

Java 2 Standard Edition

ZAMONAVIY DASTURLASH TILI

Kasb-hunar kollejlari uchun o'quv qo'llanma

Rashid Turg'unboyev

“Java 2 Standard Edition – zamonaviy dasturlash tili” o’quv qo’llanma Java 2 Standard Edition dasturlash tili haqida bo’lib, o’rta mahsus kasb-hunar kollejlari uchun mo’ljallangan. Qo’llanmada Java dasturlash tili, dasturlash muhiti, Java dasturlash tilining asosiy strukturasi, obyektga ixtisoslashgan dasturlash, obyektlar, klasslar, klasslararo bog’lanish, foydalanuvchining grafik interfeysi, istisno va xatoliklar haqida batafsil ma’lumot berib o’tilgan. Qo’llanmaning xar bir bo’limi misollar bilan boyitilgan.

Mundarija

1. KIRISH

- 1.1. Algoritm tushunchasi
- 1.2. Java dasturlash tili va Java dasturining ishlash printsipli
- 1.3. Javaning qisqacha tarixi

2. JAVA DASTURLASH MUHITI

- 3.1. Java dasturlash vositasini o'rnatish
- 3.2. Dasturlash muhitini tanlash
- 3.3. Sodda Java dasturi

3. JAVA DASTURLASH TILINING ASOSIY STRUKTURASI

- 3.4. Java birlamchi kod strukturasi va main() metodi
- 3.5. Izohlar
- 3.6. Identifikatorlar
- 3.7. O'zgaruvchilar
- 3.8. Qiymat berish operatori
- 3.9. Arifmetik operatorlar
- 3.10. Bajarilish ketma-ketligini boshqarish
- 3.11. Tsikllar
- 3.12. Massivlar

4. OBYEKTLAR

- 4.1. Obyektlarga ihtisoslashgan dasturlashga kirish
- 4.2. Klasslar
- 4.3. Metodlar
- 4.4. Obyekt konstruksiyasi
- 4.5. Obyekt destruksiyasi

5. KLASSLARARO BOG'LANISH

- 5.1. Ostki klasslar va irsiyat
- 5.2. Interfeyslar
- 5.3. Ichki klasslar
- 5.4. O'ramlar
- 5.5. Klass a'zolari va metodlarni ko'rinishi
- 5.6. Obyekt klassi

6. GRAFIK DASTURLASH

- 6.1. Swing
- 6.2. Frame yaratish va joylashtirish
- 6.3. Ma'lumotlarni Panelda ko'rsatish
- 6.4. Xodisa va amal
- 6.5. Tugmalar
- 6.6. Yozuvlar
- 6.7. Checkbox va Radio tugmalar
- 6.8. Ro'yxat va Combobox
- 6.9. Slayder va Spinner
- 6.10. Matn kiritish
- 6.11. Menyular
- 6.12. Qurollar paneli

6.13. Kontekst menyu

6.14. Dialog oynalar

6.15. Joylashuv menedjerlari

7. ISTISNO VA XATOLAR

7.1. Xatolik va istisnolar

7.2. Istisnolar bilan ishlash

KIRISH

Algoritm tushunchasi

Algoritm bu ma'lum vaqt ichida berilgan masalani bajarish uchun ketma-ket bajarilishi kerak bo'ladigan aniq ko'rsatmalar to'plamidir.

Algoritm so'zi buyuk olim, yurtdoshimiz Abu Abdulloh Muhammad ibn Musa al-Xorazmiy nomidan kelib chiqqan. 825 yilda al-Xorazmiy o'z risolasida birinchi bo'lib Xindistonda kashf qilingan o'nlik sanoq sistemasi haqida ma'lumot bergan. Al-Xorazmiy ushbu yangi sanoq sistemasida xisob ishlarini bajarishni qoidalarini ifoda etgan. XII asrning birinchi yarmida Al-Xorazmiy asarining lotin tilidagi tarjimasi Yevropaga kirib bordi. Tarjima lotincha "Algoritmi de numero Indorum" deb nomlandi. Ushbu nom "Algoritmi Xind sonlari" degan ma'noni bildirardi va bundagi "Algoritmi" so'zi Al-Xorazmiyning ismini lotinlashishini bildirar edi. Ushbu risola yordamida "algoritm" so'zi yevropa tillariga kirib bordi.

Xar qanday algoritm quyidagi talablarga javob berishi kerak:

1. Diskretlik – algoritm masalani bajarish jarayonini ma'lum bir qadamlar ketma-ketligi sifatida ko'rsatishi kerak;
2. Aniqlik – xar bir vaqt qiymati uchun keyingi bajariladigan qadam tizim xolati bilan belgilanadi;
3. Tushunarliklik – algoritm faqatgina bajaruvchiga ma'lum komandalarni o'z ichiga olishi kerak;
4. Yakunlilik – birlamchi ma'lumotlar berilganda algoritm o'z ishini chekli sondagi qadamlarda tugatishi kerak;
5. Universallik – algoritm xar xil to'plamdagi birlamchi ma'lumotlarga qo'llanishi kerak;
6. Natijalilik – algoritmni ma'lum natija bilan tugatilishi;

Algoritmni so'zlar yoki sxemalar yordamida ifoda qilish mumkin. Odatda, biron bir masalani algoritmini tuzish uchun birinchi uning algoritmi so'zlar bilan ifodalanadi. Keyinchalik, masalani amalda bajarishga yaqinlashganda algoritmni bajaruvchisiga tushunarli tilda tuziladi (masalan, mashina kodida). Algoritmni ko'rgazmali ifoda etish uchun blok-sxemalardan foydalaniladi. Algoritmni bajaruvchisi tiliga bog'liq bo'lmagan ifoda qilish usuli psevdokod xisoblanadi.

Java dasturlash tili va Java dasturining ishlash printsiplari

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

- Ob'ektlarga ixtisoslashish xususiyatlari
- Ko'pvazifalilik

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

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

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

Aksariyat zamonaviy dasturlash tillari inson uchun unda dastur yozishi va tushinishini osonlashtirib tuzilgan. Bunday dasturlar yuqori-darajali dasturlash tillari deb ataladi. Kompyuter tushunadigan tilni mashina tili deb ataladi va ular past-darajali tillar deb ataladi. Yuqori-darajali dasturlash tilida yozilgan dasturni kompyuterda ishga tushirishdan oldin uni mashina tiliga o'girib olish kerak. Bunday o'giruvchi dasturni kompilyator va o'girish jarayonini kompilyatsiya deb ataladi.

Aksariyat dasturlash tillarining bitta kamchiligi bu ularning kompilyatorlari yuqori-darajali dasturlash tilida yozilgan dasturni to'g'ridan-to'g'ri mashina tiliga o'giradi. Xar-hil kompyuterlar xar-hil mashina tilini ishlatishi sababli xar-bir kompyuter turi uchun o'zining maxsus kompilyatori zarur bo'ladi. Java dasturlash tili kompilyatsiyaga boshqacha yondashadi.

Java kompilyatori dasturni aniq bir kompyuterning mashina kodiga tarjima qilmaydi. Buni o'rniga kompilyator dasturni *bayt-kod* deb nomlanuvchi tilga o'giradi. Bayt-kod aniq bir kompyuterning mashina kodi xisoblanmaydi. Bayt-kod Java virtual mashinasining mashina kodi xisoblanadi. Java virtual mashinasi o'z navbatida bayt-kodni o'zi joylashgan kompyuterning mashina kodiga o'girib amalga oshiradi.

Java dasturini bajarish uchun quyidagilarni bajarish kerak bo'ladi. Dasturchi birlamchi kodni *java* kengaytmali faylga saqlaydi. So'ng, Java kompilyatori yordamida birlamchi kodni kompilyatsiya qilib *class* kengaytmali fayl xosil qiladi. Xosil bo'lgan faylni Java interpretatori yordamida bajaradi.

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

1-bosqich. Dasturchi dastur birlamchi kodini tuzib *Hello.java* fayliga saqlaydi. Birlamchi kodni odatda biron-bir sodda matn muharriri yordamida kiritiladi, masalan Windows operatsion tizimining Блокнот dasturi.

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



Hello.java

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

```
c:\>javac Hello.java
```



Hello.class

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

```
c:\>java Hello
```

```
Hello World!
```

Javaning qisqacha tarixi

Java dasturlash tilining tarixi 1991 yilga boradi. Sun Microsystems kompaniyasining muhandislari, Patrick Naughton va James Gosling, kabel televideniya qurilmalarida ishlatish uchun kichik dasturlash tilini yaratmoqchi qo'lishadi. Ushbu qurilmalarning hisoblash quvvati va xotirasi kichik bo'lgani uchun ularga mo'ljallangan dastur icham bo'lishi kerak bo'lgan. Bundan tashqari turli ishlab-chiqaruvchilarning qurilmalari har-xil protsessor ishlatishi mumkin bo'lganligi uchun dasturlash tili bir qurilma arxitekturasiga bog'liq bo'lib qolmasligi ham kerak edi. Projekt "Green" deb nom oldi.

Paskal dasturlash tilining asoschisi Niklaus Wirth birinchi bor portativ dasturlash tilini yaratish g'oyasi bilan chiqqan. Ushbu dasturlash tilida abstrakt mashina (ko'p hollarda bular virtual mashina deb ataladi) uchun dastur kodi generatsiya qilinib, keyin esa ushbu dastur kodi o'ziga mos interpretatori mavjud barcha mashinada ishlatilishi mumkin edi. "Green" projekt muhandislari yaratayotgan dasturlash tilida virtual mashina va Paskal dasturlash tilidan farqli o'laroq obyektlarga ixtisoslashish printsiplarini qo'llab yangidasturlash tilini yaratdilar. Yaratilgan dasturlash tili Java deb nomlandi.

1996 yilda Sun kompaniyasi Java dasturlash tilining birinchi versiyasini chiqardi. Java 1.0 versiyasi imkoniyatlari unchalik ko'p bo'lmasada, 1998 yilda keyingi Java 1.2 versiyasida oldingi kamchiliklar to'ldirilib qo'shimcha imkoniyatlar qo'shildi. Ushbu versiyadan boshlab Java dasturlash tillari qatorida barqarorlashdi va rasman Java 2 deya nomlana boshlandi. Java 1.3 va 1.4 versiyalarida muhim o'zgarishlar bo'lmasada Java 2 imkoniyatlari yaxshilandi va standart kutubxona kengaytirildi. 2004 yilda e'lon qilingan Java 1.5 versiyasida dasturlash tiliga ko'pgina kontseptual o'zgarishlar qo'shildi va rasman Java 5.0 deb nomlandi. Bundan tashqari Java dasturlash tilining shaxsiy kompyuterlar uchun "Standart versiyasi", mobil va kichik qurilmalar

uchun “Mikro versiyasi” va serverlar uchun “Korporativ versiyasi” taqdim etildi. Ushbu qo’llanmada Java dasturlash tilining Standart versiyasi yoritilgan.

Har keyingi versiyasidan Java dasturlash tilining standart kutubxonasi kengayib bordi. Quyidagi jadvalda ushbu o’zgarishlar ko’rsatilgan:

Versiya	Klass va interfeyslar soni
1.0	211
1.1	477
1.2	1524
1.3	1840
1.4	2723
1.5	3270

Java standart kutubxonasining kengayishi

Takrorlash uchun savol va topshiriqlar:

1. Algoritm deganda nima tushuniladi?
2. Algoritmilar qanday talablarga javob berishi kerak?
3. Algoritmnlarni ifodalash usullari haqida gapirib bering?
4. Java dasturlash tili qanday xususiyatlarga ega?
5. Birlamchi kod va bayt kod orasidagi farq nimada?
6. Java dasturlash tilining tarixi haqida gapirib bering.

JAVA DASTURLASH MUHITI

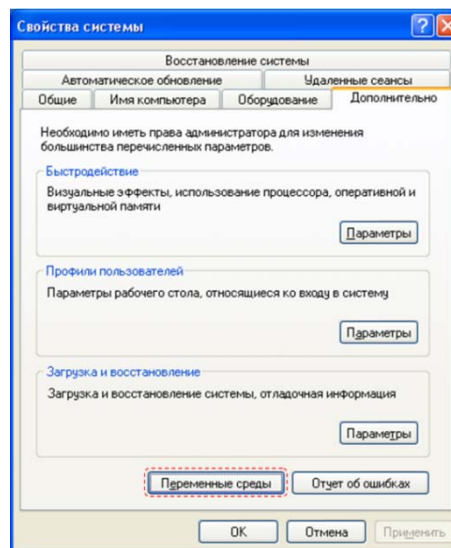
Java Dasturlash Vositasini o'rnatish

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

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

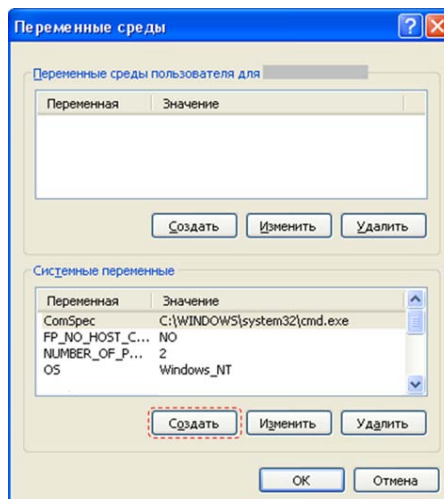
C:\Program Files\Java\jdk1.6

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



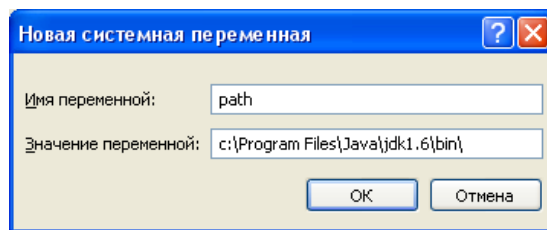
Windows operatsion tizimining xususiyatlar oynasi

Natijada muhit o'zgaruvchilari oynasi ochiladi.



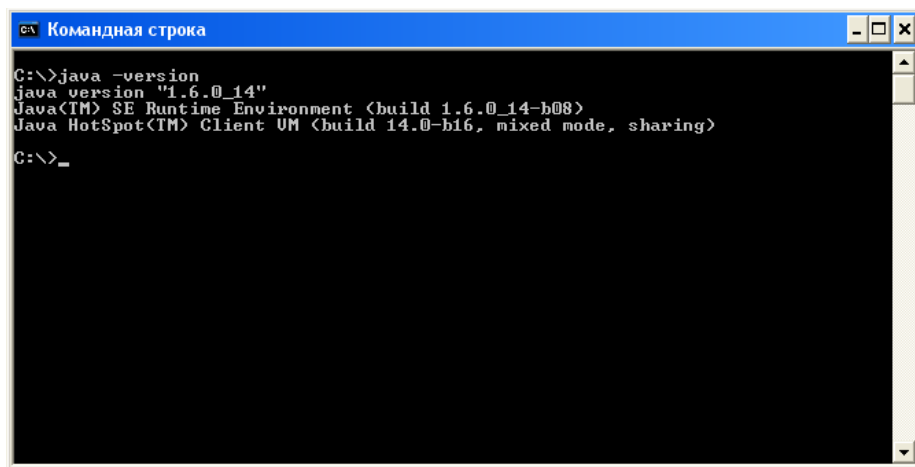
Мухит o'zgaruvchilari oynasi

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



Yangi muhit o'zgaruvchisini yaratish dialog oynasi

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



Java versiyasi haqida ma'lumot olish

Dasturlash muhitini tanlash

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

- Java Dasturlash Vositasi va biron-bir matn muxarriri (masalan, Notepad);
- Integrallashgan Dasturlash Muhitidan foydalanish (masalan, Eclipse);
- Java Dasturlash Vositasi va u bilan integrallashgan matn muxarriridan foydalanish (masalan, TextPad);

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

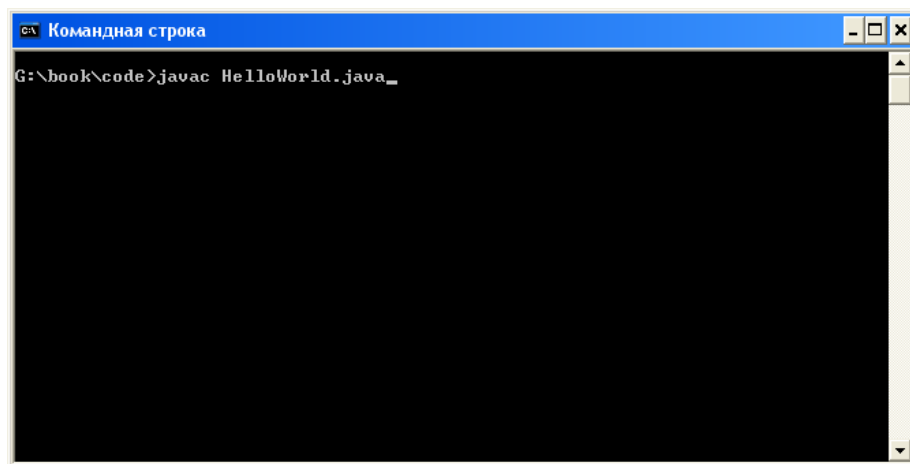
Sodda dasturlarni tuzish uchun biror-bir matn muxarriri yoki Java Dasturlash Vositasi bilan integrallashgan matn muxarriri qo'llash mumkin. Birinchi xolatda dasturning birlamchi kodi matn muxarriri yordamida kiritiladi va komanda qatori orqali kompilyatsiya qilinadi va ishga tushiriladi. Ikkinchi xolatda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

Sodda Java dasturi

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

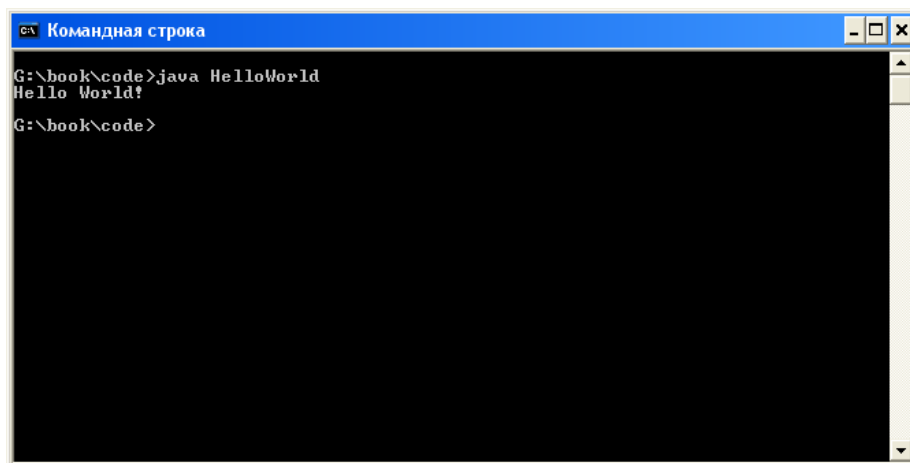
```
public class HelloWorld
{
    public static void main(String[] args)
    {
        System.out.println("Hello World!");
    }
}
```

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



Java kompilyatori yordamida birlamchi kodni kompilyatsiya qilish

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



Java Virtual Mashinasi yordamida baytkodni ishga tushirish

Korib turilganidek dastur konsol oynasiga "Hello World!" matnini chiqarib berdi.

Endi ushbu dasturning birlamchi kodini taxlillab chiqamiz. Java dasturlash tilida barcha dastur elementlari class ichida joylashadi. Bizning xolatda class quyidagicha e’lon qilingan:

```
public class HelloWorld  
{  
    ...  
}
```

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

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

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

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

Takrorlash uchun savol va topshiriqlar:

1. Java Dasturlash Vositasini o’rnatilish tartibi xaqida gapirib bering.
2. Windows operatsion tizimining *path* muhit o’zgaruvchisi nimani belgilaydi?
3. Java dasturlash tilida dastur yozish uchun qanday muhitlar mavjud?
4. “Salom dunyo” matnini konsolga chiqarib beruvchi dastur tuzing.
5. Birlamchi kod fayli qanday kengaytmaga ega bo’lishi kerak?
6. Bayt kod fayli qanday kengaytmaga ega?

JAVA DASTURLASH TILINING ASOSIY STRUKTURASI

Java birlamchi kod strukturasi va `main()` metodi

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

```
public class Program1
{
    //dastur kodi
}
```

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

```
public class Program1
{
    void go()
    {
        //metod kodi
    }
}
```

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

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

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

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

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

Izohlar

Dastur yozish mobaynida ushbu dasturda ishlatilayotgan elementlar va qo'llanilayotgan usullarga izoh yozish ehtiyoji paydo bo'ladi. Ayniqsa dastur yozish uchun uzoq vaqt talab etilsa yoki bir dasturni bir necha dasturchi yozayotgan bo'lsa ushbu izohlar ahamiyati yanada yoqori bo'ladi. Java dasturlash tilida boshqa dasturlash tillari singari izohlar dastur bajarilishiga hech qanday ta'sir qilmaydi va ular dasturchi uchun ma'lumot sifatida qo'llaniladi.

Java dasturlash tilida izoh qo'yishni uch xilusuli mavjud. Eng keng tarqalgan usuli bu izoh boshiga // simvollarini qo'yishdir. Ushbu simvollardan keyin joylashgan elementlar qatorning oxirigacha izoh xisoblanadi. Masalan:

```
System.out.println("Salom!"); //ushbu qator konsolga matn chiqarib beradi
```

yoki

```
//quyidagi operator x qiymatini bir qiymatga oshiradi
```

```
x++;
```

Agar uzunroq izoh berish kerak bo'lsa, bir necha qatorni // elemenlari bilan belgilash mumkin, yoki ikkinchi usuldan foydalanish mumkin. Bu usulda /* va */ belgisi cheklovchilaridan foydalaniladi. Birinchi belgi izoh boshiga ikkinchisi izoh oxiriga qo'yiladi. Masalan:

```
/*
```

```
    Quyidagi dastur Java dasturlash tilida yozilgan bo'lib
```

```
    U konsolga Salom so'zimi chiqarib beradi
```

```
*/
```

```
public class SalomDasturi
```

```
{
```

```
    public static void main(String[] args)
```

```
    {
```

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

```
    }
```

```
}
```

Izoh berishning uchunchi usuli dasturning izohlariga asosan xujjatlarni avtomatik xosil qilish uchun ishlatiladi. Ushbu izohlar /** va */ belgilari bilan chegaralanadi.

Identifikatorlar

Dasturda ishlatiladigan o'zgaruvchi, klass, metod yoki ob'yekt nomlari identifikatorlar deb ataladi. Java identifikatorlari xarflar, sonlar, "\$" simvoli va "_" simvolidan tashkil topishi mumkin. Masalan,

```
test, test1, _test, TEST, $test
```

barchasi to'g'ri identifikator xisoblanadi. Identifikatorlar tanlashda quyidagi qoidalarga rioya qilish kerak:

- Identifikatorlar harf, "\$" simvoli yoki "_" simvoli bilan boshlanishi kerak;
- Birinchi xarfdan keyin identifikator xarflar, "\$" simvoli, "_" simvoli yoki sonlarning istalgan kombinatsiyasini o'z ichiga olishi mumkin;
- Identifikator istalgan miqdordagi simvollaridan tashkil topishi mumkin;
- Java kalit so'zlarini identifikator sifatida ishlatish mumkin emas;
- Identifikatorlar xarf kattaligini farqlaydi. Masalan, test va Test ikkita xar-xil identifikatorlar.

To'g'ri tuzilgan identifikatorga misollar:

```
_test
$test
_____test_2
_.$
juda_batavsil_berilgan_identifikator
```

Quyidagilar esa noto'g'ri tuzilgan identifikatorlar:

```
:test
-test
test#
.test
4test
```

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

abstract	boolean	break	byte	case	catch
char	class	const	continue	default	do
double	else	extends	final	finally	float
for	goto	if	implements	import	instanceof
int	interface	long	native	new	package
private	protected	public	return	short	static
strictfp	super	switch	synchronized	this	throw
throws	transient	try	void	volatile	while
assert	enum				

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

- Klass va interfeys nomlariga: birinchi xarfi katta xarf bo'lishi va agar bir necha so'zdan iborat bo'lsa xar bir so'zning birinchi xarfi katta xarf bo'lishi kerak. Masalan:

```
Dastur
XisobRaqam
```

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

```
balansniTekshirish
xisobIshiniBajarish
```

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

```
DARAJA
MIN_NARX
```

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

O'zgaruvchilar

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

```
int elementlarSoni;

double elementYuzasi, elementHajmi;
```

Birinchi misol *elementlarSoni* o'zgaruvchisi *int* turdagi qiymatlarni olishini e'lon qiladi. *int* butun sonlarni anglatadi. Ikkinchi misol *elementYuzasi* va *elementHajmi* o'zgaruvchilarni *double* turdagi qiymatlarni qabul qilishini e'lon qiladi. *double* ratsional sonlarni anglatadi. Ushbu misoldan ko'rinadiki, bir turdagi o'zgaruvchilarni vergul bilan ajratib bir qatorda e'lon qilish mumkin.

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

Tur nomi	Qiymati	Ishlatadigan xotira	Qiymat chegarasi
boolean	true yoki false	1 byte	
char	Bitta xarf	2 byte	Barcha Unikod xarflari
byte	Butun son	1 byte	-128 dan 127 gacha
short	Butun son	2 byte	-32768 dan 32767 gacha
int	Butun son	4 byte	-2147483648 dan 2147483647 gacha
long	Butun son	8 byte	-9223372036854775808 dan 9223372036854775807 gacha
float	Ratsional son	4 byte	$-3.40282347 \cdot 10^{+38}$ dan $-1.40239846 \cdot 10^{-45}$ gacha
double	Ratsional son	8 byte	$\pm 1.76769313486231570 \cdot 10^{+308}$ dan $\pm 4.94065645841246544 \cdot 10^{-324}$ gacha

Qiymat berish operatori

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

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

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

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

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

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

```
x += 2;  
  
ifodasi  
  
x = x + 2;
```

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

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

Arifmetik operatorlar

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

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

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

```
count = count1 + count2
```

Agar `count1` va `count2` o'zgaruvchilar butun son bo'lsa `count` o'zgaruvchining qiymati ham butun son (`int`) bo'ladi. Agar `count1` yoki `count2` o'zgaruvchilarining biri ratsional bo'lsa `count` o'zgaruvchisining qiymati ratsional qiymat bo'ladi.

Inkrement va dekriment operatorlari

Java dasturlash tilida qiymatlarni bir birlikka o'zgartiruvchi operatorlar mavjud. Qiymatni bir birlikka oshiruvchi operator inkrement operatori bo'lib umumiy ko'rinishi `n++` ko'rinishida bo'ladi. Qiymatni bir birlikka kamaytiruvchi operator dekriment operatori bo'lib `n--` ko'rinishida bo'ladi. Masalan,

```
int n = 1;

int m = 4;

n++;

System.out.println("n qiymati = " + n);

m--;

System.out.println("m qiymati = " + m);
```

dastur kodi quyidagi satrlarni ekranga chiqarib beradi,

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

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

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

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

```
int n = 3;
```

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

dastur kodi quyidagi satrlarni ekranga chiqarib beradi,

```
6  
  
4
```

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

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

dastur kodi quyidagi satrlarni ekranga chiqarib beradi,

```
8  
  
4
```

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

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

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

dastur kodi quyidagi satrlarni ekranga chiqarib beradi,

```
5  
  
4
```

Aksincha,

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

```
System.out.println(n);
```

dastur kodi quyidagi satrlarni ekranga chiqarib beradi,

```
4
```

```
4
```

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

Bajarilish ketma-ketligini boshqarish

Java, boshqa dasturlash tillari singari, bajarilishni boshqarish uchun shartli ifodalar va tsikllarni qo'llaydi.

`if/else` shartli ifodasi

`if` shartli ifoda quyidagi umumiy ko'rinishga ega:

```
if (shart) ifoda
```

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

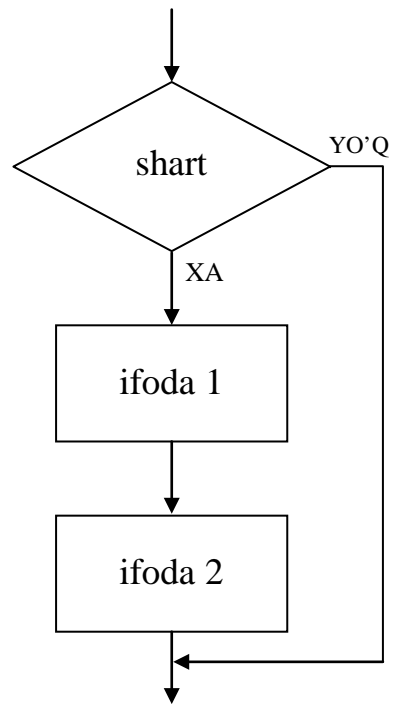
```
if (shart)
{
    ifoda 1;
    ifoda 2;
}
```

Masalan:

```
if (x>25)
{
    tanNarx = 130;
    sifat = "yaxshi";
}
```

dastur kodida agar `x` qiymati 25dan katta bo'lsa u xolda shartli ifodaning figurali qavslari ichida joylashgan barcha ifodalar bajariladi, ya'ni, `tanNarx` o'zgaruvchisining qiymati 130ga `sifat` o'zgaruvchisining qiymati "yaxshi" qiymatiga o'zgartiriladi.

Quyida `if` shartli ifodaning blok sxemasi keltirilgan.



if shartli ifodaning blok sxemasi

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

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

Ushbu ko'rinishda shartli ifodaning sharti to'g'ri bo'lsa ifoda1 bajariladi, agar shart noto'g'ri bo'lsa ifoda2 bajariladi. Agar shartning xar bir xolatida bir necha ifodalar bajarilishi kerak bo'lsa ifodalar figurali qavslar ichida bo'lishi kerak.

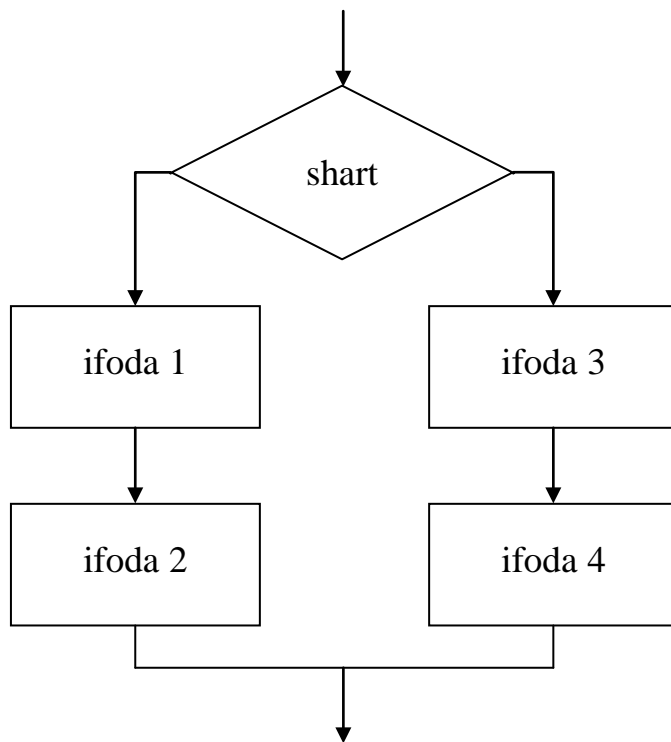
```
if (shart){
    ifoda1;
    ifoda2;
} else {
    ifoda3;
    ifoda4;
}
```

Masalan:

```
if (x>25){
    tanNarx = 130;
    sifat = "yaxshi";
} else {
    tanNarx = 50;
    sifat = "qoniqarsiz";
}
```

Dastur kodida agar x qiymati 25dan katta bo'lsa tanNarx o'zgaruvchisining qiymati 130ga va sifat o'zgaruvchisining qiymati "yaxshi"ga o'zgaradi. Agar shartli ifodaning sharti bajarilmasa, ya'ni x qiymati 25dan kichik bo'lsa, tanNarx o'zgaruvchisining qiymati 50ga va sifat o'zgaruvchisining qiymati "qoniqarsiz"ga o'zgaradi.

`if/else` shartli ifodaning blok sxemasi quyidagi ko'rinishda bo'ladi.



if/else shartli ifodaning blok sxemasi

Agar dasturda bir necha shartlarni ketma ket tekshirish kerak bo'lsa `if/else if` shartli ifodadan foydalanish mumkin:

```
if (shart1)
{
    ifoda1;
} else if (shart2) {
    ifoda2;
} else if (shart3) {
    ifoda3;
}
```

Masalan,

```
if (x>100)
{
    tanNarx = 250;
    sifat = "alo";
}
```

```

} else if (x>25) {

    tanNarx = 130;

    sifat = "yaxshi";

} else if (x<25) {

    tanNarx = 50;

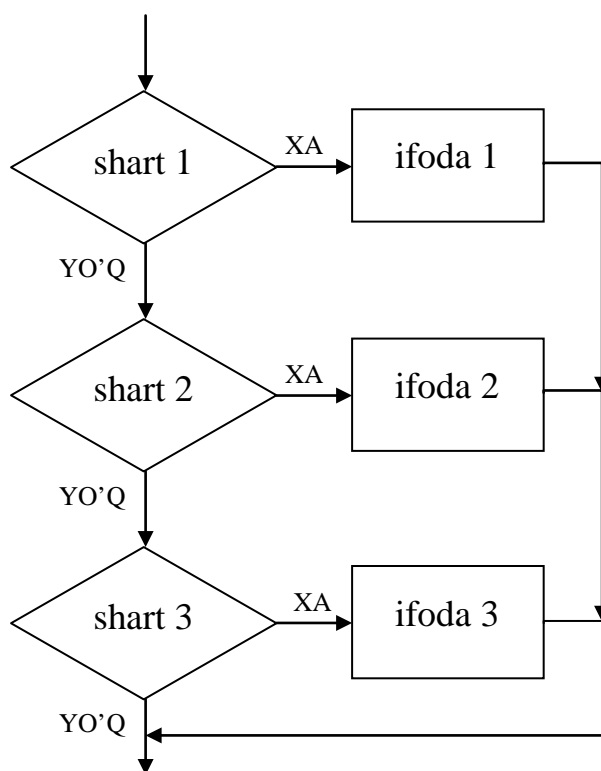
    sifat = "qoniqarsiz";

}

```

dastur kodida x qiymati 100dan katta bo'lsa tanNarx o'zgaruvchisining qiymati 250, sifat o'zgaruvchisining qiymati "alo", agar x qiymati 100dan kichik va 25dan katta bo'lsa tanNarx qiymati 130ga, sifat qiymati "yaxshi"ga, agar x qiymati 25dan kichik bo'lsa tanNarx qiymati 50ga, sifat qiymati "qoniqarsiz"ga o'zgaradi.

if/else if shartli ifodaning blok sxemasi quyidagi ko'rinishda bo'ladi.



if/else if shartli ifodaning blok sxemasi

switch shartli ifoda

switch shartli ifoda boshqaruvni blok ichida joylashgan, ma'lum belgiga ega ifodaga o'tkazib berish uchun xizmat qiladi. switch shartli ifoda quyidagi umumiy ko'rinishga ega:

```

switch (n)

{

    case a: ifoda_1;

```

```

    case b: ifoda_2;

    ...

    default: ifoda_N;
}

```

Bu yerda `n` o'zgaruvchisining turi butun son, ya'ni `int`, bo'lishi kerak. Shartli ifoda bajarilganda `n` o'zgaruvchisi qabul qiladigan qiymat `switch` shartli ifodasining `case` qatorida joylashgan qiymatiga teng bo'lsa ushbu qatordagi ifoda bajariladi. Agar o'zgaruvchi qiymati `case` qatorlarining qiymatlariga teng bo'lmasa u xolda `default` qatorining ifodasi bajariladi. `default` qatori odatda `switch` shartli ifodaning oxirida joylashadi. `default` qatori shartli xisoblanmaydi va yozilmasligi xam mumkin. Masalan,

```

int x = 2;

switch (x)
{
    case 1: System.out.println("Birinchi ifoda");

    case 2: System.out.println("Ikkinchi ifoda");
}

```

`x` o'zgaruvchisining qiymati 2 teng bo'lgani uchun 2 qiymatga ega `case` qatori bajariladi. Ushbu dastur kodi quyidagi matnni ekranga chiqarib beriladi:

```
Ikkinchi ifoda
```

Shuni ta'kidlab o'tish kerakki `case` qatorlari `switch` shartli ifodaning kirish nuqtasi xisoblanadi, ya'ni tegishli `case` qatori bajarilganda undan keyin joylashgan qatorlar xam bajariladi. Masalan:

```

int x = 2;

switch (x)
{
    case 1: System.out.println("Birinchi qator");

    case 2: System.out.println("Ikkinchi qator");

    case 3: System.out.println("Uchunchi qator");

    default: System.out.println("Default qatori");
}

```

Yuqoridagi dastur kodi bajarilganda ekranga quyidagi matn chiqarib beradi:

```

Ikkinchi qator

Uchunchi qator

Default qatori

```


Agar `switch` shartli ifodasining faqatgina bir qatori bajarilishi kerak bo'lsa ushbu qator oxirisiga bajarilish tartibini buzuvchi ifodadan, ya'ni `break` ifodasidan, foydalanish mumkin. Masalan:

```
int x = 2;

switch (x)
{
    case 1:
        System.out.println("Birinchi qator");
        break;

    case 2:
        System.out.println("Ikkinchi qator");
        break;

    case 3:
        System.out.println("Uchinchi qator");
        break;

    default: System.out.println("Default qatori");
}
```

Yuqoridagi dastur kodi quyidagi matnni ekranga chiqarib beradi:

```
Ikkinchi qator
```

Tsikllar

Java dasturlash tilida tsikllar asosan berilgan ifodani biror bir shart to'g'ri bo'lishi davomida bajaradi.

`while` tsikli shart to'g'ri bo'lishi davomida ifodani bajaradi. Agar shart to'g'ri bo'lmasa tsiklda joylashgan ifoda bajarilmaydi. `while` tsiklining umumiy ko'rinishi quyidagicha bo'ladi:

```
while (shart) ifoda
```

Masalan:

```
while (xomAshyo > 0) {
    maxsulotIshlabChiqarish();
    maxsulot++;
    xomAshyo--;
}
```

Dastur kodi `xomAshyo` tugagunga qadar, ya'ni `Oga` teng bo'lgunga qadar, `maxsulotIshlabChiqarish()` metodini ishga tushiradi. Agar `xomAshyo` bo'lmasa, ya'ni `Oga` teng bo'lsa, tsikl umuman bajarilmaydi va `maxsulotIshlabChiqarish()` metodi ishga tushirilmaydi.

`while` tsikli shartni tsikl boshida tekshiradi. Agar shart to'g'ri bo'lmasa tsikl ichidagi ifoda umuman bajarilmaydi. Agar tsikl ifodasi shart to'g'ri bo'lmasa xam kamida bir marta bajarilishi kerak bo'lsa `do/while` tsiklidan foydalanish mumkin. Uning umumiy ko'rinishi quyidagicha bo'ladi:

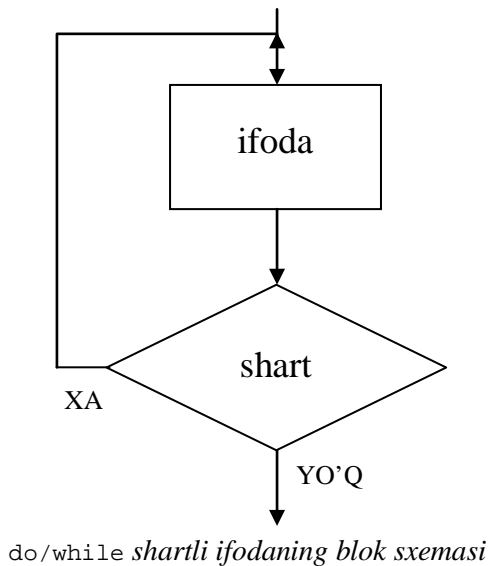
```
do ifoda while (shart);
```

Ushbu tsiklda birinchi ifoda bajariladi keyin esa shart tekshiriladi. Agar shart to'g'ri bo'lsa tsikl qaytariladi, aks xolda tsikl to'xtatiladi. Masalan,

```
do {  
    xaroratniTekshirish();  
    muzlatkichniIshlatish();  
} while (xarorat > 20);
```

Dastur kodi `xaroratniTekshirish()` va `muzlatkichniIshlatish()` metodlarini ishga tushiradi, keyin esa `xarorat` qiymatini 20dan katta-kichikligini tekshiradi. Agar `xarorat` qiymati 20dan katta bo'lsa tsikl qaytariladi, katta bo'lmasa tsikl to'xtatiladi.

`do/while` shartli ifodaning blok sxemasi quyidagi ko'rinishda bo'ladi.



Belgilangan tsikllar

`for` tsikli xisoblagich yoki xar bir iteratsiyada yangilanib turuvchi o'zgaruvchi orqali boshqariladigan iteratsiyalarni tashkil qiladi.

`for` belgilangan tsiklning umumiy ko'rinishi quyidagicha bo'ladi:

```

for (bo'lim1; bo'lim2; bo'lim3)
{
    ifoda;
}

```

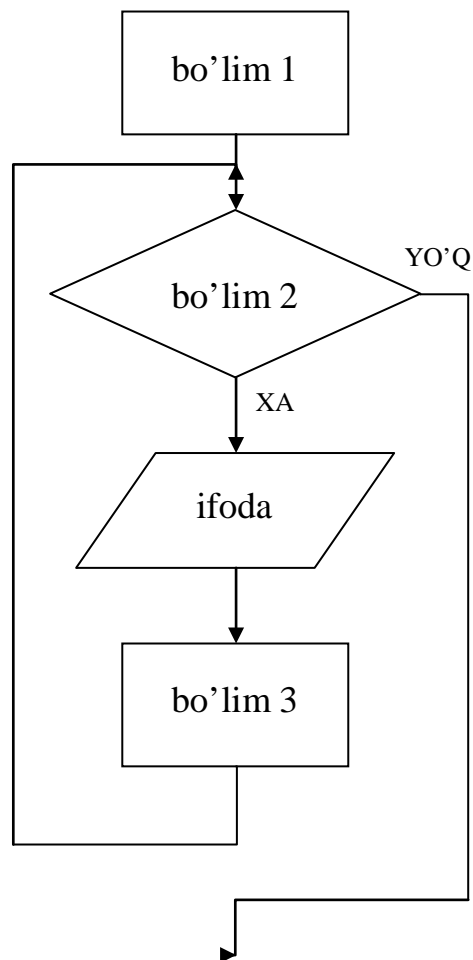
bunda, bo'lim1 xisoblagich o'zgaruvchisini e'lon qiladi va uni boshlang'ich qiymatini belgilaydi. Bo'lim2 tsiklning xar bir iteratsiyasida tekshiriladigan shartni belgilaydi. Bo'lim3 xisoblagichni yangilash qoidasini belgilaydi. Masalan, quyidagi tsikl 1 dan 10 gacha bo'lgan sonlarni ekranga chiqarib beradi.

```

for (int i=1; i<=10; i++)
    System.out.println(i);

```

Quyida for tsiklining blok sxemasi keltirilgan.



for tsiklining blok sxemasi

Shuni aytib o'tish kerakki, for tsiklida e'lon qilingan xisoblagich o'zgaruvchisi faqatgina tsikl doirasida amal qiladi.

Bajarilish tartibini buzuvchi ifodalar

Tsikllar bajarilishi davomida ulardan vaqtidan oldin chiqish yoki keyingi iteratsiyaga o'tish extiyoji paydo bo'ladi. Buning uchun `break` va `continue` ifodalaridan foydalanish mumkin.

`break` ifodasi yorliqsiz va yorliqli turlariga bo'linadi. Yorliqsiz `break` ifodasini qo'llashni quyidagi misolda ko'rib chiqamiz:

```
while (xomAshyo > 0)
{
    maxsulotIshlabChiqarish();
    maxsulot++;
    if (maxsulot > reja) break;
    xomAshyo--;
}
```

Ushbu dasturda tsikl ikki xolatda tugaydi, birinchi, agar `xomAshyo = 0` bo'lganda yoki `maxsulot = reja` bo'lganda. Demak, `break` ifodasi tsikldan shartsiz chiqib ketish uchun qo'llaniladi.

Yorliqli `break` ifodasi bir birida joylashgan tsikllardan chiqish uchun ishlatiladi. Masalan,

```
xafta:
while (kun < 6)
{
    while (soat < 8)
    {
        ishlash();
        soat++;
        if (tatil == true) break xafta;
    }
    kun++;
}
```

Ushbu dastur kodida `tatil == true` sharti to'g'ri bo'lganda `break` ifodasi joylashgan tsikldan emas balki `break` ifodasidan keyin ko'rsatilgan yorliq tsiklidan chiqiladi.

`continue` ifodasi tsiklning joriy iteratsiyasi bajarilishi to'xtatib ushbu tsiklni keyingi iteratsiyasiga o'tkazadi. Masalan,

```
for (int i=1; i<=10; i++)
{
    if (i>5) continue;
    System.out.println(i);
}
```

```
}
```

dastur kodida `i>5` bo'lganda tsiklning joriy iteratsiyasi bajarilishi to'xtatilib keyingi iteratsiyaga o'tiladi. Natijada `i>5` bo'lganda `System.out.println(i)` ifodasi bajarilmaydi.

Massivlar

Massiv bu bir turdagi qiymatlar to'plamini saqlovchi ma'lumot strukturasi. Massivda joylashgan xar bir element o'z indeksiga ega bo'lib ushbu indeksi orqali unga murojaat qilish mumkin.

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

```
int [] a;
```

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

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

Ushbu ifoda 100ta butun sonlarni saqlay oladigan massivni yaratadi.

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

```
int [] a = new int [4];
```

```
a [1] = 0;
```

```
a [2] = 1;
```

```
a [3] = 2;
```

```
a [4] = 3;
```

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

Massivdagi elementlar sonini aniqlash uchun `length` ifodasidan foydalanish mumkin. Masalan,

```
int [] a = new int[4];
```

```
System.out.println(a.length);
```

dastur kodi massiv elementlar soni 4ligini ko'rsatadi.

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

```
int [] a = {0, 1, 2, 3};
```

dastur kodi

```
int [] a = new int [4];
```

```
a [0] = 0;
```

```
a [1] = 1;
```

```
a [2] = 2;
```

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

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

Ko'p o'lchamli massivlar

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

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

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

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

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

```
double [] b1 = {1.2, 1.3, 1.4, 1.5, 1.6};
```

```
double [] b2 = {2.0, 2.1, 2.2, 2.3, 2.4};
```

```
double [] b3 = {1.1, 1.1, 1.1, 1.1, 1.1};
```

```
double [] b4 = {0.1, 0.2, 0.3, 0.4, 0.5};
```

```
balans [0] = b1;
```

```
balans [1] = b2;
```

```
balans [2] = b3;
```

```
balans [3] = b4;
```

Xuddi shu massivni quyidagi qisqa usulida ham yaratib olish mumkin:

```
double [] [] balans = { {1.2, 1.3, 1.4, 1.5, 1.6}, {2.0, 2.1, 2.2, 2.3, 2.4},  
{1.1, 1.1, 1.1, 1.1, 1.1}, {0.1, 0.2, 0.3, 0.4, 0.5} };
```

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

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

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

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

Takrorlash uchun savol va topshiriqlar:

1. `main()` metodi nima vazifani bajaradi?
2. Izohlar qanday simvollar bilan boshlanadi?
3. Identifikatorlar qanday talablar qo'yiladi?
4. Java dasturlash tilida qanday sodda qiymatlar mavjud?
5. Inkrement va dekriment operatorlari qanday ko'rinishga ega?
6. Shartli ifoda operatori xaqida gapirib bering.
7. Qanday tsikl operatorlari mavjud?
8. Massiv deganda nima tushuniladi?
9. Ko'p o'lchamli massivlar xaqida gapirib bering.

OBJEKTLAR

Obyektlarga ixtisoslashgan dasturlashga kirish

Obyektlarga ixtisoslashgan dasturlash xozirgi kunda asosiy dasturlash paradigmasi xisoblanib, 1970 yillarda yaratilgan protsedurali dasturlash texnikasini o'rniga keldi. Obyektlarga ixtisoslashgan dizayn dasturni bir necha to'laqon, bir biri bilan ishlovchi dastur komponentlari, ya'ni obyektlarga bo'lish imkoniyatini birdiradi. Obyektlarga ixtisoslashgan dasturlashni maqsadi mavjud muammoni bir necha kichik, oson xal qilib bo'linadigan muammolarga bo'lish xisoblanadi. Java dasturlash tili to'laqon obyektlarga ixtisoslashgan dasturlash tilidir.

Obyektlarni dizayn qilish uslubi bu dasturni obyektlarga bo'lish uchun mo'ljallangan qoidalar majmui. Ko'pincha bu real dunyo element va xodisalarini dastur komponentlariga bog'lash xisoblanadi. Dasturni qayta ishlatish imkoniyatiga ega bo'lgan obyektlarga bo'lishni turli xil uslublari mavjud.

Obyektlarga ixtisoslashgan dasturlashning unumdorligini asosiy sababi bu xar bir obyektning o'ziga tegishli vazifani bajarish majburiyati mavjudligi. Agar biror obyekt o'zini majburiyati bo'lmagan vazifaga bog'liq bo'lsa u xolda ushbu vazini bajaruvchi obyektga murojaat qilishi kerak. Birinchi obyekt ikkinchi obyektidan vazifani bajarishni so'raydi (Java dasturlash tilida ushbu so'rov metod chaqiruvi orqali amalga oshiriladi).

Klasslar

Klasslar Java dasturining tashkiliy elementi xisoblanadi. Klass metodlarni, o'zgaruvchilarni, izitsializatsiya kodi va boshqa klasslarni o'z ichiga olishi mumkin. U klass na'munalarini, klass strukturasini bajaruvchi obyektlarini, yaratish uchun andoza xisoblanadi. Klass class kalit so'zi orqali e'lon qilinadi. Klassning metod va o'zgaruvchilari uning figurali qavslari ichida, ya'ni tanasida, joylashadi. Masalan:

```
class Avtomobil
{
    int tezligi;

    double narxi;

    String rangi;

    public int getTezligi()
    {
        return tezligi;
    }
}
```

Yuqoridagi Avtomobil klassi uchta o'zgaruvchilarni o'z ichiga oladi, bular tezligi, narxi va rangi. Ushbu klass getTezlik() metodini xam belgilaydi. Avtomobil klassi e'lon qilingandan

so'ng `Avtomobil` obyektini yaratib olish mumkin. Buning uchun quyidagi ifodadan foydalanish mumkin:

```
Avtomobil a;  
  
a = new Avtomobil();
```

Yuqoridagi dastur kodining birinchi qatorida `Avtomobil` turidagi `a` o'zgaruvchisi e'lon qilingan. Ikkinchi qatorda `new` kalit so'zi yordamida `Avtomobil` obyekti yaratilib `a` o'zgaruvchisiga biriktirilgan. `Avtomobil` obyekti yaratilgandan so'ng uning o'zgaruvchilari va metodlariga murojaat qilish mumkin. Masalan:

```
a.tezligi = 200;  
  
int tez = a.getTezligi();
```

Metodlar

Metodlar klass tanasida joylashadi. Ular lokal o'zgaruvchilar va boshqa Java ifodalarini o'z ichiga oladi va ushbu o'zgaruvchilar metod chaqirilganda ishga tushiriladi. Metodlar ularni chaqirgan ifodaga qiymat qaytarishi mumkin. Qaytariladigan qiymat sodda qiymatlar, o'zgaruvchi yoki `void`, xech qanday qiymat qaytarilmaydi, bo'lishi mumkin. Metod e'lon qilinganda qaytariladigan qiymat metod nomidan oldin ko'rsatilishi kerak. Metodlar qiymat qabul qilishi xam mumkin va ushbu qiymat chaqiruvchi tomonidan beriladi. Metod e'lon qilinganda qabul qiladigan qiymat metodni qavslari ichida ko'rsatiladi.

Masalan:

```
class ArifmetikQiymat  
{  
  
    int xQiymat;  
  
    int yQiymat;  
  
    double xisoblash(int x, int y)  
    {  
  
        double arifQiymat = (x + y)/2;  
  
        return arifQiymat;  
  
    }  
  
}
```

Ushbu dasturda `ArifmetikQiymat` klassida ikkita butun sonlarni (`int x` va `int y`) qabul qiluvchi `xisoblash()` metodi e'lon qilingan. Ushbu metod ratsional (`double`) qiymatni `return` kalit so'zi orqali chaqiruvchi ifodaga qiymat sifatida qaytaradi.

`xisoblash()` metodi ichida `arifQiymat` nomli o'zgaruvchi e'lon qilingan. Metod ichida e'lon qilingan o'zgaruvchilar *lokal o'zgaruvchilar* deyiladi. Lokal o'zgaruvchilarni klass o'zgaruvchilardan farqi shundaki, ular vaqtincha bo'lib metod bajarilguncha axamiyatga ega.

Metod bajarilib bo'lingandan so'ng lokal o'zgaruvchilar o'z ahamiyatini yo'qotadi. Bundan tashqari lokal o'zgaruvchilariga metod tashqarisidan murojaat qilib bo'lmaydi.

Java dasturlash tilida bitta klass ichida bir xil nomga ega bir nechta metodlarni e'lon qilish xam mumkin. Ushbu printsip *metodni qayta e'lon qilish* deb nomlanadi. Faqatgina qayta e'lon qilingan metod xar xil son va turdagi argumentlarni qabul qilishi shart. Metod chaqirilganda kompilyator metod qabul qiladigan argumentlariga qarab ushbu metodlar ichidan kerakligini ishga tushiradi. Masalan,

```
public class QabulQilinganQiyamat
{
    public void qiyamat(String s)
    {
        System.out.println("birinchi metod");
        System.out.println("qabul qilingan qiyamat - " + s);
    }
    public void qiyamat(int x)
    {
        System.out.println("ikkinchi metod");
        System.out.println("qabul qilingan qiyamat - " + x);
    }
}
```

dasturida

```
qiyamat("tekshiruv");
```

metodi chaqirilsa dastur ekranga

birinchi metod

qabul qilingan qiyamat - tekshiruv

chiqarib beradi. Agar,

```
qiyamat(5);
```

metodi chaqirilsa dastur ekranga

ikkinchi metod

qabul qilingan qiyamat - 5

chiqarib beradi.

Obyekt konstruktsiyasi

Java dasturlash tilida obyektlar “heap” nomli tizim xotirasida joylashadi. Boshqa dasturlash tillaridan farqli o'laroq Java obyektlarni xotiraga joylashtirish va o'chirishni avtomatik ravishda bajaradi. Yangi obyekt yaratilganda unga xotiradan kerakli joy ajratiladi. Obyekt kerak bo'lmay qolganda u Javaning maxsus “garbage collector” yordamida o'chirilib tashlanadi.

Obyekt `new` operatori yordamida yaratiladi klass andozasi asosida yaratiladi. Masalan, quyidagi klass berilgan bo'lsin:

```
public class Kitob
{
    String kitobNomi = "1001 kecha";

    int varoqlarSoni = 324;

    String muqovasi = "Qattiq";

    public void getKitobNomi()
    {
        return kitobNomi;
    }
}
```

Ushbu klass andozasi asosida yangi obyekt quyidagicha yaratiladi:

```
Kitob k = new Kitob();
```

Ya'ni `Kitob` qiymatini qabul qiluvchi `k` o'zgaruvchisiga `new` operatori yordamida yangi `Kitob` obyektini yaratilib biriktirilgan.

Java dasturlash tilida obyektlar obyekt konstruktori yordamida yaratiladi. Konstruktor klass ichida joylashgan, klass bilan bir xil nomga ega va qiymat qaytarmaydigan maxsus metod xisoblanadi. Ushbu maxsus metod xar doim yangi klass na'munasi, ya'ni obyekt, yaratilganda chaqiriladi. Boshqa metodlar singari konstruktorlar qiymat qabul qilishi va qayta e'lon qilinishi xam mumkin. Masalan,

```
public class Avtomobil
{
    int tezlig;

    public Avtomobil()
    {
        this.tezlig = 200;
    }

    public Avtomobil(int t)
    {

```

```

        this.tezlig = t;
    }
}

```

Dasturda `Avtomobil` ikkita, qiymat qabul qilmaydigan va qiymat qabul qiladigan, konstruktorga ega. Obyekt yaratilayotganida konstruktor argumentiga qarab kerakligi ishga tuhiriladi. Masalan,

```

Avtomobil a = new Avtomobil();

Avtomobil b = new Avtomobil(250);

```

Birinchi xolatda `a` o'zgaruvchiga biriktirilgan `Avtomobil` obyektini tezlik o'zgaruvchisi 200 qiymatga ega bo'ladi. Ikkinchi xolatda `b` o'zgaruvchisiga biriktirilgan `Avtomobil` obyektini tezlik o'zgaruvchisi 250 qiymatiga ega bo'ladi.

Agar klassda xech qanday konstruktor berilmagan bo'lsa kompilyator avtomatik ravishda qiymat qabul qilmaydigan konstruktorni qo'shib beradi. Ya'ni,

```

public class Avtomobil
{
    int tezlik;

    public void getTezlik()
    {
        return tezlik;
    }
}

```

dasturi quyidagi dastur bilan bir xil

```

public class Avtomobil
{
    int tezlik;

    public Avtomobil()
    {
    }

    public void getTezlik()
    {
        return tezlik;
    }
}

```

Obyekt destruktsiyasi

Yuqoridagi bo'limda aytilganidek obyektlar "heap" tizim xotirasida joylashadi. Boshqa manbalar singari tizim xotirasi xam chegaraga ega. Demak, obyektlar bilan ishlash mobaynida xotirada joyni tejash maqsadida keraksiz obyektlarni muntazam o'chirib turish kerak. Aksariyat dasturlash tillarida dasturchi keraksiz obyektlarni ochirishni o'zi nazorat qilishi va amalga oshiruvchi dastur kodini tuzishi kerak. Java dasturlash tili keraksiz obyektlarni ochirishni avtomatik tizimiga ega bo'lib u *keraksiz obyektlarni to'plash tizimi* deyiladi.

Agar obyekt dasturning biron bir o'zgaruvchisi tomonidan ssilkaga ega bo'lmasa ushbu obyekt keraksiz xisoblanadi va keraksiz obyektlarni to'plash tizimi tomonidan o'chiriladi. Ushbu tizim obyektlarni ssilkalarini muntazam tekshirib turadi va ssilkasi qolmagan obyektlarni o'chirish uchun belgilaydi va o'chiradi, o'chirilgan obyektlar egallagan xotira manbalari tizimga qaytariladi.

Masalan,

```
Avtomobil a = new Avtomobil();  
  
a = null;
```

Dastur kodining birinchi qatorida Avtomobil obyekti yaratilib u a o'zgaruvchisi tomonidan ssilka qilinmoqda. Ikkinchi qatorda a o'zgaruvchisi tomonidan Avtomobil obyektiga qilinayotgan ssilka bekor qilingan. Ushbu qator bajarilgandan keyin Avtomobil obyekti ssilkasiz qolib keraksiz obyektga aylanadi va keraksiz obyektlarni to'plash tizimi tomonidan o'chiriladi.

Takrorlash uchun savol va topshiriqlar:

1. Obyektlarga ixtisoslashgan dasturlashning afzalliklari nimalardan iborat?
2. Klass deganda nima tushuniladi?
3. Klass qanday e'lon qilinadi?
4. Metod nima vazifani bajaradi?
5. Metod qayerda e'lon qilinadi?
6. Lokal o'zgaruvchi deganda nima tushuniladi?
7. Obyekt qanday va nima asosida yaratiladi?
8. Obyekt qachon o'chiriladi?

KLASSLARARO BOG'LANISH

Ostki klasslar va irsiyat

Java dasturlash tilida klasslar ierarxiyada joylashadi. `extend` kalit so'zi yordamida bir klass ikkinchi klassni ostki klassi sifatida e'lon qilinishi, ya'ni kengaytirishi mumkin. Ostki klass yuqori klass a'zolarini, ya'ni o'zgaruvchilari va metodlarini, meros qilib oladi va ularni o'zini a'zolari singari ishlatishi mumkin.

Masalan:

```
public class Daraxt
{
    float balandlik;

    public void bargChiqarish()
    {
    }
}

public class MevaliDaraxt extends Daraxt
{
    int mevalarSoni;

    public void mevaSolish()
    {
    }
}
```

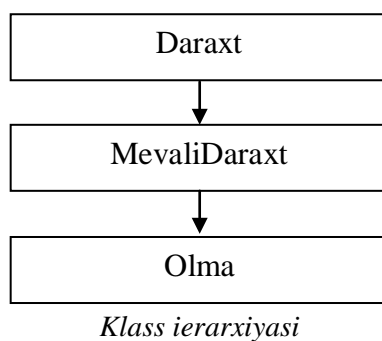
Yuqoridagi dasturda `MevaliDaraxt` klassi `Daraxt` klassini kengaytirib `Daraxt` klassi a'zolarini meros qilib oladi. Endi `MevaliDaraxt` klassi o'zi e'lon qilgan `mevalarSoni` o'zgaruvchisi va `mevaSolish()` metodi va `Daraxt` klassidan meros qilib olgan `balandlik` o'zgaruvchisi va `bargChiqarish()` metodini a'zolari sifatida o'z ichiga oladi.

Klass faqatgina bitta klassni kengaytirishi mumkin. Ostki klassni yana kengaytirish mumkin. Masalan:

```
public class Olma extends MevaliDaraxt
{
    Date gullashVaqti;

    public void gullash()
    {
    }
}
```

Ushbu dasturda Olma klassi MevaliDaraxt va Daraxt klass a'zolarini o'z ichiga oladi. Yuqoridagi klass ierarxiyasi quyidagi rasmda keltirilgan:



Klass biror bir klassni kengaytirs u kengaytirilayotgan klass metodlarini meros qilib oladi. Agar meros qilib olinayotgan metod bajarilishi to'g'ri kelmasa yoki o'zgacha bajarish kerak bo'lsa meros qilib olingan metodni *qayta yaratish* mumkin. Masalan:

```
public class Xisoblagich
{
    public float xisoblash(int a, int b)
    {
        float c = (a+b)/2;
        return c;
    }
}

public class Kalkulyator extends Xisoblagich
{
    public float xisoblash(int a, int b)
    {
        float c = (a*b)/2;
        return c;
    }
}
```

Yuqoridagi dasturda Xisoblagich klassi xisoblash() metodini e'lon qilib, ushbu metod qabul qilgan ikki butun sonni o'rta arifmetik qiymatini xisoblab qaytaradi. Kalkulyator klassi Xisoblagich klassini kengaytirib xisoblash() metodini meros qilib olib uni qayta yaratadi va endi xisoblash() metodi qabul qilgan ikki butun sonni o'rta geometrigini xisoblab beradi.

```
Kalkulyator k = new Kalkulyator();

k.xisoblash(4,5);
```

Dastur kodida xisoblash metodi 10 qiymatini qaytaradi. Shuni ta'kidlab o'tish kerakki metodni qayta yaratayotganda metod qaytarayotgan va qabul qilayotgan qiymat turlari bir xil bo'lishi kerak.

Klass ikkinchi klassni kengaytirganda u kengaytirayotgan klass metodlarini a'zo sifatida meros qilib oladi va meros qilib olgan metodlarni bevosita ishlatishi yoki qayta yaratishi mumkin. Java dasturlash tilida klasslarda `abstrakt` metodlar mavjudki, ularni meros qilib olgan ostki klasslar albatta qayta yaratishi kerak bo'ladi. Abstrakt metodlar oddiy metodlardan farqi shundaki, ularda metod tanasi mavjud bo'lmaydi. Masalan:

```
public void go()

{

}

public abstract void go();
```

Yuqoridagi birinchi metod oddiy metod xisoblanadi. Uni tanasi mavjud bo'lib figurali qavslar bilan belgilanadi. Ikkinchi metod abstrakt metod xisoblanib tanaga ega bo'lmaydi. Bitta abstrakt metodi mavjud klass xam abstrakt klass xisoblanadi. Abstrakt klasslarni abstrakt bo'lmagan metodlari xam mavjud bo'lishi mumkin. Abstrakt klasslar obyekt yaratish uchun andoza xisoblanmaydi, balki ularni kengaytirgan, abstrakt bo'lmagan klasslar uchun meros qilib oladigan metodlarni qayta yaratish shartnomasi xisoblanadi.

Masalan:

```
public abstract class Avtomobil

{

    public abstract void tezlashish();

    public void yurish()

    {

        //metod tanasidagi yurish kodi

        ...

    }

}

public class YukAvtomobili extends Avtomobil

{

    public void tezlashish()

    {

        //yuk avtomobiliga xos tezlashish kodi

        ...

    }

}
```



```

    public void yukTashish()
    {
        // yuk tashish kodi
        ...
    }
}

```

Yuqoridagi dasturda abstrakt `Avtomobil` klassi berilgan bo'lib uning bitta abstrakt va bitta oddiy metodi mavjud. `Avtomobil` klassi uni kengaytiradigan barcha klasslarda (masalan, `YukAvtomobili`, `YengilAvtomobil`) `tezlashish()` metodi mavjud bo'lish shartnomasini taklif etadi. Ya'ni `Avtomobil` klassini kengaytirgan klass `tezlashish()` metodini ushbu klassga xos amalni bajaradigan qilib qayta yaratishi kerak. Yuqoridagi misolda `YukAvtomobili` klassi `tezlashish()` metodini yuk avtomobillarga xos qilib qayta yaratgan. Bundan tashqari ushbu klassda yangi `yukTashish()` metodi xam e'lon qilingan. `YukAvtomobili` jami uchta metodni, ya'ni meros qilib olingan `yurish()` metodi, meros qilib olingan va qayta yaratilgan `tezlashish()` metodi va o'zida e'lon qilingan `yukTashish()` metodi.

Interfeyslar

Java dasturlash tilida *interfeys* deb ushbu interfeysga to'g'ri kelish uchun klasslarga qo'yiladigan talablar to'plami xisoblanadi. Interfeys klass bajarishi kerak bo'lgan metodlar to'plamini e'lon qiladi. Interfeys to'laqon abstrakt klassga, ya'ni barcha metodlari abstrakt bo'lgan klass, o'xshaydi. Interfeysni interface kalit so'zi bilan e'lon qilinadi va interfeysda abstrakt, ya'ni metod tanasi bo'lmagan metodlar e'lon qilinadi. Interfeysda e'lon qilingan barcha metodlar avtomatik `public` va `abstract` xisoblangani uchun ushbu o'zgartiruvchilarni metod nomi oldiga qo'yish shart emas. Masalan:

```

interface Driveable
{
    void dvigatelniYurgizish();
    void dvigatelniTuxtatish();
    float tezlashish(float t);
}

```

Interfeysni bajarish uchun `implements` kalit so'zidan foydalaniladi.

```

public class Mototsikl implements Driveable
{
    public void dvigatelniYurgizish()
    {
        //mototsiklga xos dvigatelni yurgizish kodi
    }
}

```

```

        ...
    }

    public void dvigatelniTuxtatish()
    {
        //mototsiklga xos dvigatelni tuxtatish kodi
        ...
    }

    public float tezlashish(float t)
    {
        //mototsiklga xos tezlashish kodi
        ...
    }
}

```

Yuqoridagi dasturda `Mototsikl` klassi `Driveable` interfeysini `implements` kalit so'zi yordamida bajararmoqda. `Mototsikl` klassi `Driveable` interfeysni uchta metodini ushbu klassga xos ravishda bajararmoqda.

Avval aytib o'tilganidek klass faqatgina bitta klassni kengaytirishi mumkin. Bundan farqli o'laroq, klass bir nechta interfeyslarni bajarishi mumkin. Masalan:

```

public class Mototsikl implements Driveable, Comparable
{
    ...
}

```

Bu xolatda `Mototsikl` klassi `Driveable` va `Comparable` interfeyslarida e'lon qilingan metodlarni bajarishi kerak.

Interfeys klass xisoblanmaydi va ular obyekt yaratish uchun andoza bo'la olmaydi.

Ichki klasslar

Shu vaqtgacha ko'rilgan barcha klasslar yuqori darajadagi klasslar xisoblanadi. Java dasturlash tili klasslarni boshqa klass ichida yoki metod ichida e'lon qilish imkoniyatini beradi. Ushbu klasslar *ichki klasslar* deb ataladi.

Masalan:

```

class Kompyuter{
    class Xotira
    {

```

```

        ...
    }
}

```

Yuqoridagi misolda `Xotira` klassi `Kompyuter` klassini ichki klassi xisoblanadi. Ichki klass, boshqa klass a'zolari singari, u e'lon qilingan qamrov darajasida klass a'zolariga murojaat qilishi mumkin. O'z navbatida ichki klass u e'lon qilingan klass yoki metod a'zosi xisoblanadi. Masalan:

```

class Kompyuter{
    class Xotira{
        ...
    }
    void amalBajarish(){
        ...
    }
}

```

Yuqoridagi misolda `Xotira` ichki klassi `Kompyuter` klassi a'zosi bo'lgan `amalBajarish()` metodini chaqirishi mumkin. Yoki `amalBajarish()` metodi `Xotira` klassi andozasi asosida obyekt yaratishi mumkin.

Boshqa klasslar singari ichki klasslar yuqori darajali klasslar tomonidan murojaat qilinishi xam mumkin. Lekin ushbu murojaatni amalga oshirish uchun murojaat qiluvchi ichki klass joylashgan yuqori darajali klassga ega bo'lishi kerak. Masalan, `Kompyuter` klassi tashqarisidan `Xotira` klassi andozasi asosida obyekt yaratish uchun `Kompyuter` obyektiga ega bolish kerak.

```

Kompyuter k = new Kompyuter();

Kompyuter.Xotira x = k.new Xotira();

```

Yuqoridagi misolning birinchi qatorida `Kompyuter` qiymatini qabul qiluvchi `k` o'zgaruvchisiga yangi `Kompyuter` obyekti yaratilib biriktirilgan. Ikkinchi qatorda `Kompyuter` klassining ichki klassi bo'lgan `Xotira` qiymatini qabulqiluvchi `x` o'zgaruvchisiga yangi `Xotira` obyekti yaratilib biriktirilgan.

Ichki klassni lokal a'zo sifatida metod ichida xam e'lon qilish mumkin. Masalan:

```

class Kompyuter
{
    void amalBajarish()
    {
        class Xotira()
        {
            ...

```

```

    }

}

}

```

Yuqoridagi misolda Xotira klassi amalBajarish() metodining lokal aʼzosi sifatida metodning boshqa aʼzolariga yoki Kompyuter klassining aʼzolariga murojaat qilishi mumkin. Ammo, Kompyuter klassining aʼzolari Xotira klassi metodning lokal aʼzosi boʻlganligi sababli unga murojaat qila olmaydi.

Oʻramlar

Java dasturlash tili klasslarni *package* nomli oʻramlarga guruhlash imkoniyatini beradi. Oʻramlarda dastur fayllarini boshqa kod bibliotekalaridan ajratib saqlash mumkin. Masalan, Java dasturlash tilining bibliotekalari *java.lang*, *java.util*, *javax.swing* kabi oʻramlarda saqlanadi.

Oʻramlarni qoʻllashni asosiy maqsadi bu klass nomini qaytarilmasligini taʼminlashdir. Aytaylik ikki dasturchi bir xil nomga ega, masalan Ishchi, klasslarni yaratdi. Ushbu klasslarni qoʻllash jarayonida ularni nomlari bir xil boʻlganligi sababli xatolik chiqishi mumkin. Buni oldini olish uchun ushbu klasslar xal oʻramlarga joylashtirilishi kerak. Class nomini qaytarilmasligini kafolatlash maqsadida Java dasturlash tili dastur tuzayotgan tashkilotni Internetdagi domen nomini teskari tartibda yozib oʻram nomi sifatida qoʻllashni taklif etadi. Masalan, *java.uz* bizning tashkilotni Internetdagi domen nomimiz boʻlsin. Demak, bizning oʻram nomi *uz.java* boʻladi.

Klassni oʻramga qoshish uchun oʻram nomi birlamchi kod faylini boshiga oʻram nomini yozish kerak. Masalan, *Ishchi* klassini eʼlon qiluvchi *Ishchi.java* birlamchi kod fayli quyidagi koʻrinishga ega boʻladi:

```

package uz.java;

public class Ishchi
{
    ...
}

```

Bundan keyin yaratilgan klass birlamchi kod faylini oʻram nomiga mos direktoriyaga joylashtirish lozim. Yuqoridagi misolda birlamchi kod fayli quyidagi direktoriyaga joylashtiriladi:

```

.
uz\
    java\
        Ishchi.java
        Ishchi.class

```

Klass o'zi joylashgan o'ramni barcha klasslarini va boshqa o'ramlarda joylashgan `public` o'zgartiruvchisi bilan belgilangan klasslarni ishlatishi mumkin. Boshqa o'ramda joylashgan `public` o'zgartiruvchisi bilan belgilangan klasslarga ikki usulda murojlat qilish mumkin. Birinchi usulda, ishlatilayotgan barcha klasslar oldiga o'ramni to'liq nomini qo'shib yozishdir. Masalan:

```
uz.java.Ishchi ishchi = new uz.java.Ishchi();
```

Ushbu usul katta dastur yozishda noqulay xisoblanadi. Osonroq va ko'p qo'llaniladigan usul bu `import` ifodasini qo'llashdir. Ushbu ifodani qo'llashdan maqsad bu o'ramda joylashgan fayllarga murojaat qilishni qisqa usulini berishdir. Masalan:

```
import uz.java.*;
```

Ushbu ifoda `uz.java` o'ramidagi barcha klasslarni `import` qiladi va ularga qisqa nom bilan murojaat qilish imkonini beradi:

```
Ishi ishchi = new Ishchi();
```

Klass a'zolari va metodlarni ko'rinishi

Obyektlarga ixtisoslashgan dezaynni asosiy xususiyatlaridan biri bu ma'lumotni berkitish, ya'ni *enkapsulyatsiya* xisoblanadi. Enkapsulyatsiyaning asosiy maqsadi bu obyekt o'zgaruvchilarini tashqi obyektlarni to'g'ridan to'g'ri murojaatidan berkitish xisoblanadi. Bu bilan obyekt o'zgaruvchilariga noto'g'ri qiymatlarni berilishi oldi olinadi. Masalan, `Avtomobil` obyektini tezlik o'zgaruvchisiga manfiy qiymat berilishi. Enkapsulyatsiyaning yana bir afzalligi bu obyekt bajaradigan amallarini qay tarzda bajarilishini berkitishdir. Masalan, `Telefon` obyektini `aloqaniUratish()` metodi yordamida tarmoq orqali aloqani o'rnatsa, ushbu metod qay tarzda aloqani o'rnatishi (ya'ni qanday tarmoqda, qaysi tarmoq protokolini qo'llab, ma'lumotlarni qay tarzda uzatib) tashqi muxitdan berkitiladi. Boshqa obyektlar `Telefon` obyektini yordamida aloqa o'natishi uchun `aloqaniUratish()` metodini chaqirishi lozim xolos.

Obyekt a'zolarini boshqa obyektlardan yashirish uchun o'zgartiruvchilar qo'llaniladi. Java dasturlash tilida to'rt turdagi o'zgartiruvchilar qo'llaniladi, bular:

1. `private`
2. o'zgartiruvchisiz (default)
3. `protected`
4. `public`

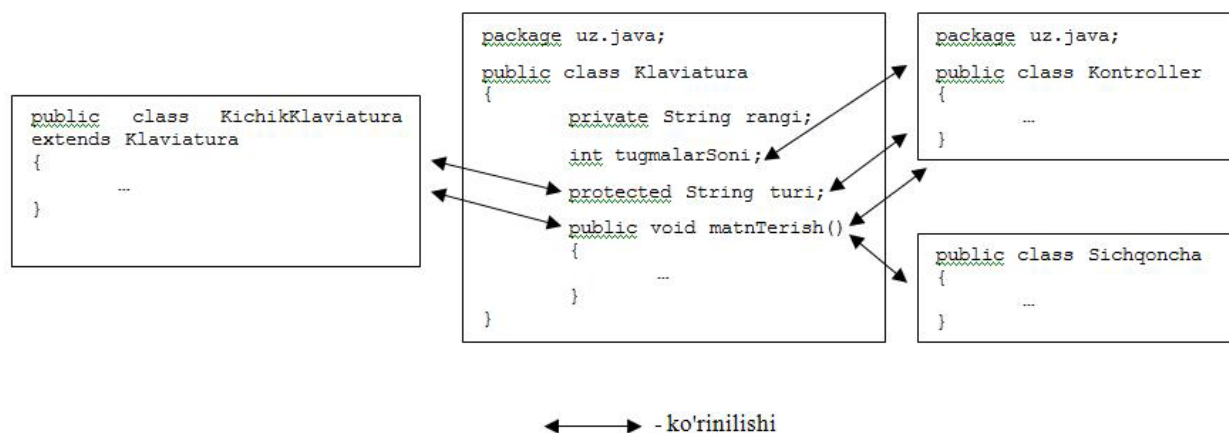
`private` o'zgartiruvchisi klass a'zosini boshqa klasslarga ko'rinmaydigan qilib qo'yadi. `private` o'zgartiruvchili klass a'zosiga faqatgina ushbu klassni boshqa a'zolari murojaat qilishi mumkin.

O'zgartiruvchisi bo'lmagan klass a'zosi klass joylashgan o'ramning boshqa klasslariga ko'rinadi. Ya'ni bunday klass a'zosiga murojaat qila olish uchun klasslar bir o'ramda bo'lishi lozim.

`protected` o'zgartiruvchisi klass a'zosini boshqa o'ramda joylashgan ushbu klassni ostki klassiga va klass joylashgan o'ramning boshqa klasslariga ko'rinarli qiladi.

public o'zgartiruvchisi klass a'zosini barcha klasslarga ko'rinarli qiladi.

Quyidagi rasmda klass a'zolarini o'zgartiruvchisiga qarab ko'rinish darajasi keltirilgan.



Klass a'zolarining o'zgartiruvchilari va ularning a'zo ko'rinishiga ta'siri

Object klassi

Java dasturlash tilida barcha klasslar Object klassini avtomatik ravishda kengaytiradi. Klass Object klassini kengaytirishi uchun ushbu klass nomidan keyin extends Object ifodasini yozish shart emas. Java ushbu ifodani avtomatik ravishda qo'shib qo'yadi.

Object klassi bir necha muxim metodlarga ega. Kengaytirish qoidasiga asosan barcha klasslar Object klassi a'zolarini (jumladan metodlarini) meros qilib oladi. Quyida Object klassini asosiy metodlarini ko'rib chiqamiz.

equals() metodi

Object klassining equals() metodi ikki obyektning tengligini tekshiradi. Metod quyidagi ko'rinishda e'lon qilingan:

```
public boolean equals(Object obj)
{
    ...
}
```

equals() metodi boshqa obyektning qabul qiladi va mantiqiy qiymatni (obyektlar teng bo'lganda true, aks holda false qiymatini) qaytaradi. Object klassida e'lon qilinishi bo'yicha equals() metodi tenglikka tekshirilayotgan obyektlar bir xil obyekt o'zgaruvchilari tomonidan murojaat qilinayotganligi tekshiriladi. Meros qilib olingan metodlarni qayta e'lon qilish imkoniyati mavjud bo'lganligi sababli, equals() metodini xam qayta e'lon qilib olish mumkin.

Masalan, agar ikkita kitob bir xil nom va varoqlar soniga ega bo'lsa ular bir xil xisoblansin. Ushbu me'zonga asosan tenglikni tekshiradigan equals() metodini quyidagicha qayta e'lon qilish mumkin:

```
public class Kitob
```

```

{
    int varoqlarSoni;

    String nomi;

    public boolean equals(Object obj)
    {
        return nomi.equals(obj.name) && varoqlarSoni == obj.VaroqlarSoni;
    }
}

```

Yuqorida qayta e’lon qilingan equals() metodi Kitob obyektlarini tengligini tekshirish uchun qayta e’lon qilingan.

hashCode() metodi

hashCode() metodi joriy obyektни maxsus kodini qaytaradi. Ushbu maxsus kod obyekt mazmuni asosida xisoblanadi. Object klassida e’lon qilinishi bo’yicha hashCode() metodi xar bir obyekt namunasiga, ya’ni bitta klass andozasi asosida yaratilgan obyektga, qaytarilmas sonni maxsus kod sifatida belgilaydi. Maxsus kod obyektlarni to’plamlarda saqlash uchun ishlatiladi.

toString() metodi

toString() metodi obyektни matn qiymatini qaytaradi. Ushbu metod asosan obyekt va uning xolati xaqida ma’lumot berish uchun qo’llaniladi. Masalan:

```

Scooter s = new Scooter();

System.out.println(s.toString());

```

Yuqoridagi dastur kodi quyidagi malumotni ekranga chiqarib beradi:

```
Scooter@42e816
```

Meros qilib olingan boshqa metodlar singari toString() metodini qayta e’lon qilib olish mumkin. Masalan, toString() metodi quyidagicha qayta e’lon qilingan bo’lsin:

```

public class Scooter
{
    public String toString()
    {
        return "Scooter bu kichik mototsikl";
    }
}

```

Bunda,

```

Scooter s = new Scooter();

System.out.println(s.toString());

```

dastur kodi quyidagi ma'lumotni ekranga chiqarib beradi:

```
Scooter bu kichik mototsikl
```

Takrorlash uchun savol va topshiriqlar:

1. Ostki klass deganda nima tushuniladi?
2. Irsiyat degani nima?
3. Biror bir klassning ostki klassi qanday xususiyatga ega bo'ladi?
4. Interfeys deganda nima tushuniladi?
5. Interfeysni bajarish uchun nima qilish kerak?
6. Ichki klass deganda nima tushuniladi?
7. Ichki klass qanday xususiyatga ega?
8. O'ram deganda nima tushuniladi?
9. Qanday ko'rinish o'zgartiruvchilari mavjud va ular qanday vazifani bajaradi?
10. Object klassi qanday klass?

GRAFIK DASTURLASH

Swing

Swing bu Java dasturlash tilining grafikli foydalanuvchi interfeysi qurollar to'plamidir. `javax.swing` o'rami interfeys elementlari xisoblanadigan oynalar, tugmalar, combo qutilari, katalog daraxtlari, jadvallar, menyular va zamonaviy grafikli dastur tuzish uchun kerak bo'ladigan boshqa elementlarni o'z ichiga oladi.

Swing qurollar to'plamidan oldin grafikli foydalanuvchi interfeysini yaratish uchun Java dasturlash tilida `Abstract Window Toolkit (AWT)` nomli qurollar to'plami ishlatilgan. AWT qurollar to'plami grafikli foydalanuvchi elementlarini yaratishni dastur mo'ljallanayotgan platformaning (Windows, Macintosh, Solaris) grafikli foydalanuvchi interfeysi qurollar to'plamiga yo'naltirgan. Natijada xar bir platforma uchun dastur ko'rinishi ushbu platformaga xos bo'lgan. Shu orqali Java dasturlarni portativligi ta'minlangan.

AWT qurollar to'plami qo'llaydigan printsiptni o'ziga xos kamchiliklari xam mavjud. Ushbu qurollar to'plami grafikli foydalanuvchi interfeys elementlarini yaratishni dastur bajarilayotgan platformaga yo'naltirayotganligi sababli xar xil platformalarda interfeys elementlari bajarayotgan amallari farq qiladi. Bundan tashqari, xamma platformalar xam bir xil grafikli interfeys elementlar to'plamiga ega emas. Shu va boshqa kamchiliklarni oldini olish maqsadida grafikli foydalanuvchi interfeys elementlari mustaqil xosil qilish bo'lib ushbu printsipt `Swing` qurollar to'plamida qo'llanilgan.

`Swing` qurollar to'plami yordamida dasturning foydalanuvchi interfeysini yaratishda oldindan tayyorlangan komponentlardan foydalaniladi. Ushbu komponentalar maxsus konteynerlar ichiga joylashtiriladi. Konteynerlar ichida komponentlarni tartibli joylashtirish uchun joylashuv menedjerlar ishlatiladi. Joylashuv menedjerlari foydalanuvchi interfeys elementlarini joylashish strategiyasini belgilab beradi.

Frame yaratish va joylashtirish

Frame Java dasturlash tilining yuqori darajali oynasi xisoblanadi. Ushbu oyna barcha grafikli foydalanuvchi interfeysi elementlarini o'z ichiga oladi. Frame `Swing` o'ramining `JFrame` klassi yordamida yaratiladi. Frame bilan ishlashni quyidagi misol orqali ko'rib chiqamiz.

```
import javax.swing.*;

public class SoddaFrameTest{

    public static void main(String[] args){

        JFrame frame = new JFrame();

        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

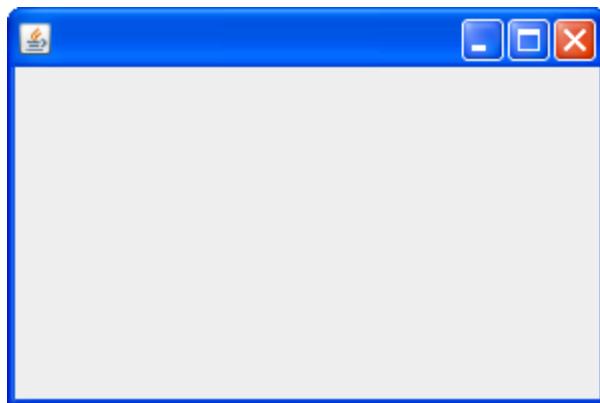
        frame.setSize(300, 200);

        frame.setVisible(true);

    }
```

```
}
```

Yuqoridagi dastur ekranga bo'sh oynani chiqarib beradi.



JFrame yordamida yaratilgan bo'sh oyna

Ushbu dasturni xar bir qatorini ko'rib chiqamiz.

Birinchi qatorda `Swing` o'rami `import javax.swing.*;` ifodasi orqali import qilingan. Keyingi qatorlarda `SoddaFrame` klassi va uning `main()` metodi e'lon qilingan. Oltinchi qatorda oynani xosil qiluvchi `frame` obyektini `Frame` klassi andozasi asosida yaratilgan. Yettinchi qatorda foydalanuvchi ushbu oynani yopganda qanday amal bajarilishi `frame` obyektini `setDefaultCloseOperation()` metodi orqali berilgan. Bizning misolda dasturdan chiqib ketish amali berilgan. Sakkizinchi qatorda oynani yangi o'lchamlari `frame` obyektini `setSize()` metodi orqali berilgan. Yangi `frame` obyektini yaratilganda oyna o'lchami `0x0` piksel bo'ladi. Yangi oyna yaratilganda u avtomatik ravishda ekranda chiqarilmaydi. Buning uchun `frame` obyektini `setVisible()` metodini chaqirish kerak bo'ladi.

`JFrame` klassi aksariyat metodlarini o'zining superklasslaridan meros qilib oladi. Asosiy metodlari quyidagilardan iborat:

- `dispose()` – oynani yopadi va u egallab turgan sistema resurslarini bo'shatadi;
- `setIconImage()` – oynaning sarlavxa qismiga yorliq chiqarib beradi;
- `setTitle()` – sarlavxa qismidagi matnni o'zgartiradi;
- `setResizable()` – oyna o'lchamini foydalanuvchi tomonidan o'zgartirilish imkonini belgilaydi;
- `setLocation()` – oynani ekranda joylashishini o'zgartiradi, metod argument sifatida (`x`, `y`) qiymatlarini qabul qilib `x` qiymati gorizontal va `y` qiymati vertikal koordinata o'qlarini bildiradi. (0, 0) koordinatali nuqta ekranni yuqori-chap qismida joylashgan.
- `setBounds()` – metod argument sifatida (`x`, `y`, kenglik, balandlik) qiymatlarini qabul qilib `x` va `y` qiymatlari oyna joylashish nuqtalarini, kenglik va balandlik qiymatlari oyna kengligi va balandligini belgilaydi.

Quyidagi misolda ushbu metodlardan ba'zilarini qo'llagan xolda oyna yaratib olamiz.

```
import javax.swing.*;

import java.awt.*;

public class SoddaFrame2
```

```

{
    public static void main(String[] args)
    {
        JFrame frame = new JFrame();

        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        frame.setTitle("Sarlavhaga ega oyna");

        Toolkit kit = Toolkit.getDefaultToolkit();

        Image img = kit.getImage("euro.png");

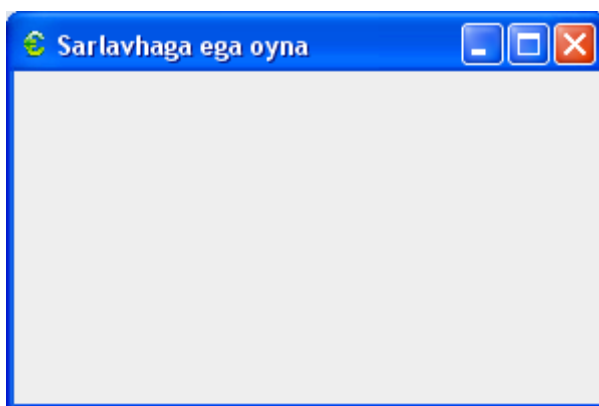
        frame.setIconImage(img);

        frame.setSize(300, 200);

        frame.setVisible(true);
    }
}

```

Dastur quyidagi sarlavha matni va sarlavha yorlig'iga ega oynani ekranga chiqarib beradi:



Sarlavhaga ega oyna

Shuni eslatib o'tish kerakki sarlavha yorlig'i uchun ishlatiladigan rasm fayli dastur klassi bilan bitta papkada joylashishi kerak.

Ma'lumotlarni Panelda ko'rsatish

Java dasturlash tilida Frame boshqa foydalanuvchi interfeysi elementlari uchun konteyner vazifasini bajaradi. Foydalanuvchi interfeysi elementlari Panel elementiga chiziladi keyin esa ushbu Panel Frame oynasiga joylashtiriladi.

Panellar `JPanel` klassi yordamida yaratilib ular quyidagi xususiyatlarga ega:

- ular ustiga elementlar chizish mumkin;
- ular o'z navbatida konteyner xisoblanib boshqa foydalanuvchi interfeysi elementlarni joylashtirishi mumkin.

Panel elementida biron bir rasm joylashtirish uchun JPanel klassini kengaytirib paintComponent() metodini qayta yaratish kerak bo'ladi. paintComponent() metodi JPanel klassini tomonidan JComponent klassidan meros qilib olingan bo'lib ushbu metod Graphics turidagi elementni argument sifatida qabul qiladi. Quyidagi misolni ko'rib chiqamiz:

```
import java.awt.*;

import javax.swing.*;

public class MeningRasmim
{
    public static void main(String[] args)
    {
        JFrame f = new JFrame();

        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        f.setSize(300,200);

        MyPanel p = new MyPanel();

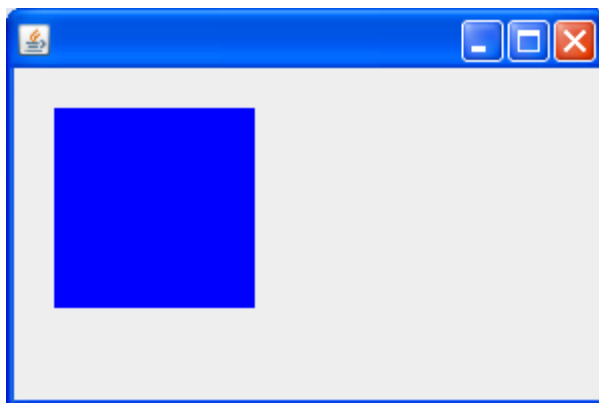
        f.add(p);

        f.setVisible(true);
    }
}

class MyPanel extends JPanel
{
    public void paintComponent(Graphics g)
    {
        g.setColor(Color.orange);

        g.fillRect(20,20,100,100);
    }
}
```

Ushbu dastur ekranga quyidagi oynani chiqarib beradi:



JPanel klassining paintComponent() metodi yordamida chizilgan rasm

MyPanel klassi JPanel klassini kengaytirib meros qilib olingan paintComponent() metodini qayta yaratadi. Graphics elementi yordamida ko'k rangli, 20,20 nuqtada boshlanadigan va 100x100 o'lchamga ega to'rtburchak chizib olinadi. MeningRasmim klassi Frame oynasini yaratadi va MyPanel klassi asosida panel yaratib Framega joylashtiradi.

JPanel klassining paintComponent() metodini qayta yaratib panelga tayyor rasmlarni xam joylashtirish mumkin. Yuqoridagi MeningRasmim dasturda MyPanel klassining paintComponent() metodini quyidagiga o'zgartiramiz:

```
public void paintComponent(Graphics g)
{
    Image rasm = new ImageIcon("sumka.png").getImage();
    g.drawImage(rasm,3,4,this);
}
```

Dastur quyidagi oynani ekranga chiqarib beradi:



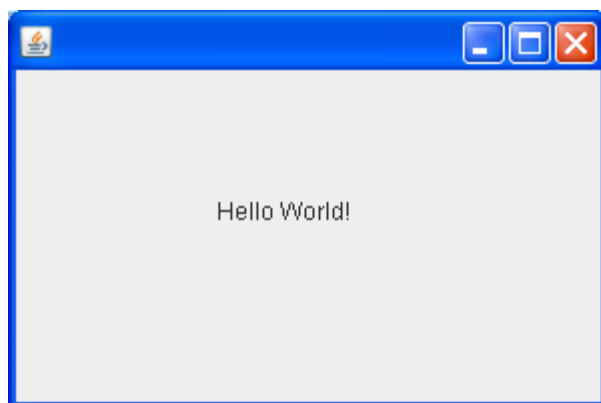
JPanel klassining paintComponent() metodi yordamida aks ettirilgan rasm

JPanel klassining paintComponent() metodi yordamida Panelga matn yozish xam mumkin. Buning uchun MeningRasmim klassida paintCpmponent() metodini quyidagiga o'zgartiramiz:

```
public void paintComponent(Graphics g)
{
    g.drawString("Hello World!",100,75);
}
```

```
}
```

Dastur quyidagi oynani ekranga chiqarib beradi:



JPanel klassining paintComponent() metodi yordamida aks ettirilgan matn

Xodisa va amal

Java dasturlash tilida foydalanuvchi xosil qilgan xodisalarni qabul qiluvchi obyektlar qabul qilib ularga belgilangan amallarni bajaradi. Obyektlarga ixtisoslashgan dasturlash tiliga xos bo'lganidek, xodisa `java.util` o'ramida joylashgan `EventObject` klassi tomonidan yaratiladigan obyekt orqali ifodalanadi. Xar bir xodisa turi uchun o'zining ostki klasslari mavjud, masalan, tugmalar uchun `ActionEvent`, oynalar uchun esa `WindowEvent`. Xodisani xosil qiluvchi foydalanuvchi interfeysi elementi xodisa manbasi deyiladi.

Xodisalarni boshqarish quyidagicha amalga oshiriladi:

1. Xodisani qabul qiluvchi obyekt xodisa nazorat qiluvchi interfeysni bajaradi;
2. Xodisa manbasi xodisani qabul qila oladigan va unga xodisa obyektini yubora oladigan obyektini belgilaydi;
3. Xodisa xosil bo'lganda xodisa manbasi xodisa obyektini belgilangan barcha xodisani qabul qiluvchi obyektlarga jo'natadi;
4. Xodisani qabul qiluvchi obyekt xodisa obyektidagi ma'lumotni bajariladigan amalni xal qilishda ishlatadi.

Xodisani qabul qiluvchi obyektini xodisa manbasiga belgilash misoli quyida keltirilgan:

```
ActionListener listener = new TugmaListener();  
  
JButton button = new JButton("OK");  
  
button.addActionListener(listener);
```

Ushbu misolning birinchi qatorida xodisani qabul qiluvchi obyekt yaratilgan. Ikkinchi qatorda xodisa manbasi yaratilgan. Uchinchi qatorda xodisa manbasiga xodisani qabul qiluvchi obyekt belgilangan. Xodisa xosil qilinganda (ya'ni foydalanuvchi tugmani bosganda) xodisa manbasi xodisani xosil qilib (ushbu xolda `ActionEvent` xodisasini) xodisa qabul qiluvchi obyektga uzatadi. Xodisani qabul qiluvchi obyekt klassi tegishli interfeysni (ushbu xolda `ActionListener` interfeysini) bajarishi kerak bo'ladi. `ActionListener` interfeysini bajarish uchun

xodisani qabul qiluvchi obyekt klassi `ActionEvent` obyektini argument sifatida qabul qiluvchi `actionPerformed()` metodiga ega bo'lishi kerak.

```
public class TugmaListener implements ActionListener
{
    public void actionPerformed(ActionEvent xodisa)
    {
        //bajariladigan amal kodi
    }
}
```

Foydalanuvchi tugmani bosganda `JButton` obyektini `ActionEvent` xodisa obyektini xosil qilib xodisani qabul qiluvchi obyektga uzatadi va `actionPerformed()` metodi chaqirilib undagi amal bajariladi.

Tugmalar va yozuvlar

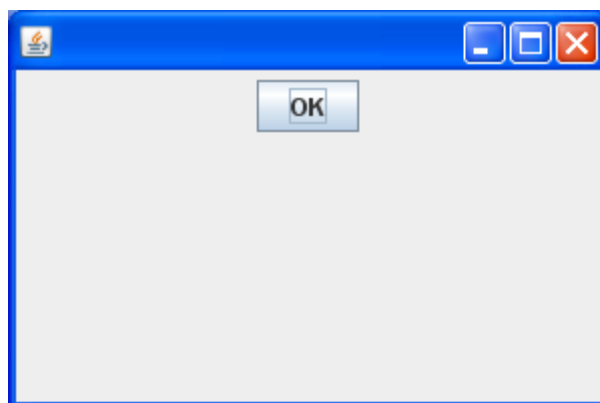
Tugma bu biron bir amalni bajarish uchun foydalanuvchi ruxsatini olish uchun ishlatiladigan element xisoblanadi. Java dasturlash tilida tugmalarni `JButton` klassi andozasi yordamida yaratib olish mumkin. Masalan:

```
JButton button = new JButton();
```

Dastur kodi yangi tugma yaratib uni button o'zgaruvchisiga biriktirib beradi. Yuqoridagi dastur kodi xech qanday yozuvga ega bo'lmagan tugmani yaratib beradi. Ammo, foydalanishni osonlashtirish maqsadida xar bir tugma bajaradigan amaliga asosan nomga ega bo'lishi kerak. Nomga ega tugmani yaratib olish uchun quyidagi dastur kodidan foydalanish mumkin:

```
JButton button = new JButton("OK");
```

Ushbu dastur kodi quyidagi tugmani yaratib beradi



JButton klassi yordamida yaratilgan tugma

`JButton` klassi tugmalarga matndan tashqari rasm joylashtirish imkonini xam beradi. Rasmni fayldan yuklash uchun `ImageIcon` klassidan foydalaniladi. Masalan:

```
import java.awt.*;
```

```
import javax.swing.*;

public class RasmliTugma{

    public static void main(String[] args){

        JFrame frame = new JFrame();

        JPanel panel = new JPanel();

        Icon icon = new ImageIcon("search.png");

        JButton button = new JButton("OK", icon);

        panel.add(button);

        frame.getContentPane().add(panel);

        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

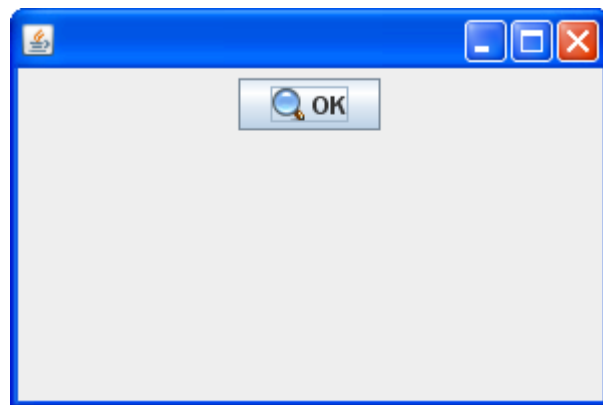
        frame.setSize(300,200);

        frame.setVisible(true);

    }

}
```

Ushbu dastur quyidagi oynani ekranga chiqarib beradi:



JButton klassi yordamida yaratilgan rasmlı tugma

Java dasturlash tilida tugma bosilganda `ActionEvent` xodisasi xosil qilinadi. Ushbu xodisani qabul qilish uchun dastur xodisalarni nazorat qiluvchi `ActionListener` interfeysini bajarishi va uning `actionPerformed()` metodini qayta yaratishi kerak bo'ladi. `actionPerformed()` metodi ichiga tugma bosilganda bajarilishi kerak bo'ladigan amallar kiritiladi. Masalan:

```
import java.awt.*;

import java.awt.event.*;

import javax.swing.*;

public class TugmaAmali
{

    JPanel panel;

    Color backgroundColor = Color.YELLOW;
```



```

JButton button;

String tugmaMatni = "Sariq";

public static void main(String[] args)
{
    TugmaAmali ta = new TugmaAmali();

    ta.go();
}

public void go()
{
    JFrame frame = new JFrame("TugmaAmali");

    JPanel panel = new JPanel();

    panel.setBackground(backgroundColor);

    button = new JButton(tugmaMatni);

    button.addActionListener(new ButtonListener());

    panel.add(button);

    frame.getContentPane().add(panel);

    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    frame.setSize(300,200);

    frame.setVisible(true);
}

public class ButtonListener implements ActionListener
{
    public void actionPerformed(ActionEvent e)
    {
        if(backgroundColor == Color.YELLOW)
        {
            backgroundColor = Color.RED;

            tugmaMatni = "Qizil";
        }else{
            backgroundColor = Color.YELLOW;

            tugmaMatni = "Sariq";
        }

        panel.setBackground(backgroundColor);

        button.setText(tugmaMatni);
    }
}

```

```

    }
}
}

```

Yuqoridagi dastur tugma bosilganda oyna rangi sariq rangdan qizil rangga o'zgaradi.



Tugma xosil qilgan xodisa asosida amal bajarish

Yozuvlar

Dastur oynasiga biron bir yozuvni qo'shish uchun `JLabel` klassidan foydalanish mumkin. Yozuvlar asosan oynada joylashgan elementlarga izoh berish yoki ularni nomlash uchun ishlatiladi. Masalan:

```

import java.awt.*;

import javax.swing.*;

public class YozuvDasturi
{
    public static void main(String[] args)
    {
        JFrame frame = new JFrame("Yozuv Dasturi");

        JLabel label = new JLabel("Yozuv joylashgan oyna");

        JPanel panel = new JPanel();

        panel.add(label);

        frame.getContentPane().add(panel);

        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

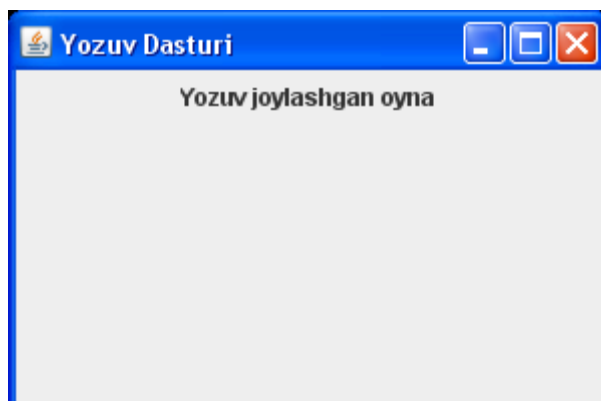
        frame.setSize(300,200);

        frame.setVisible(true);

    }
}

```

Ushbu dastur yozuv joylashgan quyidagi oynani ekranga chiqarib beradi:

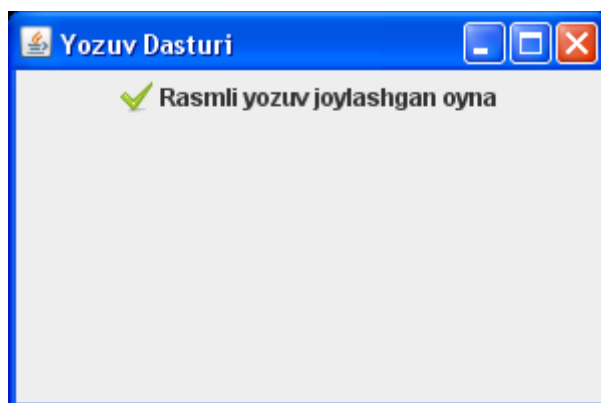


JLabel klassi yordamida yaratilgan yozuv

Yozuvni oldida joylashgan rasm bilan xam yaratish mumkin. Buning uchun ImageIcon klassi yordamida rasmni fayldan import qilib JLabel klassi konstruktoriga uzatish kerak bo'ladi. Masalan, YozuvDasturi dasturga quyidagi o'zgarishlar kiritish kerak:

```
...  
Icon icon = new ImageIcon("note.png");  
  
JLabel label = new JLabel("Rasmlı yozuv joylashgan oyna", icon,  
SwingConstants.CENTER);  
  
...
```

O'zgartirilgan dastur quyidagi oynani ekranga chiqarib beradi:



JLabel klassi yordamida yaratilgan rasmlı yozuv

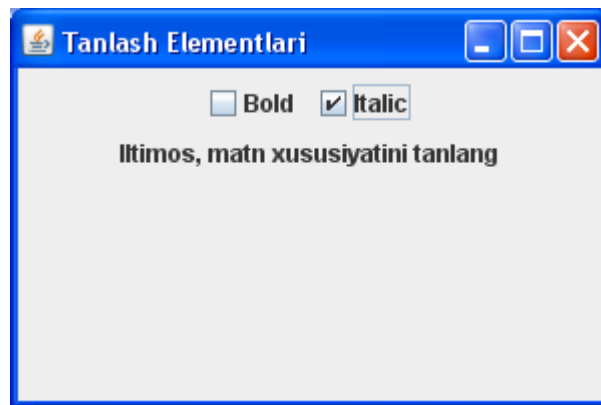
Shuni yodda tutish lozimki, note.png rasm fayli YozuvDasturi klass fayli joylashgan papkada joylashgan bo'lishi kerak.

Yozuvni oynada joylashishi SwingConstants interfeysining konstantalari orqali belgilanadi. Masalan, bizning xolda SwingConstants.CENTER konstantasi yozuvni markazga tekislab joylashtiradi.

Checkbox va Radio tugmalar

Java dasturlash ptili foydalanuvchidan ma'lumot yig'ish uchun bir necha turdagi elementlarni taklif etadi. Checkbox va Radio tugmalar grafikli foydalanuvchi interfeysining tanlash elementlari xisoblanadi va foydalanuvchidan belgilangan variantlar orasidan tanlash imkonini beradi.

Checkbox elementi foydalanuvchiga “xa” yoki “yo’q” variantlarini tanlash imkonini beradi. Checkbox elementi yozuvga ega bo’lib ushbu yozuv Checkbox maqsadini belgilaydi. Foydalanuvchi Checkbox elementini sichqon bilan bosish orqali tanlaydi yoki tanlanishni olib tashlaydi. Checkbox elementi `JCheckBox` klassi yordamida xosil qilinadi. Quyidagi oynada Checkbox elementlari ko’rsatilgan:



JCheckBox klassi yordamida yaratilgan checkbox elementi

Ushbu oynani quyidagi dastur xosil qiladi:

```
import java.awt.*;

import javax.swing.*;

public class TanlashElementlari
{
    public static void main(String[] args)
    {
        TanlashElementlari te = new TanlashElementlari();
        te.go();
    }

    public void go()
    {
        JFrame frame = new JFrame("Tanlash Elementlari");

        JPanel panel = new JPanel( );

        JCheckBox ch1 = new JCheckBox("Bold");

        JCheckBox ch2 = new JCheckBox("Italic");

        JLabel label = new JLabel("Iltimos, matn xususiyatini tanlang");
```

```

        panel.add(ch1);

        panel.add(ch2);

        panel.add(label);

        frame.getContentPane().add(panel);

        frame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );

        frame.setSize(300,200);

        frame.setVisible(true);

    }
}

```

Checkbox elementi tanlanganlik xolatini bilish uchun Checkbox klassini `isSelected()` metodi ishlatiladi. Agar Checkbox tanlangan bo'lsa metod `true` qiymatini, tanlanmagan bo'lsa `false` qiymatini qaytaradi.

Checkbox elementi bosilganda `ActionEvent` xodisasi xosil qilinadi va ushbu xodisani `ActionListener` orqali qabul qilib amal bajarish mumkin. Masalan, quyidagi dastur Checkbox tanlanganda Label elementi ustida amal bajariladi:

```

import java.awt.*;

import java.awt.event.*;

import javax.swing.*;

public class TanlashElementlari
{
    JLabel label;

    JCheckBox ch1;

    JCheckBox ch2;

    public static void main(String[] args)
    {
        TanlashElementlari te = new TanlashElementlari();

        te.go();
    }

    public void go()
    {
        JFrame frame = new JFrame("Tanlash Elementlari");

        JPanel panel = new JPanel( );

        ActionListener Listener = new Listener();

        ch1 = new JCheckBox("Bold");
    }
}

```

```

ch1.addActionListener(listener);

ch2 = new JCheckBox("Italic");

ch2.addActionListener(listener);

label = new JLabel("Iltimos, matn xususiyatini tanlang");

label.setFont(new Font("Arial", Font.PLAIN, 14));

panel.add(ch1);

panel.add(ch2);

panel.add(label);

frame.getContentPane().add(panel);

frame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );

frame.setSize(300,200);

frame.setVisible(true);

}

public class Listener implements ActionListener
{

    public void actionPerformed(ActionEvent e)
    {

        int fm = 0;

        if(ch1.isSelected()) fm = fm + Font.BOLD;

        if(ch2.isSelected()) fm = fm + Font.ITALIC;

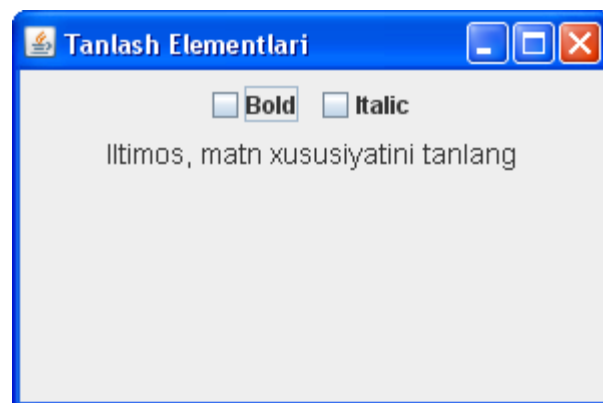
        label.setFont(new Font("Arial", fm, 14));

    }

}
}

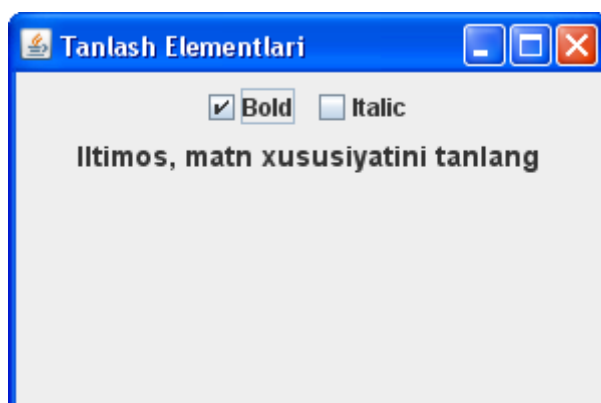
```

Ushbu dastur quyidagi oynani ekranga chiqarib beradi:



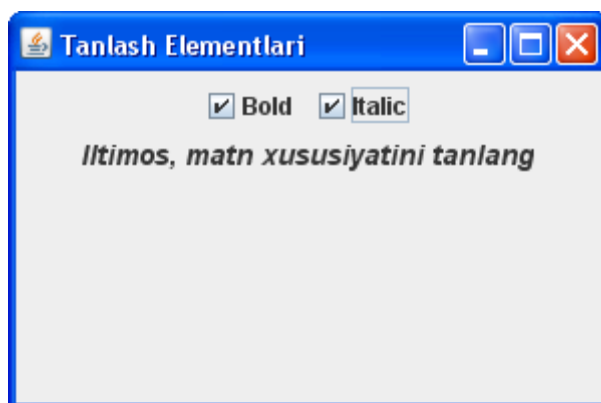
Checkbox elementi va u xosil qilgan xodisa asosida amal bajarish

Bold yozuviga ega Checkbox elementi tanlanganda “Iltimos, matn xususiyatini tanlang” matni qalin yozuvga o’zgaradi:



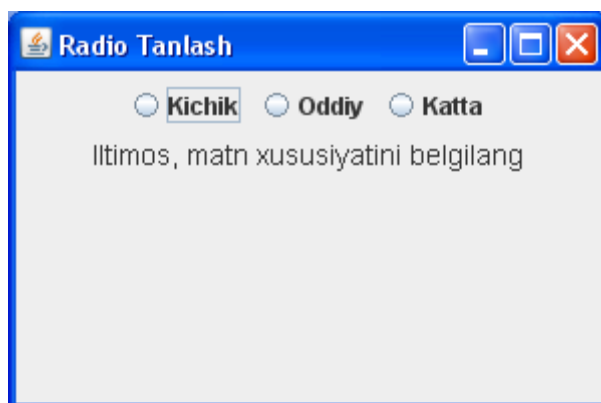
Checkbox elementi va u xosil qilgan xodisa asosida matn ko’rinishini qalinroq qilish

Bold va Italic yozuvlariga ega Checkbox elementlari tanlanganda matn qalin va kursiv yozuvga o’zgaradi:



Checkbox elementi va u xosil qilgan xodisa asosida matn ko’rinishini qalinroq va kursiv qilish

Checkbox elementi birdaniga bir nechta variantlarni tanlash imkonini berish uchun ishlatiladi. Bir nechta variantlardan faqat bittasini tanlash uchun Radio tugmalar ishlatiladi. Bir variant tanlanganda boshqa tanlangan variantlar belgilanishi bekor qilinadi. Buning uchun Radio tugmalar bir guruhda bo’lishi kerak. Radio tugmalar `JRadioButton` klassi yordamida yaratiladi. Radio tugmalar guruhi `ButtonGroup` klassi yordamida xosil qilinadi. Quyidagi oynada Radio tugmalar ko’rsatilgan:



`JRadioButton` klassi yordamida yaratilgan radio tugmalar

Ushbu oyna quyidagi dastur yordamida yaratilgan:

```
import java.awt.*;

import javax.swing.*;

public class RadioTanlash
{
    public static void main(String[] args)
    {
        RadioTanlash rt = new RadioTanlash();

        rt.go();
    }

    public void go()
    {
        JFrame frame = new JFrame("Radio Tanlash");

        JPanel panel = new JPanel( );

        ButtonGroup guruh = new ButtonGroup();

        JRadioButton rb1 = new JRadioButton("Kichik");

        guruh.add(rb1);

        JRadioButton rb2 = new JRadioButton("Oddiy");

        guruh.add(rb2);

        JRadioButton rb3 = new JRadioButton("Katta");

        guruh.add(rb3);

        JLabel label = new JLabel("Iltimos, matn xususiyatini belgilang");

        label.setFont(new Font("Arial", Font.PLAIN, 14));

        panel.add(rb1);

        panel.add(rb2);

        panel.add(rb3);

        panel.add(label);

        frame.getContentPane().add(panel);

        frame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );

        frame.setSize(300,200);

        frame.setVisible(true);
    }
}
```


Radio button element foydalanuvchi tomonidan tanlanganda `ActionEvent` xodisasi xosil qilinadi va `ActionListener` interfeysi yordamida ushbu xodisani qabul qilib biron bir amal bajarib olish mumkin. Masalan, quyidagi dastur tegishli Radio tugmasi tanlanganda oynadagi matn o'lchamini o'zgartirib beradi:

```
import java.awt.*;

import java.awt.event.*;

import javax.swing.*;

public class RadioTanlash
{
    JRadioButton rb1;

    JRadioButton rb2;

    JRadioButton rb3;

    JLabel label;

    public static void main(String[] args)
    {
        RadioTanlash rt = new RadioTanlash();

        rt.go();
    }

    public void go()
    {
        JFrame frame = new JFrame("Radio Tanlash");

        JPanel panel = new JPanel( );

        ActionListener Listener = new Listener();

        ButtonGroup guruh = new ButtonGroup();

        rb1 = new JRadioButton("Kichik");

        rb1.addActionListener(Listener);

        guruh.add(rb1);

        rb2 = new JRadioButton("Oddiy");

        rb2.addActionListener(Listener);

        guruh.add(rb2);

        rb3 = new JRadioButton("Katta");

        rb3.addActionListener(Listener);

        guruh.add(rb3);

        label = new JLabel("Iltimos, matn xususiyatini belgilang");
```

```

label.setFont(new Font("Arial", Font.PLAIN, 14));

panel.add(rb1);

panel.add(rb2);

panel.add(rb3);

panel.add(label);

frame.getContentPane().add(panel);

frame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );

frame.setSize(300,200);

frame.setVisible(true);
}

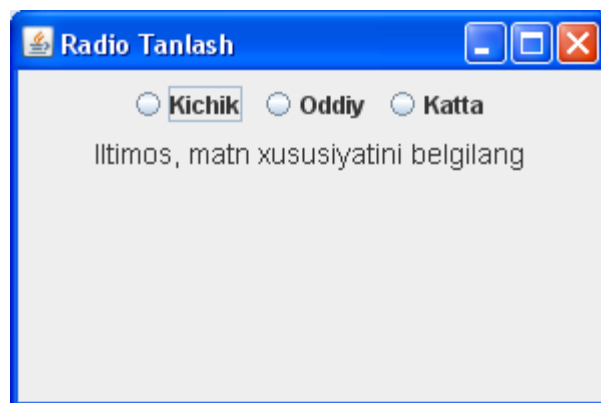
public class Listener implements ActionListener
{
    public void actionPerformed(ActionEvent e)
    {
        if(rb1.isSelected()) label.setFont(new Font("Arial", Font.PLAIN, 12));

        if(rb2.isSelected()) label.setFont(new Font("Arial", Font.PLAIN, 14));

        if(rb3.isSelected()) label.setFont(new Font("Arial", Font.PLAIN, 16));
    }
}
}

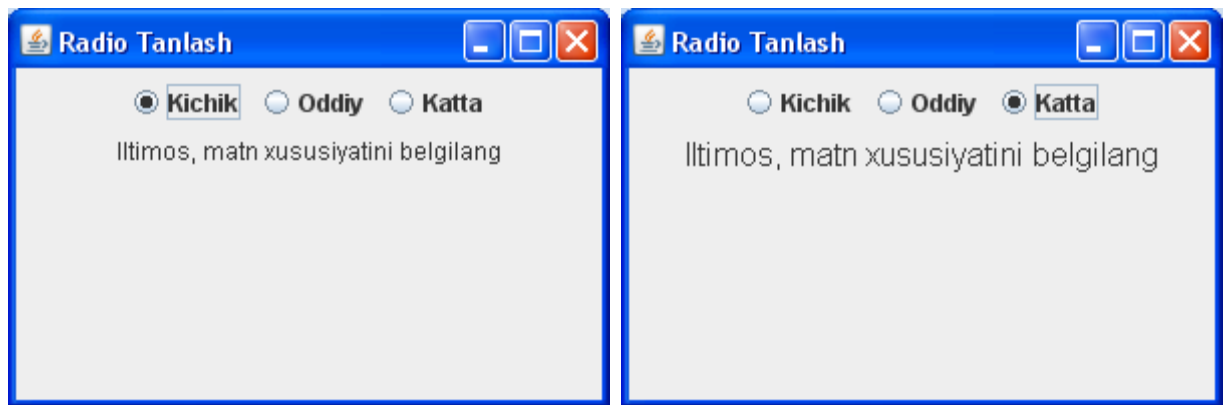
```

Ushbu dastur quyidagi oynani ekrangachiqarib beradi:



Radio tugmalar va u xosil qilgan xodisa asosida amal bajarish

Kichik va Katta yozuvlariga ega Radio tugmalari tanlanganda matn o'lchami o'zgarishi quyidagi oynalarda ko'rsatilgan:



Radio tugma tanlanganda xosil qilingan xodisa asosida matn ko'rinishini o'zgartirish

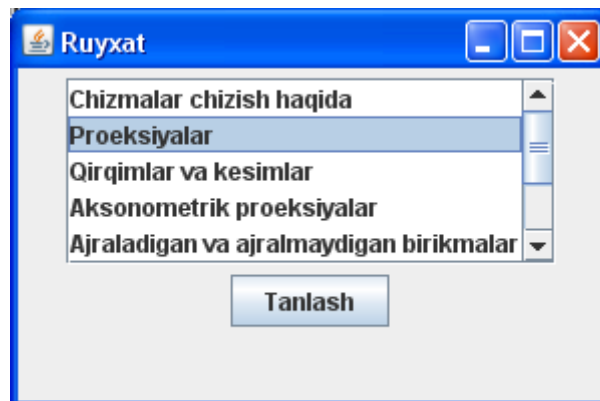
Guruhda bo'lgan Radio tugmalarning bir vaqtning o'zida faqatgina bittasi tanlangan bo'lishi mumkin.

Ro'yxat va Combobox

Royxat va Combobox tanlash elementlari xisoblanib, ko'p variantlar orasidan tanlash kerak bo'lganda oynada joyni tejash uchun ishlatiladi.

Ro'yxat `JList` klassi tomonidan yaratiladi va foydalanuvchiga belgilangan variantlardan tanlash imkonini beradi. Ro'yxatda faqatgina bitta variantni tanlash yoki bir necha variantlarni tanlash imkoniyati mavjud. Ro'yxat yaratishda variantlarni ifoda etuvchi ma'lumotlar modeli ishlatiladi. Ushbu modelni massiv yordamida ifoda etish mumkin.

Quyidagi misolda ro'yxat joylashgan oyna keltirilgan:



JList klassi yordamida yaratilgan ro'yxat

Ushbu oyna quyidagi dastur yordamida yaratilgan:

```
import java.awt.*;

import javax.swing.*;

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

```

JFrame frame = new JFrame("Ruyxat");

JPanel panel = new JPanel();

String [] malumotModeli = {"Chizmalar chizish haqida", "Proeksiyalar",
"Qirqimlar va kesimlar", "Aksonometrik proeksiyalar", "Ajraladigan va ajralmaydigan
birikmalar", "Eskizlar va texnik rasm", "Mashinasozlik chizmalari"};

JList list = new JList(malumotModeli);

list.setVisibleRowCount(5);

JScrollPane scrollPane = new JScrollPane();

scrollPane.setViewportView(list);

JButton button = new JButton("Tanlash");

panel.add(scrollPane);

panel.add(button);

frame.getContentPane().add(panel);

frame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );

frame.setSize(300,200);

frame.setVisible(true);

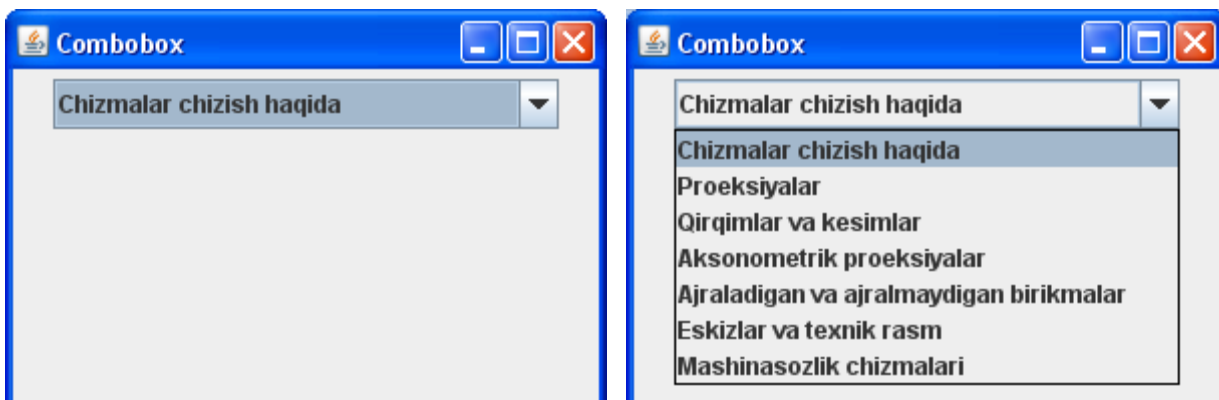
}

}

```

Ushbu dasturda ro'yxat yaratishda ma'lumotlar modeli `maluotModeli` massivi yordamida ifoda etilgan. `JList` klassini `setVisibleRowCount()` metodi ro'yxatni joriy ko'rinadigan qatorlar sonini belgilash uchun ishlatiladi. `JScrollPane` klassi ro'yxatni boshqa elementlarini ko'rib olish uchun ro'yxatga gorizontaal o'tkazgich biriktirib beradi. Tanlangan elementni indeksi yoki qiymatini olish uchun `JList` klassini `getSelectedIndex()` yoki `getSelectedValue()` metodlaridan foydalanish mumkin.

Combobox elementi ro'yxatdan farqli o'laroq tanlash variantlaridan faqatgina bittasini ko'rsatib beradi. Qolgan variantlarni ko'rib olish uchun Combobox yonboshida joylashgan tugmaga bosish kerak bo'ladi. Combobox `JComboBox` klassi yordamida yaratiladi. Quyidagi misolda Combobox joylashgan oyna va Combobox ruyxatini ochilgan xolati keltirilgan:



JComboBox klassi yordamida yaratilgan combobox elementi

Ushbu oyna quyidagi dastur yordamida yaratilgan:

```
import java.awt.*;

import javax.swing.*;

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

        JFrame frame = new JFrame("Combobox");

        JPanel panel = new JPanel();

        String [] malumotModeli = {"Chizmalar chizish haqida", "Proeksiyalar",
        "Qirqimlar va kesimlar", "Aksonometrik proeksiyalar", "Ajralladigan va ajralmaydigan
        birikmalar", "Eskizlar va texnik rasm", "Mashinasozlik chizmalari"};

        JComboBox combo = new JComboBox(malumotModeli);

        panel.add(combo);

        frame.getContentPane().add(panel);

        frame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );

        frame.setSize(300,200);

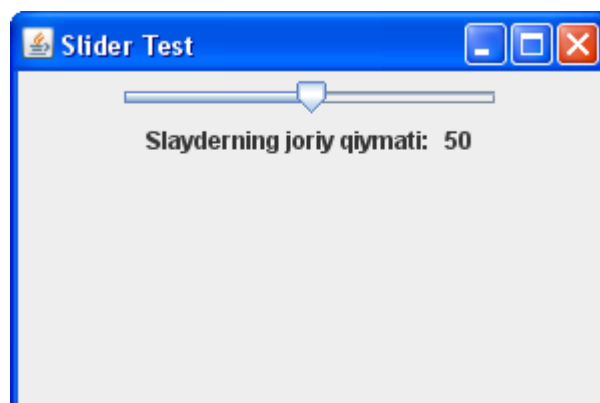
        frame.setVisible(true);

    }
}
```

Tanlangan elementni yoki uning indeksini olish uchun `JComboBox` klassini `getSelectedItem()` yoki `getSelectedIndex()` metodlaridan foydalanish mumkin.

Slayder va Spinner

Slayder elementi berilgan minimal va maksimal qiymatlar orasidan qiymat tanlash imkonini beradi. Slayder `JSlider` klassi yordamida yaratiladi. Quyidagi misolda Slayder joylashgan oyna keltirilgan:



JSlider klassi yordamida yaratilgan slayder elementi

Ushbu oyna quyidagi dastur yordamida yaratilgan:

```
import java.awt.*;

import javax.swing.*;

import javax.swing.event.*;

public class SliderTest
{
    JLabel qiymat;

    JSlider slider;

    public static void main(String[] args)
    {
        SliderTest st = new SliderTest();

        st.go();
    }

    public void go()
    {
        JFrame frame = new JFrame("Slider Test");

        JPanel panel = new JPanel();

        slider = new JSlider(0, 100, 50);

        slider.addChangeListener(new sliderChange());

        JLabel label = new JLabel("Slaydarning joriy qiymati: ");

        qiymat = new JLabel("" + slider.getValue());

        panel.add(slider);

        panel.add(label);

        panel.add(qiymat);

        frame.getContentPane().add(panel);

        frame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );

        frame.setSize(300,200);

        frame.setVisible(true);
    }

    public class sliderChange implements ChangeListener
    {
        public void stateChanged(ChangeEvent e)
        {

```

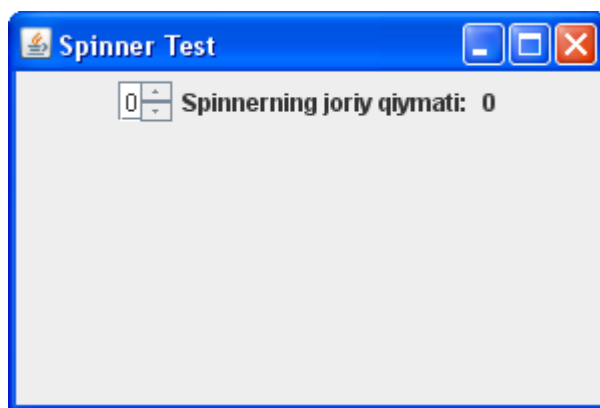
```

        qiymat.setText("" + slider.getValue());
    }
}
}

```

Slayder yurgichi xolati o'zgartirilganda `ChangeEvent` xodisasi xosil qilinadi va ushbu xodisani `ChangeListener` interfeysini bajarib qabul qilish va amal bajarish mumkin. Slayderni joriy qiymatini olish uchun `JSlider` klassini `getValue()` metodidan foydalanish mumkin.

Spinner elementi qiymat tanlash uchun ishlatilib, Slayder elementidan farqli o'laroq minimal va maksimal qiymatlari bilan chegaralanmagan. Element yonboshida joylashgan yuqoriga va pastga qaragan strelkalar bilan Spinner qiymatini oshirish yoki kamaytirish mumkin. Spinner elementi `JSpinner` klassi yordamida yaratiladi. Quyidagi misolda Spinner joylashgan oyna keltirilgan:



JSpinner klassi yordamida yaratilgan spinner elementi

Ushbu oyna quyidagi dastur yordamida yaratilgan:

```

import java.awt.*;

import javax.swing.*;

import javax.swing.event.*;

public class SpinnerTest{

    JLabel qiymat;

    JSpinner spinner;

    public static void main(String[] args)

    {

        SpinnerTest st = new SpinnerTest();

        st.go();

    }

    public void go()

    {

```

```

JFrame frame = new JFrame("Spinner Test");

JPanel panel = new JPanel();

spinner = new JSpinner();

spinner.addChangeListener(new spinnerChange());

JLabel label = new JLabel("Spinnerning joriy qiymati: ");

qiymat = new JLabel("" + spinner.getValue());

panel.add(spinner);

panel.add(label);

panel.add(qiymat);

frame.getContentPane().add(panel);

frame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );

frame.setSize(300,200);

frame.setVisible(true);

}

public class spinnerChange implements ChangeListener

{

    public void stateChanged(ChangeEvent e)

    {

        qiymat.setText("" + spinner.getValue());

    }

}

}

```

Spinner elementi belgilangan elementlar orasidan tanlash imkonini xam beradi. Buning uchun Spinner konstruktoriga ma'lumotlar modelini uzatish mumkin. Ma'lumotlar modeli massiv yordamida yaratiladi. Masalan:

```

String[] qiymatlar = {"sariq", "qizil", "oq", "qora", "zangor"};

SpinnerListModel model = new SpinnerListModel(qiymatlar);

JSpinner spinner = new JSpinner(model);

```

Matn kiritish

Foydalanuvchi matn kiritishi uchun matn qatori va matn maydoni elementlari ishlatiladi. Java dasturlash tilidi matn kiritish uchun `JTextField` va `JTextArea` klasslari mavjud. `JTextField` klassi bitta qatordan iborat matnni, `JTextArea` klassi bir nechta qatordan iborat matnni qabul qiladi.

Matn qatorini xosil qilish uchun quyidagi dastur kodidan foydalanish mumkin:

```
JTextField matnQatori = new JTextField("Kiritiladigan matn", 20);
```

Ushbu dastur kodi yangi matn qatorini yaratib unga “Kiritiladigan matn” matnini qo’shib beradi. `JTextField` konstruktoriga uzatilayotgan ikkinchi qiyat matn qatorini kengligini bildiradi, ya’ni bizning xolda matn qatori 20 simvol kenglikka ega.

Agar matn kiritish uchun foydalanuvchiga bo’sh matn qatori berish kerak bo’lsa quyidagi dastur kodidan foydalanish mumkin:

```
JTextField matnQatori = new JTextField(20);
```

Bu xolda 20 simvol kenglikka ega bo’sh matn qatori yaratiladi.

Matn qatoridagi matnni o’zgartirish uchun `setText()` metodidan, foydalanuvchi kiritgan matnni olish uchun `getText()` metodidan foydalanish mumkin.

```
matnQatori.setText("Yangi matn");  
  
String matn = matnQatori.getText();
```

Foydalanuvchi kiritadigan matn bir qatordan ortiq bo’lsa matn qatori o’rniga matn maydoni ishlatiladi. Matn maydoni `JTextArea` klassi tomonidan yaratiladi. Matn maydoniga foydalanuvchi istalgan sondagi matn qatorini kiritishi mumkin. Matn maydoni quyidagi dastur kodi orqali yaratiladi:

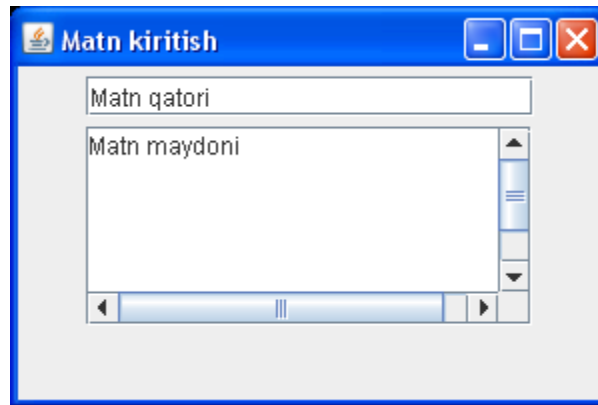
```
JTextArea matnMaydoni = new JTextArea(6, 20);
```

Ushbu dastur kodi 6 qator va 20 simvol uzunlikdagi bo’sh matn maydonini xosil qiladi. Oldindan kiritilgan matn maydonini xosil qilish uchun quyidagi dastur kodidan foydalanish mumkin:

```
JTextArea matnMaudoni = new JTextArea("Oldindan kiritilgan matn", 6, 20);
```

Matn qatori singari matn maydonidagi matnni o’zgartirish uchun `setText()` metodidan va foydalanuvchi kiritgan matnni olish uchun `getText()` metodidan foydalaniladi. Agar foydalanuvchi kiritayotgan matn qatorga sig’masa qolgan matnni keyingi qatorga avtomatik ravishda o’tkazish uchun `setLineWrap()` metodidan foydalanish mumkin. Biron bir matnni maydondagi matn oxiriga qo’shish uchun `append()` metodidan foydalanish mumkin. Matn maydonida joylashgan barcha matnni ko’rib olish uchun `JScrollPane` klassidan foydalanib maydon chetlariga gorizontal va vertikal surgichlar joylashtirib olish mumkin.

Quyidagi misolda matn qatori va matn maydoni joylashgan oyna keltirilgan:



TextField va JTextArea klasslari yordamida yaratilgan matn qatori va matn maydoni

Ushbu oyna quyidagi dastur yordamida yaratilgan:

```
import java.awt.*;

import javax.swing.*;

public class MatnKiritish
{
    public static void main(String[] args)
    {
        JFrame frame = new JFrame("Matn kiritish");

        JPanel panel = new JPanel();

        JTextField matnQatori = new JTextField("Matn qatori", 20);

        JTextArea matnMaydoni = new JTextArea("Matn maydoni", 6, 20);

        JScrollPane scrollPane = new JScrollPane();

        scrollPane.setViewportView().setView(matnMaydoni);

        panel.add(matnQatori);

        panel.add(scrollPane);

        frame.getContentPane().add(panel);

        frame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );

        frame.setSize(300,200);

        frame.setVisible(true);
    }
}
```

Menyular

Java dasturlash tili oynaga foydalanuvchi interfeysi elementi xisoblanadigan menyular joylashtirish imkoniyatini beradi. Menyu xosil qilish uchun birinchi navbata menyular ro'yxati joylashgan menyu qatori yaratiladi. Menyu qatori `JMenuBar` klassi tomonidan yaratiladi.

```
JMenuBar menyuQatori = new JMenuBar();
```

Menyu elementi `JMenu` klassi yordamida yaratilib `JMenuBar` klassining `add()` metodi yordamida menyu qatoriga qo'shiladi.

```
JMenu faylMenyusi = new JMenu("Fayl");
```

```
menyuQatori.add(faylMenyusi);
```

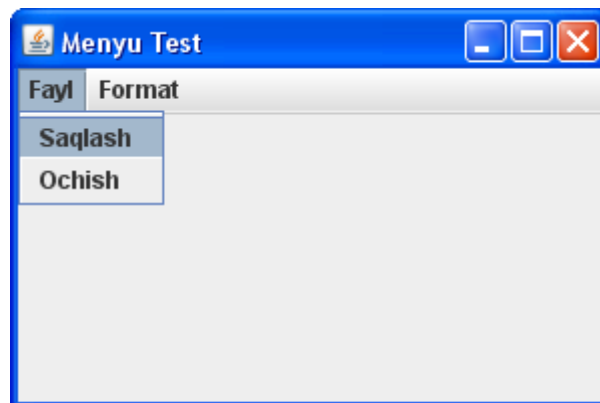
So'ngra, menyu elementlari `JMenuItem` klassi yordamida yaratiladi va `JMenu` klassining `add()` metodi yordamida menyuga qo'shiladi.

```
JMenuItem saqlashItem = new JMenuItem("Saqlash");
```

```
faylMenyusi.add(saqlashItem);
```

Menyu elementlari bosilganda `ActionEvent` xodisasi xosil qilinib uni `ActionListener` interfeysi yordamida qabul qilib ushbu menyu elementiga tegishli amal bajarish mumkin.

Quyidagi misolda menyu qatori joylashgan oyna keltirilgan:



JMenuBar, JMenu va JMenuItem klasslari yordamida yaratilgan menyu

Ushbu oyna quyidagi dastur yordamida yaratilgan:

```
import java.awt.*;

import javax.swing.*;

public class MenyuTest
{
    public static void main(String[] args)
    {
        JFrame frame = new JFrame("Menyu Test");

        JMenuBar menyuQatori = new JMenuBar();
```

```

JMenu faylMenyusi = new JMenu("Fayl");

menuQatori.add(faylMenyusi);

JMenuItem saqlashItem = new JMenuItem("Saqlash");

faylMenyusi.add(saqlashItem);

JMenuItem ochishItem = new JMenuItem("Ochish");

faylMenyusi.add(ochishItem);

JMenu formatMenyusi = new JMenu("Format");

menuQatori.add(formatMenyusi);

JMenuItem shriftItem = new JMenuItem("Shrift");

formatMenyusi.add(shriftItem);

frame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );

frame.setSize(300,200);

frame.setJMenuBar(menuQatori);

frame.setVisible(true);

}

}

```

Menyu elementlari tugmalar singari yozuvga qo'shimcha rasmga xam ega bo'lishi. Buning uchun `JMenuItem` klassining quyidagi dastur kodidan foydalanish mumkin:

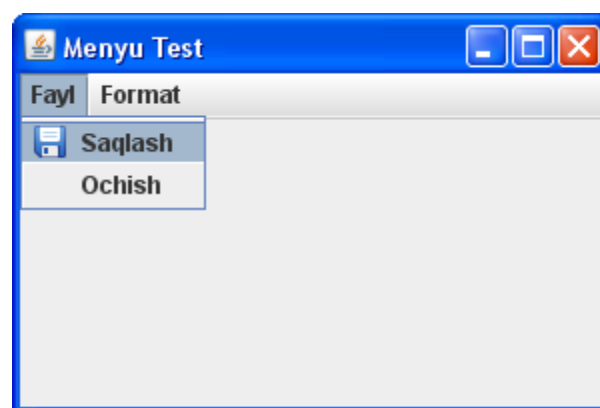
```

Icon saqlashIcon = new ImageIcon("saqlash.png");

JMenuItem saqlashItem = new JMenuItem("Saqlash", saqlashIcon);

```

`MenyuTest` dasturiga yuqoridagi o'zgarishni kiritib quyidagi o'zgarishni olamiz:



Rasmli menyu elementi

Shuni yodda tutish kerakki, `saqlash.png` fayli `MenyuTest` dasturi joylashgan papkada joylashisji kerak.

Menyuga uning elementi sifatida boshqa menyuni ostki menyu sifatida joylashtirish mumkin. Masalan, `MenyuTest` dasturida yangi Menyu yaratib uni Fayl menyusiga ostki menyu sifatida qo'shamiz.

```
JMenu importMenu = new JMenu("Import");

JMenuItem xujjatItem = new JMenuItem("Xujjatni import qilish");

importMenu.add(xujjatItem);

faylMenyusi.add(importMenu);
```

MenuTest dasturiga ushbu o'zgarishlarni kiritib quyidagini olamiz:

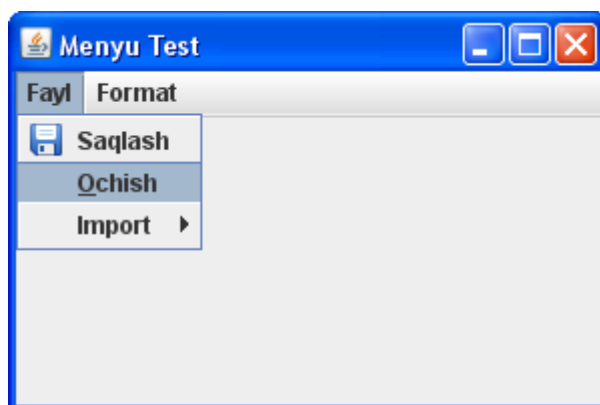


Ostki meyuga ega menyu

Foydalanuvchi menu elementlarini faqatgina sichqon yordamida emas, balki klaviatura yordamida tanlashi mumkin. Buning uchun menu elementlariga tegishli klaviatura qisqartmalarini biriktirish kerak. JMenuItem klassining quyidagi konstruktoridan foydalanish mumkin:

```
JMenuItem ochishItem = new JMenuItem("Ochish", 'O');
```

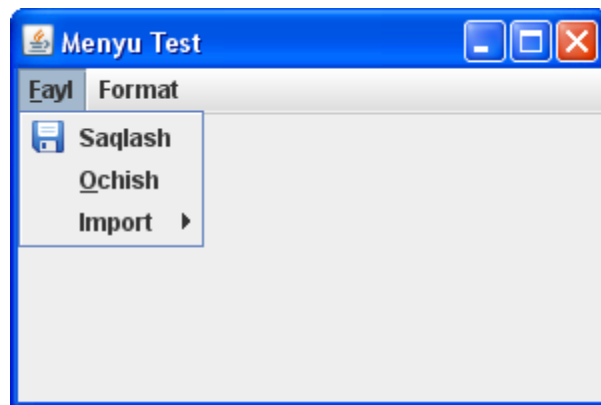
Natijada menu elementining konstruktorning ikkinchi parametrda berilgan xarfi tagiga chizilib qo'yoladi va ushbu menu tanlanishi uchun klaviaturada Alt va tagiga chizilib qo'yilgan xarf bosiladi.



Klaviatura qisqartmasiga ega menu elementi

Menu elementidan tashqari klaviatura qisqartmasini menyuning o'ziga xam belgilash mumkin. Buning uchun JMenu klassining setMnemonic() metodidan foydalanish mumkin.

```
faylMenyusi.setMnemonic('F');
```



Klaviatura qisqartmasiga ega menyu

Endi `MenyuTest` dasturida Fayl menyusining Ochish elementini tanlash uchun `Alt+F` va `Alt+O` tugmalar kombinatsiyasidan foydalanish mumkin.

Qurollar paneli

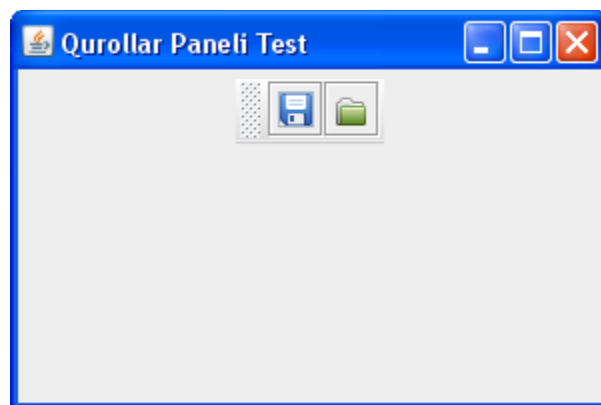
Qurollar paneli dasturning ko'p ishlatiladigan ammallarini tezda ishga tushirish imkonini beruvchi tugmalar to'plamidir. Qurollar paneli `JToolBar` klassi yordamida yaratiladi.

```
JToolBar qurollarPaneli = new JToolBar();
```

Qurollar paneliga tugma joylashtirish uchun `JToolBar` klassining `add()` metodidan foydalaniladi.

```
Icon saqlashIcon = new ImageIcon("saqlash.png");
JButton saqlashButton = new JButton(saqlashIcon);
qurollarPaneli.add(saqlashButton);
```

Quyidagi oynada qurollar paneli joylashgan oyna keltirilgan.



JToolBar klassi yordamida yaratilgan qurollar paneli

Ushbu oyna quyidagi dastur tomonidan yaratilgan:

```
import java.awt.*;
import javax.swing.*;

public class QurollarPaneliTest
```

```

{
    public static void main(String[] args)
    {
        JFrame frame = new JFrame("Qurollar Paneli Test");

        JPanel panel = new JPanel();

        JToolBar qurollarPaneli = new JToolBar();

        Icon saqlashIcon = new ImageIcon("saqlash.png");

        JButton saqlashButton = new JButton(saqlashIcon);

        Icon ochishIcon = new ImageIcon("ochish.png");

        JButton ochishButton = new JButton(ochishIcon);

        qurollarPaneli.add(saqlashButton);

        qurollarPaneli.add(ochishButton);

        panel.add(qurollarPaneli);

        frame.getContentPane().add(panel);

        frame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );

        frame.setSize(300,200);

        frame.setVisible(true);

    }
}

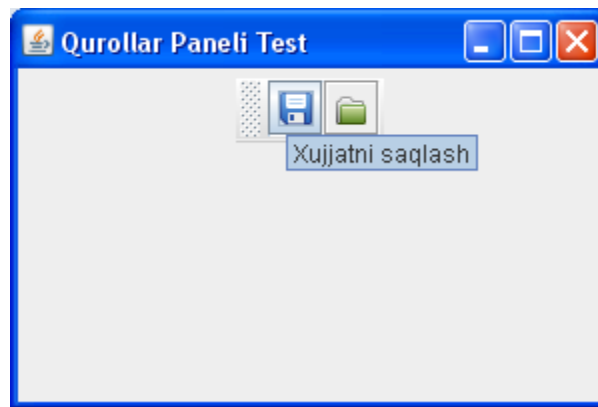
```

Qurollar panelining boshqa foydalanuvchi interfeysi elementlaridan farqi shundaki uni oyna bo'ylab istalgan joyga siljitish mumkin.

Qurollar panelining kamchiligi shundaki unda joylashgan tugmalar yozuvi odatda faqatgina rasmdan iborat va foydalanuvchi qaysi rasm qanday amalga tegishlilikini eslab qolishi qiyin. Ushbu kamchilikni oldini olish uchun tugmalarga biriktirilgan izohlardan foydalanish mumkin. Tugmaga biriktirilgan izoh sichqon ko'rsatkichi ushbu tugma ustiga joylashtirilganda paydo bo'ladi. Tugalarga biriktirilgan izohlarni `JButton` klassining `meros` qilib olingan `setToolTipText()` metodi yordamida biriktirish mumkin.

```
saqlashButton.setToolTipText("Xujjatni saqlash");
```

Tugmalarga biriktirilgan izoh quyidagi oynada ko'rsatilgan:



Tugmaga biriktirilgan izoh

Kontekst menyu

Kontekst menyu bu menyu qatoriga biriktirilmagan va foydalanuvchi tegishli sichqon tugmasini bosganda sichqon ko'rsatkichi joylashgan joyda paydo bo'luvchi menyusi xisoblanadi. Kontekst menyu `JPopupMenu` klassi tomonidan yaratiladi. Kontekst menyu oddiy menyu singari yaratiladi. Birinchi navbatda menyuning o'zi yaratilib keyin menyu elementlari yaratilib menyuga qo'shiladi.

```
JPopupMenu kontekstMenu = new JPopupMenu();

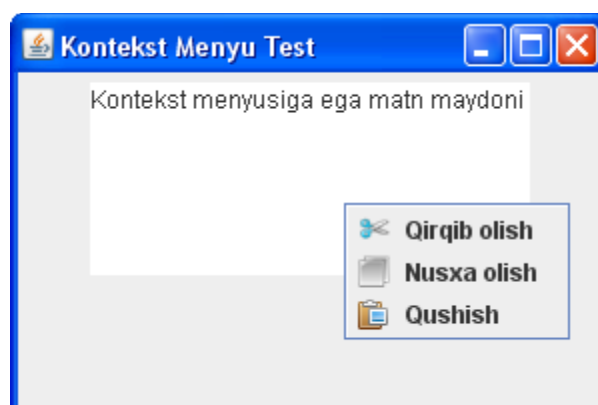
JMenuItem qirqibOlish = new JMenuItem("Qirqib olish");

kontekstMenu.add(qirqibOlish);
```

Kontekst menyu biron bir komponentga biriktiriladi. Kontekst menyusi paydo bo'lishi uchun foydalanuvchi ushbu elementga kontekst menyusini chiqarib beradigan sichqon tugmasini bosish kerak bo'ladi. Windows va Linux operatsion tizimida bu sichqonning asosiy bo'lmagan tugmasi, ya'ni o'ng tugma xisoblanadi. Masalan, kontekst menyu panel, matn maydoni, matn qatori, ro'yxat va boshqa elementlarga `JComponent` klassidan meros qilib olingan `setComponentPopupMenu()` metodi yordamida biriktirilishi mumkin.

```
matnMaydoni.setComponentPopupMenu(kontekstMenu);
```

Quyidagi oyna matn maydoniga biriktirilgan kontekst menyuni ko'rsatib beradi:



JPopupMenu klassi yordamida yaratilgan kontekst menyu

Ushbu oyna quyidagi dastur yordamida yaratilgan:


```

import java.awt.*;

import javax.swing.*;

public class KontekstMenyuTest
{
    public static void main(String[] args)
    {
        JFrame frame = new JFrame("Kontekst Menyu Test");

        JPopupMenu kontekstMenyu = new JPopupMenu();

        Icon qirqibOlishIcon = new ImageIcon("cut.png");

        JMenuItem qirqibOlish = new JMenuItem("Qirqib olish", qirqibOlishIcon);

        Icon nusxaOlishIcon = new ImageIcon("copy.png");

        JMenuItem nusxaOlish = new JMenuItem("Nusxa olish", nusxaOlishIcon);

        Icon qushishIcon = new ImageIcon("paste.png");

        JMenuItem qushish = new JMenuItem("Qushish", qushishIcon);

        kontekstMenyu.add(qirqibOlish);

        kontekstMenyu.add(nusxaOlish);

        kontekstMenyu.add(qushish);

        JPanel panel = new JPanel();

        JTextArea matnMaydoni = new JTextArea("Kontekst menyusiga ega matn maydoni", 6, 20);

        panel.add(matnMaydoni);

        matnMaydoni.setComponentPopupMenu(kontekstMenyu);

        frame.getContentPane().add(panel);

        frame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );

        frame.setSize(300,200);

        frame.setVisible(true);

    }
}

```

Oddiy menyu kabi kontekst menyu tanlanganda `ActionEvent` xodisasi xosil qilinadi va `ActionListener` interfeysini bajarib biron bir amal bajarib olish mumkin.

Dialog oynalar

Dialog oynalar foydalanuvchi interfeysining standart elementi xisoblanib foydalanuvchiga biron bir ma'lumot berish yoki foydalanuvchidan biron bir savol so'rash uchun ishlatiladi.

Dasturlarda dialog oynalar ko'p ishlatilganligi sababli Swing o'rami oldindan tayyorlangan dialoglar to'plamini o'z ichiga oladi. Dialog oynalar `JOptionPane` klassining statik metodlari yordamida yaratilishi mumkin. Ushbu klass dialog oynalarini to'rt guruhga ajratadi:

1. *Ma'lumot berish dialog oynalar* – foydalanuvchiga biron bir ma'lumot ko'rsatadi va odatda OK tugmasiga ega;
2. *Tasdiqlash dialog oynalar* – foydalanuvchidan biron bir savol so'raydi va odatda Yes, No, Cancel tugmalarga ega;
3. *Ma'lumot kiritish dialog oynalar* – foydalanuvchidan biron bir matn kiritishni so'raydi;
4. *Maxsus dialog oynalar* – istalgan elementni joylashtirish imkonini beradi.

Ma'lumot berish dialog oynasi `JOptionPane` klassining `showMessageDialog()` metodi yordamida yaratiladi. Masalan, quyidagi dastur ma'lumot berish dialog oynasini chiqarib beradi.

```
import javax.swing.*;

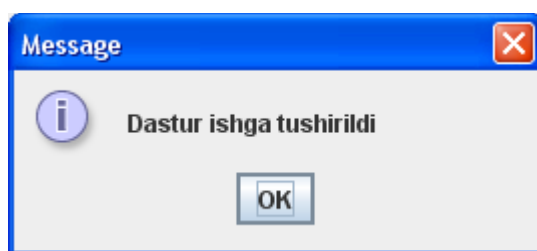
public class DialogTest1
{
    public static void main(String[] args)
    {
        JFrame frame = new JFrame();

        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        frame.setSize(300, 200);

        frame.setVisible(true);

        JOptionPane.showMessageDialog(frame, "Dastur ishga tushirildi");
    }
}
```



JOptionPane klassi yordamida yaratilgan ma'lumot berish dialog oynasi

Tasdiqlash dialog oynasi `JOptionPane` klassining `showConfirmDialog()` metodi yordamida yaratiladi. `showConfirmDialog()` metodi foydalanuvchi bosgan tugmaga asosan butun son qaytaradi. Quyidagi dastur tasdiqlash dialog oynasini chiqarib, tugmaLabel yozuvini bosilgan tugma asosida o'zgartirib beradi.

```
import javax.swing.*;

public class DialogTest2
{

```

```

public static void main(String[] args)
{
    JFrame frame = new JFrame("Dialog Test 2");

    JPanel panel = new JPanel();

    JLabel label = new JLabel("Dialog oynasida bosilgan tugma: ");

    JLabel tugmaLabel = new JLabel(" ");

    panel.add(label);

    panel.add(tugmaLabel);

    frame.getContentPane().add(panel);

    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    frame.setSize(300, 200);

    frame.setVisible(true);

    int i = JOptionPane.showConfirmDialog(frame, "Fayl saqlansinmi?");

    if(i==0) tugmaLabel.setText("Yes");

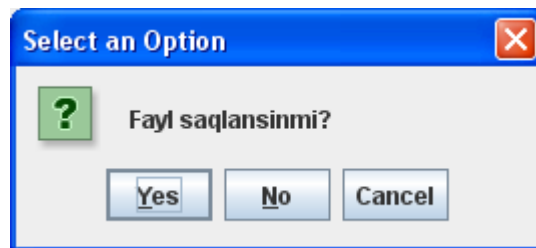
    if(i==1) tugmaLabel.setText("No");

    if(i==2) tugmaLabel.setText("Cancel");

}
}

```

Dastur chiqarib bergan tasdiqlash dialog oynasi quyida keltirilgan:



JOptionPane klassi yordamida yaratilgan tasdiqlash dialog oynasi

Ma'lumot kiritish dialog oynasi `JOptionPane` klassining `showInputDialog()` metodi yordamida yaratiladi. `showInputDialog()` metodi foydalanuvchi kiritgan matnini qaytaradi. Quyidagi dastur ma'lumot kiritish dialog oynasini chiqarib, `matnLabel` yozuvini kiritilgan matn asosida o'zgartirib beradi.

```

import javax.swing.*;

public class DialogTest3
{
    public static void main(String[] args)
    {
        JFrame frame = new JFrame("Dialog Test 3");

```

```

JPanel panel = new JPanel();

JLabel label = new JLabel("Kiritilgan matn: ");

JLabel matnLabel = new JLabel(" ");

panel.add(label);

panel.add(matnLabel);

frame.getContentPane().add(panel);

frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

frame.setSize(300, 200);

frame.setVisible(true);

String s = JOptionPane.showInputDialog(frame, "Fayl nomini kiriting");

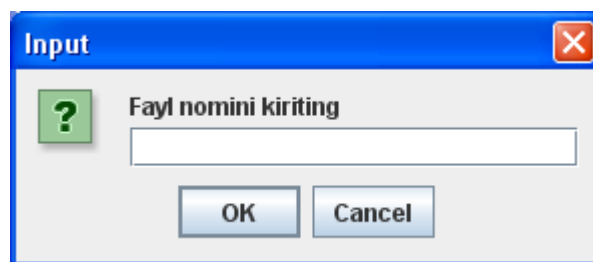
matnLabel.setText(s);

}

}

```

Dastur quyidagi ma'lumot kiritish dialogini chiqarib beardi:



JOptionPane klassi yordamida yaratilgan ma'lumot kiritish dialog oynasi

Maxsus dialog oynasi `JOptionPane` klassining `showOptionDialog()` metodi yordamida yaratiladi. Ushbu dialog oynasiga istalgan elementlarni joylashtirib olish mumkin. Quyidagi dastur avtorizatsia dialog oynasini yaratib beradi.

```

import javax.swing.*;

public class DialogTest4
{
    public static void main(String[] args)
    {
        JFrame frame = new JFrame("Dialog Test 4");

        JPanel panel = new JPanel();

        JLabel matnLabel = new JLabel(" ");

        panel.add(matnLabel);

        frame.getContentPane().add(panel);

        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}

```

```

        frame.setSize(300, 200);

        frame.setVisible(true);

        JTextField userField = new JTextField();

        JPasswordField passField = new JPasswordField();

        String message = "Foydalanuvchi nomi va parolini kiriting";

        JOptionPane.showOptionDialog(frame, new Object[] {message, userField,
passField}, "Login", JOptionPane.OK_CANCEL_OPTION, JOptionPane.QUESTION_MESSAGE, null,
null, null);

        String password = "";

        for(char ch: passField.getPassword())

            password = password + ch;

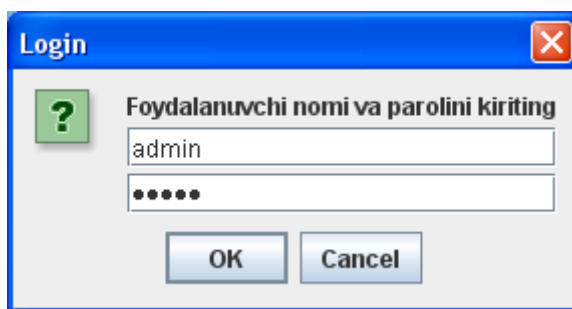
        matnLabel.setText("Foydalanuvchi: " + userField.getText() + "; Parol: " +
password);

    }

}

```

Dastur quyidagi dialog oynani chiqarib beradi:



JOptionPane klassi yordamida yaratilgan va istalgan elementlarni joylashtirish mumkin bo'lgan dialog oynasi

Dastur ishlashi davomida biron bir faylni ochish yoki saqlash extiyoji paydo bo'ladi. Ushbu maqsadda Java dasturlash tili fayllar bilan ishlovchi dialog oynasini yaratib beruvchi `JFileChooser` klassini taklif etadi. Faylni ochish dialog oynasini chiqarish uchun `JFileChooser` klassining `showOpenDialog()` metodidan, faylni saqlash dialog oynasini chiqarish uchun `showSaveDialog()` metodidan foydalanish mumkin.

```

JFileChooser chooser = new JFileChooser();

chooser.showOpenDialog(frame);

```

Fayl tanlash dialog oynasi ochilganda fayl tanlash uchun ma'lum papka ochilishi kerak bo'lsa `JFileChooser` klassining `setCurrentDirectory()` metodidan foydalanish mumkin.

```

chooser.setCurrentDirectory(new File("G:\\book\\code\\"));

```

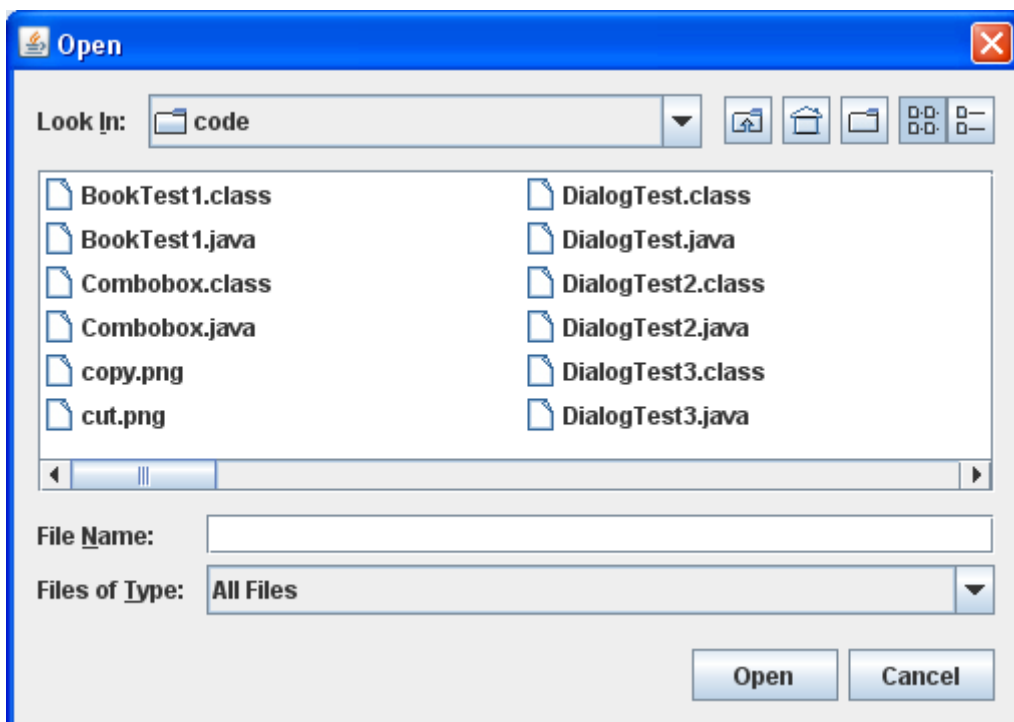
Foydalanuvchi tanlagan faylni olish uchun `getSelectedFile()` metodidan foydalanish mumkin.

```

File file = chooser.getSelectedFile();

```

Quyida fayl tanlash dialog oynasi keltirilgan:



JFileChooser klassi yordamida yaratilgan fayl tanlash dialog oynasi

Ushbu fayl tanlash dialog oynasi quyidafi dastur tomonidan yaratilgan:

```
import java.awt.*;
import javax.swing.*;
import java.io.*;

public class FileTanlash
{
    public static void main(String[] args)
    {
        JFrame frame = new JFrame("File Tanlash");
        JPanel panel = new JPanel();
        JLabel label = new JLabel("");
        JFileChooser chooser = new JFileChooser();
        chooser.setCurrentDirectory(new File("G:\\\\book\\\\code\\\\"));
        panel.add(label);
        frame.getContentPane().add(panel);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(300,200);
        frame.setVisible(true);
        chooser.showOpenDialog(frame);
    }
}
```

```

        String s = chooser.getSelectedFile().getPath();

        label.setText(s);

    }

}

```

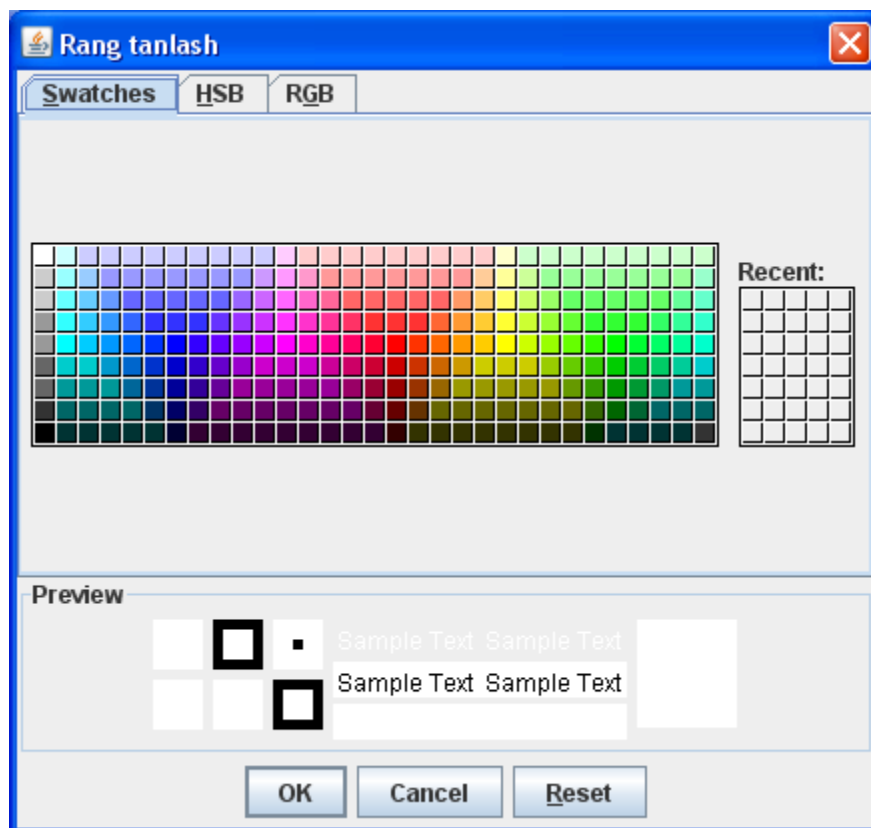
Bu yerda `Fayl` klassi diskda joylashgan faylni ifoda etadi. Fayl manzilini ko'rsatishda papkalarni ajratuvchi “\” simvoli “\\” ko'rinishda yozilishi kerak. `Fayl` klassi `java.io` o'ramida joylashgan bo'lib, ushbu o'ram dastur boshida import qilingan.

Swing o'rami rang tanlash tayyor dialog oynasini xam taklif etadi. Rang tanlash dialog oynasini `JColorChooser` klassi yordamida yaratib olish mumkin. `JColorChooser` klassning `showDialog()` statik metodi dialog oynani chiqarib beradi.

```
Color c = JColorChooser.showDialog(frame, "Rang tanlash", null);
```

`showDialog()` metodi tanlangan rangni qiymat sifatida qaytaradi. Metodning uchunchi parametri joriy tanlangan rangni belgilaydi. Joriy belgilangan rang bo'lmasa `null` qiymati berilishi mumkin.

Quyida rang tanlash dialog oynasi keltirilgan:



JColorChooser klassi yordamida yaratilgan rang tanlash dialog oynasi

Ushbu oyna quyidagi dastur yordamida yaratilgan:

```

import java.awt.*;

import javax.swing.*;

public class RangTanlash

```

```

{
    public static void main(String[] args)
    {
        JFrame frame = new JFrame("Rang Tanlash");

        JPanel panel = new JPanel();

        frame.getContentPane().add(panel);

        frame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );

        frame.setSize(300,200);

        frame.setVisible(true);

        Color c = JColorChooser.showDialog(frame, "Rang tanlash", null);

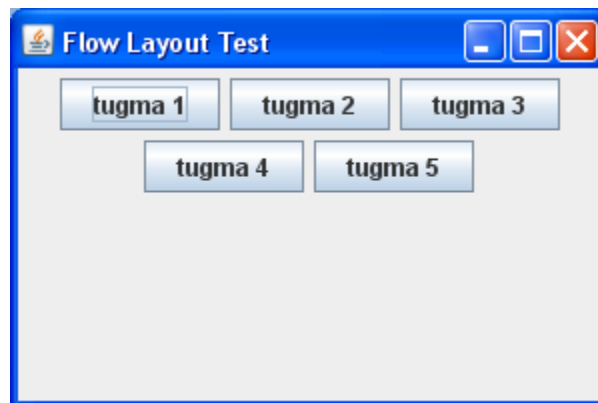
        panel.setBackground(c);
    }
}

```

Joylashuv menedjerlari

Java dasturlash tilida foydalanuvchi interfeysi elementlarini panel yoki frameda joylashtirish usullarini joylashuv menedjerlari orqali belgilnadi.

Panelda elementlar avtomatik ravishda *flow layout* joylashuv menedjeri orqali joylashtiriladi. Ushbu menedjer elementlarni bir qatorga ketma-ket, o'rtaga tekislab joylashtiriladi. Agar elementlar bir qatorga sig'masa, sig'magan elementlar ikkinchi qatorga o'tkaziladi.



Flow layout joylashuv menedjerining standart xolati

Joylashuv menedjeri panelning `setLayout()` metodi orqali belgilanadi. Flow layout joylashuv menedjeri `FlowLayout` klassi yordamida yaratiladi. Panelda elementlar joylashuvini o'rtaga tekislashdan chapga tekislashga o'zgartirish uchun quyidagilarni bajarish kerak:

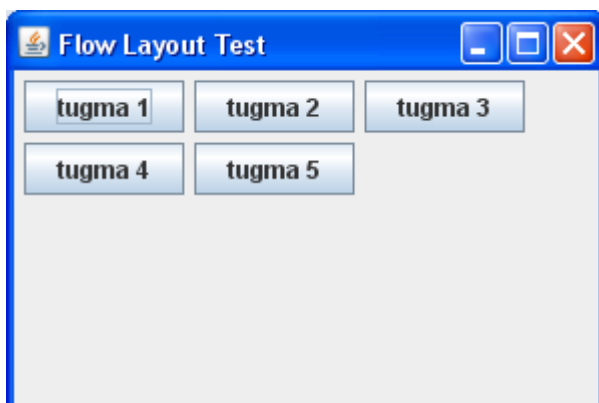
```

layout.setAlignment(FlowLayout.LEFT);

panel.setLayout(layout);

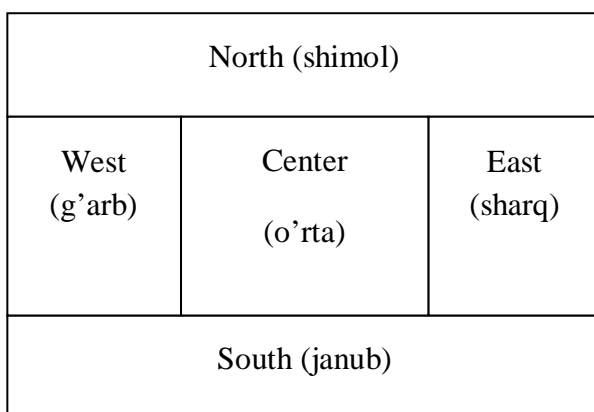
```


Natijada elementlar tekislanishi quyidagiga o'zgaradi.



Flow layout joylashuv menedjerining chapga biriktirilgan xolati

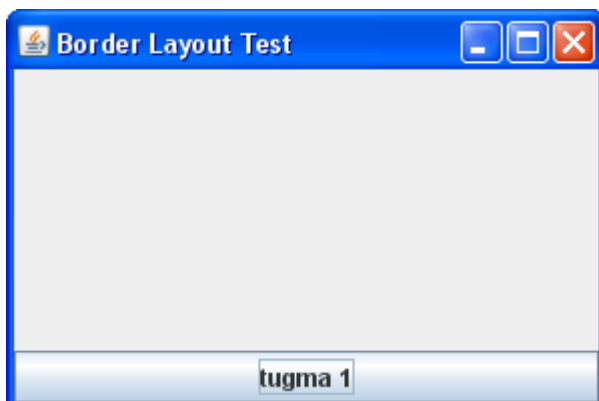
Boshqa joylashuv menedjeri bu *border layout* joylashuv menedjeridir. Ushbu joylashuv menedjeri flow layout manager joylashuv menedjeridan farqli o'laroq elementlarni panelni ma'lum joylariga joylashtirish imkonini beradi. Border layout joylashuv menedjeri panelni besh qismga (north, south, west, east) bo'lib, elementlarni ushbu qismlarga joylashtirish imkonini beradi.



Border layout joylashuv menedjerining sohalari

Border layout joylashuv menedjeri `BorderLayout` klassi tomonidan yaratiladi.

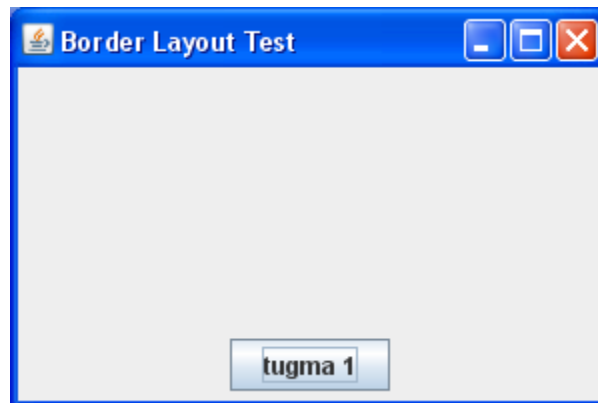
```
BorderLayout layout = new BorderLayout();  
panel.setLayout(layout);  
panel.add(button1, BorderLayout.SOUTH);
```



Border layout joylashuv menedjerining janubiy qismiga joylashtirilgan element

Flow layout joylashuv menedjeridan farqli o'laroq border layout joylashuv menedjeri joylashtirgan elementlarni asil o'lchamlarini mavjud maydonni to'ldirish uchun o'zgartiradi. Interfeys elementlarini asl o'lchamlarini saqlab qolish uchun ushbu elementlarni flow layout joylashuv menedjeri boshqaruvidagi panelga joylashtirib, keyin ushbu panelni border layout joylashuv menedjeri boshqaruvidagi panelning ma'lum qismiga joylashtirilishi mumkin.

```
JPanel ichkiPanel = new JPanel();  
  
ichkiPanel.add(button1);  
  
panel.add(ichkiPanel);
```

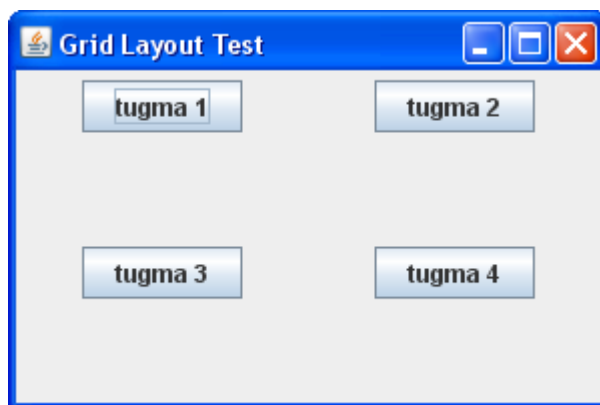


Oldindan panelga joylashtirib olingan element

Yana bir joylashuv menedjeri bu *Grid layout* joylashuv menedjeridir. Ushbu joylashtiruv menedjeri foydalanuvchi interfeysi elementlarini jadval sifatida qator va ustunlarga joylashtiradi. Jadval yacheykalari bir xil o'lchamga ega bo'ladi. Grid layout joylashuv menedjeri `GridLayout` klassi yordamida yaratiladi. Qator va ustunlar sonini klass konstruktori orqali berish mumkin. Border layout joylashuv menedjeri singari grid layout joylashuv menedjeri foydalanuvchi interfeysi elementlari o'lchamlarini mavjud joyni to'liq egallash uchun o'zgartiradi. Foydalanuvchi interfeysi elementlari o'lchamlarini saqlab qolish uchun ularni avval flow layout joylashuv menedjeri boshqaruvidagi panelga joylashtirish kerak.

```
GridLayout layout = new GridLayout(2, 2);  
  
panel.setLayout(layout);  
  
JPanel ichkiPanel1 = new JPanel();  
JPanel ichkiPanel2 = new JPanel();  
JPanel ichkiPanel3 = new JPanel();  
JPanel ichkiPanel4 = new JPanel();  
  
ichkiPanel1.add(button1);  
ichkiPanel2.add(button2);  
ichkiPanel3.add(button3);  
ichkiPanel4.add(button4);  
  
panel.add(ichkiPanel1);  
panel.add(ichkiPanel2);
```

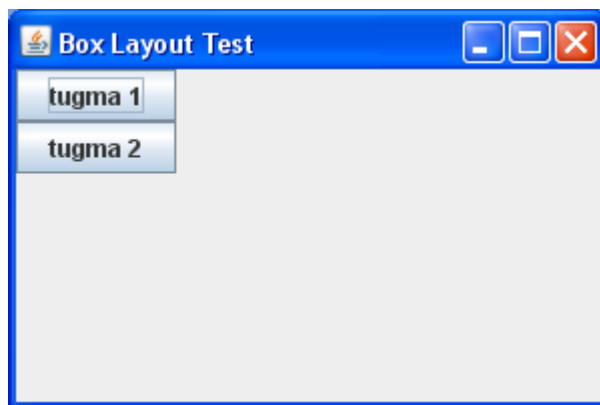
```
panel.add(ichkiPanel3);  
panel.add(ichkiPanel4);
```



Grid layout joylashuv menedjeri

Yuqoridagi barcha joylashuv menedjerlari `java.awt` o'ramiga tegishli klasslar yordamida yaratiladi. `java.swing` o'rami xam joylashuv menedjerlariga ega. Bularning bittasi *Box layout* joylashuv menedjeridir. Ushbu joylashuv menedjeri foydalanuvchi interfeys elementlarini vertikal qator yoki gorizontal ustunga joylashtirish imkonini beradi. Box layout joylashuv `BoxLayout` klassi tomonidan yaratiladi.

```
BoxLayout layout = new BoxLayout(panel, BoxLayout.Y_AXIS);  
panel.setLayout(layout);  
panel.add(button1);  
panel.add(button2);
```



Box layout joylashuv menedjeri

Takrorlash uchun savol va topshiriqlar:

1. Frame deganda nima tushuniladi?
2. Panel nima uchun ishlatiladi?
3. Xodisa deganda nima tushuniladi?
4. Xodisalar qanday boshqariladi?
5. Tugalar qanday yaratiladi?
6. Yozuvlar deganda nima tushuniladi?

7. Tanlash elementlariga nimalar kiradi?
8. Checkbox va Radio tugma o'rtasida farq nima?
9. Matn kiritish uchun qanday grafik elementlardan foydalaniladi?
10. Menyu xosil qilish tartibi qanday?
11. Qanday dialog oynalari mavjud?
12. Joylashuv menedjerlarining vazifasiga nimalar kiradi?

ISTISNO VA XATOLAR

Xatolik va istisnolar

Dastur ishlashi mobaynida ichki yoki tashqi sabablarga ko'ra xatoliklar yuzaga kelishi mumkin. Ichki sabablarga massivga murojat qilishda noto'g'ri indeksni ishlatish yoki obyekt biriktirilmagan obyekt o'zgaruvchisini ishlatish kabilar kiradi. Tashqi sabablarga noto'g'ri ma'lumot joylashgan fayl bilan ishlash yoki tarmoqdagi nosozliklar misol bo'la oladi. Agar dastur ishi xatolik tufayli to'liq bajarilmay qolsa, dastur quyidagi amallarni bajarishi lozim:

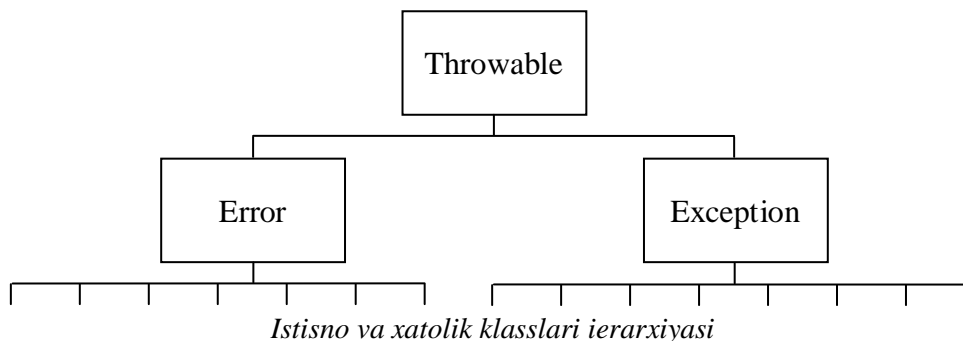
- xatosiz xolatga qaytish;
- foydalanuvchi ishini saqlab dastur ishini to'g'ri tugatish.

Istisnolarni boshqarish mexanizmining maqsadi boshqaruvni xatolik paydo bo'lgan joydan ushbu xatolik bilan ishlay oladigan dastur kodiga o'tkazishdan iborat. Istisnolar bilan ishlash uchun paydo bo'lishi mumkin bo'lgan xatoliklarni ko'rib chiqish kerak. Bular:

1. Foydalanuvchi kiritgan ma'lumotning xatoligi – kiritilgan matn xatoligi, noto'g'ri URL adresi, noto'g'ri son turi;
2. Qurilma xatoliklari – printer o'chiqligi yoki uning ishlashidagi xatolik, server ishlashidagi xatolik;
3. Resurslar cheklanishi – disk to'lib qolishi;
4. Dastur kodi xatoliklari – metodlarni noto'g'ri bajarilishi, mavjud bo'lmagan massiv indeksini ishlatish.

Java dasturlash tilida agar metod o'z ishini bajarish imkoni bo'lmasa (xatoliklar sababli) ushbu metod xatolik xaqida ma'lumotni o'z ichiga olgan obyektни xosil qiladi. Bundan tashqari, metodni chaqirgan dastur kodining bajarilishi to'xtatiladi va istisnolarni boshqaruvchi mexanizm dasturning istisnolarni boshqaruvchisini izlaydi. Istisnolarni boshqaruvchi istisno obyektini qabul qilish va xatolik yuzaga kelganda bajariladigan amallarni belgilash imkonini beradi.

Java dasturlash tili obyektlarga ixtisoslashgan dasturlash tili bo'lganligi sababli istisnolar obyekt ko'rinishida bo'ladi. Barcha xatolik va istisnolar `Throwable` klassidan kelib chiqqan klasslar orqali xosil bo'ladi. Quyida istisnolar ierarxiya daraxtining yuqori qismi keltirilgan:



`Throwable` klassini ikkita, `Error` va `Exception` klasslari kengaytiradi. Ushbu klasslarni o'z navbatida aniq bir xatoliklarni ifoda etuvchi klasslar kengaytiradi.

`Error` klassi dasturning ichki xatoliklarini va resurslar cheklanishini belgilaydi. Ushbu xatoliklarni dastur yordamida to'g'irlash mushkul bo'lib, bu xolatda foydalanuvchini xatolik xaqida ogoxlantirib dastur ishi tugatiladi.

Dastur tuzilayotganda `Exception` klassi va uni kengaytiruvchi klasslarga asosi e'tiborni qaratish kerak. `Exception` klassi o'z navbatida ikki turdagi klasslar tomonidan kengaytiriladi. Birinchi turdagi klasslar bu `RuntimeException` klassi va uni kengaytiruvchi klasslar. Ikkinchi turga boshqa klasslar kiradi.

Java dasturlash tilida `Error` va `RuntimeException` klasslaridan kelib chiqqan istisnolar *tekshirilmaydigan istisnolar* deyiladi. Barcha boshqa istisnolar *tekshiriladigan istisnolar* deyiladi. Tekshiriladigan istisnolarga dasturchi istisnolarni boshqaruvchi dastur kodini ta'minlab berishi kerak.

Metod amal bajarishi mobaynida uddalay olmaydigan xolat yuzaga kelsa istisno obykti xosil qilinadi. Agar metod bajarilishida istisno xosil bo'lish xolati mavjud bo'lsa metod e'lon qilish kodida ushbu xolat belgilanishi kerak. Istisno `throws` kalit so'zi bilan belgilanadi. Masalan, agar dasturdagi `faylniOchish()` metodi xotirada joylashgan faylni ochishi kerak bo'lsa, metod bajarilishi mobaynida ochilishi kerak bo'lgan fayl xotirada mavjud bo'lmasligi yoki mavjud fayl ochilmasligi mumkin. Ushbu xolda `IOException` istisnosi xosil bo'ladi.

```
public void faylniOchish() throws IOExceptions
{
    ...
}
```

Istisnoni atayin xosil qilish xam mumkin. Istisno `throw` kalit so'zi bilan xosil qilinadi. Odatda bu dastur ishini tekshirish uchun amalga oshiriladi. Masalan,

```
IOException e = new IOException();
throw e;
```

Istisnolar bilan ishlash

Istisno xosil bo'lgandan keyin ushbu istisno ushlanishi kerak. Istisnoni ushlash uchun `try/catch` blokidan foydalaniladi.

```
try {
    //dastur kodi
    //dastur kodi
} catch (ExceptionTuri e) {
    //istisnoni boshqaruvchi dastur kodi
}
```

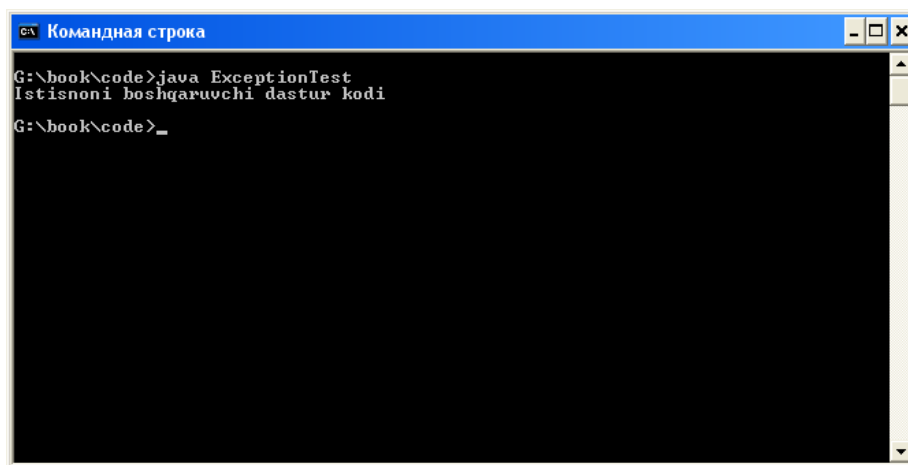
`try` bloki ichidagi dastur kodida istisno xosil bo'lsa dastur `try` bloki ichidagi qolgan dastur kod bajarilishi to'xtatilib `catch` bloki ichidagi istisnoni boshqaruvchi dastur kodi bajariladi, ya'no istisno ushlaniladi. Xosil bo'lgan istisno `catch()` ifodasida e'lon qilingan istisno yoki uni kengaytiruvchi istisno turidan bo'lishi kerak. Masalan, quyidagi dastur `try` bloki ichida `IOException` istisnosi xosil qilinadi. Xosil qilingan istisno `catch` bloki tomonidan ushlanadi va istisno boshqaruvchi kodi bajariladi.

```
import java.io.*;

public class ExceptionTest
{
    public static void main(String[] args)
    {
        try {
            IOException istisno = new IOException();

            throw istisno;
        } catch(IOException e) {
            System.out.println("Istisnoni boshqaruvchi dastur kodi");
        }
    }
}
```

Dastur komanda satriga quyidagi matnni chiqarib beradi:



```
Командная строка
G:\book\code>java ExceptionTest
Istisnoni boshqaruvchi dastur kodi
G:\book\code>_
```

Istisnolarni boshqarish tizimi

Agar `try` bloki ichidagi dastur kodida xech qanday istisno xosil bo'lmasa `catch` bloki bajarilmay ushbu blokdan keyingi dastur bajariladi. Agar xosil bo'lgan istisno `catch()` ifodasida ko'rsatilgan istisnodan farqli bo'lsa `catch` bloki bajarilmaydi va xosil bo'lgan istisno ushbu metodni chaqirgan dastur kodiga uzatiladi.

Metod xosil bo'ladigan istisnoni `try/catch` bloki bilan ushlashi yoki `throws` kalit so'zi bilan xosil bo'lishi xaqida e'lon qilinishi kerak.

Takrorlash uchun savol va topshiriqlar:

1. Istisno deganda nima tushuniladi?
2. Istisnolar qaysi xollarda xosil bo'ladi?
3. Istisnolar qanday boshqariladi?
4. Xatolik deganda nima tushuniladi?

Foydalanilgan adabiyotlar

1. Core Java 2 Volume I – Fundamentals, 7th Edition, Cay S. Horstmann, Gary Cornell, Prentice Hall PTR, 2004
2. Learning Java, 3rd Edition, Jonathan Knudsen, Patrick Niemeyer, O'Reilly, 2005
3. Head First Java, 2nd Edition, Kathy Sierra, Bert Bates, O'Reilly, 2005
4. The Java Programming Language, 4th Edition, Ken Arnold, James Gosling, David Holmes, Addison Wesley Professional, 2005