



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

Probleme rezolvate:

Scriti, 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());
    }
}
```

metode accesori

```
1 class Cerc
2 {
3     private double raza;
4     public void setRaza(double r)
5     {
6         raza = r;
7     }
8     public double getRaza()
9     {
10        return raza;
11    }
12    public double arie()
13    {
14        return Math.PI * raza * raza;
15    }
16    public double lungime()
17    {
18        return 2 * Math.PI * raza;
19    }
20 }
21
22 public class TestCerc{
23     public static void main(String[] args) {
24         Cerc cerculMeu = new Cerc();
25         cerculMeu.setRaza(10);
26         System.out.println("Raza = " + cerculMeu.getRaza());
27         System.out.println("Aria = " + cerculMeu.arie());
28         System.out.println("Lungimea = " + cerculMeu.lungime());
29     }
30 }
```

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 același 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:

metoda de tip constructor

```

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());
}
}
```

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

Execute Mode, Version, Inputs & Arguments

JDK 17.0.1

CommandLine Arguments

Execute

Result

CPU Time: 0.19 sec(s), Memory: 35532 kilobyte(s)

```
Raza = 20.0
Centrul cercului este in punctul: x = 3 y = 9
Modificarea razei cercului
Raza = 10.0
Aria = 314.1592653589793
Lungimea = 62.83185307179586
```

Supraîncărcarea metodelor constructor

Ca și metodele obișnuite, constructorii pot avea un număr diferit de parametri sau tipuri diferite pentru aceștia, deși au același nume. Folosirea mai multor constructori cu același nume dar cu parametri care diferă prin număr și/sau tip poartă denumirea de **supraincercarea metodelor constructor**. Această tehnică ne permite să creăm un obiect cu proprietățile dorite sau ne dă posibilitatea să creăm obiecte care să își seteze proprietățile pornind de la date de intrare diferite.

Cuvântul-cheie **this** pentru constructori

Multe clase dispun de mai mulți constructori care au un comportament similar. Putem folosi cuvântul-cheie **this** în cadrul unei metode constructor pentru a apela ceilalți constructori ai clasei.

Apelul unei metode constructor definite în clasa curentă, folosind **this** se face astfel:

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

unde:

- <arg1>, <arg2>, <arg3> - specifică parametrii metodei constructor.

Întotdeauna apelul lui **this** trebuie să fie prima instrucțiune din metoda constructor, celelalte instrucțiuni urmând după aceasta.

Următorul program (**TestCercCons.java**) prezintă clasa **Cerc** care are trei variabile de instanță:

- raza
- și coordonatele centrului cercului, x și y

Clasa **Cerc** folosește două metode constructor:

- a) unul în care sunt inițializate variabilele de instanță pe baza datelor furnizate de parametrii lui **new**,
- b) și unul în care coordonatele x și y sunt preluate pe baza datelor furnizate de **new** dar variabila **raza** primește valoarea prestabilită 1

Codul sursă al programului cu numele **TestCercCons.java** este următorul:

```
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;
    }
}
```

metode de tip constructor

```
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) {
        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();
    }
}

```

```

1 class Complex{
2     // date membru
3     double x,y;
4     // metoda
5     void afis()
6     {      System.out.println(x+" "+y); }
7 }
8 public class test1 {
9     public static void main(String args[])
10    {
11        // declaram un numar complex
12        Complex z1;           // constructor implicit
13        z1 = new Complex();   // Complex z1 = new Complex();
14        z1.x=3;
15        z1.y=-10;
16        // afisam datele membru
17        z1.afis();
18    }
19 }
20
21

```

Execute Mode, Version, Inputs & Arguments

JDK 17.0.1

CommandLine Arguments

Result

CPU Time: 0.19 sec(s), Memory: 35088 kilobyte(s)

3.0 -10.0

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: Definiți și implementați clasa **Triunghi**, având ca date membre **Latura1**, **Latura2** și **Latura3** și ca funcții membre: un constructor, **Aria** și **Perimetrul**.

Indicații:

Perimetrul unui triunghi este dat de relația: $\text{Perimetrul} = \text{Latura1} + \text{Latura2} + \text{Latura3}$
 Aria unui triunghi este data de relația: $\text{Aria} = \sqrt{(\text{sp} * (\text{sp} - \text{Latura1}) * (\text{sp} - \text{Latura2}) * (\text{sp} - \text{Latura3}))}$, unde am notat cu sp - semiperimetrul triunghiului.

Lab9_2: Definiți și implementați clasa **Cerc**, având ca dată membru **Raza**, un constructor și ca funcții membre **Aria** și **Circumferința**.

Indicații:

Aria unui cerc este data de relația: $\text{Aria} = \text{PI} * \text{raza} * \text{raza}$
 Circumferința unui cerc este data de relația: $\text{Circumferința} = 2 * \text{PI} * \text{raza}$

Lab9_3: Definiți și implementați clasa **Cilindru**, având ca date membre **Raza** și **Inalțimea** cilindrului și ca funcții membre: un constructor, **Aria** și **Volumul**.

Indicații:

Aria unui cilindru este data de relația: $\text{Aria} = 2 * \text{PI} * \text{raza} * \text{inalțimea}$
 Volumul unui cilindru este data de relația: $\text{Volumul} = \text{PI} * \text{raza} * \text{raza} * \text{inalțimea}$

Lab9_4: Definiți și implementați clasa **Sfera**, având ca dată membru **Raza** și ca funcții membre: un constructor, **Aria** și **Volumul**.

Indicații:

Aria unei sfere este data de relația: $\text{Aria} = 4 * \text{PI} * \text{raza} * \text{raza}$
 Volumul unei sfere este data de relația: $\text{Volumul} = 4 * \text{PI} * \text{raza} * \text{raza} * \text{raza} / 3$

Lab9_5: Definiți și implementați clasa **Patrat**, având ca dată membru **Latura** și ca funcții membre: un constructor, **Aria** și **Perimetrul**.

Indicații:

Aria unui patrat este data de relația: $\text{Aria} = \text{latura} * \text{latura}$
 Perimetrul unui patrat este data de relația: $\text{Perimetrul} = 4 * \text{latura}$

Lab9_6: Definiți și implementați clasa **Romb**, având ca date membre diagonala mare și diagonala mică și ca funcții membre un constructor, **Perimetrul** și **Aria**.

Indicații: Perimetrul unui romb este dat de formula $\text{Perimetrul} = 4 * \text{lungimea laturii}$, iar aria este dată de formula $\text{Aria} = (\text{diagonala mare} * \text{diagonala mică}) / 2$.

Lab9_7: Definiți și implementați clasa **Dreptunghi**, având ca date membre lungime și lățime și ca funcții membre un constructor, **Perimetrul** și **Aria**.

Indicații: Perimetrul unui dreptunghi este dat de formula $\text{Perimetrul} = 2 * (\text{lungime} + \text{lățime})$, iar aria este dată de formula $\text{Aria} = \text{lungime} * \text{lățime}$.

Lab9_8: Definiți și implementați clasa **Hexagon**, având ca dată membră latura și ca funcții membre un constructor, **Perimetrul** și **Aria**.

Indicații: Perimetrul unui hexagon regulat este dat de formula $\text{Perimetrul} = 6 * \text{latura}$, iar aria este dată de formula $\text{Aria} = (3 * \sqrt{3} * \text{latura}^2) / 2$.

Lab9_9: Definiți și implementați clasa **TriunghiIsoscel**, având ca date membre baza și cele două laturi egale, și ca funcții membre un constructor, **Perimetrul** și **Aria**.

Indicații: Perimetrul unui triunghi isoscel este dat de formula $\text{Perimetrul} = 2 * \text{latura egală} + \text{baza}$, iar aria este dată de formula $\text{Aria} = (\text{baza} * \text{înălțimea corespunzătoare bazei}) / 2$.

Lab9_10: Definiți și implementați clasa **Paralelogram**, având ca date membre latura și înălțimea, și ca funcții membre un constructor, **Perimetrul** și **Aria**.

Indicații: Perimetrul unui paralelogram este dat de formula $\text{Perimetrul} = 2 * (\text{latura1} + \text{latura2})$, iar aria este dată de formula $\text{Aria} = \text{latura} * \text{înălțimea}$.

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 și utilizarea calculatoarelor”, Editura Academica Brâncuși 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