

Z - JAVA

Zabil Ibayev

2014

Mündəricat

Kitab kimlər üçündür.	3
Kompüter sizi necə başa düşür və ona necə əmr verilir.	4
Biz kompüter proqramlarını necə yazma bilərik.	4
Proqramlaşdırma dilləri - nə üçün fərqlənir?	4
Java- nə ilə fərqlənir?	5
OOP –nin məntiqi və prinsipləri.	7
Obyektin təhlili.....	7
Sinifləndirmək - <i>CLASS</i> nədir?	8
Instance	8
Encapsulasiya.....	8
Abstraction	8
Inheritance(irsilik).....	8
Polymorphism(polimorfizm).....	8
Association(asosieyşn- əlaqə)	9
Aggregation(əqrəqeyşn-hissələrin birləşdirilməsi)	9
Composition(kompozisiya)	9
Java-da yazmaq üçün silahlanmaq :).....	10
Netbeans-lə tanışlıq.	10
Salam Java proqramı	13
NetBeans-də səhvlərin aydınlaşdırılması	16
JAVA-da dəyişənlər	17
String.....	17
String-lə bağlı praktiki məsələlər – “+=”	18
Bütün hərflərin böyük və ya kiçik yazılması – “.toUpperCase” və “.toLowerCase”.....	18
İki String-i müqayisə etmək – “==” və “.equals()”	19
Sətrin uzunluğunun müəyyənləşdirilməsi – “.length()”	19
Yazı daxilində axtarış vermə -“.contains()” və “.indexOf()”	20
Char	20
Rəqəm dəyişənləri	20
Integer	20
Double	20
Floating	21
Byte	21

Short	21
Long	21
Məntiq dəyişənləri və operatorları	21
Boolean	21
Rəqəm dəyişənləri ilə bağlı praktiki məsələlər	22
Riyazi operatorlar	22
Dəyişənləri qruplaşdırma və hesablama	22
Qalıqın ekrana verilməsi	23
Bir-bir artırma və azaltma	23
Riyazi operator ardıcılığı.....	23
Dəyişənlərlə hesablama və görüntü.....	25
Şərt əməlləri.....	26
Şərt operatorları – “==”, “!=”, “<”, “>”, “<=”, “>=”, “?”	26
Şərt əməlləri – if, else	26
Şərt əməlləri - switch, case, break, default	27
Dövr əməlləri - for , do, while	29
For	29
Mürəkkəb For əməlləri	29
While	30
Continue	31
Do while	31
Dövrün adlandırılması.....	31
Array	32
Birinci üsul -	33
İkinci üsul -	33
Çoxölçülü Array	34
Built-in Java Packages	35
Import.....	35
Access control(akses kontrol – girişin idarə olunması).....	36
Method nədir?	37

Kitab kimlər üçündür.

İlk olaraq onu bildirmək istəyirəm ki, kitab professional proqramçılar üçün deyil həvəskarlar üçün nəzərdə tutulub. Çünki özüm professional proqramçı deyiləm.

Azərbaycan dilində Java proqramlaşdırma dili üzrə ədəbiyyatın olmaması və ya mövcud kitabların yetərinə mənə aydın yazılmadığını nəzərə alıb digər həvəskarlarında belə çətinliklə 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 oxunuşunu yanında yazıram. Professional proqramist dostlarım təcrübələrinə əsaslanaraq proqramlaşdırmada yüksək nəticə əldə etmək üçün ingilis dili biliyinin vacib olduğunu deyirlər. Sərbəst şəkildə çalışacağınız zaman qarşınıza çıxan problemlərə həlləri nə bu kitab nə də başqası internet üzərindəki material qədər sizi qane edə bilməz. İnternetdə axtarışda daha dəqiq nəticə əldə etmək üçün isə ingilis dili bilmək vacibdir. Daha bir əmma var əməliyyat sistemi olaraq Windows OS 8.1, Java-da yazma bilmək üçün NetBeans proqramını ingilis dilində işlədirəm. Əgər heç bir ingiliscə biliyiniz yoxdursa belə narahat olmayın bu kitabdan yenə də nə isə öyrənə bilərik.

Əvvəlki eksperimentlərimi də kompüter şəbəkələrini öyrəndiyim zaman Zİ-N Kompüter şəbəkələri və Çatışmayan biliklər kimi kitablarımda bölüşmüşəm. Zİ-N Kompüter şəbəkələri kitabında da göstərdiyi kimi mənim tərəfimdən yazılan kitabların müəllifi göstərilməklə çap və satışdan gəlir əldə edilə bilər. Məncə bilik pulsuz paylaşılmalıdır.

Kompüter sizi necə başa düşür və ona necə əmr verilir.

Kompüterlər ikili say sistemi üzərində işləyir. İkili say sistemində 0 və 1 var. Bu qurğulardan hər hansı bir iş görməsi tələb edildiyində bizim verdiyimiz əmrlər 0 - yanlış, olmaz, dayan və ya 1 - doğru, olar, davam et kimi tərcümə edilir. Ümumilikdə isə belə başa düşmək olar kompüterin hansısa hissəsinin naqillərinə elektrik axımı verilsin ya yox.

Kompüterimizdəki işləri görən mikrosxem, xalq deyimi ilə kompüterin beyni CPU(si pi yu) adlanır. :) CPU(Central Processing Unit-Sentral prosessinq yunit) mərkəzi hesablayıcı qurğu anlamına gəlir. CPU-ya əmr verilməsi üçün istehsalçılar Instruction Set(instrakşn set) adlı instruksiyalar dəsti digər deyimlə qaydalar toplusu düzəldiblər. Bütün yazılan proqramlar prosessorla nə etmək istədiklərini bu qaydalar əsasında izah edirlər.

4

Biz kompüter proqramlarını necə yazmağa bilərik.

Sadə dildə desək proqram yazmaq adı 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 OS daxilindəki adi Texteditor-la və ya MSDOS-da yazıla bilər. Hər proqramlaşdırma dili fərqli üslubda müəyyən olunmuş kodlarını ikili say sistemə tərcümə edir. Windows daxilindəki CMD(command line –kommand layn-əmr sətiri) ilə DOS-da proqram yazmaq mümkündür. CMD-yə daxil olmaq üçün Start -dan sonra CMD yazaraq proqramı açın.

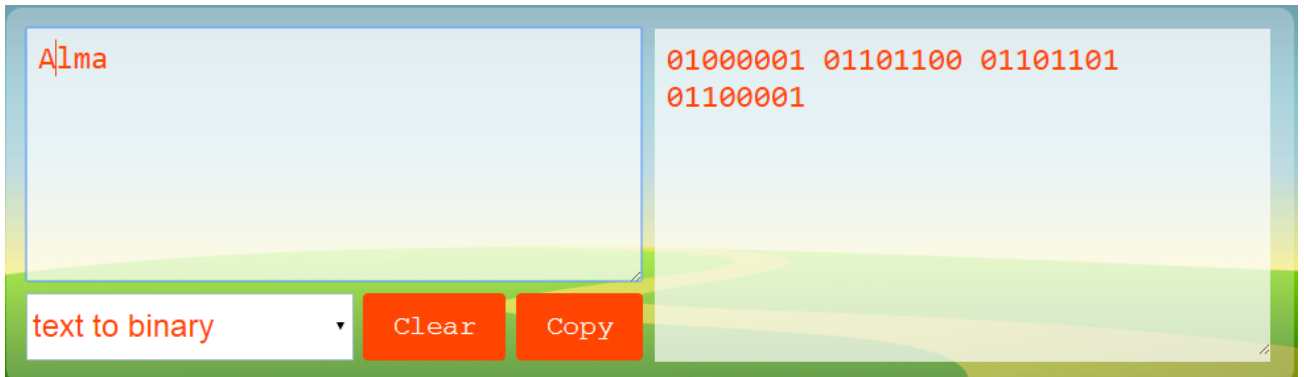


```
C:\>@echo .zabil
zabil
```

"@echo."- əmrindən sonra nə yazsanız yeni sətərə həmin yazı çıxacaqdır, sadə bir əmr. Bundan başqa DOS-da daha nə etmək olar deyirsinizsə "help" yazıb digər əmrlərə baxa bilərsiniz.

Proqramlaşdırma dilləri - nə üçün fərlənir?

Yuxarıda bildirildiyi kimi kompüter ikili say sistemindəki əmrləri anlaya bilər. Belə olan halda biz insanlar üçün bu şəkildə proqram yazmaq çox çətindir. İkili say sistemi ingilis dilində *Binary*(bayneri) adlanır. "<http://www.binarytranslator.com>" səhifəsində bunu rahat görmək olar.



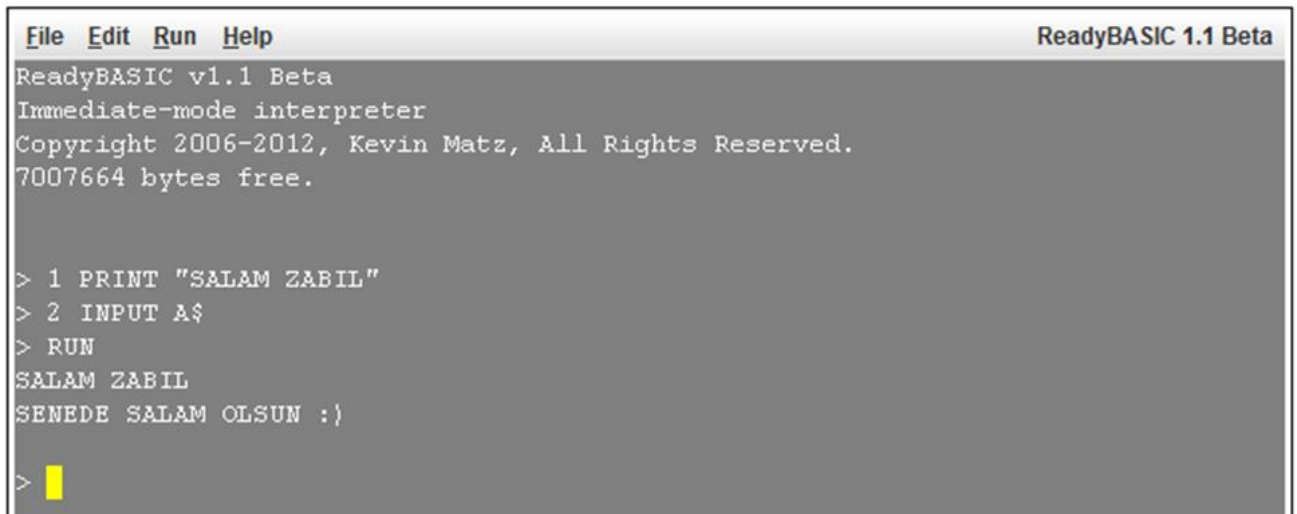
Sadəcə "Alma" sözünün sağdakı ikili say sistemindəki uzunluğuna baxın. :)

Proqramçılar bu üsulu sadələşdirmək üçün 16-lıq say sistemindən istifadə etməyə başladılar. İngiliscə 16-lıq say sistemi Hexadecimal(heksadesimal) 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* dilini misal çəkmək olar. *BASIC*-də verilmiş mətni ekrana çıxartmaq üçün tələb olunan əmrlər aşağıda təsvir olunub. Şəkildə göstərilən redaktor "http://www.readybasic.com" səhifəsindəndir. Bu səhifə Java dilində hazırlanıb.

5



```
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 sətir ayrıcalıqda oxunur sonra tərcümə və icra edilir. Bu üsulla işləyən proqrama "interpreter" - tərcümə proqramı deyilir. Yəni hər yazılan əmri kompüterin başa düşəcəyi dilə çevirir. Bu cür yaxınlaşma proqramının aşağı sürətdə işləməsinə səbəb olur. Əgər *BASIC*-də əmrlərin qabağında rəqəm yazmasan birbaşa əmrin icrasına keçir, bu səbəbdən ilk əmr *PRINT*-in (çap etmək, ekrana çıxartmaq) qarşısına rəqəm qoymuşam ki, 1-ci "Salam Zabil" sözünü ekrana çıxartsın sonra 2-ci əmri *INPUT*-u (daxil etmək) ekrana çıxartsın ki, məndə ona qarşılıq verim. *RUN*(ran)- proqramın oxunması kimi qəbul edək, proqramı işə salır. Bəs *A\$* nədir. *\$*- dəyişən deməkdir. *A* dəyişəni(*\$*) hər hansı bir məlumatla əvəz oluna bilər. Rəqəm və ya yazı. Nəticədə isə proqram məni salamlayır və məndən 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 yeni obyekt yaratmaq üçün proqramlaşdırma dili. Ən azı mənə bu cür anlamaq rahatdır.

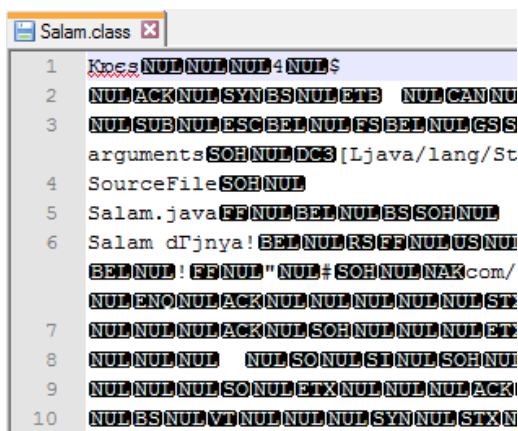
Obyekt hər hansı bir əşya, funksiyası olan qurum və ya canlı ola bilər. *OOP* əvvəlki dillərdən fərqli olaraq bir obyektə bir neçə yerdə istifadə etməyə imkan verir. Bir az texniki dillə desək, *OOP* yazdığınız proqramın

içindəki müəyyən kodları ayrı-ayrılıqda həmin və ya digər proqram daxilində obyektin xüsusiyyətlərini yenidən yaratmadan istifadə etmə imkanı yaradır.

Məsələn yaratdığınız maşın obyektini (eyni zamanda "maşın proqramını" əgər belə adlandırsaq) digər proqramın daxilindən çağırıb istədiyiniz yerə tətbiq edə bilərsiniz. Siz çağırılan obyektin kodunu tamamı ilə köçürmədən, sadəcə mənbəyini göstərməklə, ona verilən adı yeni proqramdakı müəyyən olunmuş yerə daxil edərək proqramı işə salırsınız.

Java OOP "Compile"(kompayl - komplektləşdirən, toplayan, birləşdirən) texnologiyasından istifadə edir. Java bütün *compile* etdiyi proqramlara ".class" sonluğunu əlavə edir. *Compile* olmuş proqram *bytecode*-a çevrilir. Əgər ".class" faylı görsəniz bilin ki içində *bytecode* var. *Java BASIC* kimi sadəcə *interpreter* texnologiyasından istifadə etmir. Kodları sətir-sətir və ayrı-ayrılıqda oxuyub tərcümə etmir. Əvvəlcədən proqram daxilində yazılan əmrləri müəyyən edir, sonra vahid şəkildə proqramın harada başlayıb, harada bitəcəyini, iş prosesində hansı əmrlərin yerinə yetirəcəyini anlayır. Yazacağımız "Salam dünya" proqramının *bytecode*-a çevirilmiş şəkli aşağıdakı kimidir.

6



```
Salam.class x
1 K0esNULNULNUL4NUL$
2 NULACKNULSYNBSNULETB NULCANNU
3 NULSUBNULESCBEENULFSBEENULGSS
argumentsSOHNULDCS[Ljava/lang/St
4 SourceFileSOHNUL
5 Salam.javaFEENULBEENULBSOHNUL
6 Salam dTjnya!BEENULRSFEENULUSNUL
BEENUL:FEENUL" NUL#SOHNULNAKcom/
NULBNONULACKNULNULNULNULNULST
7 NULNULNULACKNULSOHNULNULNULET
8 NULNULNUL NULSONULSINULSOHNUL
9 NULNULNULSONULETXNULNULNULACK
10 NULBSNULVTNULNULNULSYNULSTXN
```

Şəkildən də görüldüyü 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-nun versiyasından asılı olmayaraq bütün kompüterlərdə işləyə bilər.

İndi bütün yuxarıda izah etdiyim addımları bir araya toplayıb Java OOP-nin prosesi tam olaraq necə yerinə yetirdiyinə baxaq.

1. "Salam.java" proqramı yazılır;
2. "Salam.class" artıq compile olunur və fayl *bytecode*-da çevrilir;
3. JVM vasitəsilə *bytecode* binary koda çevrilir;
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

Obyekti real həyatdakı kimi virtual olaraq yaratmaq üçün onu tam incələmək, mümkün xüsusiyyətlərini anlamaq lazımdır. Təhlil üçün avtomobil obyektini götürək:

7

Avtomobil haqda bizə nə 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*(opereyşn-əməliyyatlar, funksiyalar) adlanır.

Digər bir misal olaraq bank hesabını götürə bilərik. Hesabda bizə nə bəllidir:

- 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 Metod və *Operation*-lardır. Yəqin ki artıq aydın oldu, hər hansı bir obyektin məlum olan xüsusiyyətləri *Attribute*-ları və *attribute*-ları işə salan *Method*-ları vardır. Bəzi hallarda *method*-lar *attribute*-la əlaqəli olmaya bilər.

Sınıflandırmaq - *CLASS* nədir?

Təhlil üçün avtomobil obyektini götürək. Avtomobilin ban növündən, firmasından və ya ölçüsündən asılı olmayaraq obyektəki *attribute*-lar və *method*-lar eynidir. Bu da onları eyni sinfinə 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 *attribute*-ları və *method*-ları 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(instans-instansiya, obyektin oxşarı). Məmin notbukum və Orxanın notbuku, notbuklar sinfinə, notbuk *class*-ına aiddir. Məmin və Orxanın notbuku notbuk *class*-nın *instance*-dir. Daxilində fərqli *attribute*-ları olmağına baxmayaraq ümumi göstəriciləri *class*-la eyni olan obyektlərdir.

El dilində belə deyək adından və ya sahibindən asılı olmayaraq notbuk deyəndə nə ağılına gəlir, bax elə o təsvir notbuklar sinfinə aid olması deməkdir. Kiməsə deyirsiniz mən notbuk almışam onu görməyən adam belə artıq təsəvvürü var ki notbuk sinfinə aid ola biləcək obyekt necə olmalıdır. Onun *instance*-ni fikirləşir(rəngini, ağırlığını və s.)

Encapsulasiya

Attribute-ların obyekt haqqında məlumat verdiyini artıq bilirik. Obyekt funksionallaşdıqda *encapsulasiya* *method*-lar vasitəsi ilə obyektin məlumatlarını gizlətmək rolunu oynayır. Proses sonundakı nəticə məlum olsa belə daxilə hansı *attribute*-ların qarşılıqlı əlaqəsindən nəticənin yarandığı gizlədilir.

Abstraction

Abstraction işə vacib detalları göstərməklə lazımsızları kənarlaşdırmaq funksiyasını icra edir. Misal olaraq avtomobili işə salan sahibi açarı döndürdükdə avtomobilin işə düşəcəyindən başqa obyektə gedən proseslərin necə icra olunduğunu görmür. O sadəcə açarı döndürüb avtomobilin işləməsinə gözləyir, digər proseslər ondan uzaq tutulur və ya gizlədilir.

Inheritance(irsilik)

Velosiped, avtomobil, yük maşını ayrı-ayrılıqda sınıfdırlar. Avtomobil sinfinin 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.

Polymorphism(polimorfizm)

Poly-çoxlu, *morph*-forma, davranış deməkdir. Bir neçə obyekt arasından ən düzgün olanın seçilməsində *polymorphism* *method*-dan istifadə edilir.

Bir az lori dildə izah edim. 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(əsosieyşn- əlaqə)

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

Aggregation(əqreqeyşn-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 obyektı və otaqları arasındakı əlaqəni göstərmək olar. Təxmini belə məntiqlə 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 silahlanmaq :)

JAVA-da yazmaq üçün bizə bəzi alətlər lazım olacaq. Java ORACLE(<http://www.oracle.com>) şirkətinin məhsuludur və saytlarında proqramçılar üçün NetBeans redaktoru və JAVA JDK(Java Development Kit)-Java İnkişaf Paketi pulsuz təqdim olunur.

<http://www.oracle.com/technetwork/java/javase/downloads/index.html> göstərilən ünvanndan kompüterinizin 32/64 bit olmasına baxaraq sizə uyğun versiyanı paket şəklində endirə bilərsiniz.

Accept License Agreement Decline License Agreement



JDK 8u25 & NetBeans 8.0.1

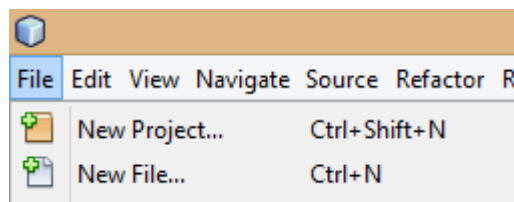
Java SE and NetBeans Cobundle (JDK 8u25 and NB 8.0.1)		
Product / File Description	File Size	Download
Linux x86	247.08 MB	jdk-8u25-nb-8_0_1-linux-i586.sh
Linux x64	243.56 MB	jdk-8u25-nb-8_0_1-linux-x64.sh
Mac OS X x64	381.18 MB	jdk-8u25-nb-8_0_1-macosx-x64.dmg
Windows x86	261.52 MB	jdk-8u25-nb-8_0_1-windows-i586.exe
Windows x64	274.73 MB	jdk-8u25-nb-8_0_1-windows-x64.exe

10

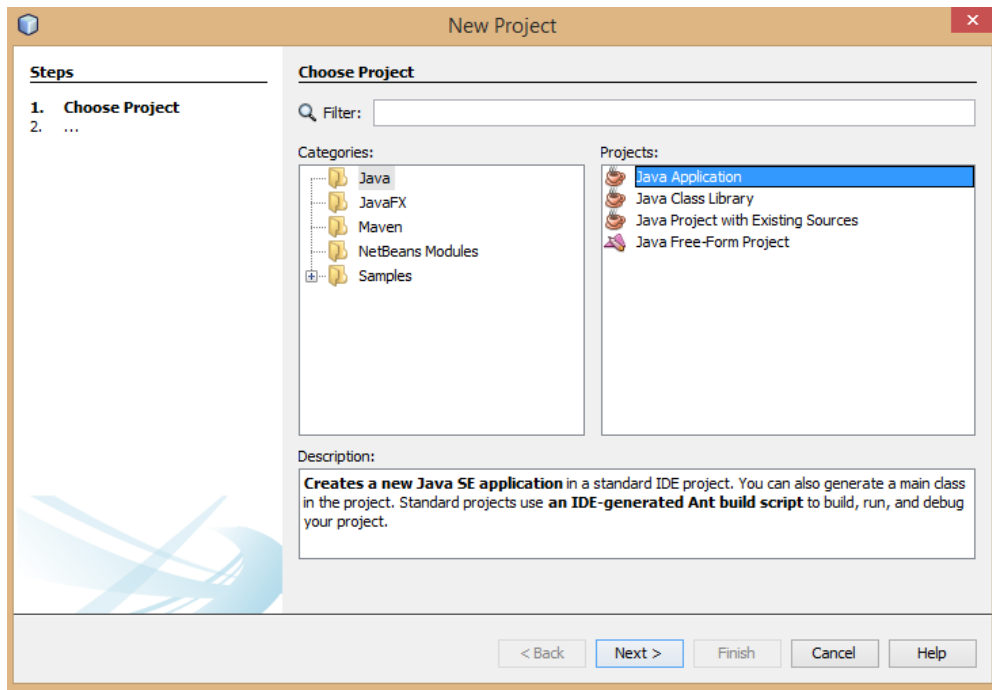
Proqramı endirdikdən sonra üzərinə sağ düymə sıxaraq Run as administrator sətirini seçin. Kompüterin administratoru kimi proqramları yükləyəndə əsasən yükləmə problemsiz başa çatır. Daha sonra şərtləri qəbul edərək Accept(aksept-qəbul etmək) yükləməyə davam edin. Hər endirilən versiyada yükləmə fərqli ola bilər. Sadəcə Finish(son) düyməsi çıxana qədər gözləyin.

Netbeans-lə tanışlıq.

Netbeans - də yeni proqram yazmaq istədiyinizdə sizə proqramın təklif etdiyi ilk olaraq yeni layihə yaratmaqdır.

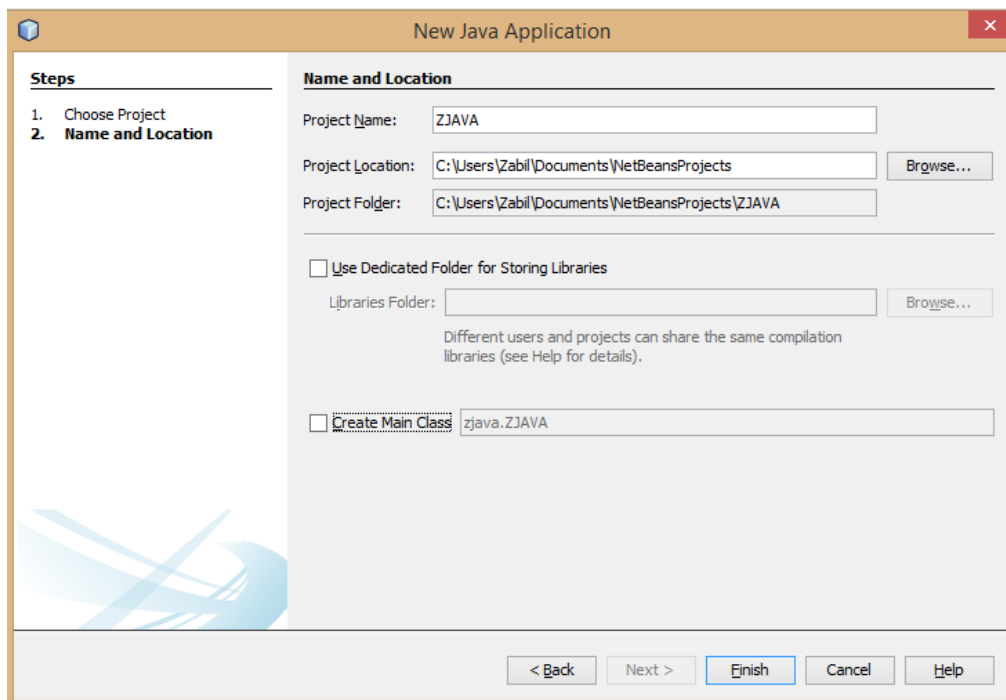



Yazacağımız proqramlar yaratdığımız layihənin tərkibində yer alacaqdır. Açılan pəncərədə bizdən nə tipli layihə yaratmaq istədiyimiz soruşulur. Biz şəkildə göstəriləndi kimi kateqoriyalardan(Categories) Java və layihələrdən(Projects) Java Application(Java Proqramı) seçirik. Sonra Next (növbəti) düyməsini sıxaraq növbəti səhifəyə keçirik.

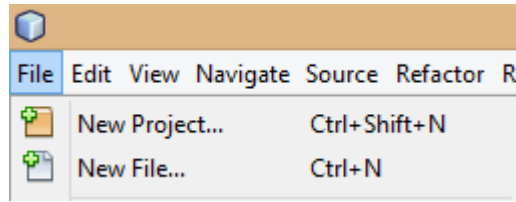


11

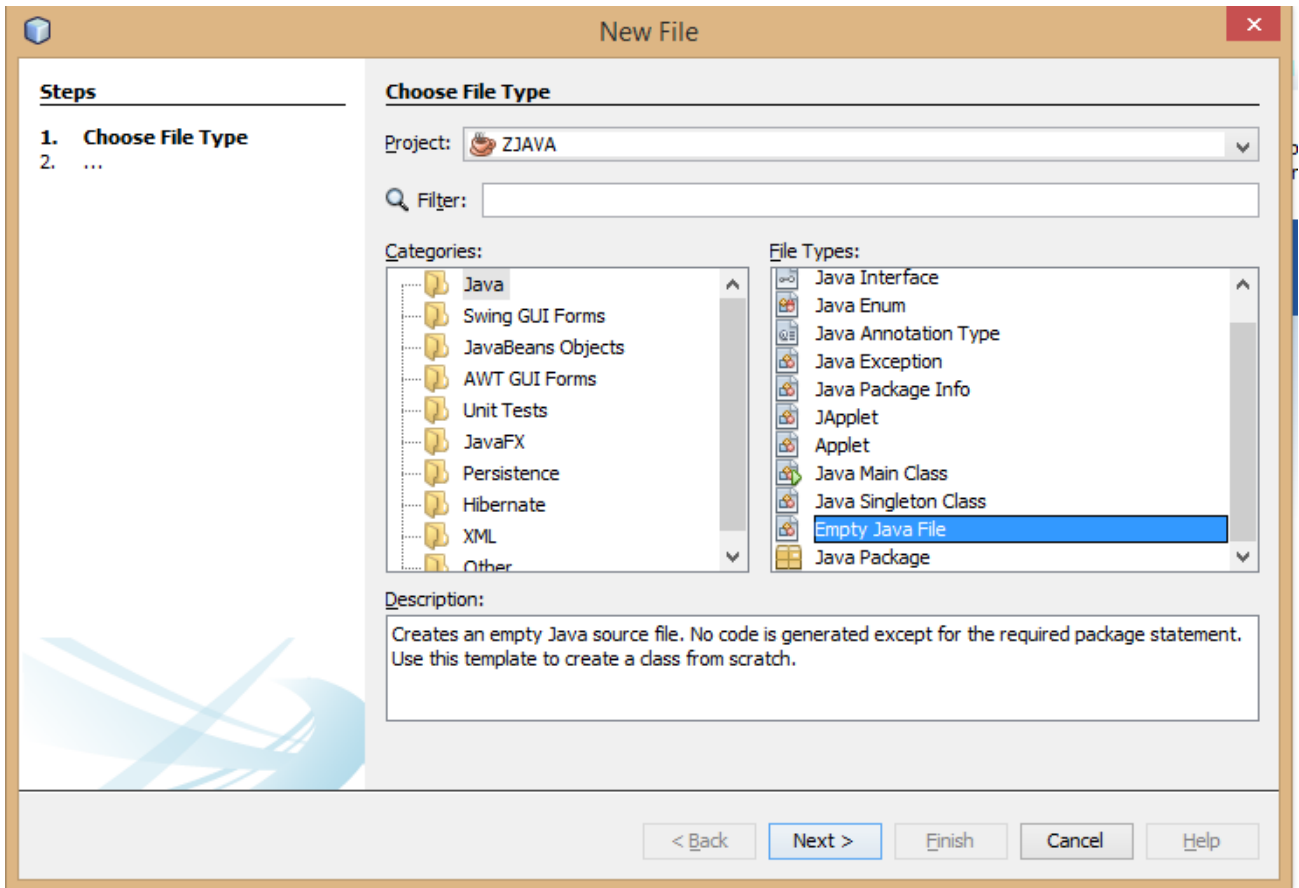
Layihəyə ZJAVA adını verirəm. Aşağıda Create Main Class qutusunu söndürün, məndə həmin hissənin qarşısında "zjava.ZJAVA" yazısı var. Sonda Finish düyməsini sıxın.



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



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

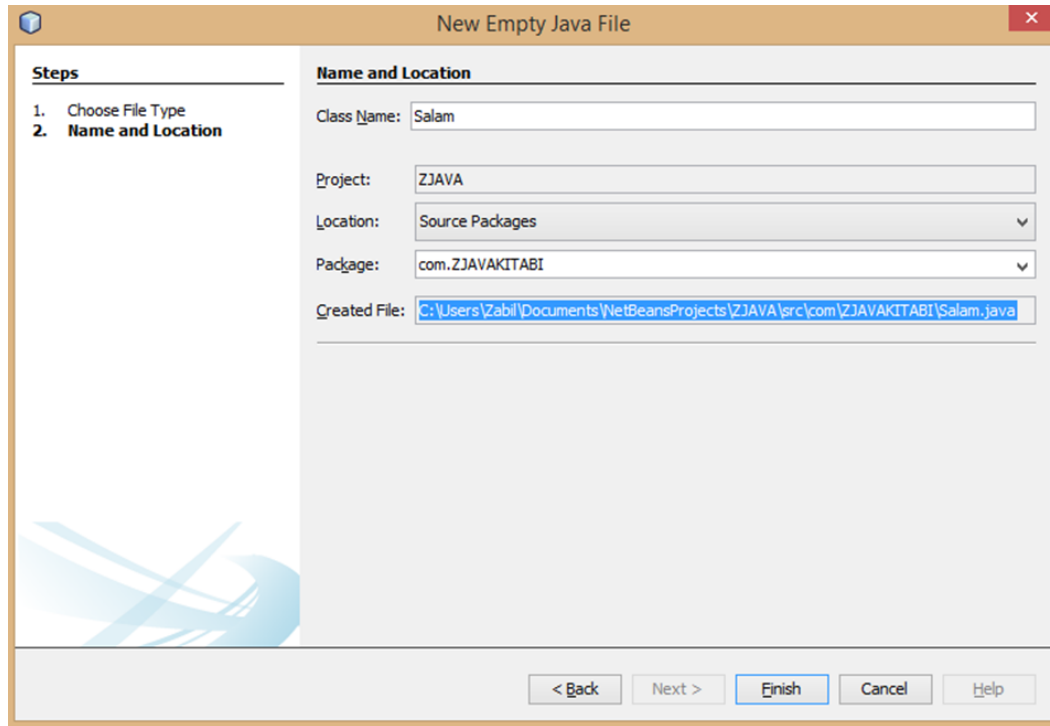


Növbəti səhifədə artıq müəyyən mənada proqram yazılışına başlayırıq.

Salam Java proqramı

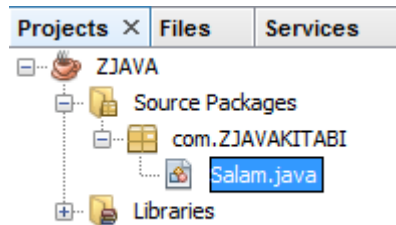
Class Name(Klas Neym - Sinfın adı) bura Salam proqramının adını yazmışam. Kompüterə əmr verirəm ki mənim proqramımın adı Salam olsun.

Package(Pəkiç-Paket) adı com.ZJAVAKITABI. Paket java proqramlarını bir arada qruplaşdırmaq üçündür. **Created File**(Kreyted - Yaradılmış) hissəsində yaratdığınız faylın harada saxlanması təyin etdiyiniz göstərilir.

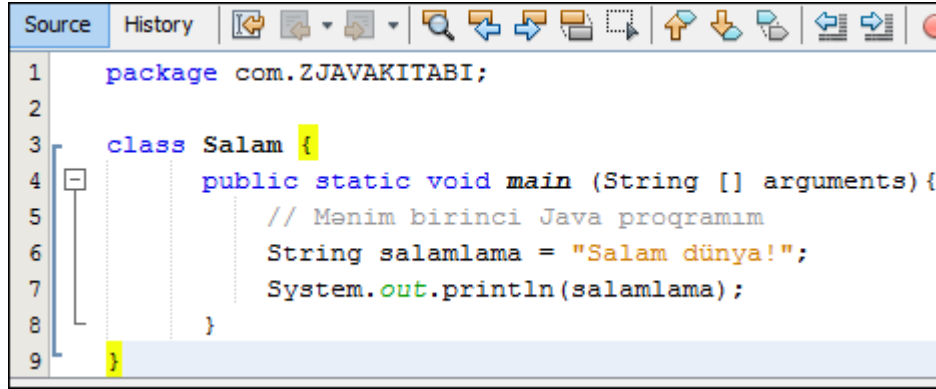


13

Finish düyməsini sıxdıqdan sonra Netbeans-də sol tərəfdə proqram faylının direktoriya üzrə necə yerləşməsi təsvir olunub.



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         // Mənim 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 özünüə lazım olacaq şəkildə dəyişdirə biləcəksiniz.

Package

Proqramları qruplaşdırmaq, Class proqrama ad vermək və onu sinifləndirmək üçündür. OOP-də izah etdiyimiz proqramı çağırmaq həmin bu Class-la bağlıdır. Proqram boyunca harada salam vermək istəsək Salam class-nı çağırıb salam verdirəcək :)))

public static void main (String [] arguments)

Bu sətiri sadəcə əzbərləyin. Java-da yazacağınız proqramlar əsasən belə başlayacaq. OOP tipli proqram yazdığımızı görə artıq bilirsinizki bir layihənin tərkibində çoxlu proqramlar ola bilər. Fərz edək ki, layihəni yazdıq və bir icon(ikona) şəklinə gətirdik və üzərinə sıxdığımızda sizcə ZJAVA layihəmiz hansı proqramı işlətməlidir. Bu zaman java sətirləri daxilində harada Main(Meyn - 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(pablik-ictimai) yaratdığımız obyektin digər proqramlar tərəfindən istifadə olunması və çağırılması üçün açıq olduğunu bildirir. Gizli deyil, ictimaiyyətə açıqdır.

Static

Static (static-sabit, dəyişməyən) instance-ı olmayan və dəyişdiyi zaman hər bir bağlantısına təsir edən class-ı bildirir. Instance-la çağırıla bilmir. Instance OOP-də ətraflı izah edilir.

Void

Heç bir dəyər geri qaytarmır. İxtisara salır.

String arguments

Bu hissə Netbeans-dən istifadə etmədən CMD vasitəsi ilə Java proqramlarını compile etmək üçün istifadə olunur. Netbeans işlətdiyimiz üçün bu hissəni ətraflı öyrənmirəm.

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, yəni o dəyəri daşıya bilər. String-in bu dəfə ki dəyişəninə "salamlama" adını vermişəm. Artıq "salamlama" = "Salam dünya!".

Dəyişənlərə istədiyiniz adı vermək mümkündür. Digər dəyişənlər və onların fərqləri proqram yazdıqca aydın olacaqdır.

Dalğalı mötərizə { }

Nə üçün istifadə olunur? Yazdığınız proqramların başlanğıc və bitmə nöqtəsini bildirir. Eyni zamanda proqram sətirlərini qruplaşdırır. Bu qruplaşdırmaya BLOCK(blok) deyilir. Bloklar digər blokların tərkibində ola bilər. NetBeans sarı rənglə blokların başlanğıc və son mötərizənin yerini görməkdə yardımçı olacaqdır.

// 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 şərhlərlə 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 stilidə dəyişmişdir. Burdakı ekrana çıxartmaq üçün istifadə olunan yazı şəklinin mürəkkəbliyi proqramın digər imkanlarının da olmasından xəbər verir. `System(sistem)`, `Out(aut-çöl)`, `println(print line(layn)-yeni sətirdən çap et)` hərfi mənada da məqsədini aydın edir “məlumatı sistemdən çölə çıxart və yeni sətirdən çap et”.



Qeyd: `System.out.println();` -bu yazını sadəcə “`sout`” yazıb sonra TAB düyməsinə basaraq sürətli şəkildə daxil edə bilərsiniz.

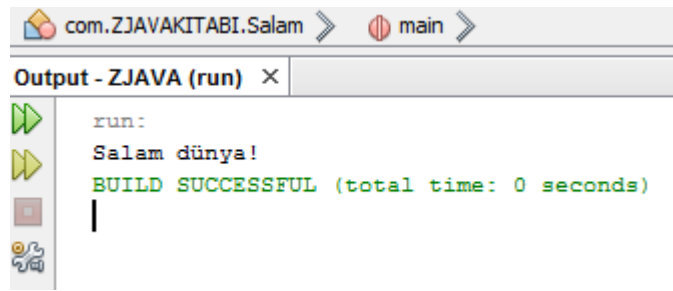
Qeyd: `Println` – yeni sətirdən çap edir, `print` isə eyni sətirdən çapı davam edir.

" ; " nöqtə vergül

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

Sonda proqramı işlətmək üçün birinci proqramı yaddaşa vermək sonra RUN etmək lazımdır. Yaddaşa vermək

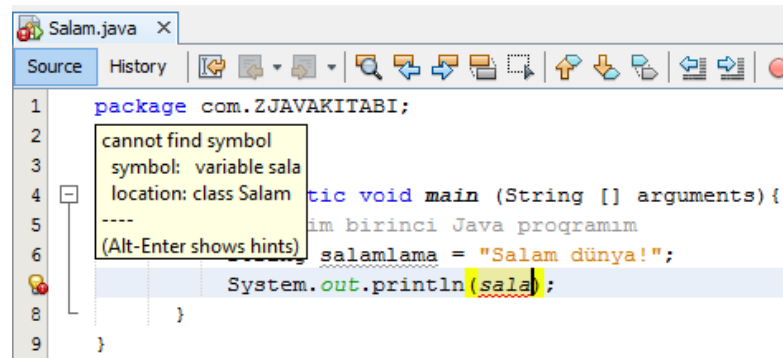
üçün CTRL+S və ya  düyməsini sıxmaq, proqramı işə salmaq üçün F6 və ya  RUN düyməsini sıxmanız yetərlidir. NetBeans-in daha bir üstünlüyü ondadır ki, yaddaşa verdiyiniz bütün fayllar avtomatik olaraq *compile* olunur. Proqramın sətirlərini düzgün yazdığınız halda aşağıdakı nəticəni əldə edəcəksiniz.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



```
com.ZJAVAKITABI.Salam > main >
Output - ZJAVA (run) x
run:
Salam dünya!
BUILD SUCCESSFUL (total time: 0 seconds)
```

NetBeans-də səhvlərin aydınlaşdırılması

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



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

cannot find symbol
symbol: variable sala
location: class Salam

(Alt-Enter shows hints)

Qeyd: İşimizi sürətləndirmək üçün artıq bəzi prosesləri təkrarən görüntülü 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 baxdığımızda nəyi nə üçün yazdığımızı anlamağa kömək edəcəkdir. NetBeans-də yazdığımız 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 özündə saxlayaraq programın iş prosesində lazım olan yerdə informasiyasını yaddaşdan çağırmaq üçündür. Artıq String-dən istifadə etmişik.

String-lə dəyişənlərə dəyər qazandırmaq üçün qoşa dırnaqdan istifadə edilir. Qoşa dırnaq daxilində yazılan bütün simvollar fərz edin ki 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. Bu səbəbdən nümunədəki rəqəmlər nəticədə sadəcə yan-yana yazılmaqla kifayətləniblər.

```
1 package com.ZJAVAKITABI;
2 class StringParametr {
3     public static void main(String [] arguments){
4         String reqem1 = "10"; String reqem2 = "10"; String cavab=reqem1+reqem2;
5         String i = "1 Zabilin \'27\' yaşı var.");//Tək dırnaq
6         String b = "2 Zabilin \"27\" yaşı var.");//Qoşa dırnaq
7         String a = "3 Zabilin \\27\\ yaşı var.");//Backslash işarəsi
8         String y = "4 Zabilin \t27 yaşı var.");//TAB -8 simvol sağa boşluq
9         String e = "5 Zabilin \b27 yaşı var.");//Backspace- Bir simvol sola silmə
10        String v = "6 Zabilin \r27 yaşı var.");//Carrige return-Sətir başına qayıtma
11        String z = "7 Zabilin \f27 yaşı var.");//Formfeed-Səhifənin başına qayıtma
12        String l = "8 Zabilin \n27 yaşı var.");//Newline- Yeni sətirdən
13        System.out.println(i);System.out.println(b);System.out.println(a);
14        System.out.println(y);System.out.println(e);System.out.println(v);
15        System.out.println(z);System.out.println(l);System.out.println(cavab);
}
```

com.ZJAVAKITABI.StringParametr > main > cavab >

Output - ZJAVA (run) x

```
run:
1 Zabilin '27' yaşı var.
2 Zabilin "27" yaşı var.
3 Zabilin \27\ yaşı var.
4 Zabilin      27 yaşı var.
5 Zabilin27 yaşı var.
27 yaşı var.
7 Zabilin 27 yaşı var.
8 Zabilin
27 yaşı var.
1010
BUILD SUCCESSFUL (total time: 0 seconds)
```

Bildiyimiz kimi String dəyişəninə mənimsədilən mətnlər qoşa dırnaq daxilində yazılır. Bəzən mətnin daxilində qoşa dırnaq və digər funksiyalar lazım gəlir, bu zaman yuxarıdakı nümunədən istifadə edə bilərsiniz. String-in müxtəlif funksiyaları mövcuddur (\' –tək dırnaq, \"qoşa dırnaq, \\- geri sləş, \t - TAB(8 simvol sağa boşluq), \b - Backspace(bəkspeys - bir simvol geriye, sola pozma), \r - Carrige return(kəric ritörn- sətirbaşına qayıtma), \f - Formfeed (formfid - formanı doldurma printerlərdə istifadə olunan köhnə funksiyadır, demək olar java-da istifadə olunmur), \n –Newline(nyulayn- yeni sətir)).

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

String-lə bağlı praktiki məsələlər – “+=”

Yuxarıda qeyd olunanlardan başqa String-in (+) operatorlarının birləşməsindən olan funksiyası vardır. (+) operatoru String-ə bir neçə dəyərin verilməsində istifadə olunur. Fərz edək ki, qeydiyyat pəncərəsi düzəldirsiniz və onun təvəllüd hissəsindəki məlumatları bir sətərə yığmaq istəyirsiniz.

```
1 package com.ZJAVAKITABI;
2
3 class Plusequal {
4     public static void main(String [] arguments){
5
6         String TEVELLUD = "";
7         TEVELLUD = TEVELLUD + "13" + "/";
8         TEVELLUD = TEVELLUD + "06" + "/";
9         TEVELLUD = TEVELLUD + "1987";
10        String Dogumtarixi = "";
11        Dogumtarixi += "13" + "/";
12        Dogumtarixi += "06" + "/";
13        Dogumtarixi += "1987";
14        System.out.println(TEVELLUD);
15        System.out.println(Dogumtarixi);
16    }
17 }
```

com.ZJAVAKITABI.Plusequal main

Output - ZJAVA (run) ×

```
run:
13/06/1987
13/06/1987
```

Yuxarıda göstərilən misalda “TEVELLUD” dəyişəninin təkrar-təkrar yazıldığını görürsünüz. (+) operatoru isə “Dogumtarixi” dəyişəninə daha az təkrar etməklə nəzərdə tutulmuş məsələni həll edir.

Bütün hərflərin böyük və ya kiçik yazılması – “.toUpperCase” və “.toLowerCase”

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

```
1 package com.ZJAVAKITABI;
2 class Boyukkiçik{
3     public static void main(String [] arguments){
4         String boyuk = "zabil ibayev";
5         String kiçik = "ZABIL IBAYEV";
6         System.out.println(boyuk.toUpperCase());
7         System.out.println(kiçik.toLowerCase());
8     }
9 }
```

Output - ZJAVA (run) ×

```
run:
ZABIL IBAYEV
zabil ibayev
```

İki String-i müqayisə etmək – “==” və “.equals()”

Fərz edək ki, qeydiyyat səhifəsi yaratmışıq. Səhifədə şifrə daxil edilir və əmin olmaq üçün ki adam şifrəni düzgün daxil edib 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.

```
1 package com.ZJAVAKITABI;
2 class Comparestring{
3     public static void main(String [] arguments){
4         String reqem1 = "15";
5         String reqem2 = "10";
6         System.out.println(reqem1.equals(reqem2));
7         System.out.println(reqem1==reqem2);
8     }
9 }
```

com.ZJAVAKITABI.Comparestring > main >

Output - ZJAVA (run) x

```
run:
false
false
```

19

Bu zaman “.equal()” metodundan və ya “==” qoşa bərabərlik operatorundan istifadə edirik. Nəticədə gördüyünüz kimi cavab “False”(fols) yalnızdır. Əgər bərabər olsa idi “True”(tru) 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()” (İqnor-fikir vermə, keys-hal, forma) hərfin böyük kiçik olmasına fikir vermədən bərabərliyi yoxla metodundan istifadə edin.

Sətrin uzunluğunun müəyyənləşdirilməsi – “.length()”

Fərz edək ki, daxil edilən şifrənin minimum 6 simvoldan ibarət olmasını istəyirsiniz. Bu zaman yazılan simvolların sayını “.length()”(lenqs) uzunluq, metodu ilə aşağıdakı kimi ölçmək olar.

```
1 package com.ZJAVAKITABI;
2 class UzunluqString{
3     public static void main(String [] arguments){
4         int minimum = 6;
5         String sifre = "İBAYEV";
6         int olcmek = sifre.length();
7         System.out.println(olcmek == minimum);
8     }
9 }
```

Output - ZJAVA (run) x

```
run:
true
```

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

Yazı daxilində axtarış vermə -“.contains()” və “.indexOf()”

Conatin(konteyn) – 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.

```
1 package com.ZJAVAKITABI;
2 class Containsof{
3     public static void main(String [] arguments){
4         String mətn1 = "Zabil İbayevin 27 yaşı var";
5         String mətn2 = "Zabil İbayevin 28 yaşı var";
6         System.out.println(mətn1.contains("28"));
7         System.out.println(mətn2.contains("28"));
8         System.out.println(mətn2.indexOf("28"));
9         System.out.println(mətn2.indexOf("30"));
}
```

Output - ZJAVA (run) ×

```
run:
false
true
15
-1
```

20

IndexOf(indeks of) 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 rəqem1= 'A';
        char _reqem2= '0';
        System.out.println(reqem1);
        System.out.println(_reqem2);
    }
}
```

Nəticə: A 0

Rəqəm dəyişənləri

Java-da rəqəmlə işləyən dəyişənlər 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-dən böyük həcmdə ədədləri saxlaya bilir. 300 simvola yaxın ədəd. Eyni zamanda float(kəsr) ədədlərini də saxlaya bilir.

Rəqəm dəyişənləri ilə bağlı praktiki məsələlər

Proqramlaşdırma dillərinin ən sadəsindən çətininə qədər hamısı riyazi məsələləri həll etmək üçün tam təmin olunmuşdur. Sadədən çətinə doğru bacardığım qədər Java-nın riyazi məsələlərdə istifadə etdiyi funksiyalarını kitabda əks etdirəcəm.

Riyazi operatorlar

+ toplama - çıxma * vurma / bölmə % qalıq göstəricisi

Dəyişənləri qruplaşdırma və hesablama

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.

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? ☺

```
1 package com.ZJAVAKITABI;
2
3 class Weight {
4     public static void main(String [] arguments) {
5         int Zabilin_kilosu, idman, tetil, indiki_kilo;
6         Zabilin_kilosu= 90;
7         tetil= 15;
8         idman= -10;
9         Zabilin_kilosu = Zabilin_kilosu+tetil+idman;
10        System.out.println(Zabilin_kilosu);
11    }
12 }
```

com.ZJAVAKITABI.Weight > main >

Output - ZJAVA (run) x

run:
95

Yuxarıdakı məsələdə hər dəyişənin qarşısında yenidən `int` yazılmayıb. Məsələdəki dəyişənlərin tam ədəd olmasını nəzərə alınıb və bir sətirdə qruplaşdırılıb.

Qalığın ekrana verilməsi

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

```
1 package com.ZJAVAKITABI;
2 class Riyazimisallar {
3     public static void main(String[] arguments) {
4         int z = 16 % 3 ; //qaliq gosterir
5         System.out.println(z);
6     }
7 }
```

23

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.

```
1 package com.ZJAVAKITABI;
2 class Birbirtoplama {
3     public static void main(String [] arguments){
4         int z, cavab1, a, cavab2;
5         z = 3; a = 3;
6         cavab1 = z++ * 10;cavab2 = ++a * 10;
7         System.out.print(cavab1 + " ");System.out.println(cavab2);
8     }
9 }
```

Output - ZJAVA (run) ×

```
run:
30 40
```

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.

```
1 package com.ZJAVAKITABI;
2
3 class HesabArdicilligi {
4     public static void main(String [] arguments){
5         int z = 5;
6         int cavab = ++z*6+4-1*10/2;
7         System.out.println(cavab);
8     }
9 }
```

Output - ZJAVA (run) ×

```
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 təkə nəticəni yox eyni zamanda toplama prosesini görüntü olaraq əks etdirin. Bunu digər riyazi operatorlarla yoxlayın.

```
1 package com.ZJAVAKITABI;
2 class Riyazimisallar {
3     public static void main(String [] arguments){
4         int eded1= 10; int eded2= 10;
5         int cavab= eded1+eded2;
6         System.out.println(eded1 + "+" + eded2 + "=" + cavab);

```

Output - ZJAVA (run) ×

```
run:
10+10=20

```

Şərt əməlləri

Şərt operatorları – “==”, “!=”, “<”, “>”, “<=”, “>=”, “?”

Qoyulmuş şərtlər təsdiqini tapırsa əmri yerinə yetirilir, əks halda heç bir əmr icra olunmur.

(A == B) – A və B dəyəri bərabərdirsə icra edilir.

(A != B) – A və B dəyəri bərabər deyilsə icra edilir.

(A < B) – A B-dən kiçikdirsə, (A > B) – A B-dən böyükdürsə icra edilir.

(A <= B) – A B-dən kiçikdirsə və ya B-yə bərabərdirsə, (A >= B) – A B-dən böyükdürsə və ya B-yə bərabərdirsə icra edilir.

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

Şərt əməlləri – if, else

If- Əgər mənasına gəlir, şərtin başlandığı əmrdir, **else(els)** - başqa, əgər olmasa başqa şərt mənasına gəlir. İstəyirəm birbaşa məsələ üzərində şərt əməllərini izah edim. Məsələ məktəbi bitirən Bilal haqqındadır ☺

```
1 package com.ZJAVAKITABI;
2
3 class Graduate {
4
5     public static void main(String[] arguments) {
6         int cari_il, dogum_ili, yaşı, TQDK_balı, ixtisas_Mühendis, ixtisas_Memar;
7         String adı_soyadı="Bilal Hacıyev";
8         cari_il=2014; dogum_ili=1995; TQDK_balı=300;
9         ixtisas_Mühendis=600; ixtisas_Memar=500;
10        if (TQDK_balı >= ixtisas_Mühendis){
11            System.out.println("Siz Mühendislik ixtisasına qəbul olmusunuz");
12        }
13        else if (TQDK_balı >= ixtisas_Memar){
14            System.out.println("Siz Memarlıq ixtisasına qəbul olmusunuz");
15        }
16        else if((yaşı=cari_il - dogum_ili) <= 18){
17            System.out.println("Sizin " + yaşı + " yaşınız var, əsgərliyə getmək üçün azdır.");
18        }
19        }
20        else {
21            System.out.println(adı_soyadı + " sən oxumadığın üçün əsgərliyə gedirsən!");
22        }
23    }
24 }
```

Find: fakülta Previous Next

Output - ZJAVA (run) X

run:
Bilal Hacıyev sən oxumadığın üçün əsgərliyə gedirsən!

Kodlardan görüldüyü 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 düzdirsə 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.

27

```
1 package com.ZJAVAKITABI;
2
3 class sualsherti {
4
5     public static void main(String[] arguments) {
6         String Cinsi="kişi";
7         System.out.print((Cinsi.equals("kişi")) ? "Cənab " : "Xanım ");
8         System.out.println("Zabil İbayev");
9     }
10 }
```

Output - ZJAVA (run) ×

```
run:
Cənab Zabil İbayev
```

Ardıcılıq aşağıdakı kimidir:

1. Şərt yazılır
2. “?” işarəsi
3. “Şərt doğrudursa 1-ci cavab dırnaqda String” və ya dırnaqsız integer
4. İkinci qətlə ilə “:” və ya
5. “Şərt doğru deyilsə 2-ci cavab 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(sviç – dəyişmək, keçirtmək)-dən istifadə edilsin. Switch() - in qarşısındakı mötərizədə String, integer, char və digər dəyişənlər yazıla bilər. Case(keys-hal)-in qarşısındakı dəyər switch-in mötərizəsindəki dəyərlə eyni olduğu zaman şərt icra edilir. Əgər eyni deyilsə o zaman növbəti case yoxlanılır. Bütün case-lər icra olunmadığı halda default(difolt) icra olunur. Default-un azərbaycan dilində düzgün tərcüməsi yoxdur və ya tapa bilmədim, amma İT-də qarşılaşdığım anlamı əsasən “standart olaraq”, “sabit seçim” kimi mənaya uyğun gəlir. Break(breyk- qırma) icra olunacaq Case-dən sonra icranı dayandır, qır mənasına gəlir. Hər bir case icra olunan sonra break yazılmasa proqram case-dən sonra yazılan digər mümkün əmrləri yerinə yetirəcəkdir.

```
1 package com.ZJAVAKITABI;
2 class Switchcase {
3     public static void main(String[] arguments) {
4         int TQDK = 0;
5         switch (TQDK) {
6             case 600:
7                 System.out.println("Mühəndislik ixtisasına qəbul oldunuz");
8                 break;
9             case 500:
10                System.out.println("Memarlıq ixtisasına qəbul oldunuz");
11                break;
12             case 300:
13                System.out.println("Aşağı bal topladığınız üçün qəbul olmadınız");
14                break;
15             default:
16                System.out.println("TQDK balınızı daxil edin");
17                break;
18        }
19    }
20 }
```

Output - ZJAVA (run) ×

```
run:
TQDK balınızı daxil edin
```

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 göstərildiyi kimi:

String ad = "Zabil"; ----- case "Zabil" **char** birhərf = ' Z '; ----- case ' Z '

İnt yaş = 27; ----- case 27

Dövr əməlləri - for , do, while

İngilis dilində dövr əməlləri loop(lup) adlanır. Dövr daxilində yazılan şərtləri və hesablamaları yuxarıdan aşağıya işləyərək şərtlərdə verilmiş son həddə çatana kimi proqramı 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 1 ədədi 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.

29

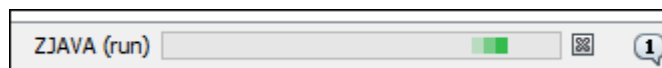
```
1 package com.ZJAVAKITABI;
2 class Forloop {
3     public static void main(String [] arguments){
4         for(int z=1; z<=15; z++) {
5             System.out.println(z);
6         }
7     }
8 }
```

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 daxilində şərtlər “;” ilə ayrılmalıdır. For-dan sonra System.out.println();-dən başqa şərtlərdə istifadə etmək mümkündür.

```
1 package com.ZJAVAKITABI;
2 class Forloop {
3     public static void main(String [] arguments){
4         for(int z=0; z<=15; z++) {
5             if (z % 3==0) System.out.println(z);
6         }
7     }
8 }
```

3-ə bölünərkən qalığı sıfıra bərabər olan ədədlər çap edilib. 0, 3, 6, 9, 12, 15

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



Mürəkkəb For əməlləri

Yuxarıda izah olunan məsələ üzərində for dövrünün quruluşunu daha rahat anlamaq mümkündür. Əslində for-un qarşısındakı mötərizədə dəyişən 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.

```

3 public static void main(String [] arguments){
4     int z=5, a;
5     for( a=0; z*a<=50; a+=1) {
6         System.out.println(z + "*" + a + "=" + (z * a)); }
    }

```

Output - ZJAVA (run) X

```

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(vayl – 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.

```

1 package com.ZJAVAKITABI;
2
3 class Whileloop {
4
5     public static void main(String[] arguments) {
6         int hava= -1;
7         while(hava<0) {System.out.println("Sobanı yandır qış gəldi");break;}
    }

```

Output - ZJAVA (run) X

```

run:
Sobanı yandır qış gəldi

```

Yuxarıda temperaturun 0 dərəcədən aşağı olduğu halda sobanın yandırılması şərti verilmişdir. Şərt doğru olan kimi loop(dövr) işə düşür. Şərt loop-un işlədiyi vaxt dəyişsə False olsa loop ekrana nəticəni çıxartmır. Şərt doğru olduğu halda sonsuz dövr gedir və loop-un bitdiyi sətirdən sonra **break** əlavə olunmaqla sonsuz dövrün qarşısı alınır. Aşağıdakı məsələdə şərtin yanlış olduğu ana qədər dövrün işləməsi göstərilibdir.

```

5 public static void main(String[] arguments) {
6     int konfet = 5;
7     int aclıq = 1;
8     while (aclıq <= konfet) {
9         System.out.print("Konfetin " + aclıq + " - " + "ni yedim. ");
10        aclıq++;
    }

```

Output - ZJAVA (run) X

```

run:
Konfetin 1 - ni yedim. Konfetin 2 - ni yedim. Konfetin 3 - ni yedim. Konfetin 4 - ni yedim. Konfetin 5

```

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

Continue

Yuxarıda yazdığım konfetlə bağlı proqramda yaqın 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 yazını ekrana verməyə kömək edir. Break yazılan sətirdə proqram tamamı ilə dayanır. Continue isə dövrün həmin sətirdən aşağıya getməsinə imkan vermir və dövrü proqramı əvvəlindən aşağı gəlməyə məcbur edir.

```
5 public static void main(String[] arguments) {
6     int konfet = 4;
7     int aclıq = 0;
8     while (aclıq <= konfet) {
9         aclıq++;
10        switch (aclıq){
11            case 3:
12                System.out.print("Konfetin " + aclıq + " - " + "nü yedim. ");
13                continue;
14            case 4:
15                System.out.print("Konfetin " + aclıq + " - " + "nü yedim. ");
16                continue;
17            default: System.out.print("Konfetin " + aclıq + " - " + "ni yedim. ");}
```

Output - ZJAVA (run) x

```
run:
Konfetin 1 - ni yedim. Konfetin 2 - ni yedim. Konfetin 3 - nü yedim. Konfetin 4 - nü yedim. Konfetin 5 - ni yedim.
```

31

Do while

Do(du- yerinə yetirmək) – demək olar ki while kimidir. Şərt True olduğu halda dövr sonsuz davam edir. False olduğu halda dövr dayandırılır. Do while -in while-dan fərqi şərtin false olduğu halda belə əyri mötərizə

```
1 package com.ZJAVAKITABI;
2 class Dowhileloop {
3     public static void main(String [] arguments){
4         do{System.out.println("Zabilin 28 yaşı var");}
5         while(27 > 28);
6     }
```

Output - ZJAVA (run) x

```
run:
Zabilin 28 yaşı var
```

daxilindəki mətn dövr yoxlandığında ən azı bir dəfə işlənir. Bu məsələdə “Zabilin 28 yaşı var” false olmasına baxmayaraq bir dəfə ekrana verilədir.

Dövrün adlandırılması

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

```

5 public static void main(String[] arguments) {
6     int ilin_gunleri = 365;
7     ayin_gunleri :
8     while (ilin_gunleri <= 365) {
9         for (int i = 1; i <= 365; i++){ System.out.print(i+" ");
10        if(i > 30){ break ayin_gunleri;}}}}

```

Output - ZJAVA (run) ×

```

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 “ayın_gunleri” while dövrü üçün seçilmiş şəri addır. Dövr 31-ə çatanda break-in qarşısında yazılan dövr dayandırılır.

Array

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

	login_toplu		pass_toplu
0	1	0	simsim
1	2	1	gizlisifre
2	3	2	555

Yuxarıdakı şəkildə göstəriləyi kimi “login_toplu” və “pass_toplu” adı altında hər birinə aid 3 dəyər qeyd edilmişdir. Burada qeydiyyatdan keçən şəxsləri cədvələ əlavə etmişik. Proqrama daxil olmaq istəyənləri array daxilindəki qeydiyyatı uyğun olaraq yoxlayırıq. Aşağıda bu proses qismən əks edilmişdir.

```

1 package com.ZJAVAKITABI;
2 class Arraytest {
3     public static void main(String[] arguments) {
4         String login = "2";
5         String pass = "gizlisifre";
6         String login_toplu[] = {"1", "2", "3"};
7         String pass_toplu[] = {"simsim", "gizlisifre", "555"};
8         if (login.equals(login_toplu[0]) && pass.equals(pass_toplu[0])) {
9             System.out.println("Salam " + login_toplu[0] + " nömrəli istifadəçi");
10        } else if (login.equals(login_toplu[1]) && pass.equals(pass_toplu[1])) {
11            System.out.println("Salam " + login_toplu[1] + " nömrəli istifadəçi");
12        } else if (login.equals(login_toplu[2]) && pass.equals(pass_toplu[2])) {
13            System.out.println("Salam " + login_toplu[1] + " nömrəli istifadəçi");
14        } } }

```

Output - ZJAVA (run) ×

```

run:
Salam 2 nömrəli istifadəçi

```

Array iki üsulla yaradıla bilər. Yuxarıdakı misalda array-in yaradılmasında əyri mötərizələrdən istifadə edilib.

Birinci üsul - Array yaradarkən onun tipi(int, String, char...) qeyd edilir. Sonra array-in dəyişəninə ad (login_toplu və ya pass_toplu) 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. Yuxarıdakı məsələdə şifrə yoxlama üsulu əslində düzgün həll deyildir. Bu üsulla hər dəyişəni ayrıca kodla yazılmalı və yoxlanmalı olsaq çox çətin və ağır proqram yazmış olarıq. Yuxarıdakı proqram indiyə qədər öyrəndiyimiz materialları təkrar etmə məqsədi daşıyır. İnşallah kitabın sonunda bir layihəni tam olaraq işləyib düzgün yanaşmaları tətbiq edəcəyik.

Qeyd: Array-lər daxilindəki elementləri 0-dan başlayaraq sayır. Birinci dəyər 1-dən başlamır.

İkinci üsulu - Array-in “new” əmrindən istifadə etməklə yaradılır. Birinci array-in tipi, adı, kvadrat mötərizələr, bərabərlikdən sonra “new” sonra array-in tipi, kvadrat mötərizə daxilində array-in sayı və nöqtə vergül yazılır. Birinci sətirin altından array dəyişəninin adı kvadrat mötərizədə elementin yerləşdiyi sıra nömrəsi və bərabərlikdən sonra elementin özü qeyd olunur.

```
1 package com.ZJAVAKITABI;
2 class Arraytest1 {
3     public static void main(String[] arguments) {
4         String beş_dildə_salam [] = new String[5];
5         beş_dildə_salam[0] = "Salam";
6         beş_dildə_salam[1] = "Hello";
7         beş_dildə_salam[2] = "Privet";
8         beş_dildə_salam[3] = "Qamarjoba";
9         beş_dildə_salam[4] = "Ciao";
10        System.out.println("Aşağıda "+beş_dildə_salam.length + " dildə salam verilir:");
11        for(int i=0;i<=4; i++){
12            System.out.print(beş_dildə_salam[i]+ " ");
13        }
14    }
15 }
```

Output - ZJAVA (run) ×

```
run:
Aşağıda 5 dildə salam verilir:
Salam Hello Privet Qamarjoba Ciao BUILD SUCCESSFUL (total time: 0 seconds)
```

Array-in sayı əvvəldə göstərilməlidir. 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.

```
1 package com.ZJAVAKITABI;
2 class Arraytest2 {
3     public static void main(String[] arguments) {
4         String deyersiz_array [] = new String[5];
5         for(int i=0;i<=4; i++){
6             System.out.print(deyersiz_array[i]+ " ");
7         }
8     }
9 }
```

Output - ZJAVA (run) ×

```
run:
null null null null null BUILD SUCCESSFUL (total time: 0 seconds)
```

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

34

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

```
4 public static void main(String[] arguments) {
5     char multidimension[][] = new char[7][7];
6     multidimension[0][2] = 'Z';
7     multidimension[1][3] = 'a';
8     multidimension[2][4] = 'b';
9     multidimension[3][5] = 'i';
10    multidimension[4][6] = 'l';
11    System.out.println("X array-inin sayı "+multidimension.length);
12    System.out.println("Y array-inin sayı "+ multidimension.length);
13    System.out.print("Kordinatlara görə Char simvolları: ");
14    for(int i=0;i<=4; i++){ int j=i;
15    System.out.print( multidimension[i][j+2]); }}
```

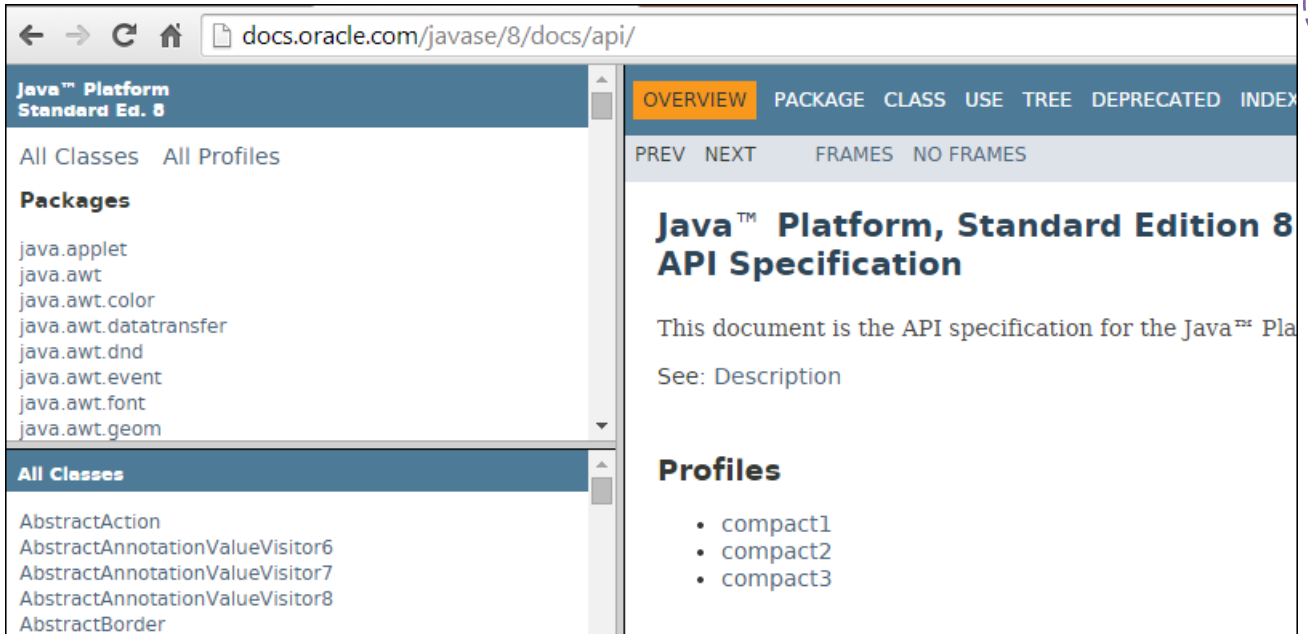
Output - ZJAVA (run) x

```
run:
X array-inin sayı 7
Y array-inin sayı 7
Kordinatlara görə Char simvolları: ZabilBUILD SUCCESSFUL (total time: 0 seconds)
```

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 etmişəm.

Built-in Java Packages

Java-da proqramlaşdırmaq üçün endirdiyimiz JAVA JDK(Java Development Kit) tərkibində Java Runtime Enviroment(JRE) - Java-nın işləməsi üçün mühiti daşıyır. JRE-yə elə JVM(Java Virtual Machine)-də demək olar. Bütünlükdə Java-nın işləməsini təşkil edən parçalardır. Oracle Java-ya bir neçə ildən bir yeni əlavələr edir. Bu əlavələrdən olan JRE-nin tərkibindəki Built-in Java Packages (daxilində Java paketləri quraşdırılmış) hazır proqramlar təklif edir. Package elə bizim yaratdığımız “com.ZJAVAKITABI” paketi ilə eyni funksiyanı daşıyır. Oracle bəzi ümumiləşmiş məsələləri yenidən yazmamaq üçün istifadəçilərə hazır yazılmış class-lar yəni hazır yazılmış proqramlar təklif edir. Hər yenilənmədə yaradılmış paketlər Oracle-ın müəyyən səhifələrində yer alır. Aşağıdakı linkdə bizim endirdiyimiz SE8(Standart Edition –Standard buraxılış) yer alır.



The screenshot shows the Oracle Java Platform Standard Edition 8 API Specification page. The browser address bar displays docs.oracle.com/javase/8/docs/api/. The page has a navigation menu with 'OVERVIEW', 'PACKAGE', 'CLASS', 'USE', 'TREE', 'DEPRECATED', and 'INDEX'. The main content area displays 'Java™ Platform, Standard Edition 8 API Specification' and 'This document is the API specification for the Java™ Platform, Standard Edition 8'. Below this, there is a 'Profiles' section with a list of profiles: compact1, compact2, and compact3. The left sidebar shows a list of packages under 'Packages' and 'All Classes'.

<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. Bu OOP-dən yadınızdadırsa Encapsulasiya-nı xatırladır. Proqramın funksiyası var amma özü görsənmir. Hazır class-ların 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.

```

1 package com.ZJAVAKITABI;
2 import java.util.Date;
3 class ImportSaat {
4     public static void main(String[] arguments) {
5         Date tarixdeyiseni = new Date();
6         System.out.println(tarixdeyiseni); }

```

Output - ZJAVA (run) ×

```

run:
Mon Mar 02 18:17:56 CET 2015

```

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(akses kontrol – girişin idarə olunması)

Access control yazdığımız class-ların, onların tərkibindəki method-ların və ümumilikdə yaratdığımız obyektlərin çağırılması və ya onlara müdaxilə olunması qaydalarını tənzim edir. Aşağıdakı cədvəldə public, private(prayvet-xüsusi) və protected(protekt-d-qorunan) access control əmrləri 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

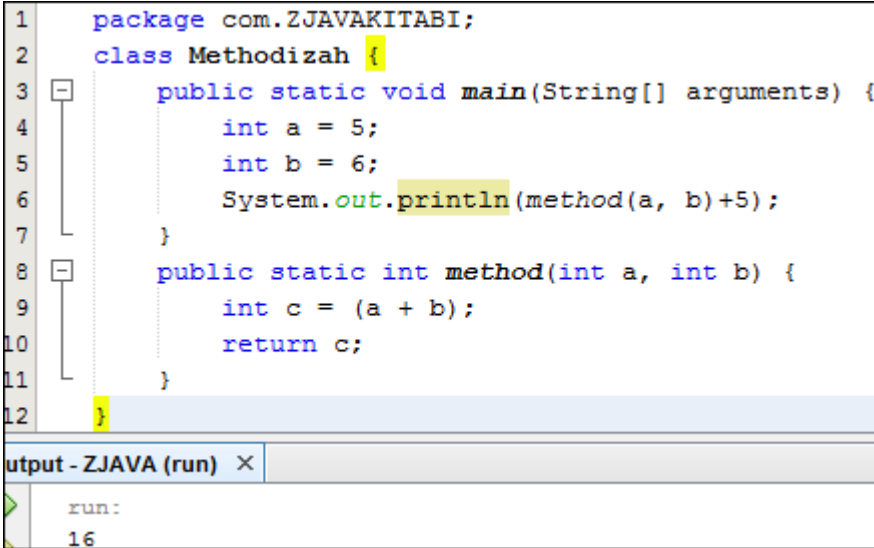
Y=yes(bəli) N=no(xeyr)

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 proqramı 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(modifayə) bəndində hər hansı bir əmr bildirilməzsə o zaman package-in ümumi access control dəyəri avtomatik mənimsənilir.

Method nədir?

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

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



```
1 package com.ZJAVAKITABI;
2 class Methodizah {
3     public static void main(String[] arguments) {
4         int a = 5;
5         int b = 6;
6         System.out.println(method(a, b)+5);
7     }
8     public static int method(int a, int b) {
9         int c = (a + b);
10        return c;
11    }
12 }
```

Output - ZJAVA (run) x

```
run:
16
```

Bildirdiyimiz kimi method-un hesablama hissəsi main-dən ayrı yazılır. İkinci “*public static int method*” yazılan hissədən başlayaraq izah edək.

- Method-da class kimi access control əmrlərini tələb edir. Method-un **public** olması onun bütün proqramlar tərəfindən çağırıla bildiyini göstərir.
- **Static** yazılması instansının olmaması deməkdir.
- Burada “**int**” return type-dır. Mötərizə daxilindəki dəyərlər integer tiplidir və return type-da istifadə olunacaqdır. Return özü proqramın sonunda yazılmasına baxmayaraq qaytaracağı nəticənin tipini əvvəlcədən qeyd etməyi tələb edir.
- Sonra method-a ad verilir.
- Ad veriləndən sonra mötərizə açılır parametrlərin tipi və adı yazılıb mötərizə bağlanır. Mötərizə daxilində main body-də olan və method-da istifadə olunacaq parametrlər yazılır. Yuxarıdakı misalda a və b parametrləri götürülüb və onların tipi integer-dir. Əgər nəticə qaytarılmayacaqsə tipin yerinə void yazılır.

Aşağıda return type-ı void ilə dəyişdirib fərqi göstərmişəm. Void yazıldığı zaman misalın tamamilən dəyişməsi lazımdır. Dəyərlər hesablanıb geri qaytarılmayacağı üçün method-un funksiyasıda anlaşılmır və aşağısından qırmızı xətt çəkilərək səhv qeyd edilib.

```
1 package com.ZJAVAKITABI;
2 class Methodizah {
3     public static void main(String[] arguments) {
4         int a = 5;
5         int b = 6;
6         System.out.println(method(a, b)+5);
7     }
8     public static void method(int a, int b) {
9         int c = (a + b);
10        return c;
11    }
12 }
```

Output - ZJAVA (run) ×

```
run:
16
```

Void-dan aşağıdakı kimi düzgün istifadə etmək olar.

```
1 package com.ZJAVAKITABI;
2 class Methodizah2 {
3     public static void main(String[] arguments) {
4         method();
5     }
6     public static void method() {
7         int a=5;
8         int b=6;
9         int c = (a + b);
10        System.out.println(c);
11    }
12 }
```

Output - ZJAVA (run) ×

```
run:
11
```

Void-lə method yaratdığımız zaman method istifadə olunan yerdə öz dəyərini əks etdirir amma onu hər hansı bir hesablamada istifadə etmək olmur çünki dəyərinin hansı tip olması bəlli deyildir.