Z - JAVA

Zabil Ibayev

2014

Mündəricat

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 yetərincə 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 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 isə ingilis dilini bilmək vacibdir.

Kompüter şəbəkələrini qurmağı öyrəndiyim zamanlarda Zİ-N Kompüter şəbəkələri və Çatışmayan biliklər kimi kitabları yazmışam. Mənim tərəfimdən yazılan kitablarda müəllif göstərilməklə çap və satışdan gəlir əldə edilə bilər.

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 naqillərinə elektrik axımı verilsin ya yox.

Kompüterimizdəki işləri görən mikrosxem, xalq deyimi ilə kompüterin beyni, CPU(Central Processing Unit) mərkəzi hesablayıcı qurğu anlamına gəlir. CPU-ya əmr verilməsi üçün istehsalçılar İnstruction Set (instruksiyalar dəsti) adlı qaydalar toplusu düzəldiblər. Bütün yazılan proqramlar prosessora 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 OS daxilindəki adi Texteditor-la və ya MSDOS-da yazıla bilər. Hər proqramlaşdırma dili fərqli üslubda müəyyən 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 birbaşa “CMD” yazıb proqramı aça bilərsiniz.

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

Proqramlaşdırma dilləri - necə fərqlənir?

Yuxarıda bildirildiyi kimi kompüter ikili say sistemindəki əmrləri anlaya bilir. Belə olan halda biz insanlar üçün bu şəkildə proqram yazmaq çox çətindir. İkili say sistemi ingilis dilində *Binary* 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 adlanır. Aşağıdakı şəkildə "Alma" sözünün 2-li və 16-lıq say sistemindəki yazılışı göstərilib.

16-lıq say sistemində proqramlaşdırmaya misal olaraq *Assembler*–i göstərmək olar. Proqramlaşdırmada növbəti mərhələ kimi vizual görünüşü daha rahat olan *BASİC* dilini misal çəkmək olar. *BASİC*-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.

BASİC-də proqrama verilən əmrlər sizin tərəfinizdən təyin olunmuş rəqəmlərin ardıcıllığına uyğun olaraq işlənir. Yazılmış hər sətir ayrıcalıqda oxunur, sonra tərcümə və icra edilir. Bu üsulla işləyən proqrama "*interpreter*" - tərcüməçi proqram deyilir. Yəni hər yazılan əmri kompüterin başa düşəcəyi dilə çevirir. Bu cür yanaşma proqramın aşağı sürətdə işləməsinə səbəb olur. Əgər *BASİC*-də əmrlərin qabağında rəqəm yazmasan birbaşa əmrin icrasına keçir, bu səbəbdən ilk əmr *PRİNT*-in (çap etmək, ekrana çıxartmaq) qarşısına rəqəm qoymuşam ki, 1-ci "Salam Zabil" sözünü ekrana çıxartsın sonra 2-ci əmri *İNPUT*-u (daxil etmək) ekrana çıxartsın ki, məndə ona qarşılıq verim. *RUN*- 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.

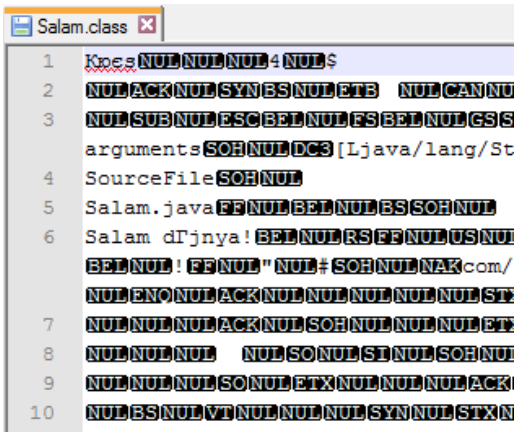
Java- nə ilə fərqlənir?

Java OOP texnologiyasından istifadə edir. Object Oriented Programming (OOP)-obyekt yönümlü proqramlaşdırma. Adından da müəyyən dərəcədə anlamaq olar, obyekt yönümlü yəni 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 obyekti bir neçə yerdə istifadə etməyə imkan verir. Bir az mürəkkəb dillə tərif versək, OOP yazdığınız proqramın içindəki müəyyən kodları ayrı-ayrılıqda həmin və ya digər proqram daxilində obyektin xüsusiyyətlərini yenidən yaratmadan istifadə etmə imkanı yaradır.

Məsələn yaratdığınız avtomobil obyektini (eyni zamanda "avtomobil proqramını" əgər belə adlandırsaq) digər proqramın 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.

Java OOP "Compile"(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 BASİC* 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 haradan başlayıb, harada bitəcəyini, iş prosesində hansı əmrlərin yerinə yetirəcəyini anlayır. Yazacağımız "Salam dünya" proqramının bytecode-a çevirilmiş şəkli aşağıdakı kimidir.



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-nun versiyasından asılı olmayaraq bütün kompüterlərdə işləyə bilir.

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 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

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

Avtomobil haqda bizə nə məlumdur:

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

bu xüsusiyyətlər ATRİBUTE-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.

Sinifləndirmək - *CLASS* nədir?

Təhlil üçün avtomobil obyektini götürək. Avtomobilin ban növündən, firmasından və ya ölçüsündən asılı olmayaraq obyektdəki *attribute*-lar və *method*-lar eynidir. Bu da onları eyni sinfinə daxil etməyə imkan verir. El dilində belə izah edim, yolda uzaqdan bir minik vasitəsini 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.

İnstance(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 attribute-ları olmağına baxmayaraq ümumi göstəriciləri class-la eyni olan obyektlərdir. Obyektin yaradılması class-ın instance-nın yaradılmasıdır.

El dilində belə deyək adından və ya sahibindən asılı olmayaraq notbuk deyəndə nə ağlına 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-nı fikirləşir(rəngini, ağırlığını və s.)

Encapsulasiya

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

Abstraction

Vacib detalları göstərməklə lazımsızları kənarlaşdırmaq funksiyasını icra edir. Misal olaraq avtomobili işə salan sahibi açarı döndürdükdə avtomobilin işə düşəcəyindən başqa obyektdə gedən

proseslərin necə icra olunduğunu görmür. O sadəcə açarı döndürüb avtomobilin işləməsini gözləyir, digər proseslər ondan uzaq tutulur və ya gizlədilir.

İnheritance(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 proqrama başqa proqramın xüsusiyyətlərini istifadə etməyə imkan yaradır.

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(əlaqə)

Obyektlər arasındakı hər hansı bir əlaqəyə association deyilir. Avtomobil obyekti və yol obyekti 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 həmdə funksiya göstərmək üçün əlaqə saxlayır.

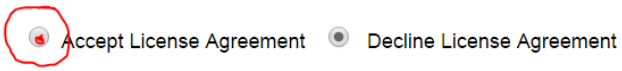
Composition(kompozisiya)

Daxilindəki obyekt və ya obyektlərlə birgə funksiya göstərmək üçün əlaqə saxlayır. Buna misal olaraq bina obyekti 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 ünvandan kompüterinizin 32/64 bit olmasına baxaraq sizə uyğun versiyanı paket şəklində endirə bilərsiniz.

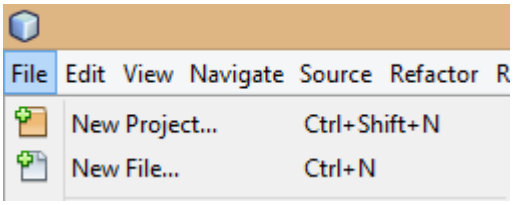


Java SE and NetBeans Cobundle (JDK 8u25 and NB 8.0.1)		
Product / File Description	**File Size**	**Download**
Linux x86	247.08 MB	jdk-8u25-nb-8_0_1-linux-i586.sh
Linux x64	243.56 MB	jdk-8u25-nb-8_0_1-linux-x64.sh
Mac OS X x64	381.18 MB	jdk-8u25-nb-8_0_1-macosx-x64.dmg
Windows x86	261.52 MB	jdk-8u25-nb-8_0_1-windows-i586.exe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

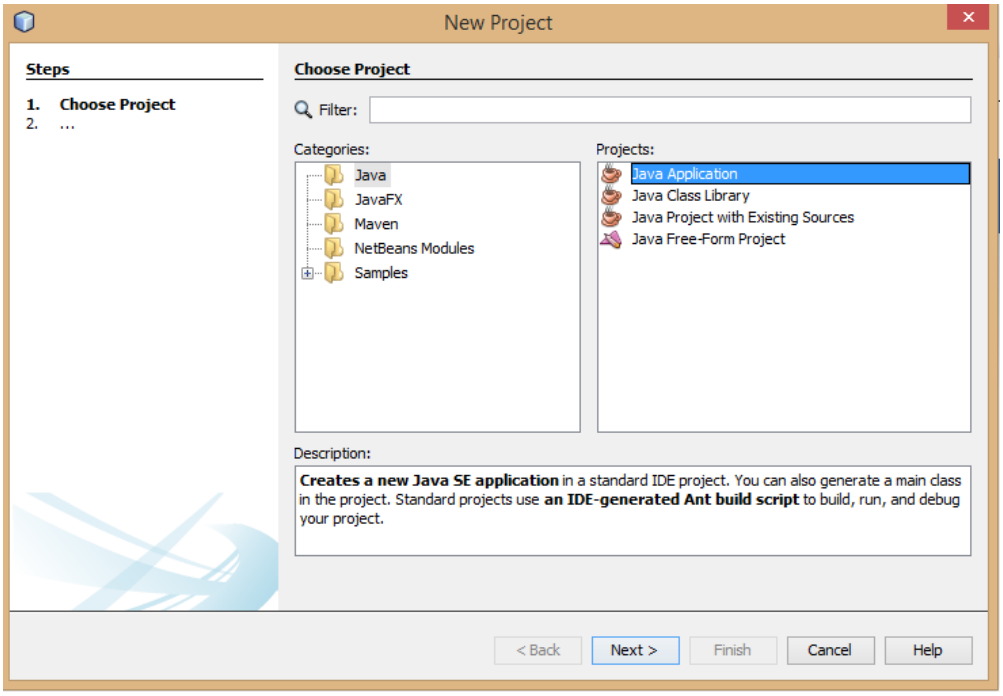
Proqramı endirdikdən sonra üzərinə sağ düymə sıxaraq Run as administrator sətrini seçin. Kompüterin administratoru kimi proqramları yükləyəndə əsasən yükləmə problemsiz başa çatır. Daha sonra şərtləri qəbul edərək Accept(qəbul etmək) yükləməyə davam edin. Hər endirilən versiyada yükləmə fərqli ola bilər.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

Netbeans-lə tanışlıq.

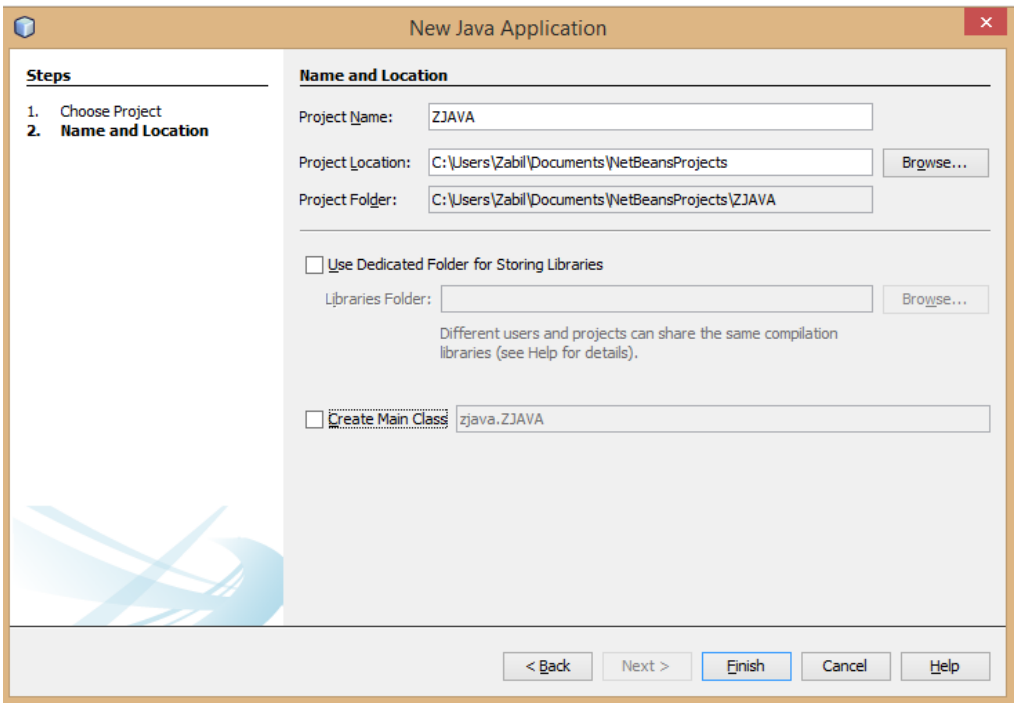
Netbeans - də yeni proqram yazmaq istədiyinizdə sizə proqramın təklif etdiyi ilk olaraq yeni layihə yaratmaqdır.



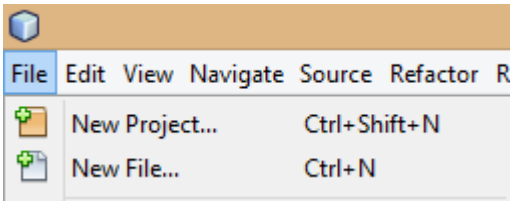
Yazacağımız proqramlar yaratdığımız layihənin tərkibində yer alacaqdır. Açılan pəncərədə bizdən nə tipli layihə yaratmaq istədiyimiz soruşulur. Biz şəkildə göstərildiyi kimi kateqoriyalardan(Categories) Java və layihələrdən(Projects) Java Application(Java Proqramı) seçirik. Sonra Next (növbəti) düyməsini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



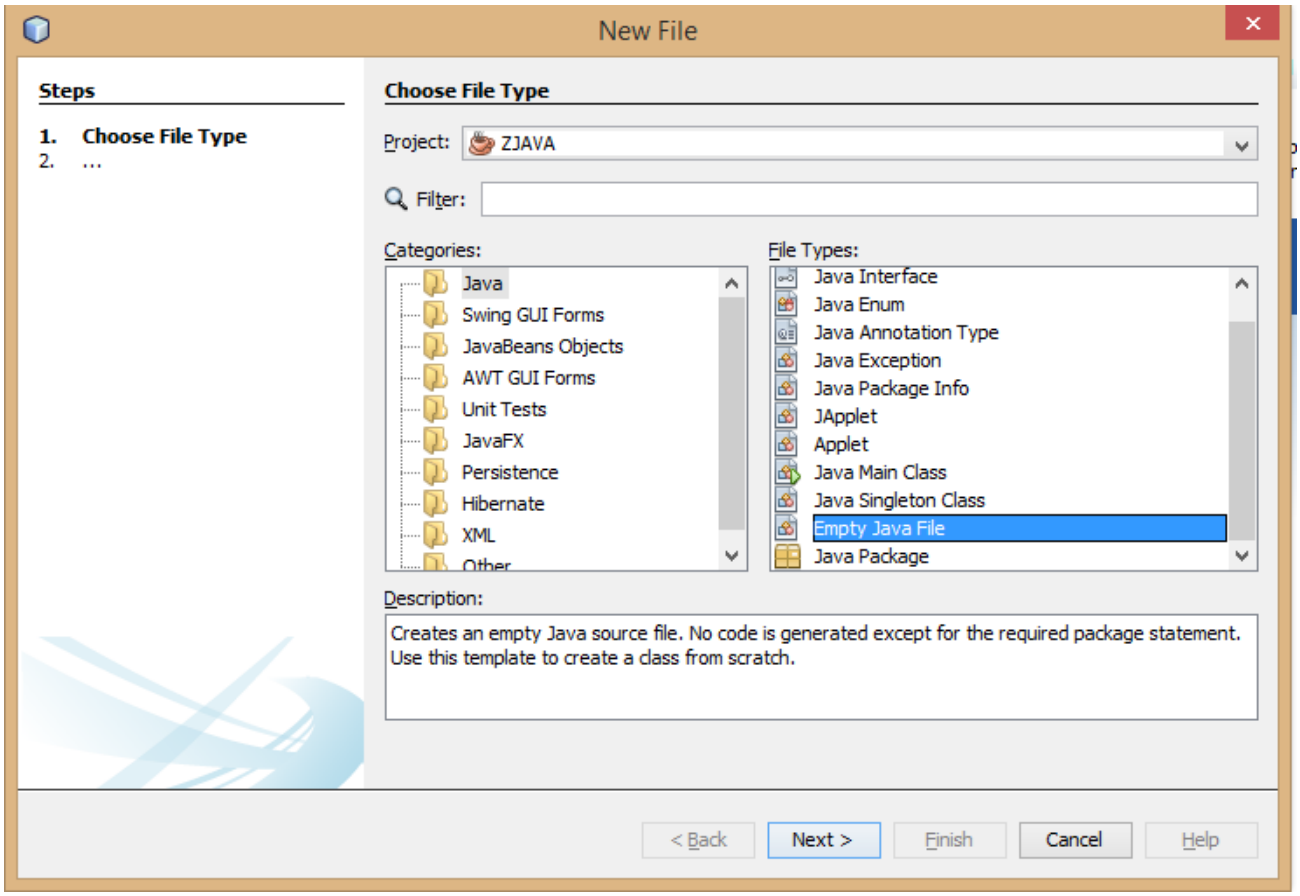
Layihəyə ZJAVA adını verirəm. Aşağıda Create Main Class qutusunu söndürün, məndə həmin hissənin qarşısında "zjava.ZJAVA" yazısı var. Sonda Finish düyməsini sıxın.



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



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

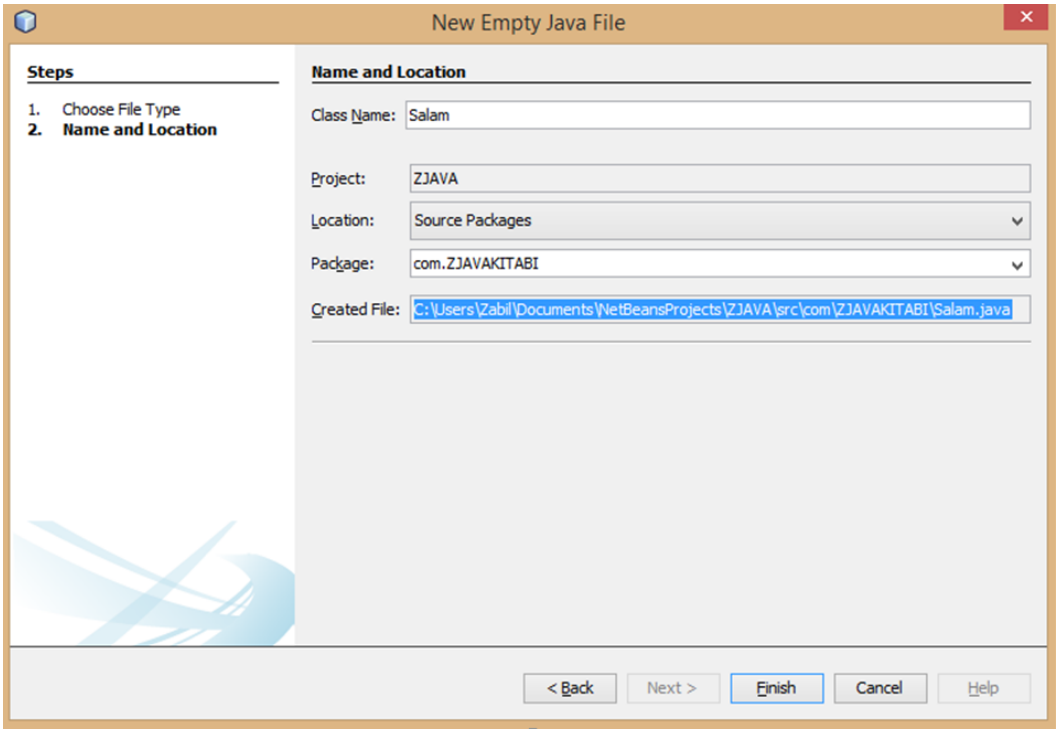


Növbəti səhifədə artıq müəyyən mənada proqram yazılışına başlayırıq.

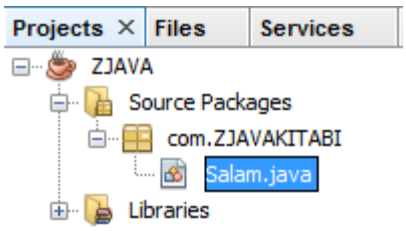
Salam Java proqramı

Class Name(Sinfin adı) bura Salam proqramının adını yazmışam. Kompüterə əmr verirəm ki mənim proqramımın adı Salam olsun.

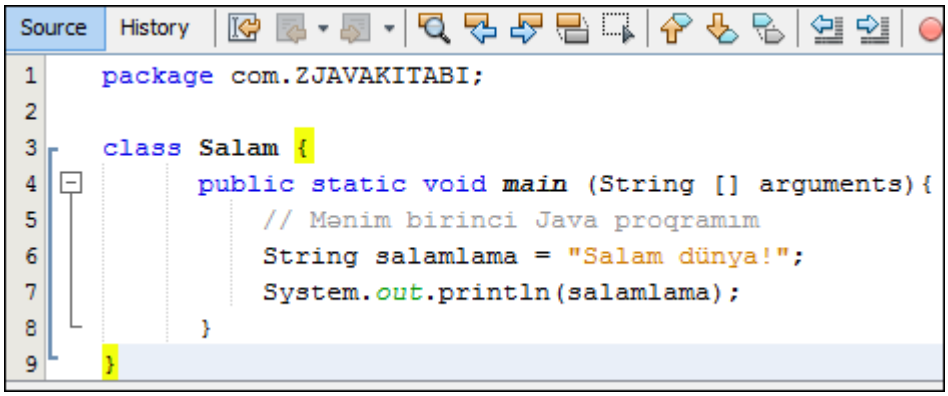
Package(Paket) adı com.ZJAVAKITABI. Paket java proqramlarını bir arada qruplaşdırmaq üçündür. **Created File**(Yaradılmış) hissəsində yaratdığınız faylın harada saxlanmasını təyin etdiyiniz göstərilir.



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



Salam.java proqramını seçdikdən sonra aşağıda göstərildiyi kimi sətirləri daxil etməyə çalışın.



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

Package

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

public static void main (String [] arguments)

Bu sətri sadəcə əzbərləyin. Java-da yazacağınız proqramlar əsasən belə başlayacaq. OOP tipli proqram yazdığımıza 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(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) yaratdığınız 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

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 dəfəki 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" sətri 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(çöl), println(print line(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.

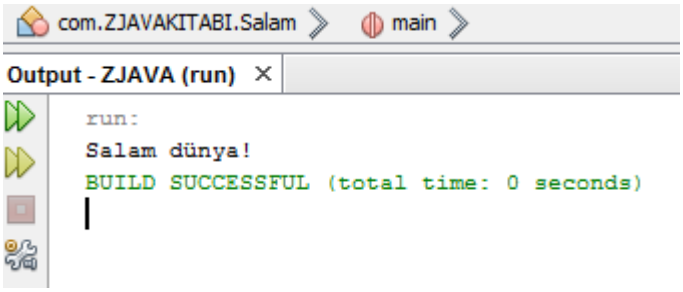
Qeyd: public static void main (String [] arguments) -bu yazını sadəcə “psvm” yazıb sonra TAB düyməsinə basaraq sürətli şəkildə daxil edə bilərsiniz.

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

" ; " nöqtə vergül

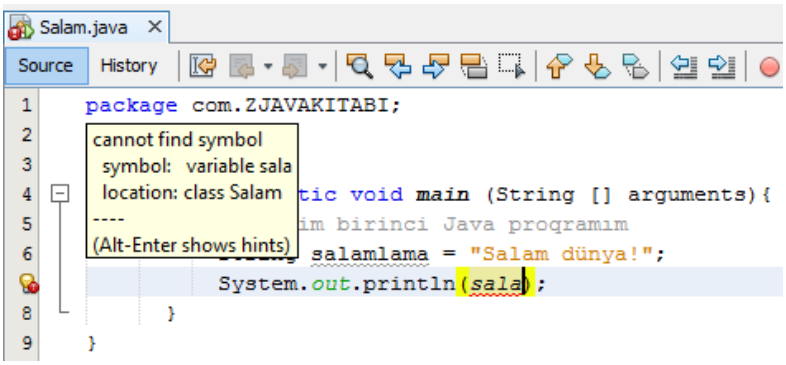
Sətirlərin sonunu göstərmək üçün proqramlaşdırmada əsasən nöqtə vergüldən istifadə edilir. Əmrin bitməsini göstərir.

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 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. Nöqtə vergül və ya adların eyni yazılması kimi səhvlər tez-tez təkrarlanır.



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.



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 ardıcıllığını özünüz əvvəldəki izahlardan təkrar etməlisiniz.

Qeyd: Növbəti proqramları əgər eyni pakage-in tərkibində yaratsanız RUN etmənizdə problem yarana bilər. Yazılan proqramı **Save** etdikdən sonra proqramın üzərinə sağ düyməni sıxıb **RUN File** sətrini 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 pakage, 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 şərtdir. 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 geriyə, sola pozma), \r - Carrige 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.

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 sətrə 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" deyişəninin 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 bilirsiz.

Bütün hərflərin böyük və ya kiçik yazılması – ".toUpperCase" və ".toLowerCase"

Proqramınıza 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
```

İki String-i müqayisə etmək "==" və ".equals()"

Fərz edək ki, qeydiyyat səhifəsi yaratmışıq. Səhifədə şifrə daxil edilir və əmin olmaq üçün ki istifadəçi 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. Bu zaman ".equal()"metodundan və ya "==" qoşa bərabərlik operatoundan 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:
false
false
```

Nəticədə gördüyünüz kimi cavab "False" yalnışdır. 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.

Sətrin uzunluğunun müəyyənləşdirilməsi – “.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.

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

Conatins – 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
```

İndexof - 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 İnteger, 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 bilir. 300 simvola yaxın ədəd. Eyni zamanda float(kəsr) ədədlərini də saxlaya bilir.

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

Byte - 128 dən 127 kimi tam ədədləri yadda saxlaya bilir.

Short - 32767 dən -32768 kimi tam ədədləri yadda saxlaya bilir.

Long - 9.22 quintliondan 9.22 quintiliona kimi tam ədədləri yadda saxlaya bilir.

Hesablamalar ilə bağlı praktiki məsələlər

Riyazi operatorlar

+ toplama
- çıxma
* vurma
/ bölmə
% qalıq göstəricisi

Dəyişənləri qruplaşdırma və hesablama ilə bağlı məsələ

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 deyişəni proqram boyunca ancaq 10-a bərabər olacaq, dəyəri dəyişməyəcəkdir.

Qalığın 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 yazılarsa 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 ardıcıllığı

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.

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

Praktiki məsələ: 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 mənaları əks etdirirlər.

(&&) - və
(==) - eyni dəyərdədir
(||) - və ya
(!) - inkar, deyil

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

Şərt əmrləri

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

Qoyulmuş şərtlər təsdiqini tapırsa əmr 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 təsdiqlənirsə icra edilir, "**?**" xüsusi şərtdir, əgər birinci şərt təsdiqləmirsə ikinci nəticəni icra edir.

Şərt əmrləri – if, else

If- Əgər mənasına gəlir, şərtin başlandığı əmrdir, **else**- başqa, digər halda mənasına gəlir. İstəyirəm birbaşa məsələ üzərində şərt əmrlərini 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 Haciyev";
        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 Haciyev 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 təsdiqlənirsə 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
```

Ardıcıllıq aşağıdakı kimidir:

1. Şərt yazılır
2. "?" işarəsi
3. "Şərt doğrudursa 1-ci cavab qoşa dırnaqda String" və ya dırnaqsız integer
4. Iki 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 olunandan sonra break yazılmasa proqram case-dən sonra yazılan digər mümkün əmrləri 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("Memarliq 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:
Memarliq 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 göstərildiyi kimi:

String ad = "Zabil"; ------ case "Zabil"

char birhərf = ' Z '; ------ case ' Z '

İnt yaş = 27; ------ case 27

Dövr əmrləri - for , do, while

İngilis dilində dövr əmrləri loop adlanır. Dövr, daxilində yazılan şərtləri və hesablamanı işləyərək şərtlərdə verilmiş son həddə çatana kimi hesablamanı təkrar edir.

For

Əsasən ədədləri ardıcıl və ya müəyyən şərtə uyğun olaraq saymaq kimi əmrləri 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 və15-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
```

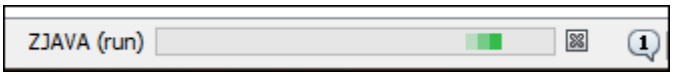
Yuxarıdakı izaha uyğun olaraq ədədlər sıralananda 2-dən başlamalıdır amma nəticədə 1-dən 15-ə qədər sıralanmışdır. For-un bu xüsusiyyətini gələcəkdəki hesablamalarınızda nəzərə almaq yaxşı olar. For daxilindəki şərtlər ";" ilə ayrılmalıdır. Aşağıdakı məsələdə bu for dövründən sonra if şərt əmrinin necə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

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 proqram 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 qarşısındaki 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 yoxlanır. 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 dayandırlması üçün break əlavə olunmalıdır. Şərt dəyişərsə yenidən loop aktiv gözləmədə ola bilər ta 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. ☺

Continiue

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 azi 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 müəyən 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 “ayın_günleri” 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

Array-lər daxilindəki elementləri 0-dan başlayaraq sayır. Birinci dəyər 1-dən başlamır. Java-da iki 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 göstərildiyi 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"};


   if (login_e.equals(login[0]) && pass_e.equals(pass[0])) {
   System.out.println("Salam " + login[0] + " nömrəli istifadəçi.");
   } else if (login_e.equals(login[1]) && pass_e.equals(pass[1])) {
   System.out.println("Salam " + login[1] + " nömrəli istifadəçi.");
   } else if (login_e.equals(login[2]) && pass_e.equals(pass[2])) {
   System.out.println("Salam " + login[2] + " nömrəli istifadəçi."); }}}
```

```
run:
Salam 2 nömrəli istifadəçi.
```

Qeyd: Yuxarıdakı məsələdə istifadəçi adı şifrələrinin yoxlanması 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.

İkinci üsulu - Array-in "new" əmrindən istifadə etməklə yaradılır. İlk olaraq array-in tipi, adı, kvadrat mötərizələr, bərabərlikdən sonra "new" sonra array-in tipi, kvadrat mötərizə daxilində array-in sayı və nöqtə vergül yazılır. Sonra altından array dəyişəninin adı kvadrat mötərizədə elementin yerləşdiyi sıra nömrəsi və bərabərlikdən sonra elementin özü qeyd olunur. Aşağıdakı məsələdə array-dən istifadə etməklə 5 dildə salamlama həyata keçirəcik.

```
package com.ZJAVAKITABI;
class Array2 {
    public static void main(String[] args) {

  String beş_dildə_salam[] = new String[6];
        beş_dildə_salam[0] = "     ";
        beş_dildə_salam[1] = "Salam";
        beş_dildə_salam[2] = "Hello";
        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 dövrdən istifadə edərək koordinatların avtomatik əldə olunmasını və elementlərin yerləşdiyi koordinatdan çağırılıb ekrana verilməsini təşkil edilib.

Obyektin yaradılması

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

```
package com.ZJAVAKITABI;
class Object {
public static void main(String[] args) {
 Ibayev Zabil = new Ibayev();
System.out.println("Zabilin saçı " + Zabil.saçı + ". Göz rəngi " + Zabil.göz);}}

class Ibayev {
    String saçı = "Qara";
    String göz = "Qəhvəyi";}
```

```
run:
Zabilin saçı Qara. Göz rəngi Qəhvəyi
```

Zabil obyektini yazdıqdan sonra CTRL+Space düyməsinə basdıqda siz JVM(Java Virtual Machine)-dən artıq müəyyən olunmuş dəyərləri çağıra bilərsiniz. Siz obyekti yaratdığınız zaman ona İbayev class-ından məlumatları götürə bilməsini mümkün edirsiniz.

Qeyd: Obyektin yaradılması class-ın instans-laşdırılmasıdır.

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.

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 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 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 proqramı istifadə olunmasını təmin edir. Digər pakage-lər onu görmür. Protected bir o qədər də istifadə olunmayan əmrdir və demək olar ki, public onu hər yerdə əvəz edə bilir. No modifier(təyin edici olmadıqda) bəndində hər hansı bir əmr bildirilməzsə o zaman pakage-in ümumi access control dəyəri avtomatik mənimsədilir.

Method nədir?

Method bir növ dəyişənə bənzəyir. Hər hansı bir hesablama, ona verilmiş ada mənimsədilir. Aşağıdakı misalda **method();** -un yazılış şəkli göstərilmişdir. Dəyişəndən fərqli olaraq method-un daxilindəki dəyər ondan aşağıda **main**-dən yəni əsas proqramdan ayrı amma eyni class-ın daxilində yazılır. Method yaratmaq üçün bizə 4 parametr 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"* bölәsini 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 əmrlərini 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 bildiriləndən sonra method-un tipi bildirilməlidir. Return type nədirsə o yazılır. Method1() tərkibində integer hesablaması olacağı üçün "int method1" yazılır. Daha sonra main-də elan olunan hər hansı bir dəyişəndən istifadə olacaqsa 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 parametrli Method 1 işləyir və nəticə ");
        return q;    }
```

- İstənilən hesablama və görüntü hazılandıqdan 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 parametrli 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 parametrli Method 1 işləyir və nəticə ");
        return q;    }

    public static int method1(int a, int b) {
        int mf = (a + b);
        System.out.print("2 parametrli 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 parametrli 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 parametrli 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 obyekti və ya obyektin instance-ını yaratmaq üçün təyin olunmuş xüsusi bir method-dur. Belə ki, konstruktor yaradan zaman iki class yaradılır. Rahat anlaşılması 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 obyekti var. Həmin obyektin atributları İbayev class-ından götürülmüşdür. "Ibayev Zabil = new Ibayev();" əmri ilə biz İbayev class-nın tipinə uyğun olaraq "new"(yeni) Zabil obyekti yaradır və atribut dəyərlərinin Ibayev class-ından götürməsini 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 obyekti yaradılsın. İbayev obyektindəki 3 parametr (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 Ibayev və Kerimov obyektlərinin Const daxilində metod çağırdığını 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 hesablamanı 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 daxildəki dəyişənlərə mənimsədirik, yəni ötürürük. Daha sonra return type void olmasını yazıb metodumuzu yaradırıq. Const daxilində daha mürəkkəb hesablamalar aparmaq mümkündür. Yuxarıda göstərilən misal prosesin ümumi təsvirini yaradır.

Qeyd: Proqram daxilində konstruktorlar method-lardan əvvəl çağırılır.

Inheritance-ın yaradılması

Bildiyimiz kimi inheritance, bir proqrama başqa proqramın xüsusiyyətlərini istifadə etməyə imkan yaradır. İnheritance aşağıdakı misalda "extend"(uzatmaq, böyütmək, artırmaq) əmri ilə irsilik ötürür. Aşağıdakı misalda A, B, C class-ları hazırlanmış, daha sonra C-yə uyğun olaraq "obyekt" yaradılmış, "obyekt" C-yə aid olmasına baxmayaraq B-nin dəyərlərini çağıra bilir.

```
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
```

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

"Super" və "This" açar sözləri

İnheritance-dan bizə müəyyən olduğu kimi, class-lar tabeliyində olduğu class-ın istənilən dəyərini çağıra bilir. 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 çağıra bilərsiniz. "Super" üstün class-ı göstərir. This(bu) açar sözü olduğu yeri, hazırki 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 class-dır 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()” method-u “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. Mənimsədiyi dəyərlərin həmişə sabit qalması kimi başa düşülür amma bu belə deyil. Statik dəyişənlər proses içində müxtəlif dəyərlərə sahib ola bilir, amma çağırıldığı obyektlərə nisbətdə dəyərləri sabitdir. Statik dəyişənlərin digər adı “class” dəyişənləridir. Bildiyimiz kimi proqram işə başladığında konstruktorlar method-lardan əvvəl çağırılır. Statik dəyişənlər proqramda compile olunanda JVM-dəki *class loader* tərəfindən müəyyən edilir və yaddaşda saxlanılır. Bundan əlavə static dəyişənlər əmrlər içində konstruktordan da əvvəl çağırılır. Aşağıdakı misalda sadə static dəyişən göstərilmişdir. Əgər static dəyişənlərə dəyər mənimsətməsək 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 mənimsətmək 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 ardıcıllığa görə yüklənirlər. Statik bloklar aşağıdakı kimi tərtib edilir. Bu misalda konstruktordan istifadə etmişəm. Konstruktorun 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 ardıcıllıqda qabaqlayır. Bundan əlavə konstruktor obyekti sadəcə main class-da yaradılmaqla nəticəni ekrana verir. Buna səbəb sonda yazılan "vahid()"-dir. Əgər mötərizə daxilində parametr verilsə həmin parametrə uyğun nəticə ekrana veriləcəkdir.

Casting

Hərhansı 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.

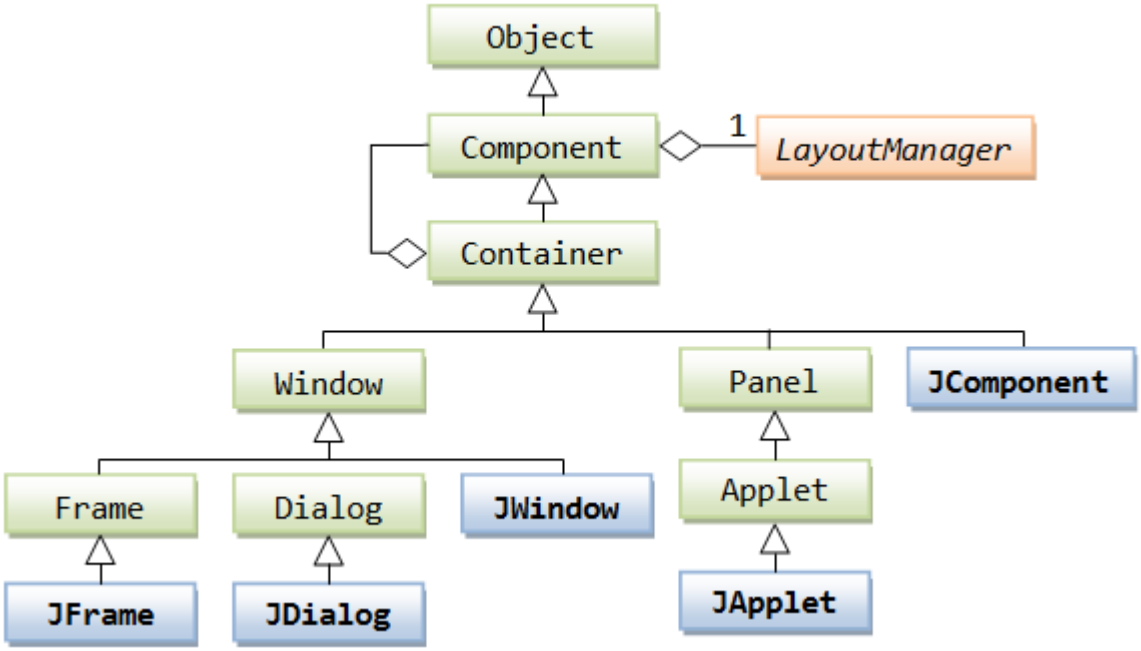
Qeyd: İnheritance əlaqəsi olan obyektlər arasında casting etmək mümkündür.

Swing GUI- Qrafiki Istifadəçi 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.

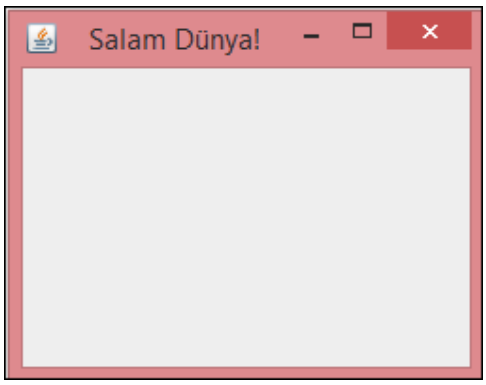
Swing-in strukturu

Interfeysin görünməsi üçün obyekt yaradılmalıdır. Aşağıdakı şəkildə Swing-in strukturu iyerarxik düzülüşdə təsvir olunmuşdur. Strukturun hər bir parçası bir araya gələrək obyekti formalaşdırır və onun üzərində toplaşır. Component və Container GUİ-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 əmrlərin 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 GUİ-ləri mənimsəməkdə daha böyük rol oynayır. Swing standart olaraq Netbeans-ə yüklənmişdir. Bu baxımdan ilk olaraq Swing-lə təcrübə keçmək ən doğrusudur.

"Salam Dünya!" proqramı qrafiki təsvirdə



Yuxarıdakı "Salam Dünya!" proqramının qrafiki görünşü 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 method-ları, proqramları və sair dəyərləri obyekt vasitəsi ilə çağırır, istədiyimiz yeni dəyərlərlə yeni forma və funksiyalar yaradırıq.

```
package com.ZJAVAKITABI;
import javax.swing.JFrame;
class Swingtest{
    public static void main(String[] args) {
        JFrame pəncərə = new JFrame("SALAM DÜNYA");
        pəncərə.setVisible(true);
        pəncərə.setSize(500, 400);
        pəncərə.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);     }}
```

Obyekt “pəncərə” yaradıldıqda proqram daxilində pəncərədən sonra “.” nöqtə qoysaq avtomatik olaraq JVM-dən JFrame-ə aid olan funksiyaları “pəncərə” obyekti 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 - bu səhifədə hər əmrə uyğun olaraq onun alt əmrləri sadə formada göstərilmişdir.