Laborator 9 - Java
Crearea claselor de obiecte.
Variabilele (campurile) clasei de obiecte

Probleme rezolvate:

Scrieti, compilati si rulati toate exemplele din acest laborator:

1. Exemplificarea domeniului de vizibilitate a variabilelor

Programul are declarate doua variabile cu acelasi nume. Prima, o variabila de instanta, are numele *test* si a fost initializata cu valoarea 10. A doua este un parametru cu acelasi nume al metodei *printTest*, insa cu valoarea 20. Parametrul *test* din cadrul metodei *printTest* ascunde variabila de instanta *test*.

Metoda printTest apelata in metoda main afiseaza parametrul test cu valoarea 20 si nu variabila de instanta.

```
public class TestDomeniu  {
        int test = 10;
        void printTest(int test) {
                System.out.println("test = " + test);}
        public static void main (String args[]) {
                TestDomeniu st = new TestDomeniu();
                 st.printTest(20);
        }
}
```

```
1  public class TestDomeniu  {
2      int test = 10;
3      void printTest(int test)
4      {
5          System.out.println("test = " + test);
6      }
7      public static void main (String args[]) {
8          TestDomeniu st = new TestDomeniu();
9          st.printTest(20);
10     }
11 }
12
```

Result

CPU Time: 0.16 sec(s), Memory: 33724 kilobyte(s)

```
test = 20
```

Se poate evita aceasta eroare folosind **referinta this** ca o referinta la obiectul curent. Astfel, *this.test* refera variabila de instanta si numele simplu *test* refera parametrul metodei printTest. Programul de mai sus se poate, astfel, modifica (TestDomeniuThis.java) pentru a afisa valoarea 10 a variabilei de instanta si nu valoarea parametrului.

```
public class TestDomeniuThis  {
        int test = 10;
        void printTest(int test)  {
                System.out.println("test = " + this.test);
        }
        public static void main (String args[]) {
                TestDomeniuThis st = new TestDomeniuThis();
                 st.printTest(20);
        }
}
```

```
1  public class TestDomeniuThis {
2      int test = 10;
3      void printTest(int test)
4      {
5          System.out.println("test = " + this.test);
6      }
7      public static void main (String args[]) {
8          TestDomeniuThis st = new TestDomeniuThis();
9          st.printTest(20);
10     }
11 }
```

Result

CPU Time: 0.14 sec(s), Memory: 32696 kilobyte(s)

```
test = 10
```

2. Modificatori de acces

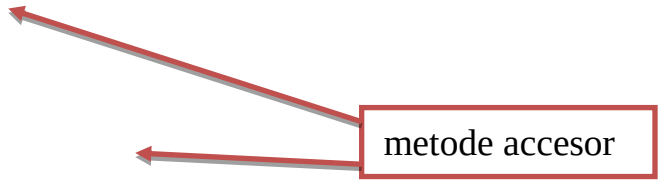
Urmatorul program (TestCerc.java) ilustreaza modul de folosire al variabilelor de instanta, precum si al metodelor de instanta.

In clasa Cerc variabila de instanta este raza care este vizibila numai in clasa in care a fost declarata (are modificatorul private).

De aceea, accesul la aceasta variabila (pentru citire si modificare) se face numai prin intermediul metodelor setRaza si getRaza care sunt publice.

Codul sursa al programului cu numele **TestCerc.java** este urmatorul:

```
class Cerc
{
        private double raza;
        public void setRaza(double r)
        {       raza = r;       }
        public double getRaza()
        {       return raza;    }
        public double arie()
        {       return Math.PI * raza * raza;  }
        public double lungime()
        {       return 2 * Math.PI * raza;     }
}
public class TestCerc
{
        public static void main(String[] args) {
        Cerc cerculMeu = new Cerc();
        cerculMeu.setRaza(10);
        System.out.println("Raza=" + cerculMeu.getRaza());
        System.out.println("Aria=" + cerculMeu.arie());
        System.out.println("Lungimea=" + cerculMeu.lungime());
         }
 }
```



```
1  class Cerc
2  {
3      private double raza;
4      public void setRaza(double r)
5      {   raza = r;   }
6      public double getRaza()
7      {   return raza;    }
8      public double arie()
9      {   return Math.PI * raza * raza;   }
10     public double lungime()
11     {   return 2 * Math.PI * raza;  }
12 }
13 public class TestCerc{
14     public static void main(String[] args) {
15         Cerc cerculMeu = new Cerc();
16         cerculMeu.setRaza(10);
17         System.out.println("Raza = " + cerculMeu.getRaza());
18         System.out.println("Aria = " + cerculMeu.arie());
19         System.out.println("Lungimea = " + cerculMeu.lungime());
20     }
21 }
22
```

Execute Mode, Version, Inputs & Arguments

JDK 17.0.1

CommandLine Arguments

Result

CPU Time: 0.15 sec(s), Memory: 33632 kilobyte(s)

```
Raza = 10.0
Aria = 314.1592653589793
Lungimea = 62.83185307179586
```

3. Metode constructor. Caracteristici

Metodele constructor au două caracteristici de bază:

a) *au întotdeauna acelaşi nume cu cel al clasei*
b) *nu returnează nici o valoare*

Urmatorul program (TestCercCons.java) prezinta clasa Cerc care are trei variabile de instanta:

- raza
- si coordonatele centrului cercului, x si y

Clasa Cerc foloseste o metoda constructor pentru a-si initializa variabilele de instanta pe baza argumentelor primite de new.

Codul sursa al programului cu numele **TestCercCons.java** este urmatorul:


```
class Cerc
{
        private double raza;
        private int x, y;

        Cerc(int coordX, int coordY, double lungRaza)
        {
                x = coordX;
                y = coordY;
                raza = lungRaza;
        }
        public void setRaza(double r) {     raza = r;       }
        public double getRaza()       {     return raza;    }
        public int getX()       {       return x;       }
        public int getY()       {       return y;       }

        public double arie()
        {       return Math.PI * raza * raza;  }
        public double lungime()
        {       return 2 * Math.PI * raza;      }
}

public class TestCercCons
{
  public static void main(String[] args) {
        Cerc cerculMeu = new Cerc(3, 9, 20);
        System.out.println("Raza=" + cerculMeu.getRaza());
        System.out.println("Centrul cercului este in punctul: x= " + cerculMeu.getX() + "
y= " + cerculMeu.getY());
        System.out.println("Modificarea razei cercului");
        cerculMeu.setRaza(10);
        System.out.println("Raza=" + cerculMeu.getRaza());
        System.out.println("Aria=" + cerculMeu.arie());
        System.out.println("Lungimea=" + cerculMeu.lungime());
         }
 }
```

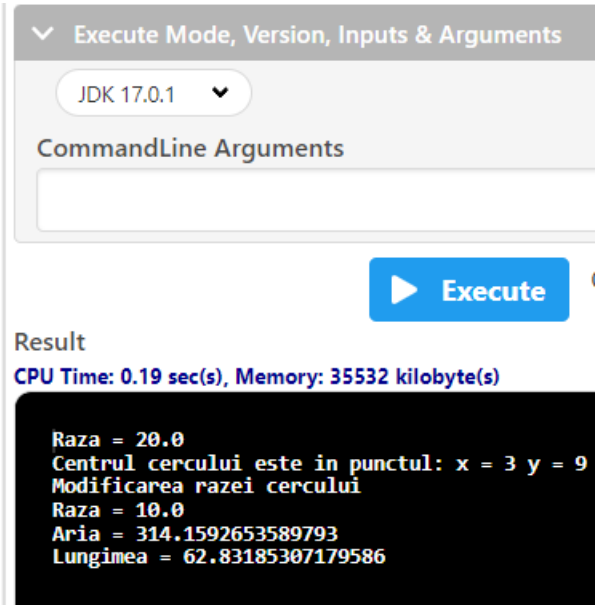
```java
class Cerc
{
    private double raza;
    private int x, y;

    Cerc(int coordX, int coordY, double lungRaza)
    {
        x = coordX;
        y = coordY;
        raza = lungRaza;
    }
    public void setRaza(double r) { raza = r;   }
    public double getRaza()     {   return raza;    }
    public int getX()   {   return x;   }
    public int getY()   {   return y;   }

    public double arie()
    {   return Math.PI * raza * raza;   }
    public double lungime()
    {   return 2 * Math.PI * raza;  }
}

public class TestCercCons
{
  public static void main(String[] args) {
    Cerc cerculMeu = new Cerc(3, 9, 20);
    System.out.println("Raza = " + cerculMeu.getRaza());
    System.out.println("Centrul cercului este in punctul: x = " + cerculMeu.getX() + " y = " + cerculMeu.getY());
    System.out.println("Modificarea razei cercului");
    cerculMeu.setRaza(10);
    System.out.println("Raza = " + cerculMeu.getRaza());
    System.out.println("Aria = " + cerculMeu.arie());
    System.out.println("Lungimea = " + cerculMeu.lungime());
     }
}

```



Supraîncărcarea metodelor constructor

Ca si metodele obisnuite, constructorii pot avea un numar diferit de parametri sau tipuri diferite pentru acestia desi au acelasi nume. Folosirea mai multor constructori cu acelasi nume dar cu parametrii care difera prin numar si/sau tip poarta denumirea de **supraincarcarea metodelor constructor**. Aceasta tehnica ne permite sa cream un obiect cu proprietatile dorite sau ne da posibilitatea sa cream obiecte care sa isi seteze proprietatile pornind de la date de intrare diferite.

*Cuvântul-cheie **this** pentru constructori*

Multe clase dispun de mai multi constructori care au un comportament similar. Putem folosi cuvantul-cheie **this** in cadrul unei metode constructor pentru a apela ceilalti constructori ai clasei.
Apelul unei metode constructor definita in clasa curenta, folosind **this** se face astfel:

this(<arg1>, <arg2>, <arg3>)

unde:
- <arg1>, <arg2>, <arg3> - specifica parametrii metodei constructor.

Intotdeauna apelul lui **this** trebuie sa fie prima instructiune din metoda constructor, celelalte instructiuni urmand dupa aceasta.

Urmatorul program (**TestCercCons.java**) prezinta clasa Cerc care are trei variabile de instanta:

- raza
- si coordonatele centrului cercului, x si y

Clasa Cerc foloseste doua metode constructor:

a) unul in care sunt initializate variabilele de instanta pe baza datelor furnizate de parametrii lui new,
b) si unul in care coordonatele x si y sunt preluate pe baza datelor furnizate de new dar variabila raza primeste valoarea prestabilita 1

Codul sursa al programului cu numele **TestCercCons.java** este urmatorul:

```
class Cerc
{
        private double raza;
        private int x, y;
        Cerc(int coordX, int coordY, double lungRaza) {
                x = coordX;
                y = coordY;
                raza = lungRaza;
        }
        Cerc(int coordX, int coordY) {
                this(coordX, coordY, 1);
        }
        public void setRaza(double r)
        {       raza = r; }
        public double getRaza()
        {       return raza; }
        public int getX()
        {       return x; }
        public int getY()
        {       return y; }
        public double arie()
        {       return Math.PI * raza * raza; }
        public double lungime()
        {       return 2 * Math.PI * raza;}
```

metode de tip constructor

```
}

public class TestCercCons
{
        public static void main(String[] args) {
         System.out.println("Crearea obiectului cu primul constructor");
         Cerc cerculMeu = new Cerc(3, 9, 20);
         System.out.println("Raza=" + cerculMeu.getRaza());
         System.out.println("Centrul cercului este in punctul: x= " +
         cerculMeu.getX() + " y= " + cerculMeu.getY());
        System.out.println("Crearea obiectului cu al doilea constructor");
        Cerc cerculMeu1 = new Cerc(3, 9);
        System.out.println("Raza=" + cerculMeu1.getRaza());
        System.out.println("Centrul cercului este in punctul: x= " + cerculMeu1.getX() +
" y= " + cerculMeu1.getY());
        System.out.println("Modificarea razei cercului");
        cerculMeu.setRaza(10);
        System.out.println("Raza=" + cerculMeu.getRaza());
        System.out.println("Aria=" + cerculMeu.arie());
        System.out.println("Lungimea=" + cerculMeu.lungime());}
 }
```

```
1  class Cerc
2  {
3      private double raza;
4      private int x, y;
5      Cerc(int coordX, int coordY, double lungRaza) {
6          x = coordX;
7          y = coordY;
8          raza = lungRaza;
9      }
10     Cerc(int coordX, int coordY) {
11         this(coordX, coordY, 1);
12     }
13     public void setRaza(double r)
14     {   raza = r; }
15     public double getRaza()
16     {   return raza; }
17     public int getX()
18     {   return x; }
19     public int getY()
20     {   return y; }
21     public double arie()
22     {   return Math.PI * raza * raza; }
23     public double lungime()
24     {   return 2 * Math.PI * raza;}
25 }
26
27 public class TestCercCons
28 {
29     public static void main(String[] args) {
30      System.out.println("Crearea obiectului cu primul constructor");
31      Cerc cerculMeu = new Cerc(3, 9, 20);
32      System.out.println("Raza=" + cerculMeu.getRaza());
33      System.out.println("Centrul cercului este in punctul: x= " +
34      cerculMeu.getX() + " y= " + cerculMeu.getY());
35     System.out.println("Crearea obiectului cu al doilea constructor");
36     Cerc cerculMeu1 = new Cerc(3, 9);
37     System.out.println("Raza=" + cerculMeu1.getRaza());
38     System.out.println("Centrul cercului este in punctul: x= " + cerculMeu1.getX() + " y= " + cerculMeu1.getY());
39     System.out.println("Modificarea razei cercului");
40     cerculMeu.setRaza(10);
41     System.out.println("Raza=" + cerculMeu.getRaza());
42     System.out.println("Aria=" + cerculMeu.arie());
43     System.out.println("Lungimea=" + cerculMeu.lungime());}
44 }
45
```

Result

CPU Time: 0.23 sec(s), Memory: 35456 kilobyte(s)

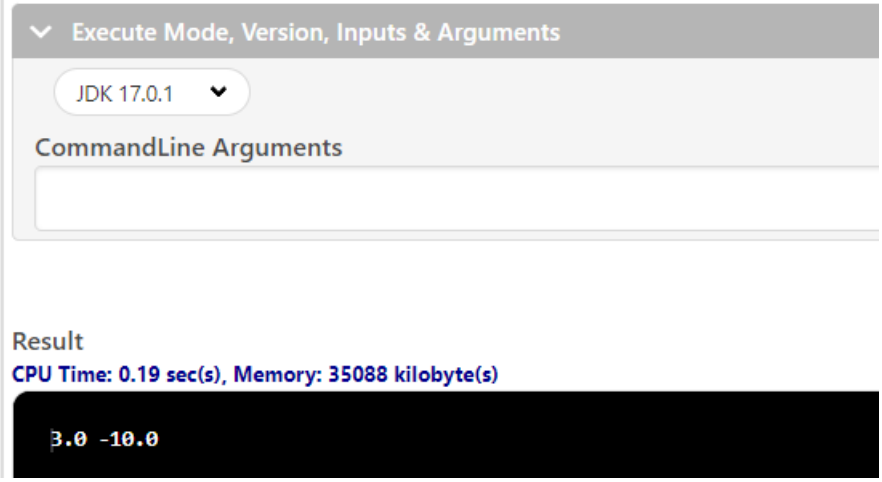
```
Crearea obiectului cu primul constructor
Raza=20.0
Centrul cercului este in punctul: x= 3 y= 9
Crearea obiectului cu al doilea constructor
Raza=1.0
Centrul cercului este in punctul: x= 3 y= 9
Modificarea razei cercului
Raza=10.0
Aria=314.1592653589793
Lungimea=62.83185307179586
```

4. Definirea unei clase Complex si a unei metode simple de afisare a datelor membru:

```
class Complex{
        // date membru
        double x,y;
        // metoda
        void afis()
        {       System.out.println(x+" "+y);  }
}
public class test1 {
        public static void main(String args[])
                {
                // declaram un numar complex
                Complex z1;                     // constructor implicit
                z1 = new Complex();             // Complex z1 = new Complex();
                z1.x=3;
                z1.y=-10;
                // afisam datele membru
                z1.afis();
                }
}
```

```java
class Complex{
    // date membru
    double x,y;
    // metoda
    void afis()
    {   System.out.println(x+" "+y);    }
}
public class test1 {
    public static void main(String args[])
        {
        // declaram un numar complex
        Complex z1;         // constructor implicit
        z1 = new Complex();     // Complex z1 = new Complex();
        z1.x=3;
        z1.y=-10;
        // afisam datele membru
        z1.afis();
        }
}

```



Avem, clasa **Complex**:

- clasa contine doua date membru, x si y, ambele de tip double
- clasa contine o metoda care are rolul de a afisa intr-un mod convenabil datele membru ale clasei.

Clasa **test1** – metoda **main()**:

- Complex z1 ; - variabila z1 este de un tip Complex. Prin, urmare, ea poate retine o referinta (adresa) catre un obiect de tip Complex.
- z1=new Complex(); - se creeaza un obiect de tip Complex si referinta catre el este retinuta de z1;
- z1.x=3; z1.y=-10; - datelor membru x si y ale obiectului a carui referinta este retinuta de z1 li se atribuie, valorile 3, respectiv -10.
- z1.afis(); - se afiseaza numarul complex retinut de obiectul referit de z1;

Probleme propuse spre rezolvare

Lab9_1: Definiţi şi implementaţi clasa **Triunghi**, având ca date membre **Latura1, Latura2** şi **Latura3** şi ca funcţii membre: un constructor, **Aria** şi **Perimetrul**.
Indicatii:
Perimetrul unui triunghi este dat de relatia: Perimetrul = Latura1 + Latura2 + Latura3
Aria unui triunghi este data de relatia: Aria=sqrt(sp * (sp - Latura1) * (sp - Latura2) * (sp - Latura3)), unde am notat cu sp - semiperimetrul triunghiului.

Lab9_2: Definiţi şi implementaţi clasa **Cerc**, având ca dată membru **Raza**, un constructor şi ca funcţii membre **Aria** şi **Circumferinta**.
Indicatii:
Aria unui cerc este data de relatia: Aria = PI * raza * raza
Circumferinta unui cerc este data de relatia: Circumferinta = 2 * PI * raza

Lab9_3: Definiţi şi implementaţi clasa **Cilindru**, având ca date membre **Raza** şi **Inaltimea** cilindrului şi ca funcţii membre: un constructor, **Aria** şi **Volumul.**
Indicatii:
Aria unui cilindru este data de relatia: Aria = 2 * PI * raza * inaltimea
Volumul unui cilindru este data de relatia: Volumul = PI * raza * raza * inaltimea

Lab9_4: Definiţi şi implementaţi clasa **Sfera**, având ca dată membru **Raza** şi ca funcţii membre: un constructor, **Aria** şi **Volumul.**
Indicatii:
Aria unei sfere este data de relatia: Aria = 4 * PI * raza * raza
Volumul unei sfere este data de relatia: Volumul = 4 * PI * raza * raza * raza / 3

Lab9_5: Definiţi şi implementaţi clasa **Patrat**, având ca dată membru **Latura** şi ca funcţii membre: un constructor, **Aria** şi **Perimetrul.**
Indicatii:
Aria unui patrat este data de relatia: Aria = latura * latura
Perimetrul unui patrat este data de relatia: Perimetrul = 4 * latura

Bibliografie:

[1] http://www.pbinfo.ro *Descrierea site-ului:"www.pbinfo.ro îți propune să rezolvi probleme de informatică, cu evaluator automat. Știi pe loc dacă soluția ta este corectă sau dacă trebuie să mai lucrezi la ea. Problemele sunt grupate după programa de informatică pentru liceu. Dar nu trebuie să fii la liceu ca să rezolvi aceste probleme. Poți fi elev de gimnaziu, student, profesor sau pur și simplu pasionat de informatică. De fapt, trebuie doar să vrei!!"*
[2] https://www.runceanu.ro/adrian
[3] Adrian Runceanu *„Programarea şi utilizarea calculatoarelor"*, Editura Academica Brâncuşi din Târgu-Jiu, 2003, ISBN 973-8436-44-3
[4] Adrian Runceanu, Mihaela Runceanu, *„Noțiuni de programare – limbajul C++"*, Editura Academica Brâncuși din Târgu-Jiu, 2012, ISBN 978-973-144-550-2
[5] Adrian Runceanu, Mihaela Runceanu, *„Algoritmi implementati in limbajul C++. Volumul I – Algoritmi elementari"*, Editura Academica Brâncuși din Târgu Jiu, 2021, ISBN 978-606-9614-06-8