi_kitob_kodlarimiz/blob/main/obyekt_java/Obyekt/Vaqt_zamon_main.java*

6.12 Static kalit so'zining boshqa joylarda foydalanish shakllari

Bu kalit so'zimizdan, metodlardan tashqari boshqa yana ikki joyda farqli bir shaklda, farqli vazifalar uchun foydalaniladi.

- Sinf ichida o'zgaruvchi berishda
- Qiymati o'zgarmaydigan o'zgaruvchilarni berishda

Sinf ichida static o'zgaruvchi berish

Sinf ichida static bir o'zgaruvchi berish uchun, o'zgaruvchi turini yozishdan avval **static** kalit so'zini yozish yetarli. Har qanday bir sinf ichkarisida joylashgan o'zgaruvchining oldida

static kalit so'zi mavjud bo'lsa, ushbu sinfdan foydalangan holda yaratilgan barcha obyektlarda bu o'zgaruvchining qiymati bir xil bo'ladi. Tushunarli bo'lishi uchun pastki qismda berilgan misolni ko'rib chiqaylik.

```
package Sinflar;

public class Static_misol_1 {
    public static int statik_ozgaruvchi = 0;
    public int normal_ozgaruvchi = 0;
    public static final double ozgarmaz
                                        = 3.14;

}
```

Ushbu sinfimizni main faylga chaqirgan holda ko'rib chiqaylik.

```
package Obyekt;

import Sinflar.Static_misol_1;
public class Static_misol_1_main {

    public static void main(String[] args) {
        Static_misol_1 son1 =
                        new Static_misol_1();
        Static_misol_1 son2 =
                        new Static_misol_1();

        son1.statik_ozgaruvchi = 5;

        System.out.println("Son1 nomli obyekt
                ichkarisidagi statik
                o'zgaruvchining qiymati :"
                +son1.statik_ozgaruvchi);

        System.out.println("Son2 nomli obyekt
                ichkarisidagi statik
                o'zgaruvchining qiymati :"
                +son2.statik_ozgaruvchi);
```

```
        System.out.println("----------------");

        son1.normal_ozgaruvchi=5;
        System.out.println("Son1 nomli obyekt
                ichkarisidagi normal
                o'zgaruvchining qiymati :"
                +son1.normal_ozgaruvchi);

        System.out.println("Son2 nomli obyekt
                ichkarisidagi normal
                o'zgaruvchining qiymati :"
                +son2.normal_ozgaruvchi);

        System.out.println("----------------");

        /* static o‘zgaruvchilarga sinf nomi
        orqali to'g'ridan-to'g’ri erisha
        olamiz*/

        System.out.println("Sinf nomini
                foydalangan holda
                o'zgaruvchiga to'g’ridan
                erishish : "
                +Static_misol_1
                .statik_ozgaruvchi);

        /*pastki qatorda yozilgan kodimiz
        ishlamaydi chunki biz bu o‘zgaruvchini
        qiymatini final so’zi orqali o‘zgarmas
        shaklga keltirgan edik */
        son1.ozgarmaz= 3.141;

    }

}
```

Ushbu kodimizni ishlaydigan holga keltirganimizdan so’ng ishga tushirganimizda pastki qismda ko’rsatilgani bir javob olamiz. Kodimiz ishlashi uchun eng oxirida yozilgan

son1.ozgarmaz= 3.141; kodimizni o’chirib qo’yishimiz kerak chunki bu kod qatori xato beradi.

```
Son1 nomli obyekt ichkarisidagi statik
o'zgaruvchining qiymati :5

Son2 nomli obyekt ichkarisidagi statik
o'zgaruvchining qiymati :5

----------------------

Son1 nomli obyekt ichkarisidagi normal
o'zgaruvchining qiymati :5

Son2 nomli obyekt ichkarisidagi normal
o'zgaruvchining qiymati :0

----------------------

Sinf nomini foydalangan holda o'zgaruvchiga
to'g’ridan erishish : 5
```

Yuqoridagi main fayl ichidagi kodimizga e’tibor bergan bo’lsangiz, sinfimiz ichkarisida statik qiymat sifatida berilgan o’zgaruvchining qiymati, son1 obyekti ichida 5 sifatida almashtirildi. Son1 obyekti ichida almashtirganimizda avtomatik bir shaklda son2 obyekti ichidagi qiymati ham 5 sifatida almashdi.

Yuqorida misol sifatida berib o’tgan kodlarimizni quyidagi GitHub manzilidan yuklay olasiz.

Static_misol_1.java kod faylimizning GitHub manzili: *https://github.com/21040001/Java_yordamchi_kitob_kodlarimiz/blob/main/obyekt_java/Sinflar/Static_misol_1.java*

Static_misol_1_main.java kod faylimizning GitHub manzili: https://github.com/21040001/Java_yordamchi_kitob_kodlarimiz/blob/main/obyekt_java/Obyekt/Static_misol_1_main.java

Qiymati o'zgarmaydigan o'zgaruvchilar berish

Qiymati o'zgarmaydigan o'zgaruvchilar **static** va **final** kalit so'zlari yordamida beriladi, yuqoridagi misolimizdagi **ozgarmaz** nomli o'zgaruvchi kabi. Bu kabi o'zgaruvchilarga bir marta qiymat beriladi va berilgan qiymatni keyin o'zgartirish mumkin emas, o'zgartirilganda kod ishlamaydi ya'ni xato beradi, yuqoridagi misolimdagi kabi.

6.13 Takrorlanuvchi metodlar

Takrorlanuvchi metodlar o'zini o'zi chaqiradigan metodlardir. Bu metodlar ma'lum bir natijaga erishganda o'zini o'zi chaqirishni to'xtatadi. Dasturlashda bir qancha problemalarni yechishda ko'p foydalaniladigan usullardan biri bu **recursion**dir (o'zini-o'zi yangilovchi). Bu mavzuni tushuntirishda eng ko'p foydalanilgan misollar, bu faktorial misollaridir.

Misol uchun 5 sonining faktoriali 5! shaklda ko'rsatiladi. 5! hisoblaydigan bo'lsak : 5! = 5x4x3x2x1=120

Bu misolimizni takrorlanuvchi yoki yangilanuvchi metodimiz orqali kodlab ko'ramiz.

```
public static long recursion(int number){

      if (number==1 || number ==0) {
          return 1;
       }

      long natija = number*recursion
                                (number-1);
      return natija;
    }
```

Yuqoridagi recursion nomidagi yangilanuvchi metodimizning ishlash shaklini tushuntirib beradigan bo'lsam. Yuqoridagi bu metodimiz o'ziga parametr sifatida berilgan qiymat 0 yoki 1ga teng bo'lganida, natija sifatida 1 qiymatini chiqaradi. Chunki 0! va 1! hisoblanganda natija 1 chiqadi.

Agar parametr sifatida berilgan qiymat 1dan katta bo'lsa, o'zini o'zi chaqirayotgan onda, o'ziga parametr sifatida berilgan qiymatni 1ga kamaytirib qaytadan o'ziga parametr sifatida beradi. Parametr qiymati har safarda 1ga kamayib boradi. Qiymat 1ga teng bo'lgan vaqtda, o'zini o'zi yangilashni to'xtadi va natija hisoblashni boshlaydi. Shu shaklda berilgan qiymat faktorialini hisoblay olamiz.

6.14 Xatolarni oldini olish (Exception Handling)

Ba'zi vaqtlarda kodlarimiz biz istagan shaklda ishlamasligi mumkin, ya'ni ishlash davomida turli xatolar berishi mumkin. Misol uchun bir o'zgaruvchi turidan boshqa bir o'zgaruvchi turiga o'tayotgan vaqtda xato(exception) olish mumkin.

Boshqa bir misol berib o'tadigan bo'lsak, foydalanuvchidan 10 dona int turida bo'lgan sonlar kirishini istayapmiz, foydalanuvchi tarafidan kirilgan 10 dona son orqali aytaylik bir list tayyorlaymiz. Bu yerda foydalanuvchining 10 dona sonni barchasini int turida bermasligi kabi, bir ehtimol bor. Bu vaziyatda biz foydalanuvchining xato qiymatlar bermasligini oldini olishimiz kerak bo'ladi. Java dasturlash tilida xatolarni oldini olish uchun **try-catch** kod bloklaridan foydalaniladi.

Try-catch blogi

Try-catch bloklari orqali xato olishimizni oldini olamiz. **Try-catch** blogi tuzilish shakli shu shaklda:

```
try{
        //….
        xato_berishi_mumkin_bo'lgan_kodlar ;
        //….
}catch(xato_sinfi yis){
        //….
        xato_vaqtida_bajariladigan_kodlar ;
        //….
}
```

Yuqorida ko'rganingiz kabi xato berishi mumkin bo'lgan kod **try** blok ichkarisida yoziladi, xato berilganda ishga tushishi kerak

bo’lgan kodlar esa **catch** blok ichkarisida yoziladi. Keling eng yaxshisi bir misol ko’rib chiqaylik:

```
public class try_catch_misol {
    public static void main(String[] args){
        int x =6;
        int y =0;
        try{
            int a = x/y;
        }catch(Exception exception){
            System.out.println("Berilgan sonni
                    0 soniga bo’lish
                    imkonsiz.");
        }
    }
}
```

Bu misolimiz try-catch uchun berilgan kichik bir misol deya olamiz. Ishlash shakli shu shaklda. birinchi bo’lib dasturimiz try blok ichkarisidagi kodlarni o’qiydi, try blok ichkarisida xato ishlashi mumkin bo’lgan kodlar yozildi degan edik. Try ichida biz berilgan x nomli o’zgaruvchining qiymatini berilgan y nomli o’zgaruvchi qiymatiga bo’lishini va chiqqan natijani a nomli o’zgaruvchiga qiymat sifatida berishini istadik. E’tibor bergan bo’lsangiz x nomli o’zgaruvchining qiymati 6, y nomli o‘zgaruvchining qiymati 0 , ya’ni 6 sonini 0 soniga bo’la olmaymiz dasturimiz ishlash davomida bir xato berishi kerak. Bu xato beradigan kodimizni **try** ichiga yozdik. Keyin **catch** blogiga o’tdik va u yerga try ichidagi kod xato berganda ishga tushishi kerak bo’lgan kodni yozdik.

Kodimizni ishga tushirganimizda shunday bir natija olamiz: *“Berilgan sonni 0 soniga bo’lish imkonsiz.”* Bu shaklda natija olishimizni sababi **try** ichiga yozgan kodimiz xato berdi bu sababli **catch** ichidagi kodimiz ishga tushgan holda bu natijani chiqarib berdi.

Throws kalit so’zi

Ba’zi programmalarda bir xatolarni xato olgan joyda emas, metod chaqirilgan yerda hal etish tanlanadi. Bunday bir vaziyatda bir metodda xato o’rtaga kelayotgan bo’lsa bu metodin xatosi bir oldingi metodin chaqirilish joyiga jo’natiladi. Bu kabi ishlarni bajarishimizda Throws kalit so’zi yordam beradi.

```
public static void throws_misol()
                throws NumberFormatException {
    Scanner in = new Scanner(System.out);
    String kirdi = in.next();
    int kirilgan_son;
    kirilgan_son = Integer.parseInt(kirdi);
    return  kirilgan_son;
        }
```

yuqoridagi metodimizga e’tibor bersangiz, **throws** so’zidan keyin NumberFormatException so’zi yozganimizni ko’rasiz. Bu metodimiz ishga tushirilganda yoki chaqirilganda NumberFormatException turida bir xato berganligi uchun, beradigan bu xatoni metod chaqirilgan joyda hal qilish majburiyati bo’ladi. Bu metodni pastdagi kabi chaqirishimiz mumkin.

```
try{
        number=throws_misol();
}catch(NumberFormatException ex){
        System.out.println("bir son
                        kirishingiz kerak");
}
```

Yuqoridagi misolimizda throws_misol nomli metodni chaqirish davomida, try-catch ichida NumberFormatException xatosini tekshirish majburiyati mavjud. Chunki, throws_misol nomli metodimizni tayyorlashda throws orqali

NumberFormatException nomli xatoni tekshirilishi kerak ekanligini aytib o'tdik.

Throw kalit so'zi

Bu kalit so'zimiz orqali, o'rtaga chiqqan xatoni to'g'ridan to'g'ri ekranga yozdira olamiz. Yuqoridagi kodimizda throw so'zi foydalanilmagan. Chunki throws bilan o'rtaga keladigan xatoni metod chaqirilgan paytda sinab ko'rish kerakligini ko'rsatib o'tganmiz. Bu kodimizni throw kalit so'zini foydalangan holda yozib ko'raylik.

```
try{
        number=throws_misol();
}catch(NumberFormatException ex){
        throw ex ;

}
```

Bu kodimizda throw orqali xatoni topib bir oldingi saviyaga jo'natdik.

Finally kalit so'zi

Agar kodlarimizdan biri xato berganda u kodimizdan so'ngra kelgan kodlar ishga tushmaydi. Bu vaziyatda oldindan bajargan ba'zi kodlarimiz yarim qolib ketadi. Misol uchun bir ma'lumotlar bazasiga bog'landik, ma'lumotlar bazasidan savollar so'rayotganimizda savollarning biri hatto berdi desak. Kodimiz xato bergandan keyingi kodlar ishga tushirilmaydi, bu sababdan bog'langan ma'lumotlar bazamizni to'liq yopa olmaymiz. Bu kabi problemalarni hal qilishda finally kalit so'zidan foydalana olamiz.

Bu kalit so’zimiz **try** kod blogi bilan yoki **try-catch** kod bloklari bilan birgalikda foydalaniladi. Pastki qismda qanday foydalanishga misol berib o’tganmiz.

```
try {
        malumotlar_bazasiga_bog’lan;
        try {
                savol_so’r ;
        }
        finally {
                bog’lantidan_ayril;
        }
}
catch (Exception ex)
{
        bog’lana_olmadik;
}
```

Yuqoridagi misolimizda ich-ichga 2 dona **try** bloklari foydalanilgandir. Tashqi **try** blokda bazaga bog‘lanishga harakat qilinadi. Agar bog’lana olmasa, tashqi tarafda joylashgan **catch** blok ichidagi kodni ichga tushiriladi. Ya’ni, birinchi try ichidagi kodlar tashlab ketiladi.

Agar ma’lumotlar bazasiga bog’lana olsa, tashqi **try** ichidagi kodlarni birma-bir ishga tushirishni boshlaydi. Bu sababdan ichkaridagi **try** kodlari ham ishga tushiriladi. Ichki qismdagi try blogimiz orqali bazadan savol so’rash paytida har qanday bir xato olsak, ichki **try** blogidan chiqiladi va **finally** blok ichidagi kodlar ishga tushiriladi.

Finally blok ichkarisida berilgan kodlar har qanday vaziyatda ishga tushiriladi. Misol uchun yuqoridagi kodimizning ichi qismida joylashgan ikkinchi **try** blogimiz xato bermaganda ham finally blok ichkarisidagi kodlar **try** ichidagi kodlar tugagandan keyin ishga tushar edi.

6.15 Paketlar (packages)

Ko‘plab sinflardan tashkil topgan Java programmalarida o’xshash sinflarni gruppalash yaxshi bir usuldir. Bu gruppalashlarni amalga oshirishimiz uchun bizga paketlar(packages) yordam beradi. Misol uchun aytaylik, ma’lumotlar bazasiga bog‘lanishga yordam beradigan barcha metodlardan tashkil topgan sinflarni boshqa sinflardan ayirishimiz kerak, buni bajarish uchun yangi bir paket ochib ma’lumotlar bazasiga oid sinflarini bu paketga qo’yish kifoya. Millionlab sinflardan tashkil topgan fayllarimiz bu shaklda nazorat qilinadi.

Paketlarning boshqa yana bir foydasi, kompilyatorlar uchun sinfning bir boshiga ifoda qilmasligi va sinf joylashgan paket ham hisobga olinishidir. Bir paket ichida bir xil ismli ikki sinf bo’lishi mumkin emas. Boshqa-boshqa paketlarda joylashgan, ikki bir xil ismli sinflar bo’lishi mumkin. Har bir Java programmalari paketlardan tashkil topishi kerak degan bir

qoida yo’q ammo, bir dona sinfdan tashkil topgan bir dastur bo’lsa ham paket ichida joylashgani yaxshi.

Paket degani azizlar bu fayl degani. Java dasturlash tilida paketlar qavatma-qavat shaklda tayyorlanadi. Har bir paket qavatidan qavatiga o’tish uchun (.) nuqta foydalaniladi. Misol uchun maktab nomli ichidagi sinf nomli faylga erishish shu shaklda: maktab.sinf

Boshqa paketdagi sinflardan foydalanish

Agar bir sinfda, boshqa bir paketda joylashgan bir sinfdan foydalanishni istasa, sinf nomini yozishdan oldin uni **import** kalit so’zi yordamida yuklab olishga majbur. Pastdagi misolimizda, **Misol1** nomli sinfimiz **Misol3** nomli paket ichida joylashgan **Misol2** nomli faylga ehtiyoji bor. Shuning uchun uni yuklab olmoqda.

```
package sinflar;

import Misol3.Misol2;

public class Misol1 {

        public static void main(String[] args){

                Misol2 nom_1 = new Misol2();

                Misol2 nom_2 = new Misol2();

                ……..

        }

}
```

Bir sinfni import kalit so’zini foydalanmagan holda chaqirish

Agar yuklab olishda import kalit so’zidan foydalanishni istamasak, faylni manzilini obyekt tayyorlanayotganda to’liq manzilni berib o’tishimiz kerak. Buni har obyekt tayyorlashda yozishga majbur bo’lamiz. Yuqoridagi kodimizni importsiz yozib ko’raylik.

```
package sinflar;

public class Misol1 {

        public static void main(String[] args){

        Misol3.Misol2 nom = new Misol3.Misol2();

        Misol3.Misol2 nom2 = new Misol3.Misol2();

        …….

        }

}
```

7 Obyektga yo'naltirilgan Javaning asoslari

7.1 Kirish

Oltinchi bo'limda sinf va obyektni o'rganib chiqdik azizlar. Bu bo'limda sizlar bilan birgalikda, obyektga yo'naltirilgan dasturlash tillaridagi asosiy qismlarni ko'rib chiqamiz. Obyektga yo'naltirilgan dasturlash tillaridagi asosiy qismlar deb, shularni ko'rsata olamiz.

- Inheritance (Meros olish)
- Polymorphism (Polimorfizm)
- Interfaces

Shuningdek bu bo'limda, obyektga yo'nalgan programmalardagi (extends, instanceof, super, interface, override vh) kalit so'zlarini va ma'nolarini o'rganib chiqamiz.

Bu bo'lim ichkarisida foydalanadigan barcha misollarimizni berilgan GitHub manzilidan yuklay olasiz.

7.2 Inheritance (Meros olish)

Inheritance, obyektga yo'naltirilgan dasturlash tillaridagi eng muhim qismlardan hisoblanadi. Inheritance, mavjud bo'lgan bir sinfning barcha xususiyatlaridan foydalangan holda, yangi bir sinf hosil qilish uchun foydalaniladi. Dasturlash olamida, asosiy muammolardan biri, haqiqiy hayotda qarshimizga chiqqan proyektlarda foydalaniladigan ob'ektlarni modellashtirish hisoblanadi.

Misol uchun aytaylik, bir bankaning programmasida foydalaniladigan kishi ma'lumotlari bilan bir sayt foydalanuvchi ma'lumotlari bir xil emas. Ammo bu kishining ismi, familiyasi, tug'ilgan kuni sayt ichida ham programma ichida ham bir xil, faqat programmadan programmaga foydalanuvchi ma'lumotlari farq qilishi mumkin. Bank programmasi ichida foydalanuvchining hisob raqamlari, sayt ichkarisida esa kasbi, oilaviy holati kabi farqli ma'lumotlar bo'lishi mumkin. Programma va sayt ichkarisida foydalanuvchiga oid bir xil ma'lumotlar ism familiya kabi ma'lumotlar mavjud. Bu bir xil ma'lumotlarni bir sinf ichkarisida yozgan holda, bu sinfni Inheritance (meros olgan) holda farqli sinflar ichida chaqirib foydalana olamiz. Ya'ni inheritance, barcha joyda bir xil bo'lgan ma'lumotlarni yoki o'zgaruvchilarni bir sinf ichkarisida berib, bu sinfni istalgan joyimizda chaqirib foydalana olishimizga yordamchi bo'ladi. Dasturchilar bir kodni qayta-qayta yozish o'rniga, inheritancedan foydalanish orqali muammoni hal qila oladi.

Yuqorida tushuntirib o'tganlarimiz tushunarli bo'lishi uchun keling bir misol ko'rib chiqamiz. Bu misolimizda **Foydalanuvchilar** nomida bir sinf beramiz. Bu misolimizni bir

onlayn do'konning ma'lumotlar saqlovchi programmasi sifatida tushuning. Onlayn do'konda sotuvchilar va xaridorlarning ma'lumotlari saqlanib turadi har doim. Bu yerda xaridor va sotuvchining bir xil turda bo'lgan ma'lumotlari mavjud ya'ni har ikkalasining ham bir ismi, bir familiyasi, bir kasbi mavjud. Bir tarafdan esa farqli turdagi ma'lumotlari ham mavjud aytaylik, sotuvchining do'konining nomi yoki foydalanuvchining uy manzili kabi. Bularni bir oz qiyin bo'layotgan bo'lishi mumkin, pastdagi misolni yaxshilab ko'rib chiqqanimizdan keyin tushunmaslik kabi bir muammo qolmaydi.

Bu misolimizdagi **Foydalanuvchilar** sinf ichkarisida sotuvchida ham xaridorda ham berilishi kerak bo'lgan ma'lumotlarni beramiz. Va extends kalit so'zidan foydalangan holda bu sinfimizni, inheritance qila olamiz.

Foydalanuvchilar.java

```
package Sinflar;

public class Foydalanuvchilar {

    String ism;
    String familiya;
    int tugulgan_sana;
    String jinsi;

    public Foydalanuvchilar(){
        ism="";
        familiya="";
        tugulgan_sana=0;
        jinsi="";
    }
    public Foydalanuvchilar(String pism ,
                            String pfamiliya,
                            int ptugulgan_sana,
```

```
                            String pjinsi) {
        this.ism = pism;
        //this.setIsm(pism);

        this.familiya=pfamiliya;
        //this.setFamiliya(pfamiliya);

        this.tugulgan_sana=ptugulgan_sana;
        //this.setTugulgan_sana
                        (ptugulgan_sana);

        this.jinsi=pjinsi;
        //this.setJinsi(pjinsi);
    }

    public Foydalanuvchilar
                        (Foydalanuvchilar k) {

        /* Pastki 3 qatordagi kod bir xil */
        this.setIsm(k.getIsm());
        //this.setIsm(k.ism);
        //this.ism=k.ism;

        /* Pastki 3 qatordagi kod bir xil */
        this.setFamiliya(k.getFamiliya());
        //this.setFamiliya(k.familiya);
        //this.familiya=k.familiya;

        /* Pastki 3 qatordagi kod bir xil */
        this.setTugulgan_sana
                        (k.getTugulgan_sana());
        //this.setTugulgan_sana
                        (k.tugulgan_sana);
        //this.tugulgan_sana=k.tugulgan_sana;

        /* Pastki 3 qatordagi kod bir xil */
        this.setJinsi(k.getJinsi());
        //this.setJinsi(k.jinsi);
        //this.tugulgan_sana=k.tugulgan_sana;
    }
```

```
    public String getIsm() {
        return ism;
    }

    public void setIsm(String ism) {
        this.ism = ism;
    }

    public String getFamiliya() {
        return familiya;
    }

    public void setFamiliya(String familiya) {
        this.familiya = familiya;
    }

    public int getTugulgan_sana() {
        return tugulgan_sana;
    }

    public void setTugulgan_sana
                        (int tugulgan_sana) {

        this.tugulgan_sana = tugulgan_sana;
    }

    public String getJinsi() {
        return jinsi;
    }

    public void setJinsi(String jinsi) {
        this.jinsi = jinsi;
    }

}
```

Sotuvchi.java

```
package Sinflar;

public class Sotuvchi extends Foydalanuvchilar
{
    String dokon_nomi;
    String savdo_turi;

    public Sotuvchi() {
        /* ust sinfda paremetr olmaydigan
        quruvchilarni chaqir */
        super();
        this.dokon_nomi="";
        this.savdo_turi="";

    }

    public Sotuvchi(Foydalanuvchilar k,
                    String psavdo_turi,
                    String pdokon_nomi ) {
        /* ust sinfdagi Foydlanuvchi turidagi
        parametr oladigan quruvchilarni
        chaqir*/
        super(k);
        this.savdo_turi=psavdo_turi;
        this.dokon_nomi=pdokon_nomi;

    }
    public Sotuvchi(Sotuvchi s) {
        super(s.getIsm(),s.getFamiliya(),
              s.getTugulgan_sana(),
              s.getJinsi());
        /*Yuqoridagi bir qator kodni
        bajaradigan ishini pastdagi 8
        qatordagi kod ham bajara oladi*/
        //super.ism=s.getIsm();
        //this.ism=s.getIsm();
        //super.familiya=s.getFamiliya();
```

```
        //this.familiya=s.getFamiliya();
       //super.tugulgan_sana=
                        s.getTugulgan_sana();
        //this.tugulgan_sana=
                        s.getTugulgan_sana();
        //super.jinsi=s.getJinsi();
        //this.jinsi=s.getJinsi();

        this.dokon_nomi=s.getDokon_nomi();
        this.savdo_turi=s.getSavdo_turi();
    }
    public Sotuvchi(String pism,
                    String pfamiliya,
                    int ptugulgan_sana,
                    String pjinsi,
                    String pdokon_nomi,
                    String psavdo_turi) {
        /* Ust sinfdagi 4 parametr oladigan
        quruvchini chaqir*/
        super(pism,pfamiliya,
                        ptugulgan_sana,pjinsi);
        this.dokon_nomi=pdokon_nomi;
        /* Pastki qatordagi kod, yuqori
        qatordagi kod bilan bir xil vazifa
        bajaradi*/
        //this.setDokon_nomi(pdokon_nomi);

        this.savdo_turi=psavdo_turi;
        /* Pastki qatordagi kod, yuqori
        qatordagi kod bilan bir xil vazifa
        bajaradi*/
        //this.setSavdo_turi(psavdo_turi);
    }

    public String getDokon_nomi() {
        return dokon_nomi;
    }

    public void setDokon_nomi
                        (String dokon_nomi) {
```

```
        this.dokon_nomi = dokon_nomi;
    }

    public String getSavdo_turi() {
        return savdo_turi;
    }

    public void setSavdo_turi
                        (String savdo_turi) {
        this.savdo_turi = savdo_turi;
    }
}
```

Xaridor.java

```
package Sinflar;

public class Xaridor extends Foydalanuvchilar{
    String manzil;
    int hisobraqami;

    public Xaridor() {
        super();
        this.manzil="";
        this.hisobraqami=0;

    }
    public Xaridor(Foydalanuvchilar k,
                   String pmanzil,
                   int phisobraqami) {
        super(k);
        this.manzil=pmanzil;
        this.hisobraqami=phisobraqami;
    }
    public Xaridor(Xaridor x) {
        super(x.getIsm(),x.getFamiliya(),
                        x.getTugulgan_sana(),
                        x.getJinsi());
```

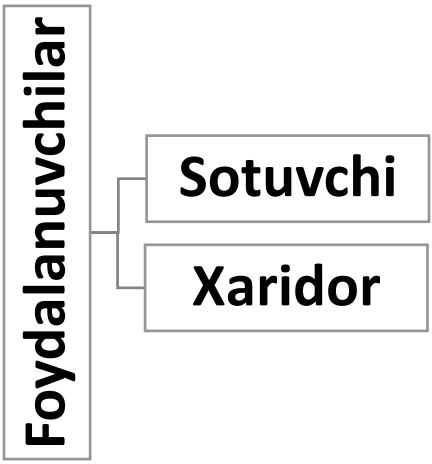
```
        this.manzil=x.getManzil();
        this.hisobraqami=x.getHisobraqami();

    }
    public Xaridor(String pism,
                   String pfamiliya,
                   int ptugulgan_sana,
                   String pjinsi,
                   String pmanzil,
                   int phisobraqami) {

        super(pism,pfamiliya,
                       ptugulgan_sana,pjinsi);

        manzil=pmanzil;
        hisobraqami=phisobraqami;
    }

    public String getManzil() {
        return manzil;
    }
    public void setManzil(String manzil) {
        this.manzil = manzil;
    }
    public int getHisobraqami() {
        return hisobraqami;
    }
    public void setHisobraqami
                        (int hisobraqami){
        this.hisobraqami = hisobraqami;
    }
}
```

Yuqoridagi 3 sinfni ko'rib chiqadigan bo'lsak, extends kalit so'zi orqali Sotuvchi va Xaridor sinflari Foydalanuvchilar sinfidan foydalanilgan holda yangidan hosil qilindi. Bu sababdan Sotuvchi va Xaridor sinflaridan Foydalanuvchilar sinfida berilgan barcha o'zgaruvchilarga erisha olamiz va bu

o’zgaruvchilarni Sotuvchi va Xaridor sinflariga tegishli bir normal o’zgaruvchi sifatida foydalana olamiz.

Super class(ust qism) va sub class (past sinf)

Hosil qilinadigan hosil qilingan sinflarni bir-biridan ayirishimiz uchun ust sinf va past sinf nomlaridan foydalanamiz. Yuqorida berib o’tgan misolimizda, **Sotuvchi** nomli sinfimiz **Foydalanuvchilar** sinfidan (**extends** kalit so’zi) yordamida hosil qilinganligi uchun **Foydalanuvchilar** nomli sinf **Sotuvchi** nomli sinfning ust sinfi hisoblanadi. Boshqacha qilib aytadigan bo’lsak, **Sotuvchi** nomli sinf **Foydalanuvchi** nomli sinfning past sinfidir.



Super() metodi bilan ust sinfni chaqirish

Bu metod maxsus bir metod siafatida bilinadi. **super ()** metodi yordami bilan har qanday bir past sinf ichkarisidan, ust sinfda joylashgan quruvchi metodlarni chaqira olamiz. Ust sinfda

birdan ko’p quruvchi metodlar (overloading) mavjud bo’lsa, bergan paremetrlarimizga ko’ra, dasturimiz avtomatik shaklda mos bo’lgan quruvchi metodni chaqiradi. Bu metoddan foydalanishga misol berib o’tadigan bo’lsak.

```
public Sotuvchi(Sotuvchi s) {
        super(s.getIsm(),s.getFamiliya(),
                s.getTugulgan_sana(),
                s.getJinsi());
        /*Yuqoridagi bir qator kodni
       bajaradigan ishini pastdagi 8
       qatordagi kod ham bajara oladi*/
        //super.ism=s.getIsm();
        //this.ism=s.getIsm();
        //super.familiya=s.getFamiliya();
        //this.familiya=s.getFamiliya();
        //super.tugulgan_sana=
                        s.getTugulgan_sana();
        //this.tugulgan_sana=
                        s.getTugulgan_sana();
        //super.jinsi=s.getJinsi();
        //this.jinsi=s.getJinsi();

        this.dokon_nomi=s.getDokon_nomi();
        this.savdo_turi=s.getSavdo_turi();
    }
```

Agar bu quruvchi metodimiz past sinfing quruvchi metodi ichkarisida chaqirilayotgan bo’lsa, muqlaqo birinchi qatorda chaqirilishi kerak. Bu sababdan yuqoridagi misolimizda:

```
super(s.getIsm(),s.getFamiliya(),
        s.getTugulgan_sana(), s.getJinsi());
```

bu kodimiz eng ust qatorda chaqirildi. Sotuvchi sinfimizning ust sinfi (superclass) Foydalanuvchilar nomli sinf bo’lganligi uchun dasturimiz, Foydalanuvchilar nomli sinfmiz ichkarisidan bergan paremetrlarimizga ko’ra mos bir quruvchi metodini chaqiradi. Foydalanuvchilar nomli sinf ichkarisida to’rt parametr oladigan

bir dona metod mavjud, bu sababdan dasturimiz quyidagi quruvchi metodni chaqiradi.

```
public Foydalanuvchilar(String pism ,
                        String pfamiliya,
                        int ptugulgan_sana,
                        String pjinsi) {
        this.ism = pism;
        //this.setIsm(pism);
        this.familiya=pfamiliya;
        //this.setFamiliya(pfamiliya);
        this.tugulgan_sana=ptugulgan_sana;
        //this.setTugulgan_sana
                        (ptugulgan_sana);
        this.jinsi=pjinsi;
        //this.setJinsi(pjinsi);
      }
```

Bu metodimiz shuningdek, bir ust sinfning o’zgaruvchilariga va metodlariga erisha olishimizga yordamchi bo’ladi.

```
public Sotuvchi(Sotuvchi s) {

        super(s.getIsm(),s.getFamiliya(),
                s.getTugulgan_sana(),
                s.getJinsi());
        /*Yuqoridagi bir qator kodni
        bajaradigan ishini pastdagi 8
        qatordagi kod ham bajara oladi*/
        //super.ism=s.getIsm();
        //this.ism=s.getIsm();
        //super.familiya=s.getFamiliya();
        //this.familiya=s.getFamiliya();
        //super.tugulgan_sana=
                        s.getTugulgan_sana();
        //this.tugulgan_sana=
                        s.getTugulgan_sana();
        //super.jinsi=s.getJinsi();
        //this.jinsi=s.getJinsi();
```

```
        this.dokon_nomi=s.getDokon_nomi();
        this.savdo_turi=s.getSavdo_turi();
    }
```

Yuqoridagi metodda super kalit so'zi orqali, bir ust sinf hisoblangan Foydlanuvchilar sinfida berilgan ism, familiya, tugulgan_sana, jinsi nomli o'zgaruvchilariga ersha oldik. Ust sinf metodlariga ham xuddi shu shaklda erisha olamiz.

This kalit so'zi

Bu kalit so'zimiz super kalit so'zining ikkinchi foydalanish shakliga o'xshab ketadi, faqat oralarida bir farq mavjud. Super kalit so'zimiz ust sinfni ko'rsatib turardi, this kalit so'zimiz esa joylashgan sinfini ko'rsatib turadi. Inheritance yordami bilan bir sinf, bir ust sinfning barcha xususiyatlaridan foydalana olganligi uchun yuqoridagi berilgan misolda, super kalit so'zining o'rniga this kalit so'zini foydalana olamiz. Faqat inheritance foydalanilmagan vaziyatlarda, this kalit so'zimiz bir sinfni ko'rsatish uchun yoki tanitish uchun foydalanilishi mumkin. Super kalit so'zi faqatgina ust sinfga erisha olish uchun foydalaniladi. Bir misol berib o'tadigan bo'lsak:

```
public Sotuvchi(Sotuvchi s) {

        super(s.getIsm(),s.getFamiliya(),
              s.getTugulgan_sana(),
              s.getJinsi());
        /*Yuqoridagi bir qator kodni
        bajaradigan ishini pastdagi 8
        qatordagi kod ham bajara oladi*/
        //super.ism=s.getIsm();
        //this.ism=s.getIsm();
        //super.familiya=s.getFamiliya();
        //this.familiya=s.getFamiliya();
```

```
        //super.tugulgan_sana=
                        s.getTugulgan_sana();
        //this.tugulgan_sana=
                        s.getTugulgan_sana();
        //super.jinsi=s.getJinsi();
        //this.jinsi=s.getJinsi();

        this.dokon_nomi=s.getDokon_nomi();
        this.savdo_turi=s.getSavdo_turi();
        }
```

Yuqoridagi metodda berilgan **Sotuvchi** turidagi parametrlarining qiymat o'zgaruvchilari, chaqirilgan metodning qiymat o'zgaruvchilariga qiymat sifatida berildi. Chaqirilgan metodning obyekt qiymat o'zgaruvchilariga to'g'ridan erisha olish uchun this kalit so'zi foydalanildi. Bu kalit so'zi yordami bilan faqatgina o'zgaruvchilarga emas sinf ichidagi barcha metodlarga ham erisha olamiz.

Inheritance uchun misol

Pastda berib o'tgan misolimizda, yuqorida tayyorlangan **Foydalanuvchi**, **Sotuvchi**, **Xaridor** nomli sinflarimiz foydalanildi. Misolimizda birinchi bo'lib, Foydalanuvchilar turida bo'lgan uch dona obyekt tayyorlandi, keyin esa bu obyektlarning oxiridagi ikkisi foydalanilgan holda Sotuvchi va Xaridor turidagi obyektlar tayyorlandi. So'ngra **malumotlarni_yoz** metodi bilan **instanceof** kalit so'zi foydalanilgan holda, parametr sifatida berilgan obyektlarning turi aniqlandi va obyektning turiga mos bo'lgan holda ekranga ma'lumotlar chiqarildi.

```
Inheritance_misol.java
```

```
package Obyekt;


import Sinflar.Foydalanuvchilar;
import Sinflar.Sotuvchi;
import Sinflar.Xaridor;


public class Inheritance_misol {
    public static void main(String[] args) {
        Foydalanuvchilar foydalanuvchi_1 =
                        new Foydalanuvchilar();
        Foydalanuvchilar foydalanuvchi_2 =
                        new Foydalanuvchilar();
        Foydalanuvchilar foydalanuvchi_3 =
                        new Foydalanuvchilar();

        // 1-foydalanuvchi
        foydalanuvchi_1.setIsm("Davronbek");
        foydalanuvchi_1.setFamiliya
                            ("Abdurazzokov");
        foydalanuvchi_1.setTugulgan_sana
                            (2003);
        foydalanuvchi_1.setJinsi("Erkak");

        // 2-foydalanuvchi
        foydalanuvchi_2.setIsm("Sherzodbek");
        foydalanuvchi_2.setFamiliya
                              ("Sultonov");
        foydalanuvchi_2.setTugulgan_sana
                              (2003);
        foydalanuvchi_2.setJinsi("Erkak");

        // 3-foydalanuvchi
        foydalanuvchi_3.setIsm("Guli");
        foydalanuvchi_3.setFamiliya
                              ("Abdullayeva");
        foydalanuvchi_3.setTugulgan_sana
                              (2005);
```

```
        foydalanuvchi_3.setJinsi("Ayol");

        /* 2- kishidan Sotuvchi obyekti
        tayyorlaymiz*/
        Sotuvchi sotuvchi_1 =
                new Sotuvchi(foydalanuvchi_2,
                        "Oziq-ovqat",
                        "Do'kon nomi-Oila");

        /* 3-kishidan ham xaridor obyekti
        tayyorlaymiz*/
        Xaridor xaridor_1 =
                new Xaridor(foydalanuvchi_3,
                "Manzili-Farg'ona", 12345);

        //tur o'zgartirishlar
        /*past sinfdan bir obyekt ust
        sinfdagiga almashtirilishi mumkin*/

        Foydalanuvchilar foydalanuvchi_4=
                (Foydalanuvchilar)sotuvchi_1;

        Foydalanuvchilar foydalanuvchi_5=
                (Foydalanuvchilar)xaridor_1;

        //tur o'zgartirishlar
        /*ust sinfdan bir obyekt past
        sinfdagiga almashtirila olinmaydi bu
        sababdan pastki qatordagi kod
        ishlamaydi*/

        /* Sotuvchi sotuvchi =
        (Sotuvchi)foydalanuvchi_1;*/

        // Ma'lumotlarni ekranga chiqarish

        malumotlarni_yoz(foydalanuvchi_1);
        malumotlarni_yoz(foydalanuvchi_2);
        malumotlarni_yoz(foydalanuvchi_3);
        malumotlarni_yoz(foydalanuvchi_4);
```

```
        malumotlarni_yoz(sotuvchi_1);
        malumotlarni_yoz(xaridor_1);

        System.out.println("—instanceof
                qarshilashtirma javoblari--");
        instanceoftest(foydalanuvchi_1,
                        sotuvchi_1, xaridor_1);
        System.out.println("----------------");

    }
    public static void malumotlarni_yoz
                                (Object s) {
        System.out.println("----------------");

        if(s instanceof Sotuvchi) {
            /* Agar objekt Sotuvchi turida
               bo'lsa, Sotuvchining
               ma'lumotlarini chiqar.*/
            Sotuvchi otuvchi = (Sotuvchi)s;
            System.out.println
                        (otuvchi.getIsm());
            System.out.println(otuvchi
                        .getFamiliya());
            System.out.println(otuvchi
                        .getTugulgan_sana());
            System.out.println(otuvchi
                        .getJinsi());
            System.out.println(otuvchi
                        .getDokon_nomi());
            System.out.println(otuvchi
                        .getSavdo_turi());

        }
        else if(s instanceof Xaridor) {
            /* Agar objekt Xaridor turida
               bo'lsa, Xaridorning
               ma'lumotlarini chiqar.*/
            Xaridor otuvchi= (Xaridor)s;
```

```
            System.out.println(otuvchi
                        .getIsm());
            System.out.println(otuvchi
                        .getFamiliya());
            System.out.println(otuvchi
                        .getTugulgan_sana());
            System.out.println(otuvchi
                        .getJinsi());
            System.out.println(otuvchi
                        .getHisobraqami());
            System.out.println(otuvchi
                        .getManzil());
        }
        else if(s instanceof Foydalanuvchilar)
{
            /* Agar objekt Foydalanuvchilar
               turida bo'lsa,
               Foydalanuvchilarning
               ma'lumotlarini chiqar.*/
            Foydalanuvchilar otuvchi=
                        (Foydalanuvchilar)s;
            System.out.println(otuvchi
                        .getIsm());
            System.out.println(otuvchi
                        .getFamiliya());
            System.out.println(otuvchi
                        .getTugulgan_sana());
            System.out.println(otuvchi
                        .getJinsi());

        }
    }

    public static void instanceoftest
                        (Foydalanuvchilar k,
                        Sotuvchi s, Xaridor x)
{
        if (k instanceof Foydalanuvchilar) {
```

```
            System.out.println("k obyekti
                        Foydalanuvchilar sinfi
                        uchun bir misol");
        }
        else {
            System.out.println("k obyekti
                        Foydalanuvchilar sinfi
                        uchun bir misol emas");
        }
        if (s instanceof Foydalanuvchilar) {
            System.out.println("s obyekti
                        Foydalanuvchilar sinfi
                        uchun bir misol");
        }
        else {
            System.out.println("s obyekti
                        Foydalanuvchilar sinfi
                        uchun bir misol emas");
        }
        if (x instanceof Foydalanuvchilar) {
            System.out.println("x obyekti
                        Foydalanuvchilar sinfi
                        uchun bir misol");
        }
        else {
            System.out.println("x obyekti
                        Foydalanuvchilar sinfi
                        uchun bir misol emas");
        }
        if (k instanceof Sotuvchi) {
            System.out.println("k obyekti
                        Sotuvchi sinfi uchun
                        bir misol");
        }
        else {
            System.out.println("k obyekti
                        Sotuvchi sinfi uchun
                        bir misol emas");
        }
```

```
        if (k instanceof Xaridor) {
            System.out.println("k obyekti
                        Xaridor sinfi uchun
                        bir misol");
        }
        else {
            System.out.println("k obyekti
                        Xaridor sinfi uchun
                        bir misol emas");

        }

    }

}
```

Yuqorida berib o’tgan misolimizni ishga tushirganimizda quyidagicha bir javob olamiz:

```
---------------

Davronbek

Abdurazzokov

2003

Erkak

---------------

Sherzodbek

Sultonov

2003

Erkak
```

```
---------------

Guli

Abdullayeva

2005

Ayol

---------------

Sherzodbek

Sultonov

2003

Erkak

Do'kon nomi Oila

Oziq-ovqat

---------------

Sherzodbek

Sultonov

2003

Erkak

Do'kon nomi Oila

Oziq-ovqat
```

```
---------------

Guli

Abdullayeva

2005

Ayol

12345

Manzili-Farg’ona

-- instanceof qarshilashtirma javoblari--

k obyekti Foydalanuvchilar sinfi uchun bir misol

s obyekti Foydalanuvchilar sinfi uchun bir misol

x obyekti Foydalanuvchilar sinfi uchun bir misol

k obyekti Sotuvchi sinfi uchun bir misol emas

k obyekti Xaridor sinfi uchun bir misol emas

---------------
```

Yuqorida berib o’tgan misolimizga e’tibor bergan bo’lsangiz, ma’lumotlarni_yoz metodida ism, familiya, tugulgan_sana, jinsi kabi a’zo o’zgaruvchilari faqatgina **Foydalanuvchilar** sinfida berilgan bo’lishiga qaramasdan, **Sotuvchi** va **Xaridor** sinfi ichkarisida ham mavjud kabi ko’rinyapti. Buning sababi

azizlar, **Xaridor** va **Sotuvchi** nomli sinflar, **Foydalanuvchilar** sinfini meros olgan holda tayyorlanganligidir.

Bu bo'lim ichkarisida foydalangan **Sotuvchi.java, Foydalanuvchilar.java, Xaridor.java, Inheritance_misol.java** nomli barcha kod fayllarimizning GitHub manzilini quyidagicha:

Sotuvchi.java:
https://github.com/21040001/Java_yordamchi_kitob_kodlarimiz/blob/main/obyekt_java/Sinflar/Sotuvchi.java

Foydalanuvchilar.java:
https://github.com/21040001/Java_yordamchi_kitob_kodlarimiz/blob/main/obyekt_java/Sinflar/Foydalanuvchilar.java

Xaridor.java:
https://github.com/21040001/Java_yordamchi_kitob_kodlarimiz/blob/main/obyekt_java/Sinflar/Xaridor.java

Inheritance_misol.java:
https://github.com/21040001/Java_yordamchi_kitob_kodlarimiz/blob/main/obyekt_java/Obyekt/Inheritance_misol.java

Object sinfi

Java dasturlash tilida ko'p foydalaniladigan sinflardan biri bu Object (Ob'ekt) sinfidir. Javadagi barcha sinflar xatoki biz tayyorlagan sinflar ham Objekt sinfiga bog'liqdir. Boshqa sinflardan hosil qilingan obyektlarni Objekt sinfiga aylantirish uchun har qanday bir **tur almashtirish** (***type casting***) ifodasiga kerak yo'q. To'g'ridan quyidagicha yozishimiz mumkin:

```
Foydalanuvchilar foydalnuvchi_1 = new Foydalanuvchilar (
                                        “Oybek”, 2003, “Erkak”);

Object s = foydalnuvchi_1;
```

Object turiga ega bo’lgan bir obyektni, boshqa bir sinfga aylantirishni istasak, u obyektni haqiqatdan ham u sinfdan tayyorlanganligini tekshirib ko’rishimiz kerak bo’ladi. bu ishni bajarish uchun instanceof kalit so’zidan foydalanamiz.

Instanceof kalit so’zi

Bu kalit so’zimiz odatda bir object turidagi bir obyektning qaysi bir sinfga tegishli ekanligi topishda foydalaniladi. Misol uchun pasta berib o’tilgan misolda, parametr a’zosi sifatida kelgan s nomli Object turidagi obyektning Sotuvchi turida mi, yo’qmi tekshirilmoqda. Agar obyektimiz Sotuvchi turida bo’lsa, metodimiz Sotuvchi sinfidagi barcha o’zgaruvchilarga erishadi va barchasining qiymatlarni ekranga chiqarib beradi.

```
if(s instanceof Sotuvchi) {
    /* Agar objekt Sotuvchi turida bo'lsa,
       Sotuvchining ma'lumotlarini chiqar.*/
    Sotuvchi otuvchi = (Sotuvchi)s;
    System.out.println(otuvchi.getIsm());
    System.out.println(otuvchi.getFamiliya());
    System.out.println(otuvchi
                       .getTugulgan_sana());
    System.out.println(otuvchi.getJinsi());
    System.out.println(otuvchi
                       .getDokon_nomi());
    System.out.println(otuvchi
                       .getSavdo_turi());

}
```

Bu kalit so’zimizning boshqa bir vazifasi inheritance (meros olinganlarni) o’rganib chiqishdir. Ya’ni, quyidagicha:

```
public static void instanceoftest
                        (Foydalanuvchilar k,
                         Sotuvchi s,
                         Xaridor x) {
    if (k instanceof Foydalanuvchilar) {
        System.out.println("k obyekti
                        Foydalanuvchilar sinfi
                        uchun bir misol");
        }
    else {
        System.out.println("k obyekti
                        Foydalanuvchilar sinfi
                        uchun bir misol emas");
        }
    if (s instanceof Foydalanuvchilar) {
        System.out.println("s obyekti
                        Foydalanuvchilar sinfi
                        uchun bir misol");
        }
    else {
        System.out.println("s obyekti
                        Foydalanuvchilar sinfi
                        uchun bir misol emas");
        }
    if (x instanceof Foydalanuvchilar) {
        System.out.println("x obyekti
                        Foydalanuvchilar sinfi
                        uchun bir misol");
        }
    else {
        System.out.println("x obyekti
                        Foydalanuvchilar sinfi
                        uchun bir misol emas");
        }
    if (k instanceof Sotuvchi) {
        System.out.println("k obyekti Sotuvchi
```

```
                    sinfi uchun bir
                misol");
        }
    else {
        System.out.println("k obyekti Sotuvchi
                    sinfi uchun bir misol
                    emas");
        }
    if (k instanceof Xaridor) {
        System.out.println("k obyekti Xaridor
                    sinfi uchun bir
                    misol");
        }
    else {
        System.out.println("k obyekti Xaridor
                    sinfi uchun bir misol
                    emas");

        }
```

Yuqorida berib o'tilgan misolimizda, Foydalanuvchi turidagi k, Sotuvchi turidagi s, Xaridor turidagi x obyektlarini qaysi sinflardan meros olingan holda tayyorlanganligi tekshirilmoqda. Kerakli parametrlar berilgandan so'ngra, metodimizni ishga tushirganimizda quyidagicha bir javob olamiz:

```
k obyekti Foydalanuvchilar sinfi uchun bir
misol

s obyekti Foydalanuvchilar sinfi uchun bir
misol

x obyekti Foydalanuvchilar sinfi uchun bir
misol

k obyekti Sotuvchi sinfi uchun bir misol emas

k obyekti Xaridor sinfi uchun bir misol emas
```

7.3 Polymorphism

Polymorphism biologiyadan obyektga yo'naltirilgan dasturlash tillariga o'tkazilgan bir tushuncha hisoblanadi. Biologiyada bir turdagi hayvonlar, bir xil turda bo'lishiga qaramasdan o'zlariga xos bo'lgan harakat ko'rsatishlari mumkin. Java dasturlash tilida ham bir sinfdan tayyorlangan hayvonlarni farqli bir shaklda harakat qilishlarini saqlay olamiz. Java dasturlash tilida ikki Polymorphism mavjud deya olamiz: **Overloading** (ortiqcha yuklanish) va **Overriding** (bekor qilish).

Overloading tushunchasini oldingi mavzularda o'rganib chiqqan edik. Bu sababdan hozir birgalikda **Overriding** tushunchasini o'rganib chiqamiz. Keling o'rganib chiqishda **Foydalanuvchilar** va **Sotuvchi** sinflaridan foydalanaylik. Birinchi bo'lib ikki sinfimizga ham pastdagi kabi **malumotlarni_yoz** bir metod qo'shaylik.

```
public class Foydalanuvchilar {

        String ism;

        String familiya;

        public void malumotlarni_yoz() {

                System.out.println(ism);

                 }

                // qolgan kodlarimiz

}
```

Foydalanuvchilar sinfimizda malumotlar_yoz metodi, faqatgina foydalanuvchi ismini ekranga chiqaradi. Agar biz **Sotuvchi** sinfimizda bu metod faqatgina ismni emas familiyasini ham chiqarishini istasak bu metodimizni yangidan berishimiz kerak bo'ladi.

Bu metod ayni vaqtda ust sinfimizda mavjud bo'lganligi uchun, @Override'dan foydalangan holda, ust sinfdagi ya'ni Foydalanuvchilar sinfidagi malumotlarni_yoz metodini bekor qilib, Sotuvchi ichidagi malumotlarni_yoz metodini ishga tushira olamiz. Pastda berilgan misoldagi kabi.

```
public class Sotuvchi extends Foydalanuvchilar {

        String dokon_nomi;

        String savdo_turi;

        @Override

        public void malumotlarni_yoz() {

                System.out.println(ism+" "+familiya);

        }

        //qolgan kodlar

}
```

Bu metodlarimizni foydalanishini Override_misol.java nomli faylimizda ko'rib chiqaylik . Misolimizda Foydalanuvchilar , Sotuvchi, Xaridor turiga mansub bo'lgan 3 obyekt uchun malumotlarni_yoz() metodi chaqirilgan.

```
package Obyekt;

import Sinflar.Foydalanuvchilar;
import Sinflar.Sotuvchi;
import Sinflar.Xaridor;

public class Override_misol {

    public static void main(String[] args) {
        Foydalanuvchilar foydalanuvchi_1 =
                        new Foydalanuvchilar();
        Foydalanuvchilar foydalanuvchi_2 =
                        new Foydalanuvchilar();
        Foydalanuvchilar foydalanuvchi_3 =
                        new Foydalanuvchilar();

        // 1-foydalanuvchi
        foydalanuvchi_1.setIsm("Davronbek");
        foydalanuvchi_1.setFamiliya
                            ("Abdurazzokov");
        foydalanuvchi_1.setTugulgan_sana
                            (2003);
        foydalanuvchi_1.setJinsi("Erkak");

        // 2-foydalanuvchi
        foydalanuvchi_2.setIsm("Sherzodbek");
        foydalanuvchi_2.setFamiliya
                            ("Sultonov");
        foydalanuvchi_2.setTugulgan_sana
                            (2003);
        foydalanuvchi_2.setJinsi("Erkak");

        // 3-foydalanuvchi
        foydalanuvchi_3.setIsm("Guli");
        foydalanuvchi_3.setFamiliya
                            ("Abdullayeva");
        foydalanuvchi_3.setTugulgan_sana
                            (2005);
        foydalanuvchi_3.setJinsi("Ayol");
```

```
        Sotuvchi sotuvchi_1 =
                new Sotuvchi(foydalanuvchi_2,
                    "Oziq-ovqat","Do'kon nomi
                     Oila");
        Xaridor xaridor_1 =
                new Xaridor(foydalanuvchi_3,
                    "Manzili-Fargona", 12345);

        foydalanuvchi_1.malumotlarni_yoz();
        sotuvchi_1.malumotlarni_yoz();
        xaridor_1.malumotlarni_yoz();


    }

}
```

Kodimizni ishga tushirganimizda quyidagicha bir javob olamiz:

```
Davronbek

Sherzodbek Sultonov

Guli
```

Ko’rib turganingiz kabi **Sotuvchi** sinfidan tayyorlangan **sotuvchi_1** obyektiga ko’ra ishga tushirilgan malumatlarni_yoz() metodi boshqalaridan farqli bir shaklda ishga tushgan , ya’ni ham ismini ham familiyani ekranga chiqargan. Chunki **Sotuvchi** nomli sinf ichkarisida, **Foydalanuvchilar** nomli sinf ichkarisida berib o’tilgan malumotlarni_yoz() metodini **Override** qilgan holda yangi metod yozgan edik.

Override (Bekor qilish)ning ham ma’lum bir qoidalari mavjud.

Bularni sanab o’tadigan bo’lsak:

- ***final*** kalit so'zidan foydalanilgan holda ko'rsatilgan sinflardagi metodlar Override qilinmaydi.
- Static sinflarni ham Override qila olmaymiz.
- Bir sinf final sifatida ko'rsatilmagan bo'lsa ham, ichida ***final*** kalit so'zidan foydalanilgan holda bir metod berilgan bo'lsa, bu metod ham Override qilinmaydi.
- ***private*** sifatida berilgan metodlar ham Override qilinmaydi.

Agar ust sinfda joylashgan biron-bir metodni Override qilishni istasangiz yuqoridagi berib o'tilganlarga e'tibor berishni unutmang.

7.4 Abstract sinflar (Mavhum sinflar)

Programmalarda ba'zan ust sinflarni to'g'ridan foydalanilishni istamaymiz. Ba'zan programmalarda, faqatgina ust sinfdan hosil qilingan sinflardan tayyorlangan obyektlarning foydalanishini istaymiz. Bu kabi vaziyatlarda bizga abstract sinflar yordam beradi. Mavhum sinflarni berishda abstract kalit so'zidan foydalanamiz. Misol uchun, Foydalanuvchilar sinfini mavhum bir sinf sifatida beraylik.

```
public abstract class Foydalanuvchilar {

        String ism;

        String familiya;

        public abstrack void malumotlarni_yoz() {

        //....

}
```

Bu kabi abstract sinfdan to'g'ridan bir obyekt tayyorlay olmaymiz:

```
Foydalanuvchilar foydalanuvchi = new Foydalanuvchilar();

/* bu xato chunki bir abstract sinfdan bu shaklda oybekt
tayyorlay olmaymiz*/
```

Abstract sinflardan obyekt tayyorlash uchun, avvalo bu sinfni boshqa bir sinfga inheritance qilishimiz kerak. Undan so'ngra inheritance qilgan sinfimizdan obyektlar tayyorlay olamiz.

7.5 Abstract metodlar(Mavhum metodlar)

Mavhum metodlar bu – ichida har qanday kod berilmagan metodlar deyishimiz mumkin. Boshqa so'z bilan aytganda metod ichkarisi bosh bo'lgan metodlar mavhum metodlar hisoblanadi. Bir metodni mavhum (abstract) qilishni istaganimizda, metod turini yozishdan oldin abstract kalit so'zini yozish va oxirida (); xarakterlarni yozishimiz kifoya. Bir misol beradigan bo'lsak:

```
public abstract void malumot_yoz( );
```

Bu metodlar nima uchun foydalaniladi? Savoliga javob berib o'taylik. Agar bir sinfda abstract bir metod mavjud bo'lsa, bu sinfdan foydalanilgan holda tayyorlanadigan barcha sinflar ichida bu metod ichkarisiga metod kodlarini yozishga majbur. Mavhum sinflar va metodlar orasidagi bog'liqliklarni aytib o'tadigan bo'lsak:

Agar bir sinfda abstract metod mavjud bo'lsa, bu sinf ham abstract bo'lishga majbur. Ammo bir sinf abstract bo'lsa, bu sinf ichkarisidagi metodlar abstract bo'lishi kerak degan bir qoida yo'q.

7.6 Interfaces (Interfeyslar)

Interfeyslarni, butin sinflari tanasiz bo'lgan abstract sinflari kabi tushunishimiz mumkin. Bir interfeys ichidagi barcha metodlar tanasiz bo'lishi kerak. Interfeyslarni tayyorlash sinflarni tayyorlashga o'xshaydi. Misol uchun:

Interfeys_misol.java

```
package Sinflar;

public interface Iterfeys_misol {

        public void malumotlarni_yoz( );

}
```

Interfeyslar bir ham Java kodlari kabi oxirida **.java** yozilgan holda saqlanadi. Yuqorida ko'rganingiz kabi tanasiz qilish uchun abstract kalit so'ziga ehtiyojimiz yo'q. Chunki interfeyslar ichidagi metodlar ichi bo'sh bo'lishga majbur. Ya'ni biz hech qanday bir o'zgarish qila olmaymiz.

Interfeyslar odatda programmalarning vazifalarini ko'rsatib turadi deya olamiz. Interfeyslar sinflarning ko'rinishini ifodalab turadi va interfeyslardan foydalangan sinflar, interfeyslarga ko'ra ishlaydi. Birdan ko'p kishilar qatnashayotgan katta proyektlarda sinflar tayyorlanishi interfeyslar yordamida bajariladi. Interfeyslarni odatda jamoa liderlari tarafidan

tayyorlanadi. Dasturchilar esa, tayyorlangan interfeyslarni dastur holatiga keltirib chiqadi. Boshqacha qilib aytganda, dasturchilar interfeyslar ishlashi uchun kerak bo‘ladigan barcha ishlarni bajaradi.

Implements kalit so’zi

Bu so’zimizdan odatda bir sinfni tayyorlashda foydalanamiz. Bir sinf, bir interfeysdagi metodlardan foydalanishni istaganda, **implements** kalit so’zidan foydalanadi. Misol uchun pastadagi **Sotuvchi** sinfiga e’tibor beraylik.

```
public class Sotuvchi extends Foydalanuvchilar
                  implements Interfeys_misol {
    String dokon_nomi;
    String savdo_turi;

    public Sotuvchi() {
        /* ust sinfda paremetr olmaydigan
        quruvchilarni chaqir */
        super();
        this.dokon_nomi="";
        this.savdo_turi="";

    }
    @Override
    public void malumotlarni_yoz() {
        System.out.println(this.getIsm());
        System.out.println(this
                           .getFamiliya());
        System.out.println(this
                           .getTugulgan_sana());
        System.out.println(this.getJinsi());
        System.out.println(this
                           .getDokon_nomi());
        System.out.println(this
                           .getSavdo_turi());
```

```
    }
        // qolgan kodlar
}
```

Misolimizga e’tibor bergan bo’lsangiz, **Sotuvch**i sinfimiz interfeys hisoblangan **Interfeys_misol** nomli sinfimizni chaqirdi. Chaqirgandan so’ng sinf ichkarisidagi **malumotlarni_yoz** nomli bir bosh metodga Sotuvchi sinfi ichkarisida kodlar yozdi. Ya’ni interfeys sifatida berilgan sinflarni chaqirganimizdan keyin, interfeys sinfimiz ichida berilgan barcha metodlar uchun sinf ichkarisida kodlarni yozishimiz kerak.

Final kalit so’zini inheritance ichkarisida foydalanish

Final kalit so’zidan **inheritance** ichida ikki xil usulda foydalanamiz.

1. Agar bir sinf **final** so’zi bilan e’lon qilingan bo’lsa, bu sinfdan meros olgan holda yangi bir sinf tayyorlay olmaymiz. Misol uchun:

```
public final class Student() {

        //Student sinf kodlari

}

/* Bu sinfimizni pastdagi kabi chaqira olmaymiz chunki Student sinfimiz final kalit so’zi bilan berilgan*/

public class university extends Student(){

}
```

2. Agar bir metod **final** so'zi bilan berilgan bo'lsa bu metodni **@Override** so'zi orqali bekor qila olmaymiz. Pastdagi misolimizda Foydalanuvchi sinfi ichkarisida malumot_yoz metodini final so'zidan foydalangan holda e'lon qilganimiz uchun, Xaridor sinf ichkarisida bu metodimizni Override orqali bekor qila olmadik.

```
public class Foydalanuvchi{
        //kodlar
        public void final malumot_yoz(){
                System.out.printl(this.ism);
        }
}
---------------------------------------------------------------------------
public class Xaridor extends Foydalanuvchi{
        //kodlar
        /* pastdagi kodimiz ishlamaydi xato beradi chunki
        malumot_yoz metodimiz final so'zi bilan
        berilgan*/
        @Override
        public void malumot_yoz(){
        System.out.println(this.ism + this.familiya);
        }
}
```

7.7 To’plamlar (Collections)

To’plamlar bu Java dasturlash tili bilan birgalikda keladigan programmalardir. Bu to’plamlarimiz, oldingi mavzularda o’rganib chiqqan massivlarimizdan murakkab hisoblanadi. Bularga ham aslida massivlar deya olamiz. Bular ham ma’lumotlar bilan ishlaydi. To‘plamlarni uchga ayira olamiz.

- List
- Set (Set list turi)
- Map (Xarita)

List

List bu aytganimiz kabi massivlarning bir oz murakkab bo’lgani yoki bir oz kelishgan varianti sifatida ko’riladi, azizlar. Dasturchilar tarafidan eng ko’p foydalanilgan list turi bu ArrayList turi hisoblanadi. List turidagi ba’zi sinflar pastda berilgandir.

- ArrayList
- AbstractList
- LinkedList
- Vector

Set

Bu turdagi list turimiz ma’lumotlarni bazaga takrorlanmagan holda saqlashimizga yordam beradi. Misol uchun sonlardan tashkil topgan bir list bo’lsa, bu list ichida birdan ko’p 2 soni qatnashgan bo’lishi mumkin. Bu holatda 2 ham bir ma’lumot turi hisoblanganligi uchun listimiz ichkarisida bir ma’lumot

takrorlangan holda foydalanilyapti. Biz esa bir ma'lumotni bir list ichkarisida bir marta foydalanmoqchimiz, bu vaziyatda bizga set list turi yordam beradi. Set list ichkarisida bir ma'lumotni ikki marta yozganimizda yoki berganimizda listimiz avtomatik bir shaklda birini o'chirib tashlaydi. Set list turining ham shakllari mavjud, ular quyidagicha:

- AbstractSet
- HashSet
- TreeSet
- LinkedHashSet

Map

Bu list turlari normalda ikki qismda tashkil topadi: kalit(key) va qiymat(value). Har bir kalit bir-biridan farqli bo'lishi kerak. Har bir kalit farqli-farqli ma'lumot saqlaydi. Kalit so'zi orqali, u kalit so'zdagi ma'lumotni erisha olamiz. Eng ko'p foydalanilgan Map turi HashMap hisoblanadi.

Eng ko'p ArrayList foydalanilgani uchun bu bo'limda ArrayListga misollar berib o'tamiz.

ArrayList

ArrayList turidagi listimiz aslida bir sinf hisoblanadi, shuning uchun bu turdagi bir list tayyorlash uchun new kalit so'ziga ehtiyoj tuyamiz. Bir ArrayListni quyidagicha hosil qila olamiz.

```
List list = new Arraylist();
```

Yuqoridagi kodimizda list nomida bir ArrayList hosil qilindi. Bu listimizga beriladigan qiymatlar yoki ma'lumotlarning turi aniq bo'lmaganligi uchun ma'lumot turini yozmadik. Agar listimizga beradigan ma'lumotlarimizning turi aniq bo'lsa < > xarakterlar orasida yozib o'tamiz. Misol uchun listimizga String turida bo'lgan ma'lumotlar beramiz.

```
List<String> ismlar= new ArrayList<String>();
```

ArrayList qiymatlarini yoki a'zolarini berish

Listimizga a'zolar qo'shish uchun *add* metodidan foydalanamiz. Pastdagi misolimizda ismlar nomli ArrayListimizga 4 dona yangi a'zolar qo'shilganini ko'rishingiz mumkin.

```
List<String> ismlar= new ArrayList<String>();

ismlar.add("Davronbek");

ismlar.add("Mo'jiza");

ismlar.add("Sherzodbek");

ismlar.add("Muhammadqodir");
```

ArrayList ichida joylashgan a'zolarni chaqirish

ArrayList ichidagi biror bir a'zo qiymatini chaqirish uchun yoki olish uchun *get* metodidan foydalanishimiz kerak.

```
String azo_1 = ismlar.get(1);

System.out.println(azo_1);

//ekranga Davronbek ismi chiqariladi
```

ArrayList ichida berilgan bir a’zoning qiymatini o’zgartirish

ArrayListimizda berilgan bir a’zoning qiymatini o’zgartirish uchun *set* metodidan foydalanamiz. Misol uchun yuqorida berib o’tgan ismlar nomli listimizdagi bir a’zoning qiymatini almashtiraylik.

```
ismlar.set(3, “Marjona”);

// uchinchi a’zomizning qiymatini Marjonaga almashtirdik
```

ArrayListda berilgan bir a’zoni o’chirib tashlash

ArrayList ichida joylashgan bir a’zoni o’chirish uchun *remove* metodidan foydalanamiz. Bu metoddan foydalanish uchun biz o’chirmoqchi bo’lgan a’zomizning indeksini, metodimizga parametr sifatida berishimiz kerak. Misol uchun pastda berilgan misolimizda listimizdagi birinchida turgan a’zoni o’chiriladi. Listimiz a’zolarni indekslashni 0 sonidan boshlar edi, bu sababli birinchi a’zoni o’chirish uchun metodimizga 0 sonini kiritdik.

```
ismlar.remove(0);
```

ArrayList ichida nechta a’zo bor ekanligini topish

ArrayList ichkarisida nechta a’zo mavjud ekanligini topish uchun *size* metodidan foydalanamiz. Misol uchun pastdagi kodimizda dasturimiz ArrayList ichkarisidagi a’zolar sonini hisoblar ekranga chiqaradi.

```
int azolar_soni = ismlar.size();

System.out.println(“A’zolar soni: ”+azolar_soni);

// A’zolar soni: 4
```

ArrayListning barcha a’zolarini o’chirib tashlash

ArrayList ichidagi barcha a’zolarni o’chirish uchun clear metodidan foydalanamiz.

```
ismlar.clear();

int azo_soni = ismlar.size();

System.out.println(“A’zolar soni:” + azo_soni);

// A’zolar soni: 0
```

ArrayList ichidagi a’zolarni tartiblash

List ichidagi ma’lumotlarni yoki qiymatlarni tartiblash uchun **Collections** sinfi ichkarisidagi metodlardan foydalanamiz. Bu sinf ichkarisida o’rganadigan birinchi metodimiz *sort* metodi. Bir listni tartiblash uchun sort metodidan foydalana olamiz.

```
Collections.sort(ismlar);
```

ArrayList ichidagi a’zolarni teskari shaklda tartiblash

List ichidagi a’zolarni teskari bir shaklda tartiblash uchun Collections sinfi ichkarisida joylashgan *reverse* metodidan foydalanamiz.

```
Collections.reverse(ismlar);
```

ArrayList misol

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

public class ArrayList_misol {

    public static void main(String[] args) {
        List<String> ismlar =
                                new ArrayList<>();
        ismlar.add("Davronbek");
        ismlar.add("Mo'jiza");
        ismlar.add("Sherzodbek");
        ismlar.add("Muhhammadqodir");
        ismlar.add("Oyshabegim");
        ismlar.add("Mustafo");

        System.out.println("Listimiz: "
                                        +ismlar);
        System.out.println("----------------");

        System.out.println("Birinchi
                           a'zomizning qiymati:
                           "+ismlar.get(0));
        System.out.println("----------------");
```

```
        ismlar.set(0,"Dilnoza");
        System.out.println("Birinchi
                           a'zomizning qiymatini
                           almashtirdik"
                           + ismlar);
        System.out.println("---------------");

        ismlar.remove(2);
        System.out.println("Ikkinchi
                            indeksdagi a'zoni
                            o'chirdik"
                            + ismlar);
        System.out.println("---------------");

        Collections.sort(ismlar);
        System.out.println("Listimiz
                            a'zolarini
                            tartiblashtirdik"
                            + ismlar);
        System.out.println("---------------");

        Collections.reverse(ismlar);
        System.out.println("Listimizni teskari
                            bir shaklda
                            tartiblashtirdik"
                            + ismlar);
        System.out.println("---------------");

        ismlar.clear();
        System.out.println("Listimiz
                           ichkarisidagi barcha
                           a'zolarni
                           o'chirdik:");
        System.out.println("Listimiz: "
                            +ismlar);

    }
}
```

Bu kodimizni ishga tushirganimizda quyidagicha bir javob olamiz:

```
Listimiz: [Davronbek, Mo'jiza, Sherzodbek, Muhhammadqodir, Oyshabegim, Mustafo]

------------------

Birinchi a'zomizning qiymati : Davronbek

------------------

Birinchi a'zomizning qiymatini almashtirdik [Dilnoza, Mo'jiza, Sherzodbek, Muhhammadqodir, Oyshabegim, Mustafo]

------------------

Ikkinchi indeksdagi a'zoni o'chirdik [Dilnoza, Mo'jiza, Muhhammadqodir, Oyshabegim, Mustafo]

------------------

Listimiz a'zolarini tartiblashtirdik [Dilnoza, Mo'jiza, Muhhammadqodir, Mustafo, Oyshabegim]

------------------

Listimizni teskari bir shaklda tartiblashtirdik [Oyshabegim, Mustafo, Muhhammadqodir, Mo'jiza, Dilnoza]

------------------

Listimiz ichkarisidagi barcha a'zolarni ochirdik:

Listimiz: []
```