



Struktura e komponenteve: Ndërtimi i metodave dhe klasave

Struktura e komponenteve: Ndërtimi i metodave dhe klasave

Objektivat:

- ⑥ Të shkruhen metoda publike dhe private të cilat marrin argumente (parametra) dhe kthejnë rezultate.
- ⑥ Të shkruhen klasa që përmbajnë metoda të cilat rishfrytëzohen sërish.
- ⑥ Të lexohen dhe shkruhen specifikime interfejsash, të cilat përshkruajnë sjelljet e klasave dhe metodave të tyre.

Metodat

Metodë: varg i emërtuar urdhërash; i destinuar për kryerjen e një detyre ose përgjegjësie specifike.

```
public class MyApplication
{
    public static void main(String[] args)
    {
        // ... Instruksionet të cilat ekzekutohen
        //      me startimin e aplikacionit
    }
}
```

Metodat – Vazhdim

```
import java.awt.*;
import javax.swing.*;
public class MyPanel extends JPanel
{ public MyPanel()
  { // ... Instruksionet për konstruktimin dhe inicializimin
    //      e panelit
  }

  public void paintComponent(Graphics g)
  { // ... Instruksionet për vizatim në panelin
  }
}
```

Metodat publike

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

Invokim metode: dërgim mesazhi i cili përfshin emrin e metodës. Mesazhi kërkon që objekti pranues të ekzekutojë metodën e emërtuar në mesazhin.

Klient: objekti i cili dërgon mesazh (invokim).

Pranues: objekti i cili pranon mesazh; ekzekuton metodën e emërtuar në mesazhin.

Metodat publike – Vazhdim

```
public void paintComponent(Graphics g) // kjo është ballina
{ URDHËRAT // ky është trupi
}
```

Përnga sintaksa, metoda përbëhet nga *ballina* (*header line*) dhe *trupi*.

Metoda publike i është në dispozicion „publikut gjeneral“, përfshirë sistemin operativ të kompjuterit si dhe objekte të krijuara nga klasë tjera.

Ca metoda publike themelore

```
/** Vizaton ASCII art */
public class AsciiArtWriter
{ /** Konstruktori */
    public AsciiArtWriter()
    { System.out.println(); }

    /** Afishon një bletë */
    public void printBee()
    { System.out.println("  ,-.");
      System.out.println("  \\_/" ); // \\ për \
      System.out.println(">{|||}-");
      System.out.println("  /  \\\");
      System.out.println("  \-^");
      System.out.println();
    }
```

Ca metoda publike themelore – Vazhdim

```
/** Afishon një fluturë */  
public void printButterfly()  
{ System.out.println("  _ \" _"); // \" për "  
  System.out.println(" (_\\|/_)");  
  System.out.println(" (/|\\)");  
  System.out.println();  
}  
  
/** Afishon një kalikuese */  
public void printLadybug()  
{ System.out.println(" `m\\'"); // \\' për '  
  System.out.println(" (|)");  
  System.out.println();  
}  
}
```


Ca metoda publike themelore – Vazhdim

```
/** DrawArt: ca Ascii art*/
public class DrawArt
{ public static void main(String[] args)
  { AsciiArtWriter writer = new AsciiArtWriter();
    writer.printBee();
    System.out.println("Test");
    writer.printLadybug();
    writer.printButterfly();
  }
}
```

Ca metoda publike themelore – Vazhdim

Specifikim metode: koment i filluar me `/**` i cili dokumenton qellimin ose sjelljen e një metode. Specifikimet mund të përpunohen me veglën `javadoc` për të gjeneruar një dokumentacion kodi.

Metodat konstruktorë

Konstruktor: metodë përbrenda klasës e cila invokohet automatikisht kur të konstruktohet një objekt në memorien qendrore të kompjuterit.

```
AsciiArtWriter writer = new AsciiArtWriter();
```

Metodat konstruktorë – Vazhdim

DrawArt

```
main
{ 1> AsciiArtWriter writer = new AsciiArtWriter();
  ...
}
```

Metodat konstruktorë – Vazhdim

DrawArt

```
main
{ AsciiArtWriter writer == ?;
  1> writer = PRIT
  ...
}
```

a1 : AsciiArtWriter

```
AsciiArtWriter()
{ 2> System.out.println(); }
public void printBee() {...}
public void printButterfly() {...}
public void printLadybug() {...}
```

Metodat konstruktorë – Vazhdim

DrawArt

```
main
{ AsciiArtWriter writer == ?;
  1> writer = a1
  ...
}
```

a1 : AsciiArtWriter

```
public void printBee() {...}
public void printButterfly() {...}
public void printLadybug() {...}
```

Metodat konstruktorë – Vazhdim

DrawArt

```
main
{ AsciiArtWriter writer == a1;
  ...
  1> writer.printBee();
  ...
}
```

a1 : AsciiArtWriter

```
public void printBee() {...}
public void printButterfly() {...}
public void printLadybug() {...}
```

Metodat konstruktorë – Vazhdim

DrawArt

```
main
{ AsciiArtWriter writer == a1;
  ...
  1> PRIT
  ...
}
```

a1 : AsciiArtWriter

```
public void printBee()
{ 2> System.out.println(...);
  ...
}
public void printButterfly() {...}
public void printLadybug() {...}
```


Metodat konstruktorë – Vazhdim

DrawArt

```
main
{ AsciiArtWriter writer == a1;
  ...
  1> System.out.println("Test");
  ...
}
```

a1 : AsciiArtWriter

```
public void printBee() {...}
public void printButterfly() {...}
public void printLadybug() {...}
```

Parametrat e metodave

Parametrat aktualë: argumentet lista e të cilëve jepet së bashku me emrin e metodës gjatë invokimit të metodës, p.sh.,

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

Parametrat e metodave – Vazhdim

Parametrat formalë: emrat e variablave lista e të cilave jepet në ballinën e metodës.

```
/** Afishon një bletë me emër.
 * @param name emri i bletës
 */
public void printBeeWithName(String name)
{ System.out.println("  ,-.");
  System.out.println("  \\_/" );
  System.out.println(">{|||}-" + name + "-");
  System.out.println("  /  \\\");
  System.out.println("  ^-^");
  System.out.println();
}
```

Parametrat e metodave – Vazhdim

Kur metoda invokohet parametrat aktualë u shoqërohen parametrave formalë përkatës ashtu që parametri formal inicializohet me parametrin aktual. Lidhja e një parametri aktual me një parametër formal funksionon sikur një inicializim i një variableje lokale.

```
AsciiArtWriter writer = new AsciiArtWriter();  
String s = "Tringa";  
writer.printBeeWithName(s);  
  
writer.printBeeWithName("Filan " + "Fisteku");
```

Parametrat e metodave – Vazhdim

DrawArt

```
main
{ AsciiArtWriter writer == a1;
  String s == "Tringa";
  1> writer.printBeeWithName(s);
  ...
}
```

a1 : AsciiArtWriter

```
public void printBee() {...}
public void printButterfly() {...}
public void printLadybug() {...}
public void printBeeWithName(String name) {...}
```

Parametrat e metodave – Vazhdim

DrawArt

```
main
{ AsciiArtWriter writer == a1;
  String s == "Tringa";
  1> writer.printBeeWithName("Tringa");
  ...
}
```

a1 : AsciiArtWriter

```
public void printBee() {...}
public void printButterfly() {...}
public void printLadybug() {...}
public void printBeeWithName(String name) {...}
```

Parametrat e metodave – Vazhdim

DrawArt

```
main
{ AsciiArtWriter writer == a1;
  String s == "Tringa";
  1> PRIT
  ...
}
```

a1 : AsciiArtWriter

```
...
public void printBeeWithName
{ 2> String name = "Tringa";
  System.out.println(...);
  ...
}
...
```

Parametrat e metodave – Vazhdim

DrawArt

```
main
{ AsciiArtWriter writer == a1;
  String s == "Tringa";
  1> PRIT
  ...
}
```

a1 : AsciiArtWriter

```
...
public void printBeeWithName
{ String name == "Tringa";
  2> System.out.println(...);
  ...
}
...
```


Format e parametrave

Çdo vlerë e cila mund t'i shoqërohet (ndahet për vlerë) një variableje mund të jetë parametër aktual. Parametri aktual duhet të jetë kompatibel me parametrin formal.

```
import java.text.*;
/** Kryen operacione matematike. */
public class MathOperations
{ /** Afishon numrin invers të një numri të plotë
 * të formatizuar në tri vende decimale.
 * @param i numri i plotë i cili invertohet */
public void printInverse(int i)
{ DecimalFormat formatter = new DecimalFormat("0.000");
  double d = 1.0 / i;
  String s = formatter.format(d);
  System.out.println(s);
}
}
```

Format e parametrave – Vazhdim

Invokimi i `printInverse` bëhet, p.sh., me

```
MathOperations calculator = new MathOperations();  
calculator.printInverse(3);
```

Invokim jokorrekt: `calculator.printInverse(0.5).`

Për `printInverse` të shkruar me ballinën

```
public void printInverse(double i)
```

korrekte do të ishin të dy invokimet vijuese:

```
calculator.printInverse(0.5);  
calculator.printInverse(3);
```

Format e parametrave – Vazhdim

```
/** Afishon numrin invers të një numri të plotë.  
 * @param i numri i plotë i cili invertohet  
 * @param pattern shablloni për formatizimin e rezultatit */  
public void printInverse(int i, String pattern)  
{ DecimalFormat formatter = new DecimalFormat(pattern);  
  double d = 1.0 / i;  
  String s = formatter.format(d);  
  System.out.println(s);  
}
```

Format e parametrave – Vazhdim

```
MathOperations calculator = new MathOperations();  
calculator.printInverse(3, "0.00000");  
calculator.printInverse(5, "0.0");
```

Shoqërimi i parametrave aktualë parametrave formalë përcaktohet nga renditja në listë e parametrave aktualë – parametrat aktualë u shoqërohen parametrave formalë me radhë.

Format e parametrave – Vazhdim

```
/** Afishon numrin invers të një numri të plotë.  
 * @param i  numri i plotë i cili invertohet  
 * @param f  objekti i cili formatizon rezultatin */  
public void printInverse(int i, DecimalFormat f)  
{ double d = 1.0 / i;  
  String s = f.format(d);  
  System.out.println(s);  
}
```

Format e parametrave – Vazhdim

Invokimi i metodës së tretë:

```
DecimalFormat fivePlaces = new DecimalFormat("0.00000");  
calculator.printInverse(3, fivePlaces);
```

Format e parametrave – Vazhdim

- ⑥ Metodatat konstruktorë mund të shfrytëzojnë parametra formalë njësoj sikurse metodatat tjera publike.
- ⑥ Metoda ballina e së cilës ka formën

```
public void EMRI_I_METODËS( )
```

është metodë me zero parametra formalë dhe duhet të invokohet nga urdhër i cili ka zero parametra aktualë:

```
OBJEKTI_PRANUES.EMRI_I_METODËS( ) ;
```

Case study: kornizë dalëse me destinim të përgjithshëm

Specifikim interfejsi (interfejs): listë emrash dhe përgjegjësish të metodave publike të një klase.

MyWriter krijon dritare grafike e cila afishon një fjali	
Konstruktori:	
MyWriter(int w, int h)	Konstrukton dritaren me gjerësi dhe lartësi të specifikuar
Metodat:	
writeSentnce(String s)	Afishon fjalinë në dritaren grafike
repositionSentence(int x, int y)	Riafishon fjalinë në pozitën e re

Tabela 1. Specifikimi i interfejsit të kornizës dalëse MyWriter

Case study: kornizë dalëse me destinim të përgjithshëm – Vazhdim

```
import javax.swing.*;
import java.awt.*;
/** Krijon dritare grafike e cila afishon një fjali. */
public class MyWriter extends JPanel
{ private int width;
  private int height;
  private String sentence = "";
  private int xPosition;
  private int yPosition;
```

Case study: kornizë dalëse me destinim të përgjithshëm – Vazhdim

```
/** Konstruktor: krijon panelin.  
 * @param w  gjerësia e dritares  
 * @param h  lartësia e dritares */  
public MyWriter(int w, int h)  
{ width = w;  
  height = h;  
  xPosition = width / 5;  
  yPosition = height / 2;  
  JFrame myFrame = new JFrame();  
  myFrame.getContentPane().add(this);  
  myFrame.setTitle("MyWriter");  
  myFrame.setSize(width, height);  
  myFrame.setVisible(true);  
}
```

Case study: kornizë dalëse me destinim të përgjithshëm – Vazhdim

```
/** Vizaton panelin.  
 * @param g penda grafike */  
public void paintComponent(Graphics g)  
{ g.setColor(Color.white);  
  g.fillRect(0, 0, width, height);  
  g.setColor(Color.red);  
  g.drawString(sentence, xPosition, yPosition);  
}
```

Case study: kornizë dalëse me destinim të përgjithshëm – Vazhdim

```
/** Afishon një fjali.
 * @param s  fjalia që afishohet */
public void writeSentence(String s)
{ sentence = s;
  this.repaint();
}

/** Riafishon fjalinë në pozitën e re.
 * @param x  pozita e re horizontale
 * @param y  pozita e re vertikale */
public void repositionSentence(int x, int y)
{ xPosition = x;
  yPosition = y;
  this.repaint();
}
}
```

Case study: kornizë dalëse me destinim të përgjithshëm – Vazhdim

```
import javax.swing.*;
/** Afishon në dritare grafike një fjali të futur. */
public class MyWriterTest
{ public static void main(String args)
  { int width = 300;
    int height = 200;
    MyWriter writer = new MyWriter(width, height);
    writer.repositionSentence(50, 80);
    String s = JOptionPane.showInputDialog("Rradhisni ca tekst:");
    writer.writeSentence(s);
  }
}
```

Case study: kornizë dalëse me destinim të përgjithshëm – Vazhdim

1. Disenjuam dhe shkruam një klasë e cila mund të shfrytëzohet si komponentë e shumë aplikacioneve.
2. Disenjuam një specifikim interfejsi i cili na ndihmoi ta shkruajmë klasën.
3. Shfrytëzuesit e klasës mund ta lexojnë specifikim e klasës për të kuptuar se si t'i shfrytëzojnë metodat e klasës; nuk ka nevojë për ta lexuar kodimin e klasës.
4. Kodimi shfrytëzoi konstruktor dhe fusha private për të ndihmuar metodat publike të sillen në mënyrë korrekte; metodat publike invokuan njëra tjetrën sipas nevojës.

Case study: kornizë dalëse me destinim të përgjithshëm – Vazhdim

- ⑥ Kur një aplikacion ndërtohet nga disa njerëz, është më lehtë për njerëzit veç e veç të shkruajnë dhe testojnë pjesët e aplikacionit në qoftë se aplikacioni ndahet në klasë.
- ⑥ Në qoftë se një pjesë aplikacioni duhet sërish të shkruhet ose të zëvendësohet, është më lehtë për t'u bërë në qoftë se pjesa është një klasë e veçantë.

Rezultatet e metodave: Funksionet

Kur një objekt klient dërgon mesazh, ndonjëherë klienti pret përgjegjje kthyese, p.sh.,

```
String input = JOptionPane.showInputDialog("Gradë Fahrenheit:");  
int f = new Integer(input).intValue();
```

Rezultat metode: përgjegjje e llogaritur nga metoda dhe e kthyer klientit.

Funksion: metodë e cila kthen rezultat.

Rezultatet e metodave: Funksionet – Vazhdim

```
/** Konverton vlera temperaturash. */  
public class TemperatureConvertor  
{ /** Konverton vlerën e dhënë.  
    * @param f  shkallët Fahrenheit  
    * @return  ekuivalenti në shkallë Celsius */  
    public double fahrenheitToCelsius(double f)  
    { double c = (5.0/9.0) * (f - 32);  
      return c;  
    }  
}
```

Rezultatet e metodave: Funksionet – Vazhdim

Teknika për kthimin e rezultatit:

- ⑥ Në ballinën e metodës zëvendësohet fjala kyçe `void` me tipin e të dhënave të rezultatit të cilin duhet ta kthejë metoda. Fjala kyçe `void` do të thotë mos kthe fare rezultat.
- ⑥ Në fund të trupit të metodës vëhet urdhër
`return SH;`
ku shprehja `SH` llogarit vlerën e cila do të kthehet.

Rezultatet e metodave: Funksionet – Vazhdim

```
import javax.swing.*;
/** ConvertorTest teston konvertorin e temperaturave */
public class ConvertorTest
{
    public static void main(String args)
    {
        String input =
            JOptionPane.showInputDialog("Gradë Fahrenheit:");
        int f = new Integer(input).intValue();
        TemperatureConvertor convert = new TemperatureConvertor();
        double c = convert.fahrenheitToCelsius(f);
        MyWriter writer = new MyWriter(300, 200);
        writer.writeSentence(f + " Fahrenheit janë " + c + " Celsius");
    }
}
```

Rezultatet e metodave: Funksionet – Vazhdim

Urdhëri `return` është pikë daljeje nga funksioni. P.sh., funksioni vijues llogarit në mënyrë jokorrekte katrorin e një numri:

```
public int square(int num)
{
    int result = 0;
    return result;
    result = num * num;
}
```

Rezultatet e metodave: FunkSIONET – Vazhdim

ConvertorTest

```
main
{ int f == 68.0
  1> TemperatureConvertor convert = new TemperatureConvertor();
  double c = convert.fahrenheitToCelsius(f);
  ...
}
```

Rezultatet e metodave: Funksionet – Vazhdim

ConvertorTest

```
main
{ int f == 68.0
  TemperatureConvertor convert == a1
  1> double c = convert.fahrenheitToCelsius(f);
  ...
}
```

a1 : TemperatureConvertor

```
public double fahrenheitToCelsius(double f) {...}
```

Rezultatet e metodave: FunkSIONET – Vazhdim

ConvertorTest

```
main
{ int f == 68.0
  TemperatureConvertor convert == a1
  double c == ?
  1> c = PRINT;
  ...
}
```

a1 : TemperatureConvertor

```
fahrenheitToCelsius
{ double f == 68.0
  double c == 20.0
  2> return c;
}
```

Rezultatet e metodave: Funksionet – Vazhdim

ConvertorTest

```
main
{ int f == 68.0
  TemperatureConvertor convert == a1
  double c == ?
  1> c = 20.0;
  ...
}
```

a1 : TemperatureConvertor

```
public double fahrenheitToCelsius(double f) {...}
```


Rezultatet e metodave: FunkSIONET – Vazhdim

Rezultati i një funksioni mund të injorohet, p.sh., në
`class FahrenheitToCelsiusWriter` kemi pasur

```
/** main: për testim */  
public static void main(String args)  
{ new FahrenheitToCelsiusWriter(); }
```

Metodat private

Metodat private mund të invokohen vetëm përbrenda klasës në të cilë paraqiten.

Metoda private shkruhen si zgjidhje në dy situata:

1. Kur ka përsëritje të një detyre.
2. Kur një formulë ose nënalgoritëm fundamental meriton të ketë emër të vetin.

Emërtimi i një nënalgoritmi me metodë private

Algoritmi për paraqitjen e dritares grafike me kufi të kaltër e qendër të bardhë:

1. Ngjyros tërë dritaren në të kaltër.
2. Llogarit madhësinë e një drejtkëndëshi të madhësisë ca më të vogël sesa tërë dritarja.
3. Ngjyros drejtkëndëshin në të bardhë.

Emërtimi i një nënalgorithmi me metodë private – Vazhdim

```
/** Vizaton panelin.
 * @param g penda grafike */
public void paintComponent(Graphics g)
{ makeBorder(g);
  g.setColor(Color.red);
  g.drawString(sentence, xPosition, yPosition);
}
/** Ngjyros kufirin e kornizës.
 * @param pen penda grafike */
private void makeBorder(Graphics pen)
{ pen.setColor(Color.blue);
  pen.fillRect(0, 0, width, height);
  int borderSize = 20;
  int rectWidth = width - 2 * borderSize;
  int rectHeight = height - 2 * borderSize;
  pen.setColor(Color.white);
  pen.fillRect(borderSize, borderSize, rectWidth, rectHeight);
}
```

Përsëritja e një aktiviteti me metodë private

Algoritmi për ngjyrosjen e një veje:

1. Llogarit lartësinë e vesë si $2/3$ e gjerësisë së saj.
2. Llogarit skajin e majtë të vesë ashtu që veja të jetë e qendërsuar në dritaren, dhe llogarit skajin e sipërm të vesë.
3. Ngjyros venë.
4. Kthe vlerën e skajit të sipërm, në rast nevojë që mëpastaj të vendoset një tjetër ve përmbi.

Përsëritja e një aktiviteti me metodë private – Vazhdim

```
import javax.swing.*;
import java.awt.*;
/** Vizaton tri ve të renditura. */
public class StackedEggWriter extends JPanel
{ private int frameWidth;
  private int frameHeight;
  private int egg1Size;
  private int egg2Size;
  private int egg3Size;
```

Përsëritja e një aktiviteti me metodë private – Vazhdim

```
/** Konstruktori.  
 * @param width  gjerësia e panelit  
 * @param height lartësia e panelit  
 * @param size1  gjerësia e vesë së poshtme  
 * @param size2  gjerësia e vesë së mesme  
 * @param size3  gjerësia e vesë së sipërme */  
public StackedEggWriter(int width, int height,  
                        int size1, int size2, int size3)  
{ frameWidth = width; frameHeight = height;  
  egg1Size = size1; egg2Size = size2;  
  egg3Size = size3;  
  JFrame myFrame = new JFrame();  
  myFrame.getContentPane().add(this);  
  myFrame.setTitle("Mullar vezësh");  
  myFrame.setBackground(Color.yellow);  
  myFrame.setSize(frameWidth, frameHeight);  
  myFrame.setVisible(true);  
}
```

Përsëritja e një aktiviteti me metodë private – Vazhdim

```
/** Mbush dritaren me vezë.  
 * @param g penda grafike */  
public void paintComponent(Graphics g)  
{ int egg1Top = paintAnEgg(frameHeight, egg1Size, g);  
  int egg2Top = paintAnEgg(egg1Top, egg2Size, g);  
  paintAnEgg(egg2Top, egg3Size, g);  
}
```


Përsëritja e një aktiviteti me metodë *private* – Vazhdim

```
/** Vizaton një ve.  
 * @param bottom pozita e skajit të poshtëm  
 * @param width  pozita e skajit të poshtëm  
 * @param pen    penda grafike  
 * @return y     pozita e skajit të sipërm */  
private int paintAnEgg(int bottom, int width, Graphics pen)  
{ int height = 2 * width / 3;  
  int top = bottom - height;  
  int left = (frameWidth - width) / 2;  
  pen.setColor(Color.pink);  
  pen.fillOval(left, top, width, height);  
  pen.setColor(Color.black);  
  pen.drawOval(left, top, width, height);  
  return top;  
}
```

Përsëritja e një aktiviteti me metodë private – Vazhdim

```
/** main për testim */  
public static void main(String args)  
{ int width = 300;  
  int height = 200;  
  new StackedEggWriter(width, height, 140, 90, 50);  
}  
}
```