UNIVERSITETI ALEKSANDËR MOISIU, DURRËS

Leksion 5

Methods

Migert XHAJA, 2018

Përmbajtja

6.1 Metodat

Analogji për të kuptuar klasat dhe përmbajtjen e klasave:

Supozojmë se duam të ngasim një makinë, dhe ta rrisim shpejtësinë duke shtypur pedalin e gazit. Para se të ngasim makinën, dikush duhet më parë ta krijojë atë.
Zakonisht nje makinë lind nga një vizatim i një inxhinieri, njëlloj si skicat që bëjnë arkitektët për një shtëpi.
Një makinë ka pedalin e gazit që e bën të ecë shpejt, por kjo nuk është mjaftueshëm që makina të ecë shpejt – shoferi duhet të shtypë pedalin e gazit.
Përfytyroni tani që makina është objekti. Dhënia e shpejtesisë është funksioni (metoda) që ne duam të realizojmë. Supozojme se jemi në klasën Shoferët. Pra përpara se ta përdorim një makinë ne duhet më parë ta krijojmë atë si më poshtë:

Shoferët makina = newShoferët();

Kryerja e një detyre në program kërkon një metodë.

Në Java, *një klasë mban të paktën një metodë*, ashtu si vizatimi i inxhinierit për makinën mban vizatimin e strukturës mekanike të pedalit të gazit.
Një klasë përfshin një ose disa metoda të dizenjuara për të kryer detyrat e klasës.

Ne duhet të ndërtojmë një objekt të një klase përpara se programi të kryejë detyrat që klasa i përshkruan si kryhen.
Kjo është arsyeja që Java njihet si gjuhë programimi e orientuar nga objektet.
Kur ngasim makinën shtypja e pedalit të gazit i dërgon një mesazh makinës që të kryejë një detyrë – të eci më shpejt.Në i dërgojmë mesazhe një objekti – çdo mesazh implementohet si thirrje metode e cila i kërkon metodës së objektit të kryejë detyrën e saj.

- Një makinë ka disa atribute (çdo objekt ka atributet e veta):
    - Ngjyra, numri i dyerve, sasia e karburantit në serbator, shpejtësia aktuale, km e përshkuar.
    - Atributet përshkruhen si pjesë e dizenjimit të makinës në diagramat inxhinierike.
    - Çdo makinë mban atributet e veta.
    - Çdo makinë di sa karburant ka në serbatorin e saj, por jo sa ka në serbatorët e makinave të tjera.

- Një objekt ka atributet e veta që mbahen nga objekti kur ai përdoret në program:
    - Specifikohen si pjesë e klasës së objektit
    - Një objekt "llogari bankare" ka një atribut "balanca" që mban sasinë e parave në llogari.
    - Çdo objekt "llogari bankare" di balancën e llogarisë që ajo përfaqëson, por jo balancën e llogarive të tjera në bankë.

Atributet specifikohen nga variablat e instances të klasës.

Deri tani kemi parë të shkruhet vetëm metoda main dhe kemi përdorur metodat që disponon vetë java.Por si mund ta ndërtojme vetë një metode?

Në përgjithësi, një metodë ka sintaksën e mëposhtme:

```
modifierTipiITeDhenesTeVleresSeKthimitemriIMetodes(lista e parametrave) {
  // Trupi i metodes;
}
```

Përcaktimi i një metode konsiston në percaktimin e kokës së metodës dhe të trupit të saj. Më poshtë janë listuar dhe shpjeguar të gjitha pjesët e një metode:

- **Modifiers**: Është opsional, dhe i tregon kompajlerit se si duhet ta therrasi metodën. Pra ai përcakton tipin e aksesimit të metodës.
- **Tipi i te dhënës për vlerën e kthyer**: Një metodë mund të kthejë ose jo një vlerë pas ekzekutimit të saj. Nëse metoda do të kthejë vlerë, këtë vlerë do ta mbajë një variabël e cila ka nevojë për përcaktimin e llojit të të dhënës që ajo do të jetë. Pra është si tip dektarimi paraprak i variablit të kthimit. Disa metoda i kryejnë punet e dëshiruara pa kthyer asnjë vlerë. Në kete rast pjesa TipiITeDhenesTeVleresSeKthimit, do të shënohetpatjetër**void**.
  Per shembull metoda qe jane void jane: metoda main(), System.out.println(), JOptionPane.showMessageDialog etj.
  Vëmendje! KurMetoda nuk kthen asnjë vlerë, ateherë ajo quhet **Procedure**. Nëse metoda kthen vlerë pas ekzekutimit të saj, atëherë ajo do të quhet **Funksion**.
- **emriIMetodes**: Ky është emri i metodës.
- **Parametrat**: Është lista e parametrave të metodës nëse është e nevojshme të ketë.Kur e thërrasim një metodë, në kalojmë vlerën e vendosur nga ne brenda kllapave () si:
  emriIMetodes(vlere e vendosur nga ne);
  Pra e kalojmë vlerën në metodë si parameter hyrës.Kjo vlerë i referohet parametrit të paracaktuar të metodës kur ajo është krijuar.Nëse ka më shumë se një parametër hyrës, ata përkojnë me datatype, pozicionin, dhe numrin e parametrave të paracaktuar të metodës.
  Kujdes! Parametrat janë opsional, kështu që nëse nuk keni parapërcaktuar asnjë parametër hyrës në krijimin e metodës, nuk duhet të vendosni asgjëbrenda kllapave () kur e thërrisni metodën. Pra:
  emriIMetodes();
  Për shembull metoda Math.random(), nuk ka parametra hyrës.
  Trupi i metodës: trupi i metodës përmban një bllok intruksionesh që do të përcaktojnë se cfarë do të realizojë metoda kur të thërritet.

6.2 Përshtatja e kodit

Metodat mund të përdoren për të reduktuar kodin e tepërt dhe për të mundësuar ripërdorim e kodit.Metodat gjithashtu mund të përdoren për të përshtatur kodin dhe për të përmirësuar cilësinë e një program.
Shembulli më poshtë tregon përshtatjen e kodit me anëtë një metode:

Një metodë e zakonshme pjestuese. Java (MetodaPMPjava)

```
import java.util.Scanner;
public class MetodaPMP {
     /** Main method */
     public static void main(String[] args) {
     // Create a Scanner
     Scanner input = newScanner(System.in);
     // Prompt the user to enter tëo integers
     System.out.print("Enter first integer: ");
     int n1 = input.nextInt();
     System.out.print("Enter second integer: ");
     int n2 = input.nextInt();
     System.out.println("The greatest common divisor for
" + n1 +
     " and " + n2 + " is " + gcd(n1, n2 );
     }
     /** Return the gcd of tëo integers */
     public static int gcd(int n1, int n2) {
     int gcd = 1; // Initial gcd is 1
     int k = 2; // Possible gcd
          while(k <= n1 && k <= n2) {
               if(n1 % k == 0 && n2 % k == 0)
               gcd = k; // Update gcd
               k++;
                }
               return gcd; // Return gcd
     }
}
```

6.3 Metoda pa parametra hyrës

Shembull – Klasa Lenda

Krijojmë një klase të re (***Lenda***):

Fjala kyce **public** është për mënyrën e aksesimit dhe tregon që klasa është e aksesueshme publikisht.Metodat publike janë të disponueshme publikisht.Pra ato mund të thërriten nga metoda të klasave të tjera.
Tipi i kthimit tregon tipin e të dhënës që kthen metoda pasi kryen detyrën e saj.
Tipi i kthimit **void** tregon që metoda do të kryejë një detyrë por nuk do t'i kthejë mbrapsht ndonjë informacion metodës që e therret atë, pasi të përfundojë detyrën e saj.

- Emri i metodës shkruhet pas tipit të kthimit
- Me marrëveshje emrat e metodave fillojne me shkronjë të vogël dhe fjalet pasuese me shkonje të madhe.
- Kllapat bosh pas emrit të metodës tregojnë që metoda nuk kerkon informacion shtesë për të kryer detyrën e saj.
- Rreshti i parë i metodës quhet Koka e Metodës
- Trupi i çdo metode kufizohet nga kllapa hapëse dhe mbyllëse.
- Trupi i metodës përmban një ose më shume instruksione që kryejne detyrën e metodës.

```
public class Lenda {//klasa
     public void afishoMesazh(){//metoda e krijuar nga ne
          System.out.printf("Ky eshte Leksioni 3");
     }
}
```

E therrasim klasën e krijuar nga një klasë tjetër **TestLenda** e krijuar po nga ne:

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

     //krijon nje objekt te tipit Lenda (klasa jone)
     Lenda Java = newLenda();//Klasa EmriIObjektit = neë
Klasa();

     //therret metoden e krijuar me lart te bej punen e saj
(ahfishimin)
     Java.afishoMesazh();
     // nuk kemi futur asnje vlere pasi metoda nuk kishte
parametra hyres
     //dhe eshte i tipit void
     }
}
```

Console
Ky eshte Leksioni 3

Vëmendje! Meqë jemi brenda te njejtës paketë e thërrasim si:

Objekt.Metode(vlera1,vlera2, …);

përndryshe do ishte:

Klase.Objekt.Metode(vlera1,vlera2, …);

- Zakonisht nuk mund ta thërrasim një metode që i përket një klase tjetër pa krijuar më parë një objekt të asaj klase.
- Çdo klasë që krijoni, bëhet një tip i ri që mund të përdoret për të deklaruar variabla dhe për të krijuar objekte.
- Fjala kyce ***new*** krijon një objekt tëri të klasës së specifikuar në të djathtë të fjalës kyce.
- Objekti përdoret për të inicializuar variabël të tipit të klasës.
- Kllapat në të djathtë të emrit të klasës janë të nevojshme.***metoda()***

6.4Metoda me parametra hyrës

```
public class Lenda {

     public void afishoMesazh(String emriIKursit){ //kemi
krijuar nje parameter

          System.out.printf("Ky eshte Leksioni 3 per lenden "
+ emriIKursit);

     }
}
```

Vlera e parametrit afishohet si pjesë e stringut

```
import java.util.Scanner;

public class TestLenda {
     public static void main(String[] args){
          //krijojme nje objekt te tipit Scanner qe te lexoje
te dhenat
          Scanner input = newScanner(System.in);
          System.out.println("Ju lutem, fusni emrin e lendes:
");
          //krijoj nje variabel qe do ruaje te dhenen e
     shkruar nga ne
          String vrbLenda = input.nextLine(); //lexon nje
     rrjesht text
          //krijon nje objekt te tipit Lenda (klasa jone)
          Lenda Java = newLenda();
          //therret metoden e krijuar me lart te bej punen e
     saj (ahfishimin)
          Java.afishoMesazh(vrbLenda);
     }
```

```
}
```

Console
Ju lutem, fusni emrin e lendes:
Java
Ky eshte Leksioni 3 per lenden Java

- Metoda **NextLine** e objektit Scanner:
  - Lexon karakteret e shtypur nga përdoruesi në një rrjesht
  - Kthen një string
  - Shtypim enter për t'i kaluar stringun programit
- Metoda **Next** e objektit Scanner
  - Lexon fjalë
  - Lexon karaktere derisa gjen hapësirë

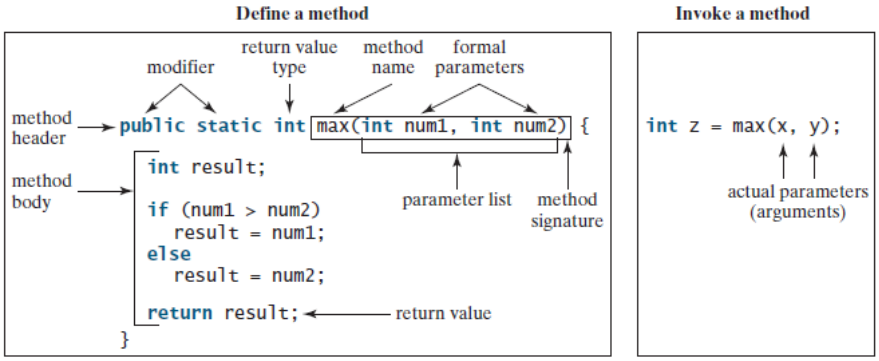
Shembull – NrMax

Krijoni nje metodë që po ti fusesh dy numra të plotë si parametra, ajo do të gjejë se cili nga numrat është më i madh. Metodën thërriteni nga një metode tjetër. Outputi duhet të jetë i trajtës:

Fusni dy numra te plote:
5 7
Numri 7 eshte me i madh

Zgjidhje:

Mënyra se si do të operohet për zgjidhjen e këtij problem,është ilustruar në figuren e mëposhtme:



Pra do të krijojmë një klas **public** (që mund të thërritet). Ajo nuk do jetë void si në shëmbullin paraardhës sepse ne duam që metoda të na kthejë një rezultat. Kështu që metoda do të ketë të specifikuar datatype e variablit të kthimit që në rastin tonëështë**int**. Metodën do ta titullojmë**max**.

Ajo do të përmbajëdy parametra hyrës që janë dy numra të plotë, të cilët ne po i quajmë**num1** dhe **num2**.
Trupi i metodës do të jetë i strukturuar sic do ushtrim tjetër qe kemi bërë më parë për të marrë rezultatin e dëshiruar. Pra do të krahasojmë numrat dhe do të ruajmë në variablin e kthimit numrin më të madh.

Metodën do ta thërrasim (krahu i djathtë i figures), duke shruar emrin e saj dhe paramtrat hyrës.

Vëmendje! **Argumentat dhe parametrat**
Numri i argumentave në thirrjen e një metodë, duhet të korrespondojë me numrin e parametrave në deklarimin e metodës.

Tipet e argumentave në thirrjen e metodës duhet të korespondojnë me tipet e parametrave në deklarimin e metodës.

Kujdes! Në kokën e metodës ne duhet të deklarojmë datatype për çdo parameter.
Për shembull:

Nuk është e lejuar

```
public class NrMax {

	public static int max(int num1,num2){

	}
}
```

Është e lejuar

```
public class NrMax {

	public static intmax(int num1,int num2){

	}
}
```

Vazhoj me krijimin e metodës për krahasimin e dy numrave të plotë:

```
public class NrMax {
	public static int max(int num1,int num2){
		//dektaroj variablin dales
		int result;
		//pjesa e kodit te krahasimit
		if(num1 > num2)
			result = num1;
		else
```

```
            result = num2;
        //i tregoj metodes se ke variabel duhet te me ktheje
        return result;

    }
}
```

Pasi krijova metodën, tani më duhet thjesht që ta therras atë . Në një klasë tjetër TestNrMax, thërras metodën max, si më poshtë:

```
import java.util.Scanner;
public class TestNrMax {

    public static void main(String[] args) {
        System.out.println("Fusni dy numra te plote:");

        Scanner input = newScanner(System.in);
        int num1 = input.nextInt();
        int num2 = input.nextInt();

        //klase.metode(parameter1,parameter2)
        System.out.println(NrMax.max(num1, num2));

    }
}
```

Pyetje: A shikoni ndonjë ndryshim në mënyren e thërritjes së metodës në këtë shëmbull krahasuar me shëmbullin paraardhes?

Vëmendje!Shohim që në këtë rast, ndryshe nga shëmbulli i mëparshëm, ne nuk kemi krijuar një objekt për ta thërritur metodën. Metodën e kemi thërritur jo si object.metode por si klase.metode. Kjo për arsyen e vetme sepse **e kemi krijuar metoden si statike** (**publicstaticint**max(**int** num1,**int** num2)).

Tipi i metodës: kur ne dizenjojmë një metode brenda klasës, nganjeherë e kemi të nevojshme që ti ndërtojmë ato të pavarura nga objektet e klasës. Në këtë mënyrë, funksionet mund të aksesohen nga vetë klasa pa patur nevojen e një objekti.Kjo do të thotë se disa nga metodat që ne dizenjojme varen nga objekti i krijuar dhe disa te tjera nga vetë klasa.

Kështu për të thërritur një metodë qëështë e pavarur nga objekti, ju mund ta thërrisni si klasa.metoda

Për shëmbull, për të llogaritur eksponencialin e një numri real, përdorni metoden exp nga klasa Math
Math.exp(numriIm);

Edhe dickatë fundit.Nëse një metode statike thërret një metode tjetër, edhe metoda që thërritet duhet të jetë gjithashtu statike.

6.5 Variablat lokale

- Variabla që deklarohen brenda trupit të një metode
- Kur përfundon metoda, vlerat e variablave të saj lokale humbin.
- Dimë që një objekt ka atribute që mbahen nga objekti për aq kohësa ai përdoret në program.
- Këto atribute ekzistojne para se metoda të thirret nga objekti, dhe pasi metoda e përfundon ekzekutimin.

Një klasë përbëhët nga një apo disa metoda që manipulojne atributet që i përkasin një objekti të klasës.
Atributet paraqiten si variabla në deklarimin e klasës.

```
public class Lenda {
     //krijoj nje veriabel private, pra qe nuk aksesohet nga
jashte
     private String emriLendes;
     //metoda per ta futur emrin e lendes
     public void setEmerLende(String emri){
          emriLendes = emri;
     }
     //metoda per ta marre ta marre emrin e lendes
     public String getEmerLende(){
          return emriLendes;
     }
     //metoda qe kryen detyren e afishimit
     public void afishoMesazh(){
          System.out.printf("Ky eshte Leksioni 3 per lenden "
+ getEmerLende());
          }
}
```

Metodat Overloading

Metodat Overloading ju mundësojnë të përcaktoni metodat me të njëjtin emër sa më gjatë pasi nënshkrimet e tyre janë të ndryshme.
Metoda **max** e përdorur më parë funksionon vetëm me tipin e të dhënave **int**. Por, çka nëse keni nevojë të përcaktoni se cili prej dy numrave në pikat lundruese ka vlerën maksimale? Zgjidhja është të krijojë një metodë tjetër me të njëjtin emër, por parametra të ndryshëm, siç tregohet në kodin e mëposhtëm:

```
public static double max(double num1, double num2) {
if (num1 > num2)
return num1;
else
return num2;
     }
```

Nëse thirni **max** me parametra **int**, do të thirret metoda maksimale që pret parametrat **int**; nëse thirni **max** me parametra të dyfishtë, do të thirret metoda **max** që pret parametra të dyfishtë. Kjo quhet metoda Overloading; që është, dy metoda kanë të njëjtin emër, por lista të ndryshme të parametrave brenda një klase. Kompilatori Java përcakton se cila metodë duhet përdorur në bazë të nënshkrimit të metodës.

Ushtrimi 5.9 `TestMethodOverloading.java`

```
 public class TestMethodOverloading {
     public static void main(String[] args) {
     // Invoke the max method with int parameters
     System.out.println("The maximum of 3 and 4 is "
     + max(3, 4));

     // Invoke the max method with the double parameters
     System.out.println("The maximum of 3.0 and 5.4 is "
     + max(3.0, 5.4));

     // Invoke the max method with three double parameters
     System.out.println("The maximum of 3.0, 5.4, and 10.14 is
"
     + max(3.0, 5.4, 10.14));
 }

     /** Return the max of two int values */
     public static int max(int num1, int num2) {
     if (num1 > num2)
     return num1;
     else
     return num2;
     }

     /** Find the max of two double values */
     public static double max(double num1, double num2) {
     if (num1 > num2)
     return num1;
     else
     return num2;
     }

      /** Return the max of three double values */
     public static double max(double num1, double num2, double
num3) {
```

```
    return max(max(num1, num2), num3);
}
    }
```

Output:

```
The maximum of 3 and 4 is 4
The maximum of 3.0 and 5.4 is 5.4
The maximum of 3.0, 5.4, and 10.14 is 10.14
```