

z i- j

# JAVA 8

Zabil İbayev

2017

Zabil İbayev

# JAVA 8

## Zİ-J

Bakı 2017



## Mündəricat

<b>Kitab kimlər üçün nəzərdə tutulub. ....</b>	<b>5</b>
<b>Kompüter sizi necə başa düşür və ona necə əmr verilir. ....</b>	<b>6</b>
<b>Biz kompüter proqramlarını necə yaza bilərik. ....</b>	<b>6</b>
<b>Proqramlaşdırma dilləri bir-birindən nə ilə fərqlənir?.....</b>	<b>7</b>
<b>Java nə ilə fərqlənir? .....</b>	<b>8</b>
<b>OOP –nin məntiqi və prinsipləri. ....</b>	<b>10</b>
Obyektin təhlili .....	10
Sinifləndirmək – “CLASS” nədir?.....	11
Instance .....	12
Encapsulasiya.....	12
Abstraction .....	12
Inheritance(irsilik).....	13
Polymorphism(polimorfizm).....	13
Association(əlaqə) .....	13
Aggregation(hissələrin birləşdirilməsi).....	14
Composition(kompozisiya) .....	14
<b>Java-da yazmaq üçün lazım olan alətlər. ....</b>	<b>14</b>
“Netbeans”-lə tanışlıq. ....	15
“Salamlama” proqramı .....	17
“NetBeans”-də səhvlərin aydınlaşdırılması .....	21
<b>“JAVA”-da dəyişənlər.....</b>	<b>23</b>
String .....	23
“+=” operatoru .....	25
“.toUpperCase” və “.toLowerCase” .....	26
“==” və “.equals()” .....	26

“.length()” .....	27
“.Contains()” və “.IndexOf()” .....	28
Char .....	28
Rəqəm dəyişənləri .....	29
Integer.....	29
Double .....	29
Floating.....	29
Byte.....	29
Short .....	29
Long.....	29
Hesablamalar ilə bağlı praktiki məsələlər.....	29
Riyazi operatorlar .....	29
Dəyişənləri qruplaşdırma.....	30
Qalığın ekrana verilməsi.....	31
Bir-bir artırma və azaltma.....	31
Riyazi operator ardıcılığı .....	32
Dəyişənlərlə hesablama və görüntü .....	33
Məntiq dəyişənləri və məntiq operatorları.....	33
Boolean.....	33
<b>Şərt əmrləri.....</b>	<b>34</b>
Şərt operatorları – (==), (!=), (<), (>), (<=), (>=), (?) .....	34
Şərt əmrləri – if, else.....	34
Sual operatoru .....	36
Şərt əmrləri - switch, case, break, default .....	36
<b>Dövr əmrləri - for , do, while .....</b>	<b>38</b>

For.....	38
Mürəkkəb For əməlləri.....	39
While .....	40
Continue .....	41
Do while .....	42
Dövrün adlandırılması.....	43
<b>Array .....</b>	<b>44</b>
Birinci üsul - .....	44
İkinci üsul - .....	45
Çoxölçülü Array.....	47
<b>Obyektin yaradılması .....</b>	<b>48</b>
<b>Built-in Java Packages .....</b>	<b>49</b>
Import .....	50
<b>Access control( girişin idarə olunması) .....</b>	<b>51</b>
<b>Method nədir? .....</b>	<b>52</b>
<b>Constructor.....</b>	<b>55</b>
<b>“Inheritance”-in yaradılması .....</b>	<b>57</b>
<b>“Super” və “This” açar sözləri.....</b>	<b>58</b>
<b>Static Variable və ya Class dəyişənlər .....</b>	<b>60</b>
<b>Casting.....</b>	<b>61</b>
<b>Swing GUI- Qrafiki istifadəçi interfeysi .....</b>	<b>62</b>
Swing-in strukturu .....	62
Qrafiki təsvirdə ilk proqram “Salam Dünya” .....	63
SWING GUI Forms.....	64
ParseInt və ValueOf .....	67
ComboBox və JOptionPane praktiki məsələ.....	68
Stream .....	70
Scanner .....	70

Delimiter .....	71
Exception , Try , Nested Try, Catch, Finally.....	72
Create, Read və Write File .....	75
Throwable, Throws və Throw .....	76
<b>MySQL database-in qurulması .....</b>	<b>78</b>
Məlumatı bazaya yazma və silmə proqramı .....	89

## **Kitab kimlər üçün nəzərdə tutulub.**

İlk olaraq onu bildirmək istəyirəm ki, kitab həvəskar proqramçılar üçün nəzərdə tutulub. Azərbaycan dilində Java proqramlaşdırma dili üzrə ədəbiyyatın olmaması və ya mövcud kitabların yetərinə mənə aydın yazılmadığını nəzərə alıb, digər həvəskarlarında belə çətinliklərlə üzləşəcəyini düşünüb, proqramlaşdırmada əldə edəcəyim təcrübəmi bu kitabda toplayıb sizinlə bölüşürəm. Yaradılmış proqramlaşdırma dillərinin ingilis dilli olması bizi eyni zamanda ingilis dilini müəyyən səviyyədə öyrənməyə məcbur edir. İngilis dilində zəif bilən və ya yeni başlayanlara kömək olsun deyə, bəzi terminlərin tərcüməsini yanında yazıram. Professional proqramçı dostlarım təcrübələrinə əsaslanaraq bildirirlərki, proqramlaşdırmada yüksək nəticə əldə etmək üçün ingilis dili biliyinin olması vacibdir. Sərbəst şəkildə çalışacağınız zamanlarda qarşınıza çıxan problemlərə uyğun həlləri tapmaqda nə bu kitab, nə də başqası internet üzərindəki material qədər sizi qane edə bilməz. İnternetdə axtarışda daha dəqiq nəticə əldə etmək üçün ingilis dilini bilmək vacibdir. Proqramlaşdırmanı öyrənmək vaxt və səbr tələb edir. Sizə uğurlar arzulayıram!



### **Kompüter sizi necə başa düşür və ona necə əmr verilir.**

Kompüterlər ikili say sistemi üzərində işləyir. İkili say sistemində 0 və 1 anlayışı var. Verdiyimiz əmrlər 0 - yanlış, olmaz, dayan və ya 1 - doğru, olar, davam et kimi tərcümə edilir. Ümumilikdə isə belə başa düşmək olar, kompüterin hansısa hissəsindəki naqillərə elektrik axımı verilsin ya yox. Kompüterimizdəki işləri görən mikrosxem, xalq deyimi ilə desək, kompüterin beyni, CPU(Central Processing Unit) mərkəzi hesablayıcı qurğu anlamına gəlir. "CPU"-ya əmr verilməsi üçün istehsalçılar Instruction Set (instruksiyalar dəsti) adlı qaydalar toplusu düzəldiblər. Bütün yazılan proqramlar prosessorla nə etmək istədiklərini bu qaydalar əsasında izah edirlər.

### **Biz kompüter proqramlarını necə yaza bilərik.**

Sadə dildə desək proqram yazmaq adi mətn yazmaq kimidir. Bu mətnlər sonda 0 və 1-ə tərcümə olunur və kompüter istədiyimiz əmri yerinə yetirir. Bəzi proqramlaşdırma dilləri xüsusi redaktə proqramları vasitəsilə, bəziləri Windows daxilindəki "Texteditor"-la və ya "MSDOS"-da yazıla bilər. Proqramlaşdırma dilləri fərqli üslubda dizayn olunmuş kodları ikili say sisteminə tərcümə edir. Windows daxilindəki CMD(command line) ilə "DOS"-da proqram yazmaq mümkündür. Windows 10 əməliyyat sistemində "Start" menyusuna daxil olaraq "CMD" yazıb "Command Prompt" proqramını açma bilərsiniz.



"@echo."- əmri ilə istədiyiniz mesajı yeni sətrdən yaza bilərsiniz. Bundan başqa "DOS"-da daha nə etmək olar deyirsinizsə "help" yazıb digər əmrlərə baxa bilərsiniz.

## Proqramlaşdırma dilləri bir-birindən nə ilə fərqlənir?

Yuxarıda bildirdiyim kimi kompüter ikili say sistemindəki əmrləri anlaya bilir. Bizim üçün ikili say sistemi ilə proqram yazmaq çox çətinidir. İkili say sistemi ingilis dilində “*Binary*” adlanır. “<http://www.binarytranslator.com>” sahifəsində təsvir olunanları rahat görmək olar.



"Alma" sözünün sağ tərəfdə tərcümə olunmuş ikili say sistemindəki uzunluğuna baxın.

Proqramçılar ikili say sisteminə keçidi sadələşdirmək üçün 16-lıq say sistemindən istifadə etməyə başladılar. İngiliscə 16-lıq say sistemi “Hexadecimal” adlanır. Aşağıdakı şəkildə "Alma" sözünün 2-li və 16-lıq say sistemindəki yazılışı göstərilib.

Binary	Hexadecimal
01000001 01101100 01101101 01100001	41 6c 6d 61

16-lıq say sistemində proqramlaşdırmaya misal olaraq “*Assembler*”-i göstərmək olar. Proqramlaşdırmada növbəti mərhələ kimi vizual görünüşü daha rahat olan “*BASIC*” dili verilmiş mətni ekrana çıxarmaq üçün tələb olunan əmrlər aşağıdakı şəkildə göstərilib. Redaktor “<http://www.readybasic.com>” sahifəsindəndir. Bunuda əlavə edim ki, sahifə Java dilində hazırlanıb.

```
File Edit Run Help ReadyBASIC 1.1 Beta
ReadyBASIC v1.1 Beta
Immediate-mode interpreter
Copyright 2006-2012, Kevin Matz, All Rights Reserved.
7007664 bytes free.

> 1 PRINT "SALAM ZABIL"
> 2 INPUT A$
> RUN
SALAM ZABIL
SENEDE SALAM OLSUN :)
>
```

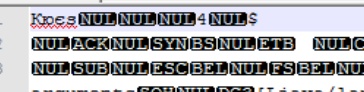
“*BASIC*”-də proqrama verilən əmrlər sizin tərəfinizdən təyin olunmuş rəqəmlərin ardıcılığına uyğun olaraq işlənir. Yazılmış hər bir sətir ayrı-ayrılıqda oxunur, sonra tərcümə və icra edilir. Bu üsulla işləyən proqramlara *interpreter*(tərcüməçi) proqramlar deyilir. Bu cür yanaşma proqramın aşağı sürətdə işləməsinə səbəb olur. Əgər “*BASIC*”-də əmrlərin qabağında rəqəm yazmasaq, proqram birbaşa əmrin icrasına keçir. Bu səbəbdən proqramımızda ilk əmr olan “*PRINT*”(çap etmək, ekrana çıxartmaq) qarşısına rəqəm qoymuşam ki, 1-ci "Salam Zabil" sözünü ekrana çıxartsın sonra 2-ci əmr “*INPUT*”(daxil etmək) ekrana çıxsın. “*RUN*” yazılmış sətirləri oxuyur və proqramı işə salır. Bəs “A\$” nədir. “\$” simvolu dəyişən deməkdir. A dəyişəni(\$) hər hansı bir məlumatla əvəz oluna bilər. Rəqəm və ya yazı. Nəticədə yazdığım kod məni salamlayır və məndən “input” vasitəsi ilə cavab aldıqda sona çatır. Daha sonra yeni sətirdən yeni əmr daxil edilməsini gözləyir.

### Java nə ilə fərqlənir?

*Java OOP* texnologiyasından istifadə edir. Object Oriented Programming (OOP) obyekt yönümlü proqramlaşdırma. Adından da müəyyən dərəcədə anlamaq olar. Obyekt yönümlü yəni obyekt yaratmaq üçün proqramlaşdırma dili.

Obyekt hər hansı bir əşya, funksiyası olan qurum və ya canlı ola bilər. *OOP* əvvəlki dillərdən fərqli olaraq bir obyektə bir neçə yerdə

Məsələn yaratdığınız avtomobil obyektini (eyni zamanda "avtomobil proqramını" əgər belə adlandırsaq) digər proqramı daxilindən çağırıb istədiyiniz yerə tətbiq edə bilərsiniz. Digər proqram dağlıq ərazi, ya şəhər ola bilər. Bunların hamısında avtomobil proqramı sərbəst işləyə biləcəkdir. Siz çağırılan obyektin kodunu tamamı ilə köçürtmədən, sadəcə mənbəyini göstərməklə, ona verilən adı yeni proqramdakı müəyyən olunmuş yerə daxil edərək proqramı işə salırsınız.



```

1  KooCs NUB NUB NUB 4 NUL$
2  NUL ACK NUL SYNBS NUL ETB NUL CANN NUL
3  NUL SUB NUL ESC BEI NUL ES BEI NUL GSS
   arguments SOH NUL DC3 [Ljava/lang/St
4  SourceFile SOH NUL
5  Salam.java FF NUL BEI NUL ES SOH NUL
6  Salam dJjnya ! BEI NUL SS FF NUL US NUL
   BEI NUL ! FF NUL NUL + SOH NUL NAK com/
   NUL ENO NUL ACK NUL NUL NUL NUL NUL ST
7  NUL NUL NUL ACK NUL SOH NUL NUL ET
8  NUL NUL NUL NUL SOH NUL NUL SOH NUL
9  NUL NUL NUL SOH NUL ETX NUL NUL ACK
10 NUL ES NUL VT NUL NUL NUL SYN NUL STX N

```

Şəkildən də göründüyü kimi kod “*compile*” olunan kimi faylın sonuna “.class” əlavə edilmişdir. Bu proses də öz növbəsində JAVA Virtual Machine (Java Virtual Maşını-JVM) işinə start verir. JVM “bytecode”-u “*binary*” koda çevirir. Java bu metodla daha sürətli və “CPU”-nün versiyasından asılı olmayaraq bütün kompüterlərdə işləyə bilər. İndi bütün yuxarıda izah etdiyim addımları bir araya toplayıb Java-nın prosesi tam olaraq necə yerinə yetirdiyinə baxaq:

1. "Salam.java" proqramı yazılır;
2. "Salam.class" compile olunur və fayl “*bytecode*”-da çevrilir;
3. JVM vasitəsilə bytecode binary koda çevirilir;
4. Nəticə əldə edilir.

### **OOP –nin məntiqi və prinsipləri.**

Həvəskar proqramçı üçün dərin nəzəri biliklərin olması vacib deyil. Bu baxımdan OOP-yə səthi yanaşaraq bizə hazırda lazım ola biləcək məqamlara toxunmuşam. OOP həyatda olan obyektləri virtual aləmdə yaratmaq üçün düşünülmüş nəzəriyyədir. Digər bir deyimlə real həyatdakı həm hərəkətli, həm hərəkətsiz obyektləri, onların digər obyektlərlə münasibətlərini virtual aləmdə yarada bilmək üçün hazırlanmış üsullar, qaydalar toplusudur. Obyekt hər hansı bir əşya, xidmət göstərən qurum və ya canlı ola bilər. Buna misal olaraq avtomobil, bank, heyvan, su, qravitasiyanı göstərmək olar.

### **Obyektin təhlili**

Obyekt real həyatdakı kimi virtual formada yaratmaq üçün onu bütünlükdə anlamaq lazımdır. Avtomobil obyektini incələyək və görək bizə hansı xüsusiyyətləri məlumdur:

- İstehsal edən firmanın adı
- Modelin ili
- Motoru
- Rəngi

- Salonu

bu xüsusiyyətlər “attribute”-lar adlanır. Bəs maşının nə kimi funksiyaları vardır?

- İşə düşür
- Sönür
- Əyləci sıxdıqda dayanır
- Qapı və baqaj hissəsi açılır/qapanır
- Şüşə silənləri hərəkət edir

Bu funksiyalar *Method*(metod) və *Operation*(əməliyyatlar) adlanır. Digər bir misal olaraq bank hesabını götürə bilərik. Bank hesabı ilə bağlı bizə nə məlumdur:

- Hesabın nömrəsi
- Hesabın kimə aid olması
- Balansı
- Gəlir faizi
- Ünvanı

Yuxarıdakı məlumatlar “*Attribute*”-lardır. Bəs bank hesabının hansı funksiyaları vardır?

- Deposit qoymaq
- Bankomatdan pul çıxartmaq
- Çek yazmaq
- Ünvanı dəyişmək
- Telefon nömrəsini dəyişmək

Bu funksiyalarda “*Method* və *Operation*”-lardır. Hər hansı bir obyektin məlum olan xüsusiyyətləri onun *atributları* və onları işə salan metodları vardır. Bəzi hallarda metodlar atributla əlaqəli olmaya bilər.

### Sinifləndirmək – “*CLASS*” nədir?

Təhlil üçün avtomobil obyektini götürək. Avtomobilin ban növündən, firmasından və ya ölçüsündən asılı olmayaraq

obyektdəki *atributlar* və *metodlar* eynidir. Bu da onları eyni sinfə daxil etməyə imkan verir. El dilində belə izah edim, yolda uzaqdan bir minik vasitəsinə görəndə siz onun markasını və ya nömrəsini görməyə bilərsiniz amma onun attribute-larına görə avtomobil sinfinə aid olduğunu deyə bilərsiniz. Daha qısa şəkildə desək *atributları* və *metodları* eyni olan obyektlər bir sinfə, bir “*class*” daxilinə salına bilərlər. Eyni zamanda heyvanları, bitkiləri və s. eyni xüsusiyyət daşıyan obyektləri bir sinfə, bir “*class*” altına salmaq olar.

**Instance**(instansiya, obyektin oxşarı). Məmin notbukum və Orxanın notbuku, notbuklar sinfinə, notbuk “*class*”-ına aiddir. Məmin və Orxanın notbuku notbuk “*class*”-nın “*instance*”-dır. Daxilində fərqli atributları olmağına baxmayaraq ümumi göstəriciləri “*class*”-la eyni olan obyektlərdir. Obyektin yaradılması “*class*”-ın “*instance*”-nın yaradılmasıdır. Başqa sözlə desək, adından və ya sahibindən asılı olmayaraq notbuk deyəndə nə ağlına gəlir? Bax elə o təsvir obyektin notbuklar sinfinə aid olması deməkdir. Kiməsə deyirsiniz mən notbuk almışam, hətta onu görməyən adamın belə artıq təsəvvürü yaranır ki, notbuk sinfinə aid ola biləcək obyekt necə olmalıdır. Onun “*instance*”-nı fikirləşir(rəngini, ağırlığını və s.)

## Encapsulasiya

Atributların obyekt haqqında məlumat verdiyini artıq bilirik. Obyekt funksionallaşdığında “*encapsulasiya*” metodlar vasitəsi ilə obyektin məlumatlarını gizlətmək rolunu oynayır. Proses sonundakı nəticə məlum olsa belə daxilə hansı atributların qarşılıqlı əlaqəsindən nəticənin yarandığı gizlədilir.

## Abstraction

Vacib detalları göstərməklə lazımsızları kənarlaşdırmaq funksiyasını icra edir. Misal olaraq, avtomobili işə salan sahibi açarı

döndərdikdə avtomobilin işə düşəcəyindən başqa obyektə gedən proseslərin necə icra olunduğunu görmür. O, sadəcə açarı döndərərək avtomobilin işləməsinə gözləyir. Digər proseslər ondan uzaq tutulur və ya gizlədilir.

### **Inheritance(irsilik)**

Velosiped, avtomobil, yük maşını ayrı-ayrılıqda sinifdirlər. Avtomobil sinifinin alt sinifləri kimi mini, VAN, SUV-ləri götürmək olar. Amma hamısı insanlar üçün yaradılmış minik vasitələrinə aiddirlər. Minik vasitəsi irsilik daşıyır və ondan sonra gələn siniflər və alt siniflər birlikdə irsilik daşıyırlar. İrsilik varsa əlaqə təbii olaraq var. Digər sözlərlə, “inheritance” bir proqramla başqa proqramın xüsusiyyətlərini istifadə etməyə imkan yaradır.

### **Polymorphism(polimorfizm)**

Poly-çoxlu, morph-forma, davranış deməkdir. Bir neçə obyekt arasından ən düzgün olanının seçilməsində “polymorphism” metodundan istifadə edilir. Fərz edin ki, şirkətdə işləyirsiniz və nəqliyyat departamentinə zəng edib deyirsiniz ki, şirkətin rəisinə digər şirkət tərəfindən orta həcmdə heykəl hədiyyə edilib və rəis deyir ki, olduğunuz yerə heykəli daşıya biləcək minik vasitəsi göndərilsin. Şirkətin balansında olan minik vasitələri sinfində sedan, SUV, yük maşını və traktor var. “Polymorphism” bu yerdə işə düşür. Əsas dəyərlər minik vasitəsinin seçilməsi, sonra orta həcmdə yük daşıya bilən avtomobilin seçimini həyata keçirir. Daha orta həcmli heykəl üçün traktor göndərmir.

### **Association(əlaqə)**

Obyektlər arasındakı hər hansı bir əlaqəyə “association” deyilir. Avtomobil obyektini və yol obyektini arasında “association” var. Belə ki, avtomobil funksiya göstərmək üçün yoldan istifadə edir.



### Aggregation(hissələrin birləşdirilməsi)

Əlaqəyə sahib olmaq kimidə qəbul edilə bilər. Avtomobil funksiya göstərmək üçün daxilindəki motordan, sükandan, təkərlərdən istifadə etməlidir. Bu cür əlaqəyə “aggregation” deyilir. Həm onlara sahibdir, həm də funksiya göstərmək üçün əlaqə saxlayır.

### Composition(kompozisiya)

Daxilindəki obyekt və ya obyektlərlə birgə funksiya göstərmək üçün əlaqə saxlayır. Buna misal olaraq bina obyektini və otaqları arasındakı əlaqəni göstərmək olar. Təxmini belə məntiqə çalışır, motor avtomobilsiz ola bilər amma otaqlar binasız ola bilməz. Eyni zamanda binanın yeri dəyişdirildikdə otaqlarında yeri avtomatik dəyişdirilir. Bir-birlərinə bağlıdırlar.

### Java-da yazmaq üçün lazım olan alətlər.

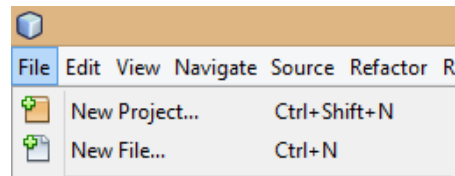
Java Oracle şirkətinin məhsuludur və saytlarında proqramçılar üçün NetBeans redaktoru ilə birgə şəkildə JDK (Java Development Kit - Java İnkişaf Paketi) pulsuz təqdim olunur. “<http://www.oracle.com/technetwork/java/javase/downloads/index.html>” ünvanından sizə uyğun versiyanı paket şəklində endirə bilərsiniz.

Java SE Development Kit (JDK) Cobundles		
<b>JDK 8 with NetBeans</b> This distribution of the JDK includes the NetBeans IDE, which is a powerful integrated development environment for developing applications on the Java platform. <a href="#">Learn more</a> ▶		<a href="#">DOWNLOAD</a> ▶
Java SE and NetBeans Cobundle (JDK 8u121 and NB 8.2)		
Product / File Description	File Size	Download
Linux x86	288.83 MB	<a href="#">jdk-8u121-nb-8_2-linux-i586.sh</a>
Linux x64	284.24 MB	<a href="#">jdk-8u121-nb-8_2-linux-x64.sh</a>
Mac OS X x64	338.8 MB	<a href="#">jdk-8u121-nb-8_2-macosx-x64.dmg</a>
Windows x86	317.35 MB	<a href="#">jdk-8u121-nb-8_2-windows-i586.exe</a>
Windows x64	326.89 MB	<a href="#">jdk-8u121-nb-8_2-windows-x64.exe</a>

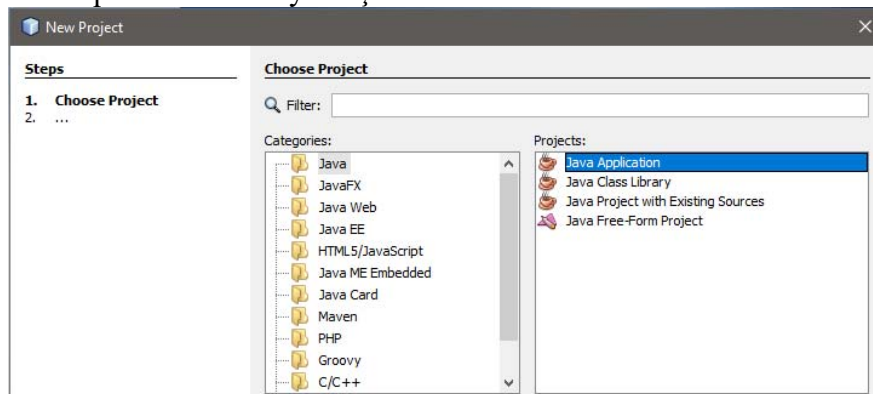
Kompüterləri Windows 32bit olanlar “x86”, 64bit olanlar “x64” versiyasını şərtləri qəbul edib endirə bilərlər. Proqramı endirdikdən sonra üzərinə sağ düymə sıxaraq “Run as administrator” sətirini seçin. Sistemin administratoru kimi proqramları yükləyəndə proses əsasən problemsiz başa çatır. Hər endirilən versiyada yükləmə addımları fərqli ola bilər. “Finish” düyməsi çıxana qədər irəliləyin.

### “Netbeans”-lə tanışlıq.

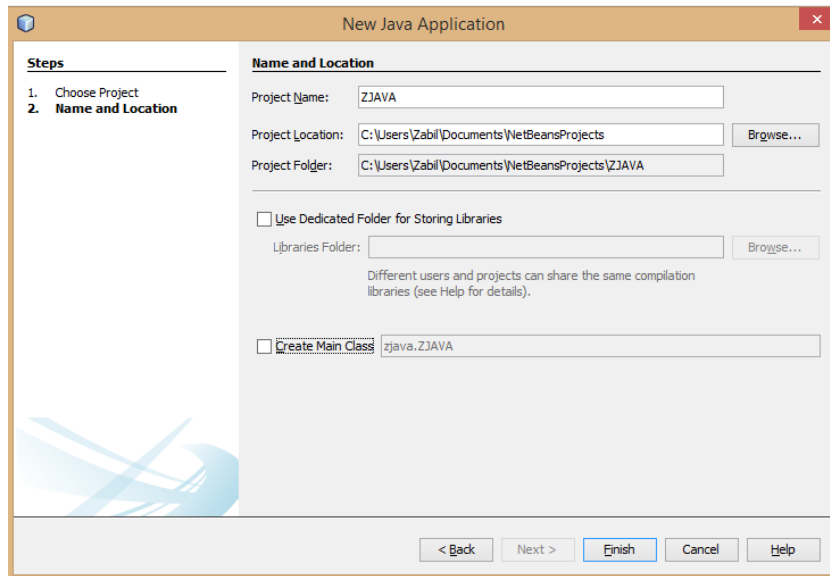
“Netbeans” -də proqram yazmaq üçün birinci yeni layihə(New Project) yaratmaq lazımdır. Yazacağımız proqramlar yaratdığımız layihənin tərkibində yer alacaqdır.




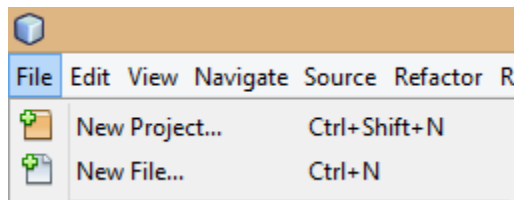
Açılan pəncərədə sizdən nə tipli layihə yaratmaq istədiyiniz soruşulur. Siz aşağıdakı şəkildə göstərildiyi kimi kateqoriyalardan(Categories) “Java” və layihələrdən(Projects) Java Application(Java Proqramı) seçin. Sonra Next(növbəti) düyməsini sıxaraq növbəti səhifəyə keçin.



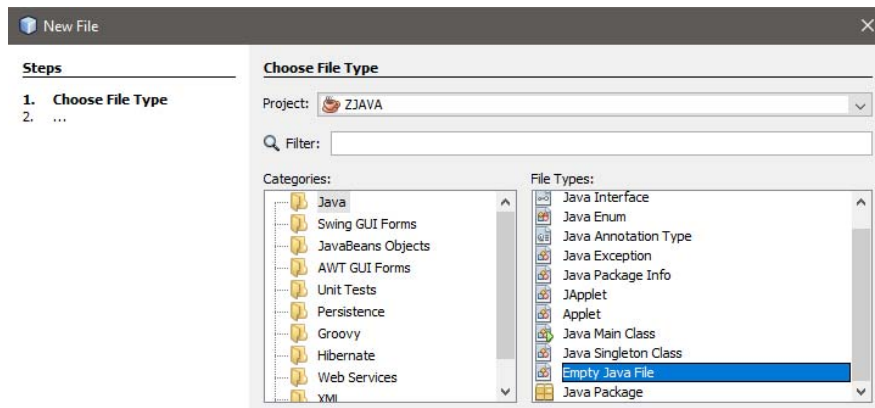
Layihəyə “ZJAVA” adını verirəm. Siz layihəni və sinifləri istədiyiniz kimi adlandırma bilərsiniz. Aşağıda “Create Main Class” qutusunu söndürün, məndə həmin hissənin qarşısında "zjava.ZJAVA" yazısı var. Sonda “Finish” düyməsini sıxın.



İndi layihəmizə Java proqramı yazmaq üçün yeni fayl əlavə edək. Bunun üçün aşağıdakı şəkildə göstərildiyi kimi  New File düyməsini sıxın.



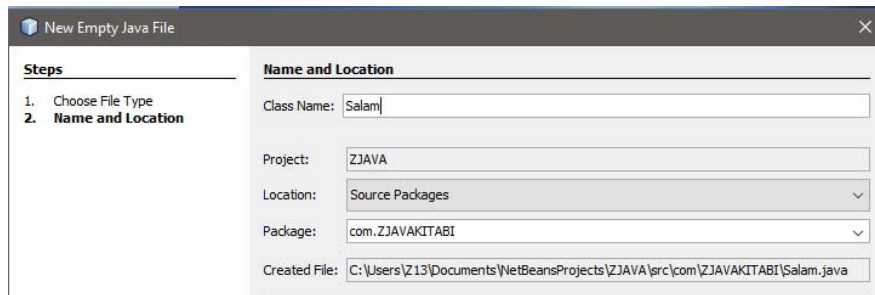
Açılan pəncərədə kateqoriyadan “Java” və File Type(Faylın tipi) bölməsindən Empty Java File(Boş Java Faylı) sətirini seçin.



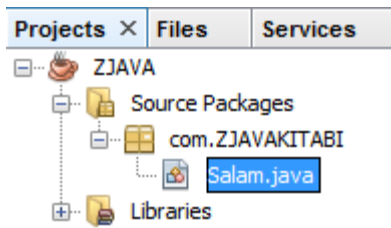
Növbəti səhifədə artıq müəyyən mənada proqram yazılışına başlayırıq.

### “Salamlama” proqramı

**Class Name**(Sınıfın adı) hissəsində proqramımızın adının “Salam” olmasını qeyd edirəm. “NetBeans”-də **Package**(Paket) java proqramlarını bir arada qruplaşdırmaq üçündür və adını “com.ZJAVAKITABI” kimi qeyd edirəm. **Created File**(Yaradılmış fayl) hissəsində yaratdığınız faylın saxlandığı ünvan göstərilir.



**Finish** düyməsini sıxdıqdan sonra “Netbeans”-də sol tərəfdə proqram faylının direktoriya üzrə necə yerləşməsi təsvir olunur.



“Salam.java” proqramını seçdikdən sonra aşağıda göstərilədiyi kimi sətirləri daxil etməyə çalışın.

```
1 package com.ZJAVAKITABI;
2
3 class Salam {
4     public static void main (String [] arguments){
5         // Menim birinci Java proqramım
6         String salamlama = "Salam dünya!";
7         System.out.println(salamlama);
8     }
9 }
```

Aşağıda proqramın strukturunu nələrin təşkil etdiyini səthi izah etmişəm. İlk baxışda qarışıq görünsə də zamanla bütün sətirləri özünüzlə lazım olacaq şəkildə dəyişdirə biləcəksiniz.

### *Package*

Proqramları qruplaşdırmaq, “class” proqrama ad vermək və onu sinifləndirmək üçündür. OOP-də izah etdiyimiz proqramı çağırmaq, “class”-la bağlıdır. Proqram boyunca harada salam vermək istəsək “Salam class”-nı çağırıb salam verdirə bilərik.

### *public static void main (String [] arguments)*

Bu sətiri sadəcə əzbərləyin. Java-da yazacağınız proqramlar əsasən belə başlayacaq. OOP tipli proqram yazdığınıza görə artıq bilərsiniz ki bir layihənin tərkibində çoxlu proqramlar ola bilər. Fərz

edək ki, layihəni yazdıq və bir icon(ikona) şəklinə gətirdik. İkona üzərinə sıxdığımızda sizcə “ZJAVA” layihəmiz hansı proqramı işlətməlidir. Bu zaman “java” sətirləri daxilində harada Main(Əsas) sözü ilə proqram qeyd olunubsa həmin proqramı birinci başladır. İlk başlanacaq proqramı “class” daxilində “public static void main (String [] arguments)” kimi qeyd edirsiniz.

### *Public*

Public(ictimai) yaratdığımız obyektin digər proqramlar tərəfindən istifadə olunması və çağırılması üçün açıq olduğunu bildirir. Gizli deyil, ictimaiyyətə açıqdır.

### *Static*

Static (sabit, dəyişməyən) “instance”-ı olmayan və dəyişdiyi zaman hər bir bağlantısına təsir edən “class”-ı bildirir. “Instance”-la çağırıla bilmir.

### *Void*

Həç bir dəyər geri qaytarmır. İxtisara salır.

### *String arguments*

Bu hissə “Netbeans”-dən istifadə etmədən “CMD” vasitəsi ilə “Java” proqramlarını “compile” etmək üçün istifadə olunur.

### *String*

“Basic”-də izah etdiyimiz “\$” işarəsi kimi dəyişəni təyin edir. “String”-in qarşısına nə yazsaq, o dəyişən müvəqqəti olaraq özünə yazı, rəqəm və başqa mümkün işarələri mənimsədə bilər. “String”-in bu dəfəki dəyişəninə “salamlama” adını vermişəm. Artıq “salamlama” = “Salam dünya!”. Dəyişənlərə istədiyiniz adı vermək mümkündür. Digər dəyişənlər və onların fərqləri proqram yazdıqca aydın olacaqdır.

### *Dalğalı, blok mötərizə { }*

Yazdığınız proqramların başlanğıc və bitmə nöqtəsini bildirir. Eyni zamanda proqram sətirlərini qruplaşdırır. Bu qruplaşdırmaya block(blok) deyilir. Bloklar digər blokların tərkibində ola bilər.

### *// comment(şərh)*

NetBeans-də "//Mənim birinci Java proqramım" sətiri boz rənglə göstərilmişdir. Cümlənin əvvəlindəki "//" şərh işarələri proqramın daxilində yazılmasına baxmayaraq kompüter üçün heç bir iş əmri vermir. Şərhlər nə üçün lazımdır? Fərz edək ki, proqramçı bir müddət sonra proqramının hansısa bir funksiyasını ayrıcalıqda götürmək istəyəndə əvvəlcədən proqramda qeyd etdiyi şərhlerle hansı sətirdə nə yazdığını anlaya bilər. Şərhlərin aşağıdakı növləri var:

1. // burda şərh yazılıb
2. /\* burda şərh yazılıb \*/
3. /\*\* burda şərh yazılıb \*/

### *System.out.println();*

Basic-də göstərdiyim nəticəni ekrana çıxartmaq üçün istifadə olunan əmr print-in artıq JAVA-dakı istifadəsini görürük. Proqramlaşdırma dilinin imkanları genişləndikcə yazı stilində dəyişmişdir. Burdakı ekrana çıxartmaq üçün istifadə olunan yazı şəklinin mürəkkəbliyi proqramın digər imkanlarının da olmasından xəbər verir. System(sistem), Out(çöl), println(print line(yeni sətirdən çap et) hərfi mənada da məqsədini aydın edir. Məlumatı sistemdən çölə çıxart və yeni sətirdən çap et anlamını verir.

**Qeyd:** System.out.println(); -bu yazını sadəcə "sout" yazıb sonra TAB düyməsinə basaraq sürətli şəkildə daxil edilə bilər.

**Qeyd:** public static void main (String [] arguments) -bu yazını sadəcə "psvm" yazıb sonra TAB düyməsinə basaraq sürətli şəkildə daxil edə bilərsiniz.


**Qeyd:** Println – yeni sətirdən çap edir, print isə eyni sətirdən çapı davam edir.

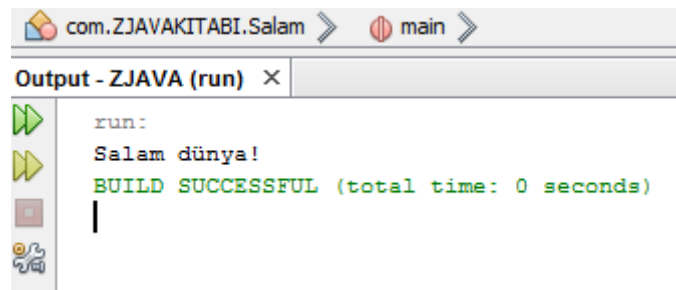
### *" ; " nöqtəli vergül*

Sətirlərin sonunu göstərmək üçün proqramlaşdırmada əsasən nöqtə vergüldən istifadə edilir. Əmrin bitməsini göstərir.

Ən sonda proqramı işlətmək üçün proqramı yaddaşa vermək sonra “RUN” etmək lazımdır. Yaddaşa vermək üçün “CTRL+S” və ya



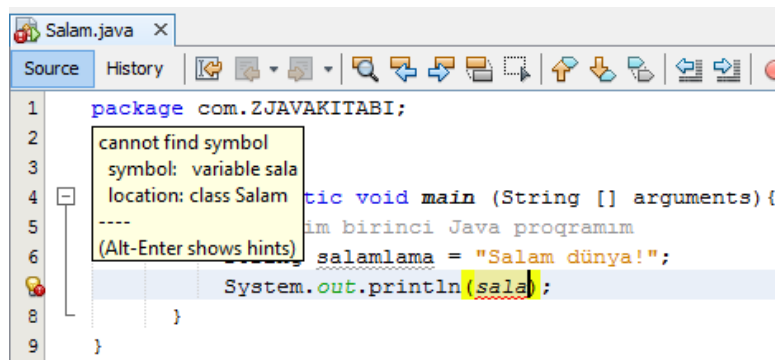
düyməsini sıxmaq, proqramı işə salmaq üçün F6 və ya  “RUN” düyməsini sıxmağınız yetərlidir. “NetBeans”-in daha bir üstünlüyü ondadır ki, yaddaşa verdiyiniz bütün fayllar avtomatik olaraq “*compile*” olunur. Proqramın sətirlərini düzgün yazdığınız halda aşağıdakı nəticəni əldə edəcəksiniz. Əgər hər hansısa bir səhv varsa sadəcə sətirləri yoxlayın. Nöqtəli vergül və ya adların eyni yazılması kimi səhvlər tez-tez təkrarlanır.



### **“NetBeans”-də səhvlərin aydınlaşdırılması**

Yazdığımız proqramda bilərəkdən cavabda göstərməli “*salamlama*” dəyişənini “sala” ilə əvəz edirəm, bu zaman proqram “RUN” ola bilməyəcək və solda qırmızı lampa bizə səhvin nə olduğunu göstərəcəkdir.





**Qeyd:** İşimizi sürətləndirmək üçün artıq bəzi prosesləri təkrarən görüntülə izah etməyəcəm. Buna misal olaraq yeni java faylı yarat dedikdə faylı yaratmaq üçün **CTRL+N** və ya “File/New file/Empty Java file” ardıcılığını özünüz əvvəldəki izahlardan təkrar etməlisiniz.

**Qeyd:** Növbəti proqramları əgər eyni “package”-in tərkibində yaratsanız “RUN” etməyinizdə problem yarana bilər. Yazılan proqramı **Save** etdikdən sonra proqramın üzərinə sağ düyməni sıxıb **RUN File** sətirini seçin və ya **Shift+F6**(Shift-in üzərinə sıxıb saxla, eyni zamanda F6 düyməsi sıx deməkdir) düymələrini sıxın. Proqram “RUN” ola bilmədikdə səhvlərin hansı sətirdə olmasına baxın.

**Qeyd:** Müəyyən əmrlərin çağırılması və ya daha başqa hansı funksiyaların mümkün olduğunu görmək istəyirsinizsə **CTRL+Space**(boşluq) düymələrini birlikdə sıxın, açılan pəncərədən seçilmiş əmri **ENTER** düyməsini sıxmaqla daxil edə bilərsiniz.

**Qeyd:** Proqramın səliqəli tərtibi müəyyən müddət sonra proqrama qayıdıb baxdığınızda nəyi nə üçün yazdığınızı anlamağa kömək edəcəkdir. “NetBeans”-də yazdığınız proqramın üzərinə sağ düyməni və ya **Alt+Shift+F** düymələrini sıxaraq yazıları Format(formaya salmaq) edə bilərsiniz. Şəkillər çox yer tutduğu üçün bəzi proqramların səliqəsiz sıxlıqda yazılmasını və bəzi əyri

mötərizələrin göstərilmədiyini bildirirəm. Yuxarıdakı misaldan da bəlli olduğu kimi package, class, public, dəyişənlər, print müəyyən aralıqda yazılmalıdır. Mötərizələr və tələb olunan simvollar şəkildə qeyd olunmayıbsa error veriləndə artıq qayda olaraq açılmış mötərizələrin qapanması, əmrlərin sonunun “;” ilə bitməli olmasından xəbərdarsınız.

### “JAVA”-da dəyişənlər

Dəyişənlər hər hansı bir məlumatı proqram boyunca özündə saxlayır. Proqramın iş prosesi boyunca lazım olan yerdə informasiyasını yaddaşdan çağırılmasını təşkil edir. Proqramlaşdırmada yadda saxlanılan məlumatın tipi göstərilməsi şərtidir. Artıq String dəyişənindən istifadə etmişik.

**String**-lə dəyişənlərə dəyər qazandırmaq üçün qoşa dırnaqdan istifadə edilir. Qoşa dırnaq daxilində yazılan bütün simvollar şəkil kimi qəbul edilir. Rəqəmləri rəqəm kimi deyil, rəqəmlərlə yazılmış bir görüntü kimi yadda saxlayır. Belə olduğu halda siz String dəyişəni ilə hesablama apara bilmirsiniz. Bildiyimiz kimi String dəyişəninə mənimsədilən mətnlər qoşa dırnaq daxilində yazılır. Bəzən mətnin daxilində qoşa dırnaq və digər funksiyalar lazım gəlir, bu zaman aşağıdakı nümunədən istifadə edə bilərsiniz. String-in müxtəlif funksiyaları mövcuddur (\' –tək dırnaq, \"qoşa dırnaq, \\- geri sləş, \t - TAB(8 simvol sağa boşluq), \b - Backspace(bir simvol geriye, sola pozma), \r - Carriage return(sətirbaşına qayıtma), \n –Newline(yeni sətir)). Yeni Java faylı yaradıb istədiyiniz kimi adlandırın və diqqət edin ki class-ın adı faylın adı ilə eyni olsun. Yoxsa proqram RUN olanda error verəcək.

```
package com.ZJAVAKITABI;
class StringParametr {
    public static void main(String[] arguments) {
        String reqem1 = "10";
    }
}
```

```

String reqem2 = "10";
String cavab = reqem1 + reqem2;
String i = "1 Zabilin \'28\' yaşı var."; //Tək dırnaq
String b = "2 Zabilin \"28\" yaşı var."; //Qoşa dırnaq
String a = "3 Zabilin \\28\\ yaşı var."; //Backslash işarəsi
String y = "4 Zabilin \t28 yaşı var."; //TAB - 8 simvol sağa boşluq
String e = "5 Zabilin \b28 yaşı var."; //Backspace - Bir simvol sola silmə
String v = "6 Zabilin \r28 yaşı var."; //Carrige return - Sətir başına qayıtma
String z = "7 Zabilin \n28 yaşı var."; //New line - yeni sətirdən
System.out.println(i);
System.out.println(b);
System.out.println(a);
System.out.println(y);
System.out.println(e);
System.out.println(v);
System.out.println(z);
System.out.println(cavab); }}
-----
-run:
1 Zabilin '28' yaşı var.
2 Zabilin "28" yaşı var.
3 Zabilin \28\ yaşı var.
4 Zabilin      28 yaşı var.
5 Zabilin28 yaşı var.
28 yaşı var.
7 Zabilin
28 yaşı var.
1010
-----

```

**QEYD:** (+) işarəsi String daxilində rəqəmləri və dəyişənləri görüntü olaraq yan-yana eyni sətirdə çapa vermək üçün istifadə edilir.

**QEYD:** Javada “String” class-dır, təkcə dəyişən tipi deyil.

### “+=” operatoru

(+=) operatoru String-ə bir neçə dəyərin verilməsində istifadə olunur. Fərz edək ki, qeydiyyat pəncərəsi düzəldirsiniz və onun təvəllüd hissəsindəki məlumatları bir sətərə yığmaq istəyirsiniz.

```
-----  
package com.ZJAVAKITABI;  
class PlusEqual {  
    public static void main(String[] arguments) {  
        String TEVELLUD = "Birinci variant" + " : ";  
        TEVELLUD = TEVELLUD + "13" + "/";  
        TEVELLUD = TEVELLUD + "06" + "/";  
        TEVELLUD = TEVELLUD + "1987";  
        String Dogumtarixi = "Ikinci variant" + " - ";  
        Dogumtarixi += "13" + "/";  
        Dogumtarixi += "06" + "/";  
        Dogumtarixi += "1987";  
        System.out.println(TEVELLUD);  
        System.out.println(Dogumtarixi); } }  
-----  
run:  
Birinci variant : 13/06/1987  
Ikinci variant - 13/06/1987  
-----
```

Yuxarıda göstərilən misalda “TEVELLUD” dəyişəninin təkrar-təkrar yazıldığını görürsünüz. (+=) operatoru isə “Dogumtarixi” dəyişənini daha az təkrar etməklə nəzərdə tutulmuş məsələni həll edir. Təvəllüd tarixini tam şəkildə bir sırada yazmaq mümkündür(TEVELLUD= “13/06/1987”). Yuxarıdakı misal digər dəyişənlərin sonradan necə əlavə olunduğunu göstərir. Bu misalda String-ə hər yeni mənimsədilən məlumat necə ardıcıl düzülür onu görə bilərsiniz.

### “.toUpperCase” və “.toLowerCase”

Programınıza daxil edilmiş yazıları sadəcə “.toUpperCase()” və “.toLowerCase()” metodlarını yazmaqla bütün hərflərini böyüdə və ya kiçildə bilərsiniz.

```
package com.ZJAVAKITABI;  
class BoyukKicik {  
    public static void main(String[] arguments) {  
        String boyuk = "zabil ibayev";  
        String kicik = "ZABIL IBAYEV";  
        System.out.println(boyuk.toUpperCase());  
        System.out.println(kicik.toLowerCase());  
    }  
}
```

run:

ZABIL IBAYEV

zabil ibayev

### “==” və “.equals()”

Fərz edək ki, qeydiyyat səhifəsi yaratmışıq. Səhifədə şifrə daxil edilir. İstifadəçinin şifrəni düzgün daxil etməsindən əmin olmaq üçün ondan şifrəni təkrar yazmasını tələb edirik. Bu zaman bizdən asılı olan məsələ iki ayrı xanada daxil edilmiş şifrənin bir-biri ilə eyni olmasını yoxlanılmasını təşkil etməkdir. Bu zaman “.equals()” metodundan və ya “==” qoşa bərabərlik operatorundan istifadə edirik.

```
package com.ZJAVAKITABI;  
class CompareString {  
    public static void main(String[] arguments) {  
        String reqem1 = "15";  
        String reqem2 = "10";  
        System.out.println(reqem1.equals(reqem2));  
        System.out.println(reqem1 == reqem2);  
    }  
}
```

```
-run:
-true
>false
```

Nəticədə gördüyünüz kimi cavab “False” yalnızdır. Əgər bərabər olsa idi “True” doğrudur olacaqdı.

**QEYD:** “.equal()” yazıların yoxlanılmasında böyük və kiçik hərfə fikir verir. Əgər “Zabil” və “zabil” sözlərini yoxlasanız, cavab eyni deyil “False”-dur. Kompüter hər əmri necə var elə anlayır, böyük və kiçik hərfləri fərqli görür. Əgər hərflərin kiçik və ya böyük olmasına baxmayaraq mətni yoxlamaq istəyirsinizsə o zaman “.equalsIgnoreCase()” (Ignore-fikir vermə; Case-hal, forma) hərfin böyük kiçik olmasına fikir vermədən bərabərliyi yoxla metodundan istifadə edin.

### “.length()”

Daxil edilən şifrənin minimum 6 simvoldan ibarət olmasını istəyirsiniz. Bu zaman yazılan simvolların sayını “length()” - uzunluq, metodu ilə aşağıdakı kimi ölçmək olar. “.length()” String class-nın metodudur.

```
-package com.ZJAVAKITABI;
-
-|class UzunluqSringCompare {
-|public static void main(String[] arguments) {
-|    int minimum = 6;
-|    String sifre = "IBAYEV";
-|    int olcmek = sifre.length();
-|    System.out.println(olcmek == minimum); }}
-
-|
```

```
-run:
>true
```

Ölçü olaraq 6 simvol qoymuşduq və “İBAYEV” yazısı 6 simvoldur ona görə nəticə True- yəni doğrudur.

### “.Contains()” və “.indexOf()”

**Contains** – saxlamaq, daxilində olma mənasına gəlir. Verilmiş mətn daxilində müəyyən bir String-i axtara bilir. Əgər verilmiş String mətn daxilində varsa True, yoxdursa False nəticə verir.

```
package com.ZJAVAKITABI;
class ContainsOF {
    public static void main(String[] arguments) {
        String mətn1 = "Zabil Ibayevin 28 yaşı var";
        String mətn2 = "Zabil Ibayevin 29 yaşı var";
        System.out.println("1-ci cavab :" + mətn1.contains("29"));
        System.out.println("2-ci cavab :" + mətn2.contains("29"));
        System.out.println("3-cü cavab :" + mətn2.indexOf("29"));
        System.out.println("4-cü cavab :" + mətn2.indexOf("30")); } }
run:
1-ci cavab :false
2-ci cavab :true
3-cü cavab :15
4-cü cavab :-1
```

**indexOf** - indeksi neçədir, axtarılan String neçənci simvoldan başlayırsa o nöqtəyə kimi olan simvolların sayını bildirir. Əgər String ümumiyyətlə yoxdursa “-1” cavabını verir.

**Char** yaddaşında bir simvol saxlaya bilir. Bir hərf və ya bir ədəd. Tək dırnaq daxilində yazılır. String-dən digər bir fərqi dəyişənləri ilə riyazi hesablamalar etmək mümkündür.

```
package com.ZJAVAKITABI;
class Char {
    public static void main(String[] arguments) {
        char reqem1 = 'A';
        char reqem2 = '0';
```

```
System.out.println(reqem1);  
System.out.println(reqem2); }}
```

```
run:
```

```
A
```

```
0
```

## Rəqəm dəyişənləri

Java-da rəqəmlə işləyən dəyişən tipləri miqyaslarına görə bölüşdürülmüşdür. Bunlar Integer, Double, Float, Byte, Short və Long-dur.

**Integer(int)** tam ədəd deməkdir. Öz yaddaşında 2.14 milyard müsbət (2147483647) və 2.14 milyard mənfi (-2147483648) tam ədəd saxlaya bilər.

**Double** - integer-dəndə böyük həcmdə ədədləri saxlaya bilər. 300 simvola yaxın ədəd. Eyni zamanda float(kəsr) ədədlərini də saxlaya bilər.

**Floating** – Bu ədədlər hər hansı bir yerində nöqtə ilə bölünmüş olması deməkdir. Kəsr ədədlərə bənzəyir. 38 simvola qədər yadda saxlaya bilər. Float ədədləri təyin edərkən sonlarına “F” əlavə edərək fərqləndirə bilərsiniz.

**Byte** - 128 dən 127 kimi tam ədədləri yadda saxlaya bilər.

**Short** - 32767 dən -32768 kimi tam ədədləri yadda saxlaya bilər.

**Long** - 9.22 quintliondan 9.22 quintiliona kimi tam ədədləri yadda saxlaya bilər.

## Hesablamalar ilə bağlı praktiki məsələlər

### Riyazi operatorlar

+ toplama

- çıxma

\* vurma

/ bölmə

% qalıq göstəricisi



## Dəyişənləri qruplaşdırma

Zabilin çəkisi normalda 90 kilodur. Tətilə çıxandan sonra çəkisi 15 kilo artmışdır. Tətildən sonra idmanla məşğul olmuş çəkisi 10 kilo azalmışdır. Zabilin indiki kilosu neçədir?

```
package com.ZJAVAKITABI;
class Weight {
    public static void main(String[] arguments) {
        int Zabilin_kilosu, idman, tetil, indiki_kilo;
        Zabilin_kilosu = 90;
        tetil = 15;
        idman = -10;
        Zabilin_kilosu = Zabilin_kilosu + tetil + idman;
        System.out.println(Zabilin_kilosu); }}
run:
95
```

Yuxarıdakı məsələdə hər dəyişənin qarşısında yenidən `int` yazılmayıb. Məsələdəki dəyişənlərin tam ədəd olması nəzərə alınıb və bir sətirdə qruplaşdırılıb. Əgər aşağıdakı kimi fərqli tiplər olsa zaman ayrı-ayrılıqda yazılmalıdır.

```
package com.ZJAVAKITABI;
class Reqemdeyishenler {
    public static void main(String[] arguments) {
        int tam_ədəd = 10;
        final int SABITƏDƏD = 10;
        float kəsr_ədəd = 2.5F;
        double cavab = tam_ədəd + SABITƏDƏD + kəsr_ədəd;
        System.out.println(cavab); }}
run:
22.5
```

Yuxarıdakı misalda double-ın tərkibində integer-i və float-ı saxlaya bilməsi göstərilib. Eyni zamanda final əmri ilə hər hansı bir dəyişənə constanat(sabit) ədəd verilməsi qaydası təsvir edilib. Integer qarşısında final yazılıbsa, o dəyişənin dəyərini sabit saxlayır və dəyişməsinə imkan vermir. Adətən konstanatlar böyük hərflə yazılır. SABİTEDED dəyişəni proqram boyunca ancaq 10-a bərabər olacaq, dəyəri dəyişməyəcəkdir.

#### Qalıqın ekrana verilməsi

Ədəd tam bölünəndən sonra qalan ədəd ekrana verilir. Burda nəticə 1 göstərilir.

```
-----  
package com.ZJAVAKITABI;  
class Riyazimisallar {  
    public static void main(String[] args) {  
        int z = 16 % 3; //qalıq göstərir  
        System.out.println(z);  
    }  
}
```

```
run:
```

```
1  
-----
```

#### Bir-bir artırma və azaltma

Proqramlaşdırma dillərində populyar olan funksiya, bir-bir artırma və ya azaltma. Dəyişənin yanına “++” və ya “--” yazmaqla əldə edilir. Artırma və ya azaltma işarələri dəyişənin önündə yazılırsa hesablamadakı yerindən asılı olmayaraq birinci dəyərini dəyişəcəkdir. Əgər dəyişəndən sonra yazılsa hesablama daxilində ilk verilən dəyərini daşıyacaqdır.

```
-----  
package com.ZJAVAKITABI;  
class Birtoplama {  
    public static void main(String[] args) {  
        int z, cavab1, a, cavab2, i, x;  
    }  
}
```

```

| z = 3; a = 3; x = 2; i = ++x;
| cavab1 = 10 * z++;
| cavab2 = 10 * ++a;
| System.out.println("1-ci: " + cavab1);
| System.out.println("2-ci: " + cavab2);
| System.out.println("3-cü: " + i); }}

```

```

- run:

```

```

| 1-ci: 30
| 2-ci: 40
| 3-cü: 3

```

### Riyazi operator ardıcılığı

1. Bir-bir artırma və ya azaltma
2. Vurma, bölmə və faiz
3. Toplama və çıxma
4. Müqayisə
5. Axırda "=" hesablanmış nəticəni dəyişənə mənimsədir.

```

package com.ZJAVAKITABI;
class HesabArdicilligi {
public static void main(String[] args) {
int z, cavab;
z = 5;
cavab = ++z * 6 + 4 - 1 * 10 / 2;
System.out.println(cavab); }}

```

```

- run:

```

```

- 35

```

**Qeyd:** Mötərizə daxilindəki hesablamalar ayrılıqda hesablanır.  
Daha sonra çöldəki hesaba qatılır.

$$Z = 5 * (3 + 2) = 5 * (5) = 25$$

## Dəyişənlərlə hesablama və görüntü

2 dəyişəni toplayıb, cavabda təkcə nəticəni yox eyni zamanda toplama prosesini görüntü olaraq əks etdirin. Bunu digər riyazi operatorlarla yoxlayın.

```
package com.ZJAVAKITABI;  
class HesablamaGorunus {  
    public static void main(String[] args) {  
        int eded1 = 5, eded2 = 20;  
        System.out.println(eded1 + "+" + eded2 + "=" + (eded1 + eded2)); }  
}  
run:  
5+20=25
```

## Məntiq dəyişənləri və məntiq operatorları

Boolean məntiq dəyişəninin iki tipi var : true (doğru) və false (yalnış). Məntiq operatorları əsasən şərtlərin (if, while, do, else if) yazılmasında istifadə olunur. Məntiq operatorları aşağıdakı mənalara əks etdirirlər.

(&&) - və

(||) - və ya

(==) - eyni dəyərdədir    (!) - inkar, deyil

**QEYD:** public, class, true, false –dan başqa bütün adları dəyişənlərə vermək olar. Dəyişən adları bir hərfə, ( ) aşağıdan xətlə və ya (\$) dollar işarəsi ilə başlaya bilər. Dəyişənlərin böyük və ya kiçik hərfə yazılmasına diqqət edin. Əgər hər hansı əməliyyatda, ən adi println-də belə dəyişəni böyük-kiçik hərf səhvi ilə yazsanız program error(səhv) verəcək.

### Şərt operatorları – (==), (!=), (<), (>), (<=), (>=), (?)

Yuxarıda izah olunan operatorlar təsdiqlənilsə icra edilir, “?” xüsusi şərtidir, əgər birinci şərt təsdiqlənilmərsə ikinci nəticəni icra edir.

**If-** Əgər mənasına gəlir, şərtin başlandığı əməldir, **else-** başqa, digər halda mənasına gəlir. İstəyirəm birbaşa məsələ üzərində şərt əməllərini izah edim. Məsələ məktəbi bitirən Bilal haqqındadır.

```
package com.ZJAVAKITABI;
class IfElse {
    public static void main(String[] args) {
        int cari_il,  dogum_ili,  yasi,  TQDK_bali,  ixtisas_Muhendis,
        ixtisas_Memar;
        String adi_soyadi = "Bilal Hacıyev";
        cari_il = 2014;
        dogum_ili = 1995;
        TQDK_bali = 300;
        ixtisas_Muhendis = 600;
        ixtisas_Memar = 500;

        if (TQDK_bali >= ixtisas_Muhendis) {
```

```

System.out.println("Siz      mühəndislik      ixtisasına      qəbul
olunmusunuz");}

else if (TQDK_bali >= ixtisas_Memar) {
System.out.println("Siz Memarlıq ixtisasına qəbul olunmusunuz");}

else if ((yasi = cari_il - dogum_ili) <= 18){
System.out.println("Sizin "+yasi+"yaşınız var, əsgərliyə getmək
üçün azdır"); }

else { System.out.println(adi_soyadi+"sən oxumadığın üçün
əsgərliyə gedirsən!"); }}}
-----
run:
Bilal Hacıyev sən oxumadığın üçün əsgərliyə gedirsən!
-----

```

Kodlardan göründüyü kimi bir neçə integer dəyişənim var. Universitetə qəbul ballarının şərtlərinə uyğun olaraq Bilalın hansı ixtisasa qəbul olacağı müəyyənləşdirilir, əgər TQDK balı uyğun deyilsə və yaşı əsgərə(müddətli həqiqi xidmət) getməyə uyğundursa onu göndəririk dağlar qoynunda xidmətə ☺ Bir az texniki şəkildə belə deyə bilərik:

**Əgər (şərt) {əgər şərt düzdürsə çap edilir}  
if (5>0) System.out.println("5 sıfırdan böyükdür");.**

Sadəcə bir if şərti ilə məsələ bitə bilər. Şərt təsdiqlənsə çapa verilir, əgər təsdiqlənmirsə heç bir cavab ekrana çıxmır. Şərtin düz olmadığı halda nəticə görmək istəyirsinizsə o zaman else if (yeni şərt yazılır){istənilən nəticə çapa verilir}.

### Sual operatoru

Fərz edək ki, proqramınızdan qeydiyyatdan keçən şəxsə Cənab və ya Xanım kimi müraciət etmək istəyirsiniz bu zaman “?” operatorundan aşağıdakı kimi istifadə etmək olar.

```
package com.ZJAVAKITABI;  
class Sualsherti {  
    public static void main(String[] args) {  
        String Cinsi = "kişi";  
        System.out.print((Cinsi.equals("kişi"))? "Cənab " : "Xanım ");  
        System.out.println("Zabil Ibayev"); } }  
run:  
Cənab Zabil Ibayev
```

Ardıcılıq aşağıdakı kimidir:

1. Şərt yazılır
2. “?” işarəsi
3. “Şərt doğrudursa 1-ci cavab qoşa dırnaqda String” və ya dırnaqsız integer
4. İki nöqtə ilə “:” və ya
5. “Şərt doğru deyilsə 2-ci cavab qoşa dırnaqda String” və ya dırnaqsız integer

### Şərt əmrləri - switch, case, break, default

If və else ilə yazılan proqramlarda şərtlər çox olanda else if ()-dən bir neçə dəfə istifadə etməli oluruq. Şərtləri çox olan məsələlər üçün daha sadə olar ki switch-dən istifadə edilsin.

**Switch( )**- birindən digərinə keçmə, dəyişmə mənasına gəlir. Qarşısındakı mötərizə daxilində String, integer, char və digər dəyişənlər yazıla bilər.

**Case** – hal, proses mənasına gəlir. Qarşısındakı dəyər switch-in mötərizəsindəki dəyərlə eyni olduğu halda şərt icra edilir. Əgər eyni deyilsə o zaman növbəti case yoxlanılır.

**Break** - fasilə vermək anlamındadır. Case-dən sonra icranı dayandırır. Hər bir case icra olundandan sonra break yazılmasa proqram case-dən sonra yazılan digər mümkün əməlləri yerinə yetirəcəkdir. Qısacası lazım olan şərt icra olundu dayan əmrini verir

**Default** - yerinə yetirə bilmədikdə mənasını daşıyır. Bütün case-lərin şərtləri uyğun gəlmədiyi halda standard həll icra olunur.

```
package com.ZJAVAKITABI;
class SwitchCase{
    public static void main(String[] args) { int TQDK = 500;
    switch (TQDK){
        case 600:
            System.out.println("Mühəndislik ixtisasına qəbul oldunuz");
            break;
        case 500:
            System.out.println("Memarlıq ixtisasına qəbul oldunuz");
            break;
        case 300:
            System.out.println("Aşağı bal topladığınız üçün qəbul olmadınız");
            break;
        default:
            System.out.println("TQDK balınızı daxil edin"); break; }}}
run:
Memarlıq ixtisasına qəbul oldunuz
```

**Qeyd:** Switch daxilində hansı tip dəyişən (integer, string, char) varsa case-də də həmin tip dəyişən istifadə olunmalıdır. Əks halda compile olmayacaq. Digər bir məsələ dəyişənlərin adları yox onların dəyəri case-ə yazılmalıdır. Aşağıda göstəriləni kimi:



```
String    ad = "Zabil";    -- case "Zabil"
char simvol = ' Z ';      -- case ' Z '
int  yaş = 29;             -- case 29
```

## Dövr əməlləri - for , do, while

İngilis dilində dövr əməlləri “loop” adlanır. Dövr, daxilində yazılan şərtləri və hesablamaları işləyərək şərtlərdə verilmiş son həddə çatana kimi hesablamaları təkrar edir.

### For

Əsasən ədədləri ardıcıl və ya müəyyən şərtə uyğun olaraq saymaq kimi əməlləri yerinə yetirir. Müəyyən olunmuş həddə dövr ilə çatmaq üçün dizayn olunmuşdur. For dövrünün daxilində başlanğıc ədəd, şərt və şərtə çatmaq üçün hesablama funksiyası olur. Aşağıdakı misalda şərt olaraq 15-dən kiçik və ona bərabər olan ədədləri sıralamaq istəyirəm. Bunun üçün başlanğıcda z dəyişəni şərt ilə yoxlanılır 15-ə bərabər olmadığı üçün “z++” ilə üzərinə bir ədəd əlavə edilir. Sonra həmin ədəd təkrar yoxlanılır və 15-ə bərabər olana qədər davam edir.

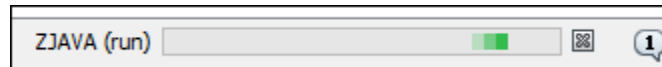
```
-----
package com.ZJAVAKITABI;
class ForLoop {
public static void main(String[] args) {
for (int z = 1; z <= 15; z++) {
System.out.print(z+" "); }}
-----
run:
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
-----
```

Yuxarıdakı izaha uyğun olaraq ədədlər sıralananda 2-dən başlamalıdır amma nəticədə 1-dən 15-ə qədər sıralanmışdır. For-un bu xüsusiyyətini gələcəkdəki hesablamalarınızda nəzərə almaq yaxşı olar. For daxilindəki şərtlər “;” ilə ayrılmalıdır. Aşağıdakı məsələdə “for” dövründən sonra “if” şərt əmrinin necə istifadə

olunması götürilib. Məsələnin şərti belədir: 15-ə qədər olan ədədlərdən hansı 3-ə qalıqsız bölünür.

```
-----  
package com.ZJAVAKITABI;  
class ForLoop2 {  
public static void main(String[] args) {  
for (int z = 0; z <= 15; z++) {  
if ( z % 3==0) System.out.print(z+" "); }  
System.out.println("- Soldakı rəqəmlər 3-ə bölünərkən qalığı 0  
olanlardır");}}  
-----  
run:  
0 3 6 9 12 15 - Soldakı rəqəmlər 3-ə bölünərkən qalığı 0 olanlardır  
-----
```

**QEYD:** Yazdığınız program sonsuz dövrə girdiyi halda NetBeans-də nəticə göstərilən pəncərənin altında RUN proses zolağına x basaraq dövrü dayandıra bilərsiniz.



### Mürəkkəb For əmrləri

Yuxarıda izah olunan məsələ üzərində for dövrünün quruluşunu daha rahat anlamaq mümkündür. Əslində for-un qarşısındakı mötərizədə dəyişənin tipini yazmamaq və ya bir neçə dəyişəni eyni vaxta daxil etmək mümkündür. Aşağıdakı misalda 5-in vurma cədvəlini bu üsulla əldə etmişəm. Z=5 mötərizədən çöldə yazılıb.

```
-----  
package com.ZJAVAKITABI;  
class ForLoop3 {  
public static void main(String[] args) {  
int z=5, a;  
for (a = 0; z*a <= 50; a+=1) {  
System.out.println(z+"*"+a+"="+ (z*a)); } } }  
-----
```

```

run:
5*0=0
5*1=5
5*2=10
5*3=15
5*4=20
5*5=25
5*6=30
5*7=35
5*8=40
5*9=45
5*10=50

```

## While

While(nə qədər ki; o vaxta qədər ki;) for-dan fərqli olaraq bir mümkün vəziyyətin dəyişəcəyi vaxta qədər şərti icra etmir. Əgər şərt doğrudursa(true) dövr sonsuz davam edir və şərtlər təkrar-təkrar yoxlanılır. Aşağıdakı məsələdə havanın temperaturu 0 dərəcədən aşağı olduğu halda sobanın yandırılması şərti verilmişdir.

```

package com.ZJAVAKITABI;
class WhileLoop {
public static void main(String[] args) {
int hava = -1;
while (hava < 0) {
System.out.println("Sobanı yandır qış gəldi");
break; } } }
run:
Sobanı yandır qış gəldi

```

Şərt true olan kimi loop(dövr) işə düşür. Əgər false olsa heç bir addım atılmır. Bildiyimiz kimi şərt true olduğu halda sonsuz dövr gedir. Belə halda loop-un dayandırılması üçün break əlavə

olunmalıdır. Şərt dəyişərsə yenidən loop aktiv gözləmədə ola bilər ki şərt yenidən true olsun. Aşağıdakı məsələdə şərtin yanlış olduğu ana qədər dövrün işləməsi göstərilibdir.

```
-----
package com.ZJAVAKITABI;
class WhileLoop2 {
    public static void main(String[] args) {
        int konfet= 5, aclıq=1;
        while (aclıq <= konfet) {
            System.out.println("Ac idim və konfetin "+aclıq+"-"+"ni yedim. ");
            aclıq++;}}
-----
-run:
Ac idim və konfetin 1-ni yedim.
Ac idim və konfetin 2-ni yedim.
Ac idim və konfetin 3-ni yedim.
Ac idim və konfetin 4-ni yedim.
Ac idim və konfetin 5-ni yedim.
-----
```

Ac olanda cibindəki 5 konfeti yeyən adamın hesabatını görürsünüz. “While” daxilindəki şərtlərin və digər funksiyaların istifadə olunma qaydasını təsvir edən sadə proqram.

### Continue

Yuxarıda yazdığım konfetlə bağlı proqramda yəqin diqqət yetirmişiniz ki 3, 4 ilə qurtaran rəqəmlərin sonluğu düzgün yazılmayıb. “3-ni”, “4-ni” yazılıb. “Continue” bizə 3 və 4 ədədi yarandıqda həmin case-ə aid olan şəkilçini ekrana verməyə kömək edəcək. “Break” yazılan sətirdə proqram tamamı ilə dayanır. “Continue” isə dövrün həmin sətirdən aşağıya getməsinə imkan vermir və dövrü proqramı əvvəlindən aşağı gəlməyə məcbur edir.

```
-----
package com.ZJAVAKITABI;
class WhileCaseContinue{

```

```

| public static void main(String[] args) {
|     int aclıq=0, konfet= 5;
|     while (aclıq <= konfet) {
|         aclıq++;
|         switch (aclıq) {
|             case 3: System.out.println
| ("Ac idim və konfetin "+aclıq+"-"+nü yedim. ");
|             continue;
|             case 4: System.out.println
| ("Ac idim və konfetin " + aclıq+ "-"+nü yedim. ");
|             continue;
|             case 6: System.out.println
| ("Ac idim və konfetin " + aclıq+ "-"+nı yedim. ");
|             continue;
|             default : System.out.println
| ("Ac idim və konfetin " + aclıq+ "-"+ni yedim. "); } } } }
| -----
| -run:
| Ac idim və konfetin 1-ni yedim.
| Ac idim və konfetin 2-ni yedim.
| Ac idim və konfetin 3-nü yedim.
| Ac idim və konfetin 4-nü yedim.
| Ac idim və konfetin 5-ni yedim.
| Ac idim və konfetin 6-nı yedim.
| -----

```

## Do while

Do(yerinə yetirmək) – demək olar ki while kimidir. Şərt “True” olduğu halda dövr sonsuz davam edir. “False” olduğu halda dövr dayandırılır. “Do while” -ın “while”-dan fərqi şərtin “false” olduğu halda belə, ən azı bir dəfə nəticəni ekrana verir.

```

| -----
| package com.ZJAVAKITABI;
| class DoWhile{
|     public static void main(String[] args) {

```

```

int bu_il=2016;
System.out.print("Zabilin 29 yaşı var.");
do{System.out.println(" Növbəti il Zabilin 31 yaşı olacaq.");}
while(bu_il>2017); }
run:
Zabilin 29 yaşı var. Növbəti il Zabilin 31 yaşı olacaq.

```

Bu məsələdə “Növbəti il Zabilin 31 yaşı olacaq” false olmasına baxmayaraq bir dəfə ekrana verilibdir.

### Dövrün adlandırılması

Dövrün adlandırılması dedikdə müəyyən bir dövrün qarşısına ona vermək istədiyiniz adı və “:”-ni yazırsınız. Daha sonra dövr müəyyən həddə çatanda adı olan dövrü qeyd edərək ona istədiyiniz əmri verə bilərsiniz.

```

package com.ZJAVAKITABI;
class ayin_gunleri{
public static void main(String[] args) {
int ilin_gunleri= 365;
ayin_gunleri:
while (ilin_gunleri <= 365){
for (int i=1; i<=365; i++) {System.out.print(i+ " ");
if (i > 30){ break ayin_gunleri; }}}} }
run:
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
26 27 28 29 30 31

```

Yuxarıdakı proqramda ilin günlərini dövrə salmışam və hər 31 gündən sonra aya bölgü aparmaq istəmişəm. While-ın yuxarısında yazılan “ayin\_gunleri” while dövrü üçün seçilmiş şərti addır. Dövr 31-ə çatanda break-in qarşısında yazılan dövr dayandırılır.

## Array

Array – matris, düzülüş deməkdir. Bir ad altında bir neçə dəyişənin birləşməsi üçün istifadə edilir. Bəzən daxil edilən məlumatlar həddindən artıq çox olur. Hər bir dəyər üçün yeni bir dəyişən yazmaq mümkün olmur. Bu səbəbdən eyni tipli dəyişənləri bir ad altında massiv formasında yadda saxlayırıq.

	Login		Password
0	1	0	simsim
1	2	1	gizlişifrə
2	3	2	555

Massivlər daxilindəki elementləri 0-dan başlayaraq sayır. Birinci dəyər 1-dən başlamır. İki ölçülü array iki üsulla yaradıla bilər.

**Birinci üsul** - Array yaradarkən onun tipi(int, String, char...) qeyd edilir. Sonra array-in dəyişəninə ad (login və ya password) verilir. Sonra array olmasını göstərən “ [ ] ” kvadrat mötərizələr daxil edilir. Bərabərlikdən sonra “ { } ” əyri mötərizələr daxilində qeyd olunan dəyərlər “element” adlanır və aralarında vergül qoyulmaq şərti ilə ardıcıl qeyd olunurlar. Aşağıdakı məsələdə istifadəçi adı və şifrəsinin massiv şəklində yadda saxlanıldığı göstərilir. Məsələ qeydiyyatdan keçən şəxsləri matris cədvəlinə uyğun olaraq yoxlamaqdır. Yuxarıdakı cədvəldə göstəriləyi kimi “login” üzrə 3 və “password” üzrə 3 məlumat qeyd edilmişdir.

```
package com.ZJAVAKITABI;
class Array {
public static void main(String[] args) {
String login_e = "2";
String pass_e = "gizlişifrə";
String login[] = {"1", "2", "3"};
String pass[] = {"simsim", "gizlişifrə", "555"};
```





```

| beş_dildə_salam[3] = "Privet";
| beş_dildə_salam[4] = "Bonjour";
| beş_dildə_salam[5] = "Ciao";
|
| System.out.println("Ümumi matris sayı " + beş_dildə_salam.length
| + " olmasına baxmayaraq " + (beş_dildə_salam.length - 1) + " dildə
| salam verilir: ");
| for (int i = 1; i < 6; i++)
| {System.out.println (i + ": " + beş_dildə_salam [i] + " "); }}
|-----
|run:
|Ümumi matris sayı 6 olmasına baxmayaraq 5 dildə salam verilir:
|1: Salam
|2: Hello
|3: Privet
|4: Bonjour
|5: Ciao
|-----

```

**Qeyd:** Array-in sayı başlanğıcda göstərilməlidir. Array-lərə əvvəlcədən nəzərdə tutulmamış element əlavə etmək istəsəniz mütləq ümumi array sayında dəyişiklik etməlisiniz. İstifadə etdiyimiz “**.length**” metodu array daxilindəki elementlərin sayını göstərir. Əgər array-in elementləri qeyd olunmazsa default olaraq elementlər aşağıdakı dəyərləri qazanacaqlar.

```

|-----
|package com.ZJAVAKITABI;
|
|class Array3 {
| public static void main(String[] args) {
| String dəyəri_olmayan_array[] = new String[6];
| for (int i = 0; i <= 5; i++) {
| System.out.println(dəyəri_olmayan_array[i] + " "); }}
|-----
|run:
|null
|null
|-----

```

```
null
null
null
null
```

Dəyişən tipinə görə array boş qaldığı zaman aşağıdakı nəticələri verir.

**String** – null, **boolean** – false, **Char** – ‘\0’  
**integer, double, long, short**(bütün rəqəm dəyişənləri) – 0

### Çoxölçülü Array

Çoxölçülü array bir dəyərdən əlavə dəyərlərə sahib olması ilə fərqlənir. Dəyərləri verilmədikdə çoxölçülü array-də eynən tək ölçülü kimi default dəyərlərə sahib olur. Aşağıdakı şəkildə çoxölçülü array təsvir edilib.

	0	1	2	3	4	5	6
0	a	a	Z	j	y	z	l
1	z	d	g	a	h	f	c
2	x	c	d	d	b	j	f
3	f	j	l	j	c	i	v
4	f	z	k	d	f	g	l
5	v	s	z	y	x	c	f
6	l	o	i	u	y	t	r

Əsasən çoxölçülü array koordinat nöqtələrinin(x, y) göstərilməsini tələb edən məsələlərdə istifadə edilir.

```
package com.ZJAVAKITABI;
class Array4 {
    public static void main(String[] args) {
        char multidimension[][] = new char[7][7];
        multidimension[0][2] = 'Z';
        multidimension[1][3] = 'a';
    }
}
```

```

| multidimension[2][4] = 'b';
| multidimension[3][5] = 'i';
| multidimension[4][6] = 'l';
| System.out.println("X arrayinin sayı " + multidimension.length);
| System.out.println("Y arrayinin sayı " + multidimension.length);
| System.out.print("Müəyyən olunmuş kordinatlara görə şifrə : ");
|
| for (int i = 0; i <= 4; i++) { int j = i;
| System.out.print(multidimension[i][j + 2]); } } }
|-----|
|run:
|X arrayinin sayı 7
|Y arrayinin sayı 7
|Müəyyən olunmuş kordinatlara görə şifrə : Zabil
|-----|

```

Yuxarıdakı məsələdə iki koordinat nöqtəsinə uyğun olaraq Char elementlər daxil edilib. Daha sonra dövrədən istifadə edərək koordinatların avtomatik əldə olunmasını və elementlərin yerləşdiyi koordinatdan çağırılıb ekrana verilməsini təşkil edilib.

### Obyektin yaradılması

OOP baxımından hər bir adam obyektidir. Müəyyən atributlara sahibdir və həmin atributlarla bağlı funksiyaları vardır. Belə ki, İbayev ailəsində Zabil obyektini var. Həmin obyektin atributları İbayev class-ından götürülmüşdür. “İbayev Zabil = new İbayev();” əmri ilə biz İbayev class-nın tipinə uyğun olaraq “new”(yeni) Zabil obyektini yaradır və atribut dəyərlərinin İbayev class-ından götürülməsini qeyd edirik.

```

|-----|
|package com.ZJAVAKITABI;
|class Object {
|public static void main(String[] args) {
|İbayev Zabil = new İbayev();
|-----|

```

```

System.out.println("Zabilin saç ı " + Zabil.saç ı + ". Göz r ęngi " +
Zabil.göz);}}
class Ibayev {
String saç ı = "Qara";
String göz = "Q ęhv ęyi";}
run:
Zabilin saç ı Qara. Göz r ęngi Q ęhv ęyi

```

Zabil obyektini yazdıqdan sonra CTRL+Space düym ęsin ę basdıqda siz JVM(Java Virtual Machine)-d ęn art ıq m ¼ ęyy ęn olunmuş d ¼ y ęrl ęri aęıra bil ęrsiniz. Siz obyekt ı yaratdıęınız zaman ona İbayev class-ıdan m ¼ lumatları g ¼ t ¼ r ę bilm ęsini m ¼ m k ¼ n edirsiniz.

**Qeyd:** Obyektin yaradılması class-ın instans-laşdırılmasıdır.

## Built-in Java Packages

Java-da proqramlaşdırmaq ¼ ç ¼ n endirdiyimiz JAVA JDK(Java Development Kit) t ¼ r kibi nd ę Java Runtime Enviroment(JRE) - Java-n ı n iřl ę m ę si ¼ ç ¼ n m ¼ hiti daşıyır. JRE-y ę el ę JVM(Java Virtual Machine)-d ę dem ęk olar. B ¼ t ¼ n l ¼ k d ę Java-n ı n iřl ę m ę sin ę t ¼ řkil ed ę n paralardır. Oracle Java-ya bir ne ę ild ę n bir yeni ęlav ę l ę r edir. Bu ęlav ę l ę rd ę n olan JRE-nin t ¼ r kibi nd ę ki Built-in Java Packages (daxilind ę Java paketl ę ri quraşdırılmıř) hazır proqramlar t ¼ klif edir. Package el ę bizim yaratdıęımız “com.ZJAVAKITABI” paketi il ę eyni funksiyanı daşıyır. Oracle b ę zi ¼ m ¼ m il ę řmiř m ę s ę l ę l ę ri yenid ę n yazmamaq ¼ ç ¼ n istifad ę il ę r ę hazır yazılmıř class-lar y ę ni hazır yazılmıř proqramlar t ¼ klif edir. H ę r yenil ę nm ę d ę yaradılmıř paketl ę r Oracle-ın m ¼ ę yy ę n s ę hif ę l ę rind ę yer alır. Ařaęıdaki linkd ę bizim endirdiyimiz SE8(Standart Edition – Standard buraxılıř) yer alır.



<http://docs.oracle.com/javase/8/docs/api/>

Səhifədə solda birinci pəncərədə paketlərin siyahısını görürsünüz və onun altındakı pəncərədə paketlərin tərkibindəki class-lara baxa bilərsiniz. Sağ tərəfdə isə seçdiyiniz class-ın tərkibi izah edilir.

**Qeyd:** Belə sual yarana bilər, əgər built-in package-lər varsa niyə Netbeans-də “com.ZJAVAKITABI” paketi kimi solda görünmürlər. OOP-dən yadınızdadırsa Encapsulasiya-nı xatırladır. Proqramın funksiyası var amma özü görsənmir. Hazır class-ların sayının çox olduğunu nəzərə alsaq Oracle-ın bu üslubu niyə seçdiyini anlamaq çətin deyil.

## İmport

Built-in Package-lər İmport(idxal etmək, daxil etmək) əmri vasitəsilə istifadə olunur. Çağırılmış paketlərin tərkibində funksiya göstərməyə hazır proqramlar və tətbiq olunmağı gözləyən method-lar mövcuddur. Aşağıdakı misalda təqvim və tarixi əldə etmək üçün “java.util.java” paketindən istifadə olunmuşdur.

```

package com.ZJAVAKITABI;
import java.util.Date;
class ImportSaat{

```

```

| public static void main(String[] args) {
| Date tarixdeyiseni = new Date();
| System.out.println(tarixdeyiseni); }

```

```

- run:

```

```

- Tue Jan 31 00:27:25 AZT 2017

```

İmport –dan sonra çağırmaq istədiyiniz paketin adı yazılır. Paketin adı yazıldıqdan sonra onun daxilindəki method-lar çağırılmağı gözləyir. Belə ki, bizdə sadə şəkildə “tarixdeyiseni”-nə həmin dəyərləri vermişik. Hər bir paketin və ya method-un istifadə olunma qaydası var. Bunlar Oracle-ın səhifəsində hər paket üzrə ayrıca izah edilmişdir.

### Access control( girişin idarə olunması)

Access control yazdığınız class-ların, onların tərkibindəki method-ların və ümumilikdə yaratdığınız obyektlərin çağırılması və ya onlara müdaxilə olunması qaydalarını tənzim edir. Modifier təyin edici, modifikator deməkdir. Aşağıdakı cədvəldə public, private(xüsusi) və protected(qorunan) access control əməlləri təsvir edilib.

Modifier	Class	Package	Subclass	World
public	y	y	y	y
protected	y	y	y	n
no modifier	y	y	n	n
private	y	n	n	n

Y -yes N -no

Əsasən proqramlar private və ya public olaraq təyin edilir. Belə ki, public əmrini istifadə etsək proqram bütün siniflər üzrə açıq olur.

Private sadəcə class daxilində programı istifadə olunmasını təmin edir. Digər package-lər onu görmür. Protected bir o qədər də istifadə olunmayan əmrdir və demək olar ki, public onu hər yerdə əvəz edə bilər. No modifier(təyin edici olmadıqda) bəndində hər hansı bir əmr bildirilməzsə o zaman package-in ümumi access control dəyəri avtomatik mənimsədir.

## Method nədir?

Method bir növ dəyişənə bənzəyir. Hər hansı bir hesablama, ona verilmiş ada mənimsədir. Aşağıdakı misalda **method()**; -un yazılış şəkli göstərilmişdir. Dəyişəndən fərqli olaraq method-un daxilindəki dəyər ondan aşağıda **main**-dən yəni əsas programdan ayrı amma eyni class-ın daxilində yazılır. Method yaratmaq üçün bizə 4 parametrlə lazımdır.

- 1.Access control modifikatorlarından hər hansı biri – public, private, protected
- 2.Return type göstərilməlidir – void(method-dan nəticə qaytarmır), return(hesablayıb nəticə qaytarır)
- 3.Name – hər hansı bir ad verilir
- 4.Parameter – istifadə edəcəyi parametrlər(dəyərlər)

Aşağıdakı nümunədə olan “*public static void main*” bölümünü nəzərdən keçirək.

- İnterger dəyərləri elan edilib –( int a = 1, b = 2, e = 3;)
- Return type olan method1, method2 və method5 nəticələrinin ekrana verilməsi üçün “System.out.println”-dən istifadə edilir.

```
System.out.println(method1(a, b, e));
System.out.println(method2());
System.out.println(method5(a,b,e));
```

- Void type olan method3 və method4 main daxilində birbaşa yazılır.

```

method3();
method4();

```

- Method-da class kimi access control əməllərini tələb edir. Method-un **public** olması onun bütün proqramlar tərəfindən çağırıla bildiyini göstərir.
- **Static** yazılması instansının olmaması deməkdir.
- **Method**-a istənilən ad verilə bilər.
- Public static bildirildəndən sonra method-un tipi bildirilməlidir. Return type nədirsə o yazılır. Method1() tərkibində integer hesablama olacağı üçün “int method1” yazılır. Daha sonra main-də elan olunan hər hansı bir dəyişəndən istifadə olacaqsə həmin dəyişənlərin tipi yazılmaqla adları göstərilir.

```

public static int method1(int a, int b, int e) {
    int q = (a + b + e);
    System.out.print("3 parametrlı Method 1 işləyir və nəticə ");
    return q; }

```

- İstənilən hesablama və görüntü hazırlandıqdan sonra sonda **return** yazılmalı və dəyişən göstərilməlidir.
- Methodlarda “overloading” -həddindən artıq yüklənmə deyilən bir üsul var. Bu funksiya eyni ad daşıyan method-lara fərqli parametrlər verməklə yaradılır. Method1 birinci 3 parametrlı olaraq hesablanıb və daha sonra method5 tərkibində 2 parametrlə çağırılaraq hesablama aparılıb.
- Method2-də intergerlər maindən götürülməyib və fərqli adda dəyişəndən istifadə edilib.
- Method3 və method4 daxilində gedən proseslər **void** tipi daşıdığı üçün method-ların daxilindəki hesablamaların nəticələri geri qaytarılmır amma nəticənin method daxilində olduğu bilinir. Həmin nəticələrin ekrana verilməsi üçün “System.out.println()” istifadə etmişəm.



- Method5 daxilində method1-dən istifadə etməyin yolu göstərilib. Method1-in istifadə edəcəyi dəyişənlər return type-da elan edilib və bundan əlavə method5 daxilində başqa bir integer dəyişəni yaradılıb. Sonda hesablama aparılıb və nəticə ekrana verilib.

```

package com.ZJAVAKITABI;
class Method {
    public static void main(String[] args) {
        int a = 1, b = 2, e = 3;
        System.out.println(method1(a, b, e));
        System.out.println(method2());
        method3();
        method4();
        System.out.println(method5(a, b)); }

    public static int method1(int a, int b, int e) {
        int q = (a + b + e);
        System.out.print("3 parametrlı Method 1 işləyir və nəticə ");
        return q; }

    public static int method1(int a, int b) {
        int mf = (a + b);
        System.out.print("2 parametrlı Method1 işləyir. Nəticə: ");
        return mf; }

    public static int method2() {
        int a = 4, b = 6;
        int z = a + b;
        System.out.print("Method 2 işləyir və nəticə ");
        return z; }

    public static void method3() {
        System.out.println("Method 3 işləyir, Void nəticə vermir"); }

```

```

public static void method4() {
    int v = 5, m = 5;
    int n = v + m;
    System.out.println("Method 4 işləyir, hesablama nəticəsi " + n); }

public static int method5(int a, int b) { int u = 9;
    System.out.print("Method 5 və daxilindəki ");
    int s = method1(a, b) + u + 2;
    return s; }}

```

---

```

-run:
3 parametrlı Method 1 işləyir və nəticə 6
Method 2 işləyir və nəticə 10
Method 3 işləyir, Void nəticə vermir
Method 4 işləyir, hesablama nəticəsi 10
Method 5 və daxilindəki 2 parametrlı Method1 işləyir. Nəticə: 14

```

## Constructor

Yeni öyrənənlər üçün konstruktorun yaradılması prosesi bir qədər çətinlik yaradır. Bir az qarışıq prosesdir. Bu səbəbdən çalışacağam mümkün qədər prosesi xırdalayıb izah edim. Konstruktor yeni obyektı və ya obyektin instance-ını yaratmaq üçün təyin olunmuş xüsusi bir method-dur. Belə ki, konstruktor yaradan zaman iki class yaradılır. Rahat anlaşılməsi üçün onları 1-ci class-ı “konstruktorun çağırıldığı” və 2-ci class-ı “konstruktor” özü kimi nəzərimdə saxlayıram. Obyektlərin yaradılma bölməsində Zabil obyektini yaratmışdım. İndi bu misala bənzər adam sinfini yaradıb daha sonra atributlara əsaslanaraq onların instance-larını çağıracağam. Obyektin yaradılmasını bir daha nəzərdən keçirək ki, prosesi daha rahat qavramağımıza kömək olsun.

Hər bir adam müəyyən atributlara sahibdir və həmin atributlarla bağlı funksiyaları vardır. Belə ki, İbayev ailəsində Zabil obyektı var. Həmin obyektin atributları İbayev class-ından götürülmüşdür. “İbayev Zabil = new İbayev();” əmri ilə biz İbayev class-nın tipinə

uyğun olaraq “new”(yeni) Zabil obyektı yaradır və atribut dəyərlərinin İbayev class-ından götürməsinı qeyd edirik. Konstruktor yaradılmasında da oxşar misaldan istifadə edəcəyik.

```
:- - - - -
package com.ZJAVAKITABI;
public class Conscall{
    public static void main(String[] args) {
        Const İbayev = new Const("Qara", "şabalıdı", 181);
        Const Kerimov = new Const("Şabalıdı", "qara ", 175);
        System.out.println("Zabilin atributları aşağıdakılardır:");
        İbayev.Sac();
        İbayev.Goz();
        İbayev.Boy();
        System.out.println("Orxanın atributları aşağıdakılardır:");
        Kerimov.Sac();
        Kerimov.Goz();
        Kerimov.Boy();
    }
}:- - - - -
```

Cosnt(konstruktor) class-nın tipinə uyğun olaraq İbayev obyektı yaradılsın. İbayev obyektindəki 3 parametir (qara saçlı, şabalıdı gözlü və boyu 181) bir adamın yaradılması üçün “Conscall” class-ından Const class-na ötürülür. Const class-ı parametrləri uyğun şəkildə dizayn eləyib nəticəni hazırlayır. Obyektlərin Const class-nın instance-ı olması bizə həmin class daxilindən method çağırmanı mümkün edir. Aşağıda qeyd olunan İbayev və Kerimov obyektlərinin Const daxilində metod çağırıldığını görürük.

İndi keçək Conts class-ı daxilində yazılana ki, bütünlükdə prosesi anlaya bilək. Konstruktor adı class adı ilə eyni olmalıdır. Əgər konstruktoru class adından fərqli adlandırsanız o zaman konstruktor yox method yaratmış olarsınız. Konstruktoru yaradarkən ona işlədə biləcəyi parametrləri təyin etməsəniz java bir qayda olaraq həmin dəyərləri konstruktor üçün özü təyin edəcək. Bundan başqa konstruktorun method-lardan seçilən əsas fərqi “return type”-ının

olmamasıdır. Yəni daxilində hesablamaları ekrana vermək üçün konstruktorun void type olmasını nəzərə almalısınız.

```
package com.ZJAVAKITABI;
public class Const {
    String sac_rengi; String goz_rengi; int boy;
    Const(String s, String g, int b) {
        sac_rengi = s; goz_rengi = g; boy = b; }
    void Sac() {System.out.println("Saç rəngi:" + sac_rengi); }
    void Goz() {System.out.println("Göz rəngi:" + goz_rengi); }
    void Boy() {System.out.println("Boyu:" + boy); }}
```

“Const” “class”-ı daxilində iki “string” tipi və bir “int” tipli dəyişən yaratmışam. Bu dəyişənlər (sac\_rengi, goz\_rengi, boy) “Const” mötərizəsi daxilindəki dəyişənlərdən(String s, String g, int b) yəni “Constcall” “class”-ından gələn məlumatların götürülməsi prosesi üçündür. “Constcall” “class”-ında verdiyimiz parametrlərə uyğun (Const class-ında) tiplər və dəyişənlər yaratmalıyıq. “Constcall”-dan gələn dəyişənlərin (String s, String g, int b) birbaşa konstruktor(Const) daxilində işlətmək mümkün deyil. Bu səbəbdən onları daxilindəki dəyişənlərə mənimsədirik, yəni ötürürük. Daha sonra return type void olmasını yazıb metodumuzu yaradıırıq. “Const” daxilində daha mürəkkəb hesablamalar aparmaq mümkündür. Yuxarıda göstərilən misal prosesin ümumi təsvirini yaradır.

**Qeyd:** Proqram daxilində konstruktorlar method-lardan əvvəl çağırılır.

### “Inheritance”-in yaradılması

Bildiyimiz kimi “inheritance”, bir proqrama başqa proqramın xüsusiyyətlərini istifadə etməyə imkan yaradır. “Inheritance” aşağıdakı misalda “extend”(uzatmaq, böyütmək, artırmaq) əmri ilə irsiliyi ötürür. Aşağıdakı misalda A, B, C “class”-ları hazırlanmış,

daha sonra C-yə uyğun olaraq obyekt yaradılmışdır. Obyekt C-yə aid olmasına baxmayaraq B-nin dəyərlərini çağırır.

```
package com.ZJAVAKITABI;
public class Inher { public static void main(String[] args) {
    programC obyekt = new programC();
    obyekt.Bnigoster(); } }
class programA {int a=5;
    public void Anigoster(){
        System.out.println("A-nın nəticəsiyəm");}}
class programB extends programA {int b=10;
    public void Bnigoster(){
        System.out.println("B-nin nəticəsiyəm");
        System.out.println("B-dən A rəqəmini görürəm = " + a);
        Anigoster();}}
class programC extends programB {int c=15;
    public void Cnigoster(){
        System.out.println("C-nin nəticəsiyəm" + c );
        Anigoster();
        Bnigoster();}}

run:
B-nin nəticəsiyəm
B-dən A rəqəmini görürəm = 5
A-nın nəticəsiyəm
```

Ardıcılığa görə programC programB-nin və programB programA-nın əlavəsidir. Başqasının əlavəsi və ya uzantısı olmayan A **superclass**-dır. B və C **subclass**-dır.

### “Super” və “This” açar sözləri

“Inheritance”-dən bizə müəyyən olduğu kimi, “class”-lar tabeliyində olduğu “class”-ın istənilən dəyərini çağırır. Bəzən alt sinif(subclass) və üst sinif(superclass) eyni adda dəyişənə sahib

olur. Bu zaman “super” əmrindən istifadə edərək üst sinfə aid olan dəyişənin proqramdakı yerini təyin edə bilirik. Super sözünü qeyd etdikdən sonra “CTRL+SPACE” düymələrini sıxaraq “JVM”-də “class”-la bağlı olan bütün mümkün dəyişənləri və funksiyaları çağırma bilərsiniz. “Super” üstün “class”-ı göstərir. This(bu) açar sözü olduğu yeri, hazırkı sinfi müəyyən edir. Aşağıdakı misalda hər iki açar söz göstərilmişdir.

```
- - - - -  
package com.ZJAVAKITABI;  
public class SuperThis {  
public static void main(String[] args) {  
    progC obyekt = new progC();  
    obyekt.Cavab();}  
class progA {int a = 5;}  
class progB extends progA {int b = 10;}  
class progC extends progB {int a = 15;  
public void Cavab() {System.out.println(super.a + this.a + super.b +  
    b + a);}}  
- - - - -  
run:  
55  
- - - - -
```

Yuxarıdakı misalda “proA” və “proB” super sinifdir və bu səbəbdən onların dəyişənləri (“Super.a” və “super.b”) “subclass”-da “super” açar sözü ilə çağırılır. ProC isə “subclass”-dır. “This.a”-nın istifadə olunduğu “Cavab()” metodu “progC” class-ına aiddir. ProA və ProC dəyişəni eyni adda olduğu üçün onları ayırmaqda super.a və this.a açar sözlərindən istifadə olunub. “Super.b” dəyişəni və “b” dəyişəni eynidir. Çünki “b” adında digər dəyişən yoxdur. Burada sonda istifadə olunan “a” dəyişəni “this” əmrinə bərabərdir çünki ProC daxilində yəni öz class-ında istifadə olunur.

## Static Variable və ya Class dəyişənlər

Statik dəyişənlərin birbaşa mənası yanlış anlaşıla bilər. Mənim sədiyi dəyərlərin həmişə sabit qalması kimi başa düşülür amma bu belə deyil. Statik dəyişənlər proses içində müxtəlif dəyərlərə sahib ola bilər, amma çağırıldığı obyektlərə nisbətə dəyərləri sabitdir. Statik dəyişənlərin digər adı “class” dəyişənləridir. Bildiyimiz kimi program işə başladığında konstruktorlar method-lardan əvvəl çağırılır. Statik dəyişənlər programda compile olunanda JVM-dəki *class loader* tərəfindən müəyyən edilir və yaddaşda saxlanılır. Bundan əlavə static dəyişənlər əmlər içində konstruktordan da əvvəl çağırılır. Aşağıdakı misalda sadə static dəyişən göstərilmişdir. Əgər static dəyişənlərə dəyər mənimləməsək rəqəmlər üçün “0”, boolean üçün “false”, char üçün “0”, obyekt üçün “null” göstəriləcəkdir.

```
-----  
package com.ZJAVAKITABI;  
class Staticvar {  
public static void main(String[] args) {  
System.out.println(sabit.id); }  
class sabit {static int id;}  
-----  
run:  
0  
-----
```

Burada bir java faylın içində 2 class yaratmışam. Birinci class main-dir. Son nəticəmizdə oradan çapa verilir. İkinci class-ımızın vəzifəsi sadəcə static dəyəri təqdim etməkdir. Static dəyişənlərə mürəkkəb funksiyalardan alınan dəyərləri mənimləmək üçün statik bloklardan istifadə edilir.

```
-----  
package com.ZJAVAKITABI;  
class Staticvar2 {  
public static void main(String[] args) {  
vahid konstruktor = new vahid(); }  
-----
```

```

class vahid {
static {System.out.println("Bu hesablama statik bloka aiddir"); }
vahid(){System.out.println("Bu yazı konstruktora aiddir"); }
static {int a = 5; int b = 5;
while (a == b) {System.out.println("A B-yə bərabərdir"); break; }}}
run:
Bu hesablama statik bloka aiddir
A B-yə bərabərdir
Bu yazı konstruktora aiddir

```

Əgər class daxilində iki statik dəyişən varsa JVM-ə yazıldıqları ardıcılığa görə yüklənilirlər. Statik bloklar aşağıdakı kimi tərtib edilir. Bu misalda konstruktordan istifadə etmişəm. Konstrukturun funksiyası class-da olan dəyərləri bir class-dan digərinə instance yaradaraq ötürməkdir. Heç bir dəyişəndən istifadə etmədən class-ın tərkibindəki iki static və bir konstruktor dəyərini main class-da yeni obyektə mənimsədərək nəticəni əldə etmişəm. Nəticədən də aydın olduğu kimi konstruktor sıralamada ikinci olsa da onun üçüncü yazılan static dəyişən ardıcılıqda qabaqlayır. Bundan əlavə konstruktor obyektı sadəcə main class-da yaradılmaqla nəticəni ekrana verir. Buna səbəb sonda yazılan “vahid()”-dir. Əgər mötərizə daxilində parametr verilsə həmin parametərə uyğun nəticə ekrana veriləcəkdir.

## Casting

Hər hansı məlumatın tipinin dəyişdirilməsinə casting deyilir. Casting dəyişənin və ya obyektin dəyərini deyil, onların int, float, char, long, short, double, byte kimi tiplərini dəyişir.

**float** mənbə = 7.06F;

**int** nəticə = (int) mənbə;

Yuxarıda göstərilən misalda float tipində olan dəyişən ondan aşağıda yazılan üsulla integer tipinə çevrilmişdir.



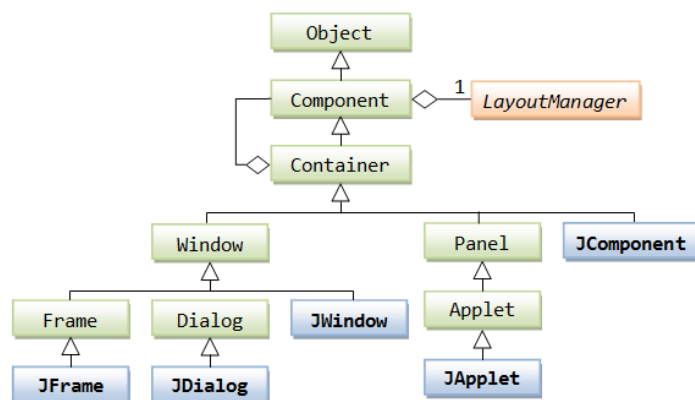
**Qeyd:** Inheritance əlaqəsi olan obyektlər arasında casting etmək mümkündür.

## Swing GUI- Qrafiki İstifadəçi İnterfeysi

Swing Graphical User Interface(GUI) – Swing Qrafiki İstifadəçi İnterfeysi Java-da arxa planda(backend) yazılan proqramları, istifadəçilərə qrafiki təsvir şəklində(frontend) təqdim etmək üçün nəzərdə tutulub. Swing arxa planda yazılan kodlamaya qrafiki görünüş verir.

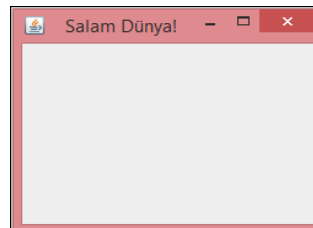
## Swing-in strukturu

İnterfeysin görünməsi üçün obyekt yaradılmalıdır. Aşağıdakı şəkildə Swing-in strukturu iyerarxik düzülüşdə təsvir olunmuşdur. Strukturun hər bir parçası bir araya gələrək obyektə formalaşdırır və onun üzərində toplaşır. Component və Container GUI-nin yaradılması üçün əsas obyektlərdir. Komponent individual elementlərdən(düymələr, yazı qutuları, və s.) ibarətdir. Konteynerlər də komponentdirlər amma onlar bir neçə komponenti bir arada tutmaq üçün istifadə olunurlar.



Yuxarıdakı təsvirdən də müəyyən olunduğu kimi, konteynerlərin tərkibində Window(pəncərə), Panel(lövhə), “JComponent” kimi komponentlər vardır. Swing-in tərkibindəki bütün elementləri və onunla bağlı əməllərin hamısını sadalamaq mümkün deyil. Hər yenilənmədə yeni elementlər və funksiyalar əlavə oluna bilər. Bu səbəbdən işləmə məntiqini anlamaq daha vacibdir. Praktiki məsələlər üzərində məşq etmək “GUI”-ləri mənimsəməkdə daha böyük rol oynayır. Swing standart olaraq “Netbeans”-ə yüklənmişdir. Bu baxımdan ilk olaraq Swing-lə təcrübə keçmək ən doğrusudur.

### Qrafiki təsvirdə ilk proqram “Salam Dünya”



Yuxarıdakı “Salam Dünya!” proqramının qrafiki görünüşü “swing”-lə tərtib edilmişdir. “Backend”-də, “Swing” paketini çağıraraq onun tərkibindəki JFrame(çərçivə, korpus, gövdə, pəncərə) proqramını istifadə etmişik. Proqramdan istifadə etmək üçün obyekt yaratmışıq və onun tərkibindəki metodları, proqramları və sair dəyərləri obyekt vasitəsi ilə çağırır, istədiyimiz yeni dəyərlərlə yeni forma və funksiyalar yaradıırıq.

```
package com.ZJAVAKITABI;  
import javax.swing.JFrame;  
class Swingtest{  
    public static void main(String[] args) {  
        JFrame pəncərə = new JFrame("SALAM DÜNYA");  
    }  
}
```

```

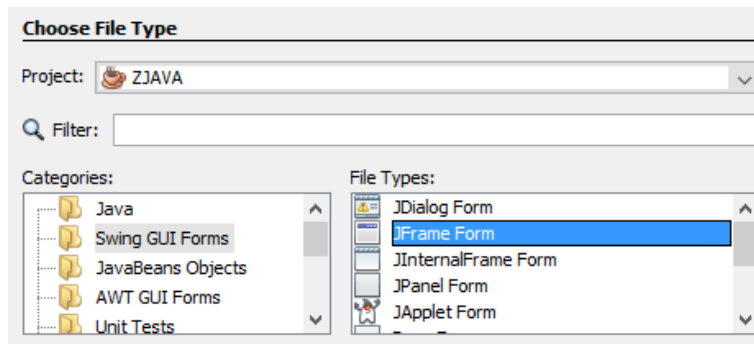
| pencərə.setVisible(true);
| pencərə.setSize(500,400);
| pencərə.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);}}
|

```

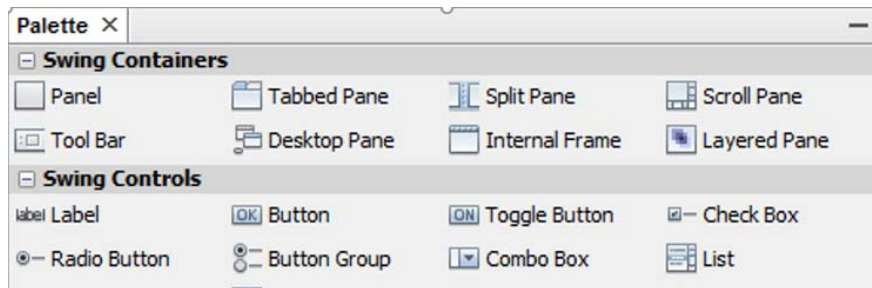
Obyekt “pəncərə” yaradıldıqda proqram daxilində pəncərədən sonra “.” nöqtə qoysaq avtomatik olaraq “JVM”-dən “JFrame”-ə aid olan funksiyaları “pəncərə” obyektinə üçün çağıracaqdır. Obyektin ekranda görünməsi üçün ilk əmrimizi “pəncərə.setVisible” (set-təyin etmək)(visible-aydın, görünən) yazırıq. Bu funksiya boolean dəyər(true və ya false) qəbul edir. “True”- doğru, təsdiqləmək mənasını verir və burada pəncərənin görünməsinə razılıq veririk. Daha sonra “pəncərə.setSize(500, 400)” əmrləri ilə pəncərənin hansı ölçüdə olması müəyyən edilir. “pəncərə.setDefaultCloseOperation” əmri pəncərənin standart olaraq qapanmasını və bu hərəkətin “JFrame”-in tərkibindəki “Exit\_ON\_CLOSE” əmri ilə yaradılacaq pəncərədə sağ üst tərəfdəki “x” düyməsinə basarkən pəncərəni bağlamasını əmr edir. Swing barədə ümumi təsəvvür yarandığını güman edirəm. Swing-in tərkibindəki bütün əmrləri əzbərləmək mümkün deyildir. Ümumilikdə proses swing paketini daxil etməklə tərkibindəki proqramları çağırır və istədiyimiz interfeys dizaynını tərtib edirik. Swing kimi müxtəlif qrafiki interfeyslər(GUI) var. Swing əmrləri ilə bağlı müxtəlif səhifələr mövcuddur. [http://www.tutorialspoint.com/swing/swing\\_controls.htm](http://www.tutorialspoint.com/swing/swing_controls.htm) - bu səhifədə hər əmrə uyğun olaraq onun alt əmrləri sadə formada göstərilmişdir.

## SWING GUI Forms

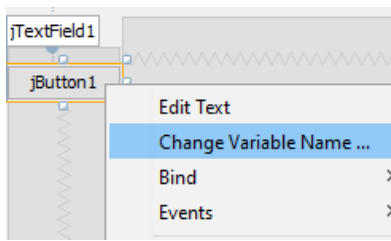
Bildiyimiz kimi Swing yazdığımız proqramlara görünüş və funksionallıq vermək üçün istifadə olunur. İlk olaraq Netbeans-də yeni fayl yaratmaq düyməsini sıxaraq və “ZJAVAKİTABİ” paketimiz daxilində “Swing GUI Forms” kateqoriyasından “JFrameForm” fayl tipini seçirik.



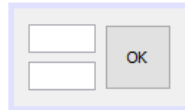
Növbəti səhifədə JFrameForm -a istədiyimiz Class adını veririk. Ortada açılan Design bölməsindəki formanın üzərinə “Palette” bölməsinə iki “Text Field”(yazı sahəsi) və bir “Button”(düymə) atırıq.



Yaratdığınız “JButton1” düyməsinin üzərinə sağ düymə ilə sıxaraq açılan menyudan “Edit text” variantını seçin və düymə üzərində görmək istədiyiniz yazını qeyd edin. Dəyişdiyimiz yazı komponentin görünən tərəfidir. Proqram tərəfi üçün komponentlərin adını dəyişmək üçün sağ düymə ilə açılan menyudan “Change Variable Name”(dəyişənin adını dəyişmək) variantını seçin. Proqram daxilində yazılan dəyişən adlarının yazılış üslubu qısa və müəyyən məntiqi dizayna sahib olmalıdır. Məsələn: “button” üçün “btn”, “Text Field” üçün “txt” kimi qısaltmalar istifadə edə bilərsiniz.



Dəyişən adları “txt1”, “txt2” və “btn” olan dörd swing komponenti yaratdım. Məqsədim “txt1” -də veriləcək yazını “txt2”-də göstərməkdir. Beləliklə proqramı aşağıdakı kimi dizayn edirəm.



Daha sonra OK düyməsinin üzərinə sıxaraq açılan “Source” (mənbə) bölməsində proqramımızı yazmağa hazırlaşırıq. Açılan pəncərədə əvvəllər gördüyümüz kimi təmiz səhifə bizi gözləmir. Aşağıda göstərilən “private void btnActionPerformed” hissəsinə gələrək düymənin basılacağı anda edəcəyi əməlləri yazırıq. Yəni proqramımızı yazmağa başlayırıq.

```
private void btnActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
}
```

Kodlama hissəsində olan “// TODO add your handling code here:” bu şərhə silə bilərsiniz. Hər bir Swing komponentinin özünə uyğun JVM-dən çağırıldığı əməllər toplusu var. TextField- bütün ona verilən dəyərləri String olaraq qəbul edir. Proqramın frontend tərəfində daxil ediləcək dəyərləri backend tərəfdə əldə etmək üçün TextField “getText()” metodundan istifadə edir.

```
private void btnActionPerformed(java.awt.event.ActionEvent evt){  
    String s=txt1.getText();  
    txt2.setText(s); }
```

Bu üsulla txt1-də verilən yazını dəyişənə mənimsədiyib daha sonra həmin dəyişəni txt2-yə ötürmüşəm. Bu misalda bir dəyişəni frontend-dən necə almağı və necə Backend-dən frontend-ə məlumat ötürməyi göstərmişəm. Yuxarıdakı proqramın daha düzgün yazılışı aşağıdakı kimidir:

```
private void btnActionPerformed(java.awt.event.ActionEvent evt){
txt2.setText(txt1.getText()); }
```

Proqramı yazdıqdan sonra yadda saxlayın və Source hissəsindən Run edin.

### ParseInt və ValueOf

Bildiyimiz kimi TextField daxil olan dəyərləri String olaraq qəbul edir. Belə olduğu halda rəqəmləri toplamaq mümkün olmur. Deməli String dəyərini Integer-ə çevirməyi öyrənməliyik. Üç textfield və bir button yaradaraq onlara daxil olan rəqəmləri toplamağa çalışaq. Button düyməsinin action performed hissəsinə daxili olaraq orada aşağıdakı kodu yazırıq.

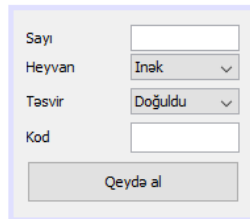
```
private void btnActionPerformed(java.awt.event.ActionEvent evt){
int nmbr1 = Integer.parseInt(txt1.getText());
int nmbr2 = Integer.parseInt(txt2.getText());
txt3.setText(String.valueOf(nmbr1 + nmbr2));}
```

Birinci Integer class-ından “parseInt()” metodunu çağırır və txt daxilindəki string-i integer-ə çeviririk. Hesabalama aparıldıqdan sonra TextField-ə string olaraq nəticəni qaytarmaq üçün String class-ından “valueOf()” metodunu çağırırıq.

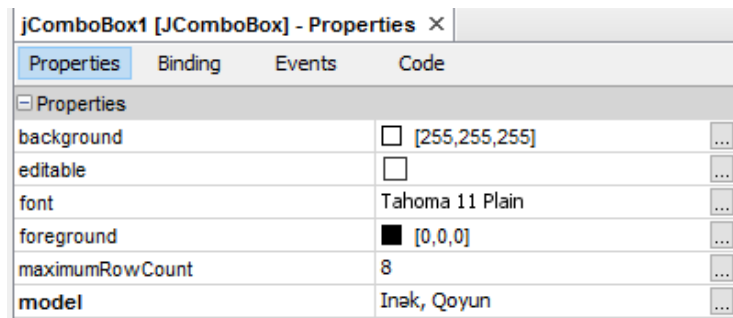
**QEYD:** SetText məlumatın TextField-ə mənimsədir. GetText məlumatı TextField-dən götürür. Demək olar ki bütün Swing komponentləri bu məntiqə işləyir.

### ComboBox və JOptionPane praktiki məsələ.

Ferma daxilindəki heyvanların artımı və ya azalmasını nəzarətdə saxlamaq üçün bir proqram hazırlayaq. Bizə iki ədəd TextField, iki ədəd ComboBox, dörd ədəd Label və bir ədəd Button lazımdır. Label- etiket deməkdir və bu proqramda “Sayı, Heyvan, Təsvir və Kod” yazıları ilə komponentlərin adlandırılmasında istifadə olunublar. Bizə iki ComboBox lazım olduğunu bildirmişdim. Biri heyvan növünün siyahısını, digəri onların hazırkı vəziyyətinin təsviri üçün istifadə olunacaqdır. TextField-lər isə istifadəçi tərəfindən daxil ediləcək məlumatları qəbul edəcəkdir.



ComboBox-u seçin və daxilindəki items(bəndləri) əlavə etmək üçün sağ alt küncdə yerləşən JComboBox1-Properties-dən model sözü yazılan hissəni tapın. Qarşısındakı üç nöqtəli düyməni sıxın. Variantları(İnək, Qoyun, Keçi, Ördək) alt-alta yazın və OK düyməsini sıxın.



Daha sonra ikinci ComboBox üçün eyni addımları atın. Doğuldu və ya Kəsildi/Öldü variantlarını əlavə edin. Eynən əvvəlki proqramda olduğu kimi Button üzərinə iki dəfə sıxaraq action performed hissəsində göstərilən qaydada proqramı yazın.

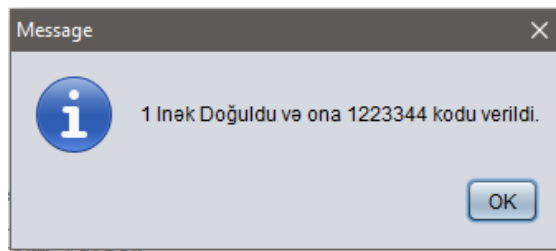
```
private void jButton1ActionPerformed(java.awt.event.ActionEvent  
    evt) {  
    String x = jTextField2.getText();  
    String code = jTextField1.getText();  
    String animal = String.valueOf(jComboBox1.getSelectedItem());  
    String expl = String.valueOf(jComboBox2.getSelectedItem());  
    JOptionPane.showMessageDialog(null,x+" "+ animal +" "+ expl +"  
    və ona "+ code+" kodu "+ "verildi."); }  
-----
```

Frontend-dən TextField-lərə daxil edilən məlumatları “getText()” methodu vasitəsi ilə əldə edib daxili dəyişənə mənimsətmişəm. “GetSelectedItem()” methodu ilə ComboBox-larda olan məlumatları əldə edib, String-ə çevirib, dəyişənə mənimsətmişəm.

```
package com.ZJAVAKITABI;  
import javax.swing.JOptionPane;  
-----
```

JOptionPane dialoq paketini import edirik. JVM-dən ona aid olan methodlardan “showMessageDialog()” çağırır, məlumatları ekrana dialoq pəncərəsi vasitəsi ilə ötürürük. Proqramı Run etdikdən sonra açılan proqramda lazım olan məlumatları daxil edin. “Qeydə al” düyməsini sıxın. Açılan “Message” informasiya pəncərəsi JOptionPane dialoqudur.





## Stream

Stream axın deməkdir. Bu termin Java-da məlumat axını ilə bağlı istifadə olunur və iki yerə ayrılır:

1. “InputStream” məlumatı hansısa mənbədən oxumaq
2. “OutputStream” məlumatı təyin olunan yerə yazmaq

JVM işə salındığı andan məlumatın təyinatı ilə bağlı iş görən üç axın komponenti “System.in”, “System.out” və “System.err” əməlləri icra etməyə hazır olur. Bunlardan ən çox “System.out.print” əmrini işlətməmişik.

## Scanner

“Scanner” daxil edilmiş və ya daxil ediləcək mətinlərlə işləmək üçün ən populyar class-dır. “Java.util.Scanner;” paketini import etməklə class-ı aktivləşdirə bilərsiniz. “Scanner” class-ının method-ları aşağıdakılardır:

String next() ; String nextLine(); byte nextByte(); short nextShort();  
int nextInt(); long nextLong(); float nextFloat(); double nextDouble();

```
-----  
package com.ZJAVAKITABI;  
import java.util.Scanner;  
public class Scann {  
public static void main(String args[]) {  
Scanner sc = new Scanner(System.in);  
System.out.println("Qeydiyyat nömrənizi yazın");  
}
```

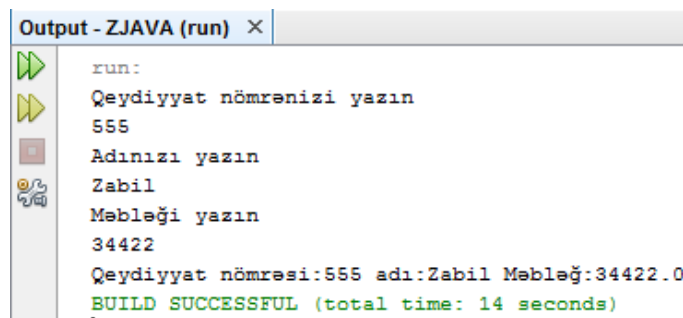
```

int reg = sc.nextInt();
System.out.println("Adınızı yazın");
String name = sc.next();
System.out.println("Məbləği yazın");
double fee = sc.nextDouble();
System.out.println("Qeydiyyat nömrəsi:" + reg + " adı:" + name + "
Məbləğ:" + fee);
sc.close(); } }

```

Yuxarıdakı misalda Scanner class-nın instance-ı olan “sc” obyektini yaradırıq. “System.in”-dən daxil edilən məlumatları necə qəbul etməsini method-lar vasitəsi ilə göstəririk. Ən sonda obyekt ilə işimizin bitdiyini “sc.close();” əmri ilə bildiririk.

**QEYD:** “System.in” yəni sistemə məlumatın daxil edilmə bölməsi NetBeans-də aşağıda Output hissəsindədir. Nəticə göstərilən hissədə məlumatda daxil edilə bilər.



```

run:
Qeydiyyat nömrənizi yazın
555
Adınızı yazın
Zabil
Məbləği yazın
34422
Qeydiyyat nömrəsi:555 adı:Zabil Məbləğ:34422.0
BUILD SUCCESSFUL (total time: 14 seconds)

```

## Delimiter

Scanner class-ı hər bir məlumatı simvol kimi qəbul edir. Hər hansı bir mətn daxilində müəyyən bir hissəni qeyd edərək, ona oxşar digər variantları işarələmə kimi tapşırıqları “Scanner” class-nın “delimiter” method-u ilə yerinə yetirə bilərsiniz.

```
String input = "1 balıq 2 balıq qırmızı balıq göy balıq";
Scanner s = new Scanner(input).useDelimiter("\\s*balıq\\s*");
System.out.print(s.nextInt()+" ");
System.out.print(s.nextInt()+" ");
System.out.print(s.next()+" ");
System.out.println(s.next());
s.close(); }
```

run:

```
1 2 qırmızı göy
```

Gördüyümüz kimi “balıq” sözünü “useDelimiter()” method daxilində istifadə edərək bütün mətdən çıxarılmasına nail olduq.

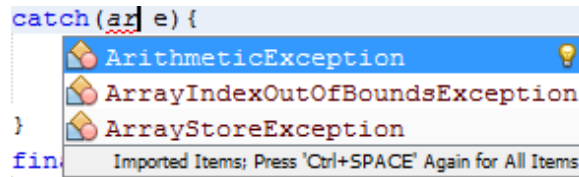
### Exception , Try , Nested Try, Catch, Finally

Program kodunu yazdığımız zaman onun iş prinsipində qarşısına çıxma biləcək xətlərin hamısını görmək mümkün olmur. Xətlər müxtəlif səbəblərdən olur. Hər hansı səbəb və haldan asılı olmayaraq xəta hadisəsinin baş vermə ehtimalına qarışı “Exception” (istisna hal) program daxilində nəzərdə tutulmalıdır. “Exception” halları “Try”(cəhd) bloku vasitəsilə hazırlanır. Try blokunda deyilir ki, bu kodu icra elə. Əgər nəzərdə tutulmayan hal baş verərsə(exception), o zaman səhvi(error) tut (catch) və verilən parametrlər üzrə həll et(handle).

```
try { kod hissə } catch (Exception e){ }
```

**Catch** blokunda səhvlər tutulub “e” dəyişəninə saxlanılma üçün mənisədir. Bu bir növ səhvin düzəldilməsi üçün atılmış bir addımdır və “error handling”(səhvlərin ələ alınması, idarə olunması) kimi qəbul edilir. Try blokunda istifadə olunan obyektədən asılı olaraq catch-in exception parametri dəyişir. Hər obyekt import olunduğu package-ə uyğun olaraq exception parametri tələb edir. Neatbeans-də catch blokunun exception hissəsində

düzgün parametr yazılmadığı halda xəbərdarlıq göstərilir. JVM (CTRL+Space) vasitəsi ilə paket adına uyğun variantı seçib səhvi düzəldə bilərsiniz.



**Finally**(son, yekun) bloku bir növ try blokundan asılıdır. Catch blokunun exception tutmasından asılı olmayaraq finally bloku demək olar ki həmişə icra olunur. Try blokunun içində “System.exit” əmri verilərsə bu zaman try bloku bağlanır. Sadəcə bu halda finally bloku icra oluna bilmir. Finally bloku method daxilində istifadə olunduğunda return dəyərini sonda dəyişdirə bilmə kimi üstünlüyü var. Catch istisna hal olduğunda aktivləşir. Finally bəzi proseduraları avtomatik yerinə yetirməyə şərait yaradır. Catch-dən asılı olmadan, həmişə try-la birgə icra olunur.

```
public class TryCatchFinally {  
    public static void main(String[] args) {  
        try { // int a=100/0;    int a=100/1; }  
        catch(ArithmeticException e) {  
            System.out.println("Catch bloku aşağıdakı səhvi tutub");  
            System.out.println(e); }  
        finally {System.out.println("Finally bloku çağırılıb"); } } }
```

Yuxarıdakı misalda “int a=100/0” variantını yoxlasanız, sıfıra bölünmə mümkün olmadığı üçün catch və finally-nin hər ikisi nəticə verəcək. “int a=100/1” variantında səhv olmadığı üçün sadəcə finally bloku nəticə verəcək.

**Nested Try**(yuvalanmış, iç-içə keçmiş) - Try bloku içində Try bloklarının olması-na deyilir.

```

public class NestedTry {public static void main(String args[]){
try{ try{
System.out.println("29 sıfıra bölünə bilməz");
int z =29/0;
} catch(ArithmeticException e){System.out.println(e);}
try{int i =30/0;
} catch(ArithmeticException e){System.out.println(e);}
System.out.println("30-da sıfıra bölünə bilməz");
} catch(Exception e){ System.out.println
("Əsas Try-da səhv olsa bu mesaj ekrana verilər");}
System.out.println("Səhvlər çıxdı amma həll edildi"); } }

```

```

run:
29 sıfıra bölünə bilməz
java.lang.ArithmeticException: / by zero
java.lang.ArithmeticException: / by zero
30-da sıfıra bölünə bilməz
Səhvlər çıxdı amma həll edildi

```

**Try-with-resources(mənbə)-** bloku bir neçə mənbədən(obyektdən) istifadə etməyə imkan yaradır. Sonuncu bloku yaxşı anlamaq üçün try bloku ilə bağlı alt blokları ümumilikdə incəliyək. Yazdığımız proqramlarda aldığımız nəticələri fayla yazmaq istəsək hansı addımları atmalıyıq? Məlumatları yazmaq üçün faylı tapmalı, yoxdursa yenisini yaratmalı, öncə faylı açmalı, yazdığımız məlumatı yaddaşda saxlamalı və faylı bağlamalıyıq. Bizdə yuxarıda sadalanan addımların tam yerinə yetirilməsi üçün proqram kodlarını, try blokları vasitəsilə nəzarətdə saxlamalıyıq. Əgər fayla məlumat yazıla bilmirsə və ya hər hansı bir xəta baş verirsə catch bloku səhvi tutur və vəziyyəti ələ alır. Catch və finally bloklarında açılan faylın bağlanması nəzərdə tutmaq şərtidir. İş prosesində istisna hal olmasına baxmayaraq, finally blokunda yazılan əmr icra edilir. Bu qəbildən, exception olsa belə fayl finally blokunda bağlana bilər. Try blokları içində istifadə olunan fayllar açıldıqda

onların bağlanması üçün finally bloku hər obyektə uyğun ayrıca bağlama əmri yazmağı tələb edir. Try with resources blokunun üstünlüyü catch və finally bloklarından azad şəkildə işləyə bilməsidir. Faylların bağlanma prosesi bu bloka aid “AutoCloseable” obyektə vasitəsi ilə avtomatik yerinə yetirilir. Obyektlərin bağlanması üçün finally blokundan istifadə etmək lazım gəlmir.

### Create, Read və Write File

Yaratdığımız hər hansı proqramın nəticəsini fayla yazmaq istəsək bu zaman “java.io” paketlərindən istifadə etməliyik.

```
-----  
package com.ZJAVAKITABI;  
import java.io.BufferedWriter;  
import java.io.File;  
import java.io.FileWriter;  
import java.io.IOException;  
public class CreateReadWrite {  
    public static void main(String[] args) {  
        File newfile = new File("C:/Users/Z13/Desktop/ZJAVA.txt");  
        if (newfile.exists()) {  
            System.out.println("Fayl daha əvvəl yaradılmışdı"); } else {  
            try {newfile.createNewFile();} catch (IOException e) { }  
            try {FileWriter fileW = new FileWriter(newfile);  
                try (BufferedWriter buffW = new BufferedWriter(fileW)) {  
                    buffW.write("Bu yazı Java proqramı tərəfindən yazılıb.");  
                    buffW.write(" Buffered writer fərqi."); }  
                System.out.println("File uğurla yaradıldı"); }  
            catch (IOException e) { } } } }  
-----
```

İlk olaraq “newfile” obyektini yaradıırıq. Sonra şərt qoyub “java.io.File;” -in method-u ilə yoxlayırıq əgər “newfile” obyektə mövcuddursa(exists()); “Fayl daha əvvəl yaradılmışdı ” yazısını ekrana verir. Digər halda “newfile” obyektinin parametrinə uyğun

olaraq “createNewFile” yeni fayl yarat. Səhv baş verərsə “e”-yə mənimsəd vəziyyəti ələ al. “java.io.FileWriter;” paketindən istifadə edərək fayla mətin yazan “fileW” obyektini yarat və parametrlərdə göstərilən “newfile”-a yaz. “java.io.BufferedWriter;” paketi “buffW” obyektini yarat və onda olan məlumatı “fileW” obyektinə mənimsəd. “BuffW” obyektini “write()” method-u ilə göstərilən mətni yaz. Faylın uğurlu yazılması ilə bağlı məlumat ekrana verilir. Səhv yoxdursa proqram başa çatır. Yuxarıda yazılan bir məqama diqqətinizi çəkmək istəyirəm. BufferedWriter məlumatı FileWriter ötürür. FileWriter-in öz obyektinə də məlumat mənimsədilib sonra fayla yazıla bilər.

```
-----  
FileWriter fileW = new FileWriter(newfile);  
! fileW.write("FileWriter mətni");  
! fileW.close();  
-----
```

FileWriter-in çatışmayan cəhəti hər string-i ayrı-ayrılıda çağırır fayla yazmasıdır. Burada istifadə olunan BufferedWriter-in funksiyası bir neçə mətni buffer yaddaşına yükləyir. Buffer yaddaşının həcmi 8192 simvoldan ibarətdir və mənimsədilən bütün mətnləri eyni anda FileWriter-in yazması üçün göndərir.

### Throwable, Throws və Throw

Java-da baş verə biləcək bütün proqram xətalalarının həll edilməsi üçün “**Throwable**” (java.lang.Throwable) class-ı mövcuddur. Throwable class-ı iki hissəyə bölünür. Exception və Error. Exception-lar həll oluna biləcək səhvlərdir və istisna hal kimi qəbul olunurlar. Error-lar isə həll olunmayan səhvlərdir. Həll olunmayan səhvlər bir növ fiziki vəziyyətlə bağlı olurlar. Fərz edək ki, yaddaş kartına musiqi yazmaq istəyirsiniz. Kartdakı həcm 2 gigabaytdır və siz 5 gigabaytlıq musiqi yazmaq istəyirsiniz. Proqram fiziki yaddaşı artırma bilmir. Bu səbəbdən səhv ortaya çıxır və error həll oluna bilmir. Exception-lar özündə iki hissəyə ayrılır. Checked və

Unchecked. Checked exception-lar java tərəfindən anlaşılan istisnalardır. Bu istisna hallar Throwable class-ının vəziyyətə uyğun həll təklifləri ilə aradan qaldırıla bilər. Unchecked exception-lar bilinməyən səhvlərdir. Əsasən Runtime(programın işə salınması) zamanı baş verə bilər. Java bu səhvlərin haradan yarandığını anlaya bilmir.

**Throws** (atır, tullayır) səhvlərin handle olunmadan atılmasını təşkil edir. Catch vasitəsi ilə error düzəldilib proqrama qaytarıla bilər amma throws yazıldığı zaman səhv proqramdan düzəldilmədən atılır. Bütün throws exception-ları class-ının irsiliyini daşıyır. Keçək throws-un istifadə olunduğu məqamlara. Fərz edək biz proqram yazırıq və proqram daxilindəki hər bir xırda exception-ı handle etmək istəmirik. Bu zaman proqramın əvvəlində, yəni main hissəsində bildiririk ki, istisna hal varsa onu əvvəlcədən susdur, səhvin üzə çıxmasına imkan vermə. Bunun üçün throws əmrini aşağıdakı kimi istifadə edirik.

```
-----  
public static void main(String[] args) throws Exception  
-----
```

Java-da bir neçə exception eyni anda atıla bilər. Proqramda istifadə olunan paketlərə uyğun olaraq exception-lar main-də ard-arda yazılır.

```
-----  
public static void main(String[] args) throws SQLException,  
ArithmeticException, ClassNotFoundException  
-----
```

**Throw** əmri “throws” əmrindən fərqli olaraq proqram daxilində bizə error-ların ortaya çıxardılmasında köməklik edə bilər. İndi fikirləşəcəksiniz ki, bəs exception tutmaq və həll etmək catch-in işi idi? Bəli istisna halları həll etmək catch-in işidir. Throw bir növ proqram daxilində müxtəlif yerlərdə səhvin çıxma bilmə ehtimalını yoxlayır. Səhv ortaya çıxandan sonra catch vasitəsilə onu tutulub həll edilməsini təşkil edir.



```

package com.ZJAVAKITABI;
public class Throw {
public static void main(String args[]) {
int a = 5, b = 5;

try { if (a == b) {
throw new ArithmeticException(); }
} catch (AbstractMethodError e) {
System.out.println("Abstract method səhvi");
} catch (ArithmeticException e) {
System.out.println("Hesablama səhvi tutulub");
} catch (Exception e) {
System.out.println("İstisna hal tutulub");
} finally { System.out.println("Proqramın sonu"); }}}

run:
Hesablama səhvi tutulub
Proqramın sonu

```

Yuxarıdakı misalda gördüyümüz kimi müxtəlif səhvlərin baş vermə ehtimalına qarşı fərqli exception-lardan istifadə etmişik. Try bloku içində şərt vermişik və düz olduğu halda ArithmeticException-in atılmasını istəmişik. Təbii olaraq atılmış istisna hala uyğun olaraq catch onu tutur və bizə müvafiq mesajı verir.



### MySQL database-in qurulması

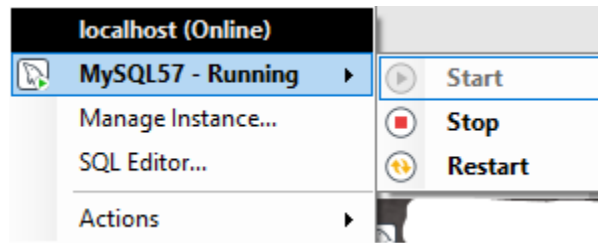
MySQL məlumat bazası opensource(açıq mənbəli) layihədir. Yəni bu bazadan istifadə etmək üçün pul xərcləməli olmayacaqsınız. Beləliklə Netbeans-lə MySQL-i bir-birinə bağlamaq üçün ilk olaraq <https://dev.mysql.com/downloads/installer/> -linkinə daxil olmalısınız. Açılan səhifədə aşağıdakı şəkildə göstərilən iki yükləmə variantından birini seçməlisiniz.



Məsləhətdir ki, əlavə problemlər yaşamamaq üçün “mysql-installer-community” variantını seçəsiniz. MySQL installer açıldıqdan sonra “Developer default” seçilmiş kimi saxlayın. Web community kimi yükləmələrdə bəzi faylların internetdən yüklənməsi tələb olacaq. Bu zaman “Execute” düyməsini sıxın. “I have read and accepted terms” seçin və təklif olunan proqramları yükləyin. Növbəti addımlar üçün Next düyməsini sıxın. Əgər hansısa əlavə komponentin yüklənməsində problem göstərərsə, o zaman yükləmədən növbəti səhifəyə keçmək üçün xəbərdar edici dialoq pəncərəsindəki “Yes” düyməsini sıxın. Dialoq pəncərəsi əsasən standart MySQL funksiyalarına mane ola biləcək komponentlərə bağlı olmur.

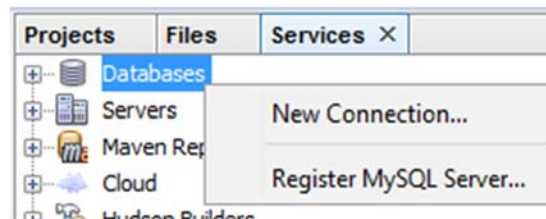
Type and Networking bölməsində Config type(Konfigurasiya tipi), Development Machine(Proqram hazırlama maşını) seçilmiş qalsın. TCP-İP(3306) və Open Firewall port seçilmiş olsun. Növbəti səhifədə Root Account(kök, əsas hesab) üçün şifrə təyin edirsiniz. MySQL Root Password(əsas şifrə) təyin edin və təkrar yazın. Sonra next (ola bilsin execute) sıxıb finish-ə gə bilərsiniz. Məsləhətdir ki, şifrəni harasa qeyd edəsiniz. İtirildiyi zaman şifrəni bərpa etmək çox vaxt aparır. Windows-da sağ alt küncdə MySQL Notifier-in ikonası çıxır. Onun üzərinə sıxaraq çıxan menyudan MySQL-ə start

və ya stop əmrini verə bilərsiniz. Əgər Start verilibsə ağ , stop əmri verilibsə qırmızı rəngdə  ikona təsvir olunacaqdır.

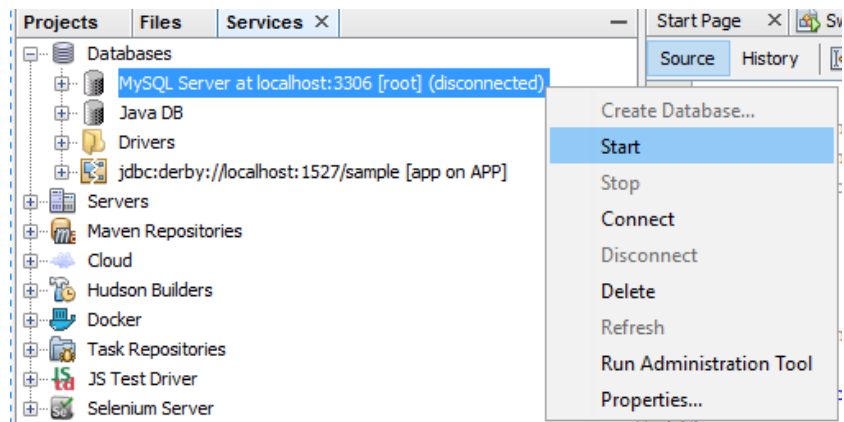


MySQL Workbench pəncərəsi açıldığı zaman orada Local Instance düyməsini sıxın və açılan xırda pəncərədə root şifrənizi daxil edin. Aşağıda olan “Remember password” qutusunu qeyd edin. Əks halda hər dəfə sizdən şifrə daxil etməyi tələb edəcək. İşlədiyi halda LocalInstanceMySQL pəncərəsi açılacaq. Pəncərəni bağlayın və Netbeans-ə keçin.

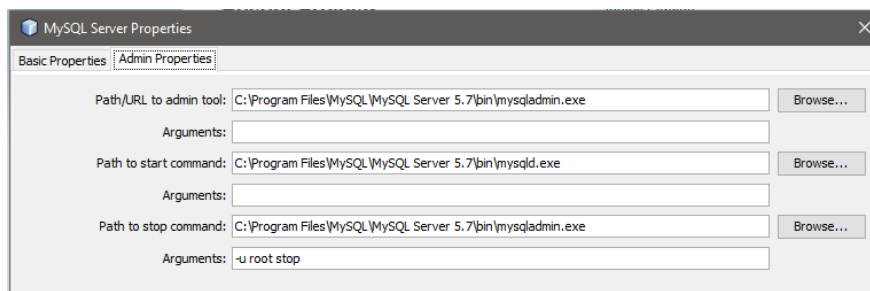
NetBeans-də sol tərəfdə Project-lər olan hissədə Services yazılan bölməyə keçin və Databases yazısının üzərinə sağ düymə sıxın. Açılan variantlardan Register MySQL Server seçin.



Açılan pəncərədə aşağıda şifrənizi yazırsınız. Sonra Ok düyməsini sıxırsınız. Artıq MySQL Netbeans-lə bağlanıb. Hazırda Databases hissəsini açın və sağ düymə ilə MySQL serverə start verin.

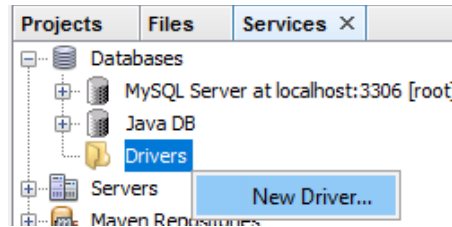


Növbəti qarşınıza çıxacaq pəncərədə MySQL start əmri üçün yolun seçilmədiyi göstəriləcək.

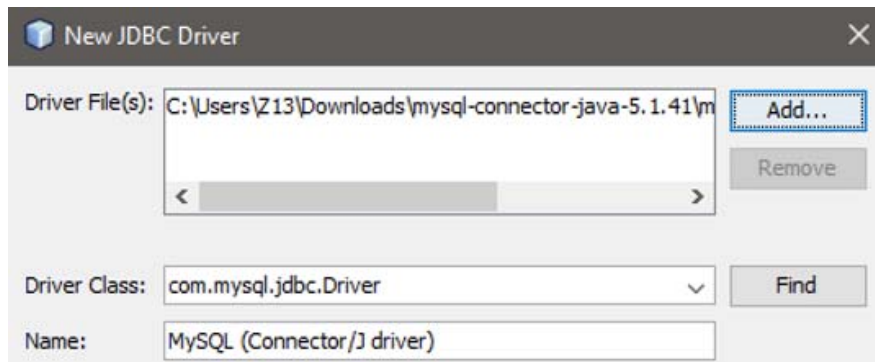


Əməliyyat sistemindən asılı olaraq MySQL yükləmə faylları fərqli yerlərdə yerləşdirilə bilər. Məndə “mysqld” faylının yerləşdiyi yer belədir(“C:\Program Files\MySQL\MySQL Server 5.7\bin\mysqld.exe”). Faylı qeyd etdikdən sonra database işləməlidir. “Mysqld.exe” bizə məlumat bazasına qoşulma əmrini verilməsi üçündür. Eyni zamanda bizə məlumat bazası ilə işimiz bitdiyində stop əmri vermək üçün “Path stop command” hissəsinə “mysqladmin.exe” proqramını əlavə etmək və Arguments hissəsində “u root stop” əmrini yazmaq lazımdır.

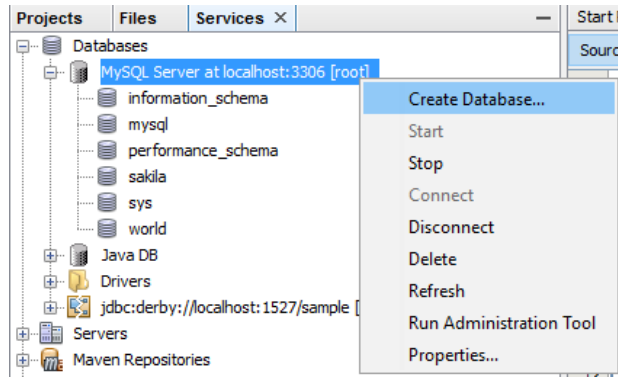
Bəzən “JDBC Driver for MySQL is not registered” səhvi göstərilir. Bu zaman (<https://dev.mysql.com/downloads/connector/j/>) göstərilən ünvandan “mysql-connector-java” zip faylı endirin. Faylı unzip etdikdən sonra, içərisindəki “mysql-connector-java-5.1.41-bin.jar” faylını tapın.



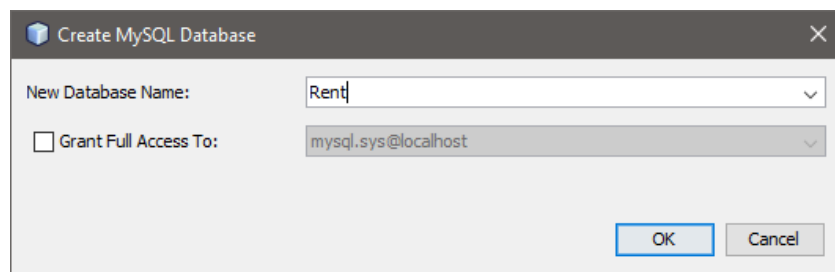
Yuxarıda göstərilən Services bölməsindəki Drivers qovluğunun üzərinə sağ düymə sıxın və “New Driver” düyməsini seçin.



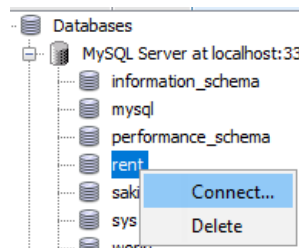
Add düyməsi ilə “Mysql-connector-java-5.1.41-bin.jar” faylını tapın və ok düyməsini sıxın. Fayl əlavə olunduqdan sonra Netbeans-də Service-lər bölməsindən “MySQL Server at localhost” yazısının üzərinə sağ düymə basaraq “Start” əmrini verin. İndi yeni məlumat bazası yaratmaq üçün solda yerləşən Service-lər bölməsində Database qovluğunun üzərinə sağ düyməni sıxırıq və “Create Database” əmrini veririk.



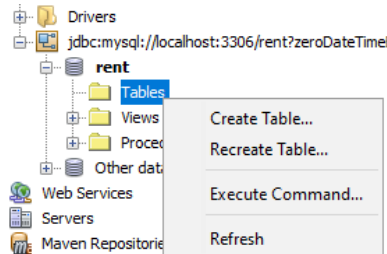
Açılan pəncərədə yeni yaradılacaq məlumat bazasına ad verirsiniz.



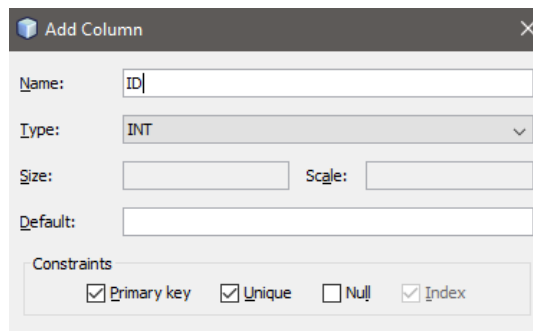
Daha sonra həmin məlumat bazası “MySQL Server at localhost” - un alt siyahısına əlavə olunur. Siyahıdan yaratdığınız bazanı tapıb üzərinə sağ düymə sıxın və “Connect ”(qoşulma) bazaya qoşulma əmrini verin.



“jdbc:mysql://localhost:3306/rent?zeroDateTimeBehavior=convert ToNull [root on Default schema]” yazısı göstərir ki, baza quraşdırılıb. Baza məlumatları cədvəlini yaratmaq üçün “Tables” (cədvəl) hissəsinin üzərinə sağ düymə ilə sıxın və “Create Table” əmrini verin.



İlk olaraq bizə bütün “Relational database”-lər üçün mühüm olan “Primary key”-i (Əsas açar) yaradırıq. Unique (unikal, təkrəredilməz) olmalıdır. ID (Identification - müəyyənədiç) column (sütun), tipi (INT) integer təyin edilməlidir.



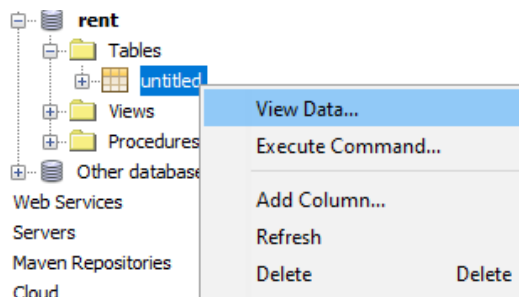
Daha sonra “CustName” və “Car” sütunu üçün VARCHAR, size 50 təyin edirəm. “Price” sütununu INT kimi yaradırəm.

Create Table

Table name:

Key	Index	Null	Unique	Column name	Data type	Size	Scale
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	ID	INT	0	0
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	CustName	VARCHAR	50	0
<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Car	VARCHAR	50	0
<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Price	INT	0	0

Program icarəyə verilən maşınların bazası ilə işləmək üçün nəzərdə tutulub. Lazım olan sütunlar əlavə edildikdən sonra OK düyməsi sıxılır. Sol tərəfdə yaratdığımız baza cədvəlinə baxmaq üçün cədvəl üzərinə sağ düymə sıxaraq View Data variantını seçirik.



Gördüyümüz kimi NeatBeans-də sağ tərəfdə MySQL qoşulması göstərilir və bizdən baza ilə işləmək üçün yeni əmr almağı gözləyir.

```

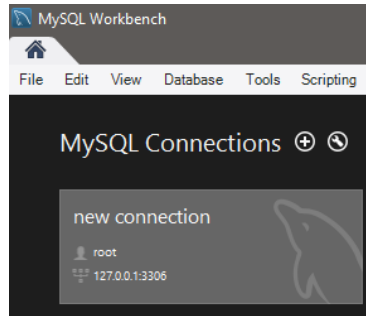
Start Page x SQL 1 [jdbc:mysql://localhost:33...] x
Connection: jdbc:mysql://localhost:3306/rent?zeroDateTimeBeha
1 SELECT * FROM untitled LIMIT 100;
2

```

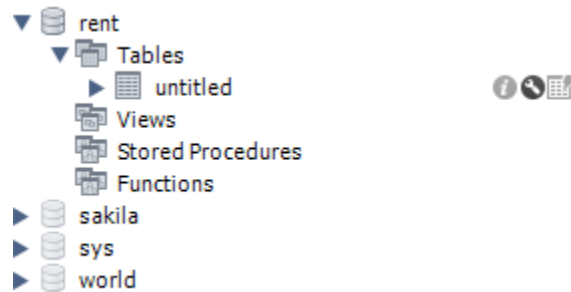
Bir məqama nəzərinizi çəkmək istəyirəm. Yaratdığımız cədvəldə “Create Table” bölməsində yaratdığımız cədvələ maraqlı bir ad verməyi unutmuşuq. “Untitled”(başlıqsız) sözünü dəyişmək üçün



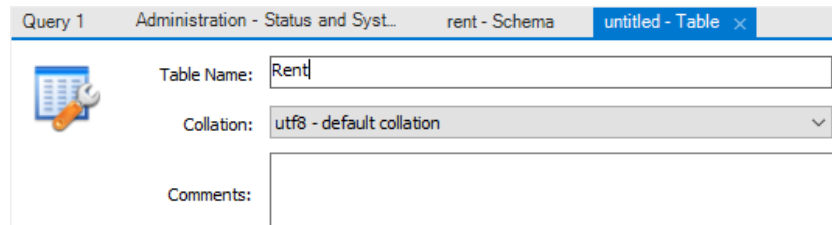
MySQL-in WorkBench programından istifadə edək. Programı Start menyusundan tapın və “new connection” düyməsini sıxın.



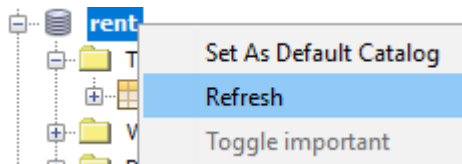
Açılan programın sol alt hissəsindəki Tables bölməsi daxilində yaratdığımız cədvəli tapın.



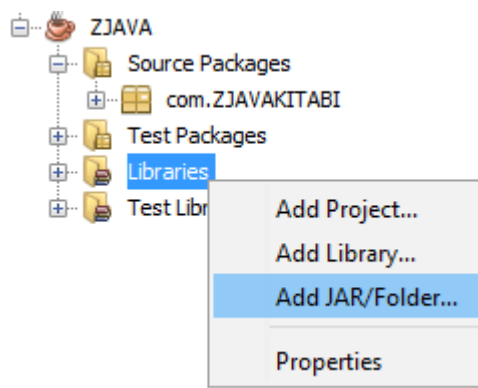
“Untitled” sözünün sağında görünən açar düyməsini sol düymə ilə sıxın.



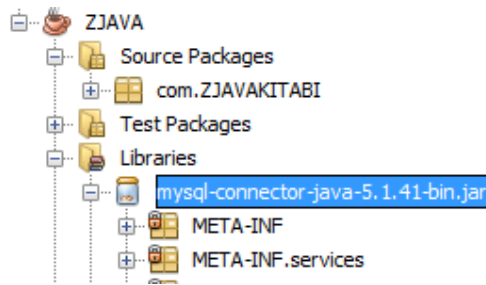
Yuxarıda cədvəlin adını dəyişdirdikdən sonra “Apply” düyməsini sıxın. Bizə cədvəlin kiçik hərflə yazılmalı olduğu mesajı göstəriləcəkdir. Açılacaq yeni pəncərədə də eynən Apply və Finish düyməsini sıxın. Proqramı bağlayıb qayıdaq Netbeans-ə.



Dəyişikliyi əldə etmək üçün “rent” bazasının üzərinə sağ düymə sıxaraq “Refresh” düyməsini seçin. İndi keçək Netbeans-də Projects olan hissəyə.



Burada Library(kitabxana) bölməsini tapın və üzərinə sağ düymə sıxmaqla (Services bölməsindəki Driver-ləri əlavə etmək hissəsində bizə tanış olan) “mysql-connector-java-5.1.41-bin.jar” faylını əlavə edək.



Fayl əlavə olunduqda onunla birgə Java-nın MySQL ilə işləyə biləcəyi bütün Class-ları və Driver-ləri siyahıda sıralanır. MySQL ilə işləmək üçün mühit yaratdıqdan sonra onu test eleməyə vaxt gəldi çatdı. SQLtest adında Java Class faylı yaradaq.

```
-----  
package com.ZJAVAKITABI;  
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.SQLException;  
  
public class SQLtest {  
    private static final String USERNAME = "root";  
    private static final String PASSWORD = "root1";  
    private static final String CONN_PATH =  
    "jdbc:mysql://localhost:3306/rent?useSSL=false";  
  
    public static void main(String[] args) throws SQLException {  
        Connection conn = null;  
        conn = DriverManager.getConnection(  
        CONN_PATH, USERNAME, PASSWORD);  
        System.out.println("MySQL bazasına qoşuldu");  
    }  
}  
-----  
run:  
MySQL bazasına qoşuldu  
-----
```

Yuxarıda gördüyümüz misalda MySQL-in çalışması üçün lazım olan paketləri import edirəm. Paketlərin əvvəlində “java.sql” yazılıb. Yəni Java-nın SQL ilə işləmək üçün paketi. Əgər hər hansı səbəbdən bizə paketin içində olan bütün class-lar lazımdırsa o zaman “ java.sql.\* ” yazıb bir əmrlə bütün paketi import eləyirik. Qoşulmanı test etmək üçün bizə “DriverManager”, “Connection” və “SQLException” class-ları olsa yetərlidir. Baxamayaraq ki, biz MySQL bazasına qoşulmaq üçün bu qədər mərhələ keçmişik, Java bizim ondan nə istədiyimizi hələdə bilmir. DriverManager class-ı Java proqramı ilə məlumat bazası arasında əlaqə saxlaya bilməsi üçün driver-ləri idarə etməyə kömək edir. Həmin class-ın tələbinə uyğun olaraq bazanın yeri(jdbc:mysql://localhost:3306/rent) olaraq göstərir. Eyni zamanda həmin bazaya daxil olmaq üçün istifadəçi adı(root) və şifrəni(root1) təqdim edirik. Connection class-ının “conn” obyektı isə baza ilə işləmək üçün müxtəlif method-lar təklif edir. Bu vəsilə ilə SQL bazalarına qoşulma zamanı Connection class-ının obyektı DriverManager class-nın qurduğu bağlantı üzərindən SQL bazası ilə işləyə bilir. Bundan əlavə proqramın main hissəsində bildiririk ki, bizə SQL ilə bağlı exception göstərməsin. Sonda qoşulma uğurlu olursa bizə məlumat verir.

### **Məlumatı bazaya yazma və silmə proqramı**

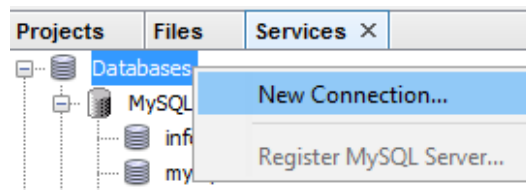
Artıq müəyyən səviyyədə proqram yazma bilirik. Son olaraq bildiklərimizi cəmləyib yekun proyektimizi yazmaq. Kirayə maşın verən bir müəssisə üçün məlumat bazası ilə işləyə bilən bir sadə proqram hazırlayaq. İlk olaraq bir JFrame Form yaradıırıq. Proqram aşağıdakı dizaynda ola bilər. Siz öz zövqünüzdə görə proqramı dizayn edə bilərsiniz.

Hesabın kodu	<input type="text"/>	Bazaya əlavə et
Müştərinin adı	<input type="text"/>	
Maşının təsviri	<input type="text"/>	Bazadan sil
Xidmətin dəyəri	<input type="text"/>	

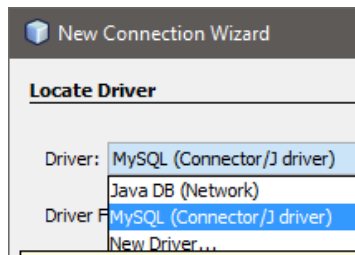
Təsvir olunan sahələrə uyğun MySQL bazasında cədvəl hazırlayırıq.

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
ID	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
CustName	VARCHAR(50)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
Car	VARCHAR(50)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
Price	VARCHAR(50)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL

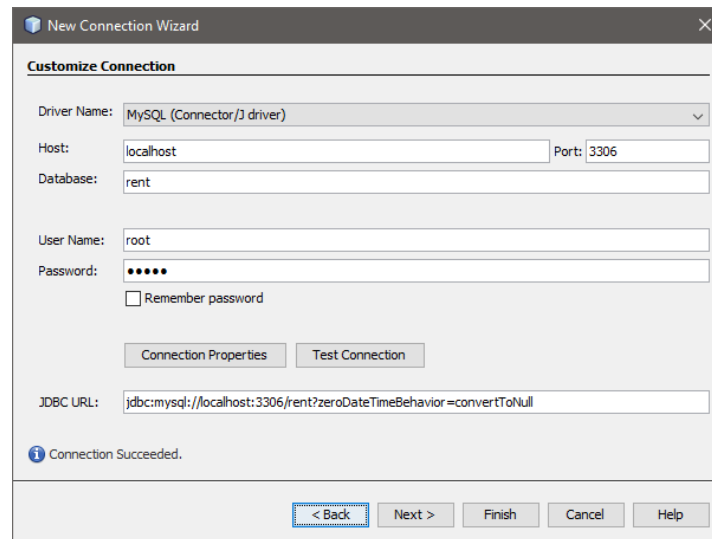
Mən cədvəli MySQL Workbench ilə hazırlamışam. Daha sonra onu yeni bağlantı kimi Java-da əlavə etmişəm.



Yeni bağlantı kimi əlavə etmək üçün Services bölməsindən Databases sözünün üzərinə sağ düymə sıxaraq orada New connection əmrini seçirsiniz.



Daha sonra açılmış pəncərədə MySQL(Connector/J driver) variantını seçib Next düyməsini sıxırsınız. Açılan pəncərədə qoşulacağınız Database adını dəyişib yazırsınız. Şifrənizi də daxil etdikdən sonra Test Connection düyməsini sıxıb məlumat bazası ilə bağlantınızı yoxlayırsınız. Uğurlu qoşulma olduğu zaman Connection Succeeded mesajı göstəriləcəkdir. Finish düyməsini sıxın və JFrameForma qaydın.



JFrameForm-un Source hissəsinə daxil olun və aşağıda göstərilən paketləri import edin.

```

package com.ZJAVAKITABI;
import java.sql.PreparedStatement;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import javax.swing.JOptionPane;
public class Rent extends javax.swing.JFrame {
    private static final String USERNAME = "root";
    private static final String PASSWORD = "root1";
    private static final String CONN_PATH =
    "jdbc:mysql://localhost:3306/rent?useSSL=false";

```

Məlumat bazasına bağlanmaq üçün lazım olan parametrləri elan edin. Bazaya qoşulma ünvanı “jdbc:mysql://localhost:3306/rent”-dir. MySQL-in 5.5-ci versiyasından sonrakı versiyalar daxil olmaqla serverin SSL sertifikatı ilə bağlı qaydaları var. Bu qaydalarda SSL sertifikatı tanınmayan serverlərdə “?useSSL=false” istifadə olunması tələb olunur. Siz əgər “jdbc:mysql://localhost:3306/rent”-olaraq ünvanı göstərsəniz proqram yenə də işləyəcək. Lakin xırda bir xəta olduğu elan ediləcəkdir. Rent bazasına daxil edilmiş məlumatların saxlanması üçün yaratdığım “Bazaya əlavə et” düyməsi variable olaraq adı “btn\_add”-dir. Beləliklə həmin düymənin üzərinə iki dəfə sıxdıqda bizi aşağıdakı göstərilən kod yazma hissəsinə gətirir. Yaratdığım Textfield-lərdən götürdüyüm məlumatları daxili dəyişənlərə mənimsədirəm və try blokunun içində istifadə etmək üçün hazırlayıram. Connection class-ını artıq tanıyıırıq. O bizə DriverManager class-ı ilə birgə MySQL bazasına qoşulmanı və onunla işləməni təşkil edir. PreparedStatement isə MySQL əmrlərini əvvəlcədən hazırlayaraq MySQL bazasında istifadə etməyə imkan yaradır.

```

private void btn_addActionPerformed(java.awt.event.ActionEvent
evt) {
String custname= txt_custname.getText();
String id= txt_id.getText();
String car=txt_car.getText();
String price= txt_price.getText();
Connection conn=null;
PreparedStatement pstmt=null;
try{ conn = DriverManager.getConnection(
CONN_PATH, USERNAME, PASSWORD);
pstmt=conn.prepareStatement("insert into rent values(?,?,?,?)");
pstmt.setString(1, id);
pstmt.setString(2, custname);
pstmt.setString(3, car);
pstmt.setString(4, price);
int i=pstmt.executeUpdate();
if (i>0){JOptionPane.showMessageDialog(
null, id + " nömrəli hesab yaradıldı"); }
}catch(SQLException e){ JOptionPane.showMessageDialog(
null, "Məlumat yazıla bilmədi. Xətanın təsviri" + e); } }

```

PreparedStatement dəyişəni Connection dəyişəni vasitəsilə 4 sual işarəsi göstərib MySQL-dəki bazamızda 4 sahənin olduğunu bildirir. Hazırlanmış kodla bildiririk ki, insert(daxil et) rent bazasının içinə(into) values(dəyərləri) hansı ki, “pstmt” dəyişəni göstərir. Bazada cədvəlin birinci kvadratına “id”, ikinci kvadratına “custname”, üçüncü kvadratına “car” və dördüncü kvadratına “price” məlumatlarını təyin edirik. “i” dəyişəni “pstmt” dəyişəninin icra olunma və yenilənməsini mənimsəyir. Şərtlə bildiririk ki, əgər “i” sıfırdan böyükdürsə o zaman dialog pəncərəsini aç və “id” nömrəli hesabın yaradılmasını bildir. Əks halda səhvi tut və müvafiq məlumatı ver. Məlumatın silinməsində eyni üsuldən istifadə etmişəm. Əgər “id” dəyişənindəki məlumat bazada varsa o



zaman məlumatı sil. Məlumatı silmə qaydası da MySQL kodu ilə göstərilmişdir.

```
private void btn_removeActionPerformed(java.awt.event.ActionEvent  
evt) {  
String id= txt_id.getText();  
Connection conn=null;  
PreparedStatement pstmt=null;  
try{ conn = DriverManager.getConnection(  
CONN_PATH, USERNAME, PASSWORD);  
pstmt=conn.prepareStatement(  
"DELETE FROM rent WHERE ID = ?");  
pstmt.setString(1, id);  
int i=pstmt.executeUpdate();  
if (i>0){ JOptionPane.showMessageDialog(  
null, id + " nömrəli hesab silindi");  
}else{JOptionPane.showMessageDialog(  
null,"Silmək istədiyiniz" + id + " nömrəli hesab mövcud deyil"); }  
}catch(Exception e){ JOptionPane.showMessageDialog(null, e);} }
```

Demək olar ki, yuxarıdakı misallarda baza ilə işləmək üçün başlanğıc məlumatlar verilmişdir. Öyrəndiyimiz addımlar digər proqramçılar tərəfindən yazılan kodları anlamaqda bizə yardımçı olacaqdır. Professional proqram yazmaq üçün uzun müddət müxtəlif layihələr üzərində praktiki biliklər öyrənmək və populyar həll üsullarından istifadə etməyi öyrənmək lazımdır.

\*\*\*\*\*