Lucrare de laborator nr. 2
Clase elementare (II)

1. Scopul lucrării

În această lucrare se studiază clase elementare complete, care conţin atât variabile de instanţă cât şi metode. Se prezintă specificatorii de acces (*private* şi *public*) , modul în care se instanţiază obiectele dintr-o clasă cu ajutorul operatorului *new* , se prezintă noţiunea de constructor şi modul de apelare al metodelor publice din afara clasei. Se prezintă şi clasa de biblioteca *Random* şi metoda publică *nextInt()* din această clasă. Se arată şi modul de definire şi de folosire al constantelor în Java.

2. Breviar teroretic

a. Specificatori de acces

Prin membrii unei clase înţelegem variabile de instanţă şi metode. Accesul la aceşti membrii este controlat prin intermediul unor **specificatori / modificatori de acces**.

Avem următorii patru specificatori de acces :

1) **private**
2) **public**
3) **protected**
4) accesul default (implicit).

♦ Specificatorul de acces **private**:

Exemplu:

- pentru o variabilă de instanţă:

private int raza;

- pentru o metodă:

private int calcul( )

O variabilă de instanţă privată nu poate fi accesată decât din metode ale clasei respective.

O metodă privată nu poate fi apelată decât de o altă metodă din clasa respectivă.

Deci, membrii **private** ai unei clase nu pot fi accesaţi din afara clasei respective.

♦ Specificatorul de acces **public**:

Membrii **public** ai unei clase pot fi accesaţi atât din clasa respectivă, cât şi din orice altă clasă.

♦ Specificatorul de acces **protected**:

Este legat de relaţia de moştenire între clase. Va fi studiat ulterior.

♦ Specificatorul de acces implicit (atunci când în faţa declaraţiei unui membru nu este scris nici unul dintre cei trei specificatori de acces anteriori).

Exemplu:

int raza;

- membrii impliciţi pot fi accesaţi din clasa respectivă, dar şi din orice altă clasă care face parte din **acelaşi pachet** cu clasa respectivă.
- membrii cu accesul implicit nu pot fi accesaţi din alte clase ce fac parte din alte pachete diferite de pachetul clasei respective.

b. Constructorul

Constructorul este o metodă specială a unei clase care are acelaşi nume cu clasa, nu are în semnătură câmpul pentru tipul valorii returnate şi este apelată în mod automat (implicit) la instanţierea unui obiect din clasa respectivă. În mod tipic, un constructor face iniţializările variabilelor de instanţă.

O clasă poate să aibă în mod explicit definiţi zero, unul sau mai mulţi constructori (polimorfism parametric).

Exemplu:

Constructorul clasei *Random* are semnătura:

public Random()

c. Instanţierea unui obiect dintr-o clasă

Se face folosind operatorul **new.** Trebuie cunoscută de asemenea semnătura constructorului clasei (lista de parametrii).

Exemplu:

Pentru a instanţia obiectul r din clasa Random, procedăm astfel:

Random r=new Random( );

Pentru a instanţia obiectul c din clasa Cerc, trebuie să ştim semnătura constructorului clasei Cerc. Să presupunem că are următoarea semnătură:

public Cerc(int raza)

În acest caz pentru a instanţia un obiect Cerc cu raza 20, procedăm astfel:

Cerc c=new Cerc(20);

d. Modul de accesare pentru membrii publici ai unei clase

Din afara clasei respective, membrii publici sau cei cu accesul implicit, se accesează in doi paşi:

1. instanţiem un obiect din clasa respectivă:

2. accesăm membrul clasei cu sintaxa generală:

obiect.numeMembruDeAccesat.

Exemplu:

În clasa *Random*, al cărei constructor are semnătura:

public Random()

se află definită metoda *nextInt()* care are următoarea semnătură:

public int nextInt(int N)

Această metodă returnează un număr aleator în gama 0…N-1.

Pentru a genera un număr aleator în gama 0..99, folosind această metodă, mai întâi instanţiem un obiect r din clasa Random, astfel:

Random r=new Random();

Apoi apelăm metoda nextInt():

int nr=r.nextInt(100);

Exemplu:

În clasa *Cerc* al cărei constructor are semnătura:

public Cerc(int raza)

se află definită metoda *afisare()* care afişează raza cercului, şi care are următoarea semnătură:

public void afisare( )

Pentru a apela metoda publică *afisare(),* din afara clasei *Cerc* procedăm astfel:

Cerc c=new Cerc(20);
c.afisare( );

Exemplu:

În clasa A este definită variabila publică de instanţă x, astfel:

public int x;

Constructorul clasei A are semnătura:

public A( )

Pentru a scrie în variabila publică x valoarea 7, într-o metodă dintr-o altă clasă, procedăm astfel:

A a=new A( );

a.x=7;

Este greşită scrierea directă:

x=7; //GREŞIT

În general se recomandă ca variabilele de instanţă ale unei clase să fie declarate **private**, iar accesul la ele să se facă prin metode **public** de tipul:

set( ) – pentru a scrie în ele

get( ) – pentru a citi

Deşi se scrie mai mult cod, folosind nişte metode verificate, programul este mai fiabil.

e. Constante

Constantele sunt variabile care au un conţinut fix.

Constantele se declară în Java cu ajutorul cuvântului cheie **final**.

Exemplu:

final int NR_ELEVI=30;

Valoarea unei constante nu se modifică pe parcursul programului.

~~NR_ELEVI=31;~~ eroare la compilare

Se recomandă ca numele constantelor să fie scris cu litere mari.

3. Probleme rezolvate

Problema 1

Să se afişeze din câte încercări se generează trei numere aleatoare egale, în gama 0..19 . Se va folosi clasa Random, şi metoda nextInt().

```
import java.util.*;
class NrIncercari{
public static void main(String args[]){
        final int GAMA=20;//constanta simbolica
        Random r=new Random();
        int contor=0;
        for(;;){
                int a=r.nextInt(GAMA);
                int b=r.nextInt(GAMA);
                int c=r.nextInt(GAMA);
                contor++;
                if((a==b)&&(b==c))break;
        }
```

```
        System.out.println(contor);
}
}
```

Problema 2

Folosind clasa BigInteger să se calculeze 2^{1000}. Vom folosi metoda *pow( )* din această clasă. Metoda *pow( )* are semnătura:
public BigInteger pow(int exponent)

Obiectul prin intermediul căruia se apelează metoda, face parte din clasa *BigInteger* , şi constructorul clasei *BigInteger* are ca parametru baza puterii, având semnătura:

public BigInteger(int baza)

```
import java.math.*;
class CalculPutere
{
 public static void main(String args[])
 {
  BigInteger baza=new BigInteger("2");
  BigInteger rezultat=baza.pow(1000);
  System.out.println("rezultat="+s);
}
}
```

Problema 3

Să se construiască clasa Umbrela, ce are ca variabilă de instanţă privată, un număr întreg diametru, ce reprezintă diametrul unei umbrele. În această clasă avem ca metode:

- constructorul, ce face iniţializarea diametrului;
- metoda *set ()* ce setează diametrul
- metoda *get()* ce returnează valoarea diametrului
- metoda *esteMare()*, ce returnează *true* dacă diametrul umbrelei este mai mare ca 100;

Scrieţi şi o clasă de test pentru clasa Umbrela, în care se citeşte de la tastatură diametrul unei umbrele, se instanţiază un obiect Umbrela cu acest diametru şi se afişează dacă este mare.

```
import java.util.*;
class Umbrela
{
```

```
 private int diametru;
 public Umbrela(int x)
 {
   diametru=x;
 }
 public void set( int d )
 {
  diametru=d;
 }
 public int get( )
 {
  return diametru;
 }
 public boolean esteMare()
 {
        final int LIMITA=100;
        if(diametru>LIMITA)return true;
        else return false;
 }
}
class TestUmbrela
{
 public static void main (String args[])
 {
  Scanner sc=new Scanner(System.in);
  System.out.print("diametrul umbrelei = ");
  int d=sc.nextInt();
  Umbrela u=new Umbrela(d);
  boolean este=u.esteMare();
  if(este==true)System.out.println("este mare");
  else System.out.println("nu este mare");
  }
}
```

Problema 4

Să se construiască clasa Cerc, ce are ca variabilă de instanţă privată, un număr întreg raza, ce reprezintă raza unui cerc. În această clasă avem ca metode:

- constructorul, ce face iniţializarea razei;

- metoda *set ()* ce setează raza
- metoda *get()* ce returnează valoarea razei
- metoda *calculPerimetru()*, ce returnează perimetrul cercului;
- metoda *calculArie()*,ce returnează aria cercului;
- metoda *afisare()* ce afișează raza

Scrieţi şi o clasă de test pentru clasa Cerc, în care se instanţiază un obiect Cerc cu raza 3 şi i se calculează perimetrul şi aria.

```
class Cerc
{
 private int raza;
 public Cerc(int x)
 {
   raza=x;
 }
 public void set( int r )
 {
  raza=r;
 }
 public int get( )
 {
  return raza;
 }
 public double calculPerimetru()
 {
   return 2*Math.PI*raza;
 }
 public double calculArie()
 {
  return Math.PI*raza*raza;
 }
 public void afisare( )
 {
  System.out.println("raza="+raza);
 }

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

```
{
 Cerc c=new Cerc(3);
 System.out.println("Perimetru= "+c.calculPerimetru());
 System.out.println("Aria= "+c.calculArie());
 }
}
```

Problema 5

Să se construiască clasa *Dreptunghi*, ce are ca variabile de instanţă private, două numere întregi a şi b, ce reprezintă lungimile laturilor unui dreptunghi. În această clasă avem ca metode:

- constructorul, ce face iniţializările;
- metoda *calculPerimetru()*, ce returnează perimetrul dreptunghiului;
- metoda *calculArie()*, ce returnează aria dreptunghiului;
- metoda *estePatrat()*, ce returnează *true* dacă dreptunghiul este pătrat;
- metoda *suntEgale()*, ce are ca parametru un dreptunghi d şi scoate ca rezultat *true* dacă dreptunghiul curent (cel pentru care se apelează metoda) este egal cu dreptunghiul d.

Scriem şi o clasă de test pentru clasa *Dreptunghi*.

```
class Dreptunghi
{
 private int a;
 private int b;
 public Dreptunghi(int x,int y)
 {
   a=x;
   b=y;
 }
 public int calculPerimetru()
 {
   return 2*(a+b);
 }
 public int calculArie()
 {
   return a*b;
 }
```

```
 public boolean estePatrat()
 {
        if(a==b)return true;
        else return false;
 }

 public boolean suntEgale(Dreptunghi d)
 {
        if ((this.a==d.a)&&(this.b==d.b))return true;
        else return false;
 }
}
class TestDreptunghi
{
  public static void main (String args[])
  {
   Dreptunghi d=new Dreptunghi(5,7);
   System.out.println("Primetrul este "+d.calculPerimetru());
   System.out.println("Aria este "+d.calculArie());
   System.out.println("Dreptunghiul este patrat= "+d.estePatrat());
   Dreptunghi d1=new Dreptunghi(5,7);
   System.out.println("Sunt egale= "+d.suntEgale(d1));
  }
}
```

Problema 6

Să se construiască clasa *Unghi*, ce are ca variabilă de instanţă privată un număr întreg x, măsura în grade a unui unghi, şi ca metode:

- constructorul;
- *suntComplementare()*, ce are ca parametru un alt unghi u, şi care returnează *true* dacă unghiul u este complementar cu unghiul curent;
- *conversieRadiani()*, ce returnează valoarea exprimată în radiani a unghiului curent x.

Scriem şi o clasă de test pentru clasa Unghi.

```
class Unghi
{
 private int x;
 public Unghi(int x)
```

```
{
  this.x=x;
 }
 public boolean suntComplementare(Unghi u)
 {
  if(this.x+u.x==90) return true;
  else return false;
 }
 public double conversieRadiani()
 {
  return (Math.PI*x)/180;
 }

}
class TestUnghi
{
  public static void main (String args[])
  {
   Unghi a=new Unghi(30);
   System.out.println("Radiani= "+a.conversieRadiani());
   System.out.println("sunt  complementare="
                          +a.suntComplementare(new Unghi(60)));
  }
}
```

Problema 7

Să se construiască clasa *Fractie,* ce are ca variabile de instanţă private de tip int, numărătorul şi numitorul unei fracţii, şi ca metode: *setNumarator()* ce setează valoarea numărătorului fracţiei cu o valoare dată ca parametru.

- *setNumitor()* ce setează valoarea numitorului fracţiei cu o valoare dată ca parametru. Se interzice valoarea 0 pentru numitor. Metoda returnează *false* dacă s-a încercat setarea numitorului la valoarea 0.
- *getNumarator()* care returnează valoarea numărătorului
- *getNumitor()* care returnează valoarea numitorului
- *inversa()* care returnează un obiect *Fractie,* inversa (răsturnata) fracţiei curente
- *suma()* ce are ca parametru un obiect *Fractie* f şi care

returnează un obiect *Fracţie*, suma fracţiei curente şi a fracţiei f. Fracţia rezultat, va fi simplificată.

- *afisare()* în care se afişează fracţia curentă.

Să se scrie şi o clasă de test pentru clasa *Fractie*.

```
import java.util.*;
class Fractie
{
 private int numarator;
 private int numitor;

 public void setNumarator( int x )
 {
  numarator=x;
 }
 public boolean setNumitor( int x )
 {
  if(x==0)return false;
  numitor=x;
  return true;
 }
 public int getNumarator( )
 {
  return numarator;
 }
 public int getNumitor( )
 {
  return numitor;
 }
 private int cmmdc(int a, int b)
 {
          while(a!=b)
            if(a>b)a=a-b;
            else b=b-a;
          return a;
 }
 public Fractie suma(Fractie f)
 {
          int numitorSuma=this.numitor*f.numitor;
          int numaratorSuma=
```

```
            this.numarator*f.numitor+this.numitor*f.numarator;
        //simplificare fractie cu cmmdc:
        int c=cmmdc(numitorSuma,numaratorSuma);
        numaratorSuma=numaratorSuma/c;
        numitorSuma=numitorSuma/c;
        Fractie rezultat=new Fractie();
        rezultat.setNumarator(numaratorSuma);
        rezultat.setNumitor(numitorSuma);
        return  rezultat;
 }
 public Fractie inversa()
 {
        if(numarator==0)return null; //nu se poate inversa
        Fractie rezultat=new Fractie();
        rezultat.setNumarator(numitor);
        rezultat.setNumitor(numarator);
        return  rezultat;
 }
 public void afisare()
 {
        System.out.println(numarator+"/"+numitor);
 }
}
class TestFractie
{
 public static void main (String args[])
 {
  Fractie f1=new Fractie();
  f1.setNumarator(5);
  f1.setNumitor(6);
  Fractie f=f1.inversa();
  System.out.println("Fractia inversata:");
  f.afisare();
  Fractie f2=new Fractie();f2.setNumarator(1); f2.setNumitor(2);
  Fractie fSum=f1.suma(f2);
  System.out.println("suma celor doua fractii:");
  fSum.afisare();
  }
}
```

4. Probleme propuse

Problema 1

Să se scrie clasa *Cerc* ce are ca variabile de instanţă pe lângă raza şi coordonatele centrului x0 şi y0. În această clasă avem ca metode:

- constructorul, ce face iniţializarea razei și a coordonatelor
- metoda *setR()* ce setează raza
- metoda *setX0()* ce setează coordonata x0
- metoda *setY0()* ce setează coordonata y0
- metoda *getR()* ce returnează valoarea razei
- metoda *getX0()* ce returnează valoarea x0
- metoda *getY0()* ce returnează valoarea y0
- metoda *calculArie()*,ce returnează aria cercului
- metoda *afisare()* ce afișează raza și coordonatele

Problema 2

Ce trebuie adăugat în clasa *Dreptunghi* astfel încât pentru un obiect *Dreptunghi* să-i putem afişa lungimile laturilor ?

Problema 3

Să se scrie clasa *Patrat* ce are ca variabilă de instanţă latura pătratului şi ca metode:

- constructorul, ce face iniţializările
- metoda *calculPerimetru()*, ce returnează perimetrul pătratului
- metoda *calculArie(),* ce returnează aria pătratului
- metoda *afisare(),* ce afişează lungimea laturii pătratului
- metoda *suntEgale(),* ce are ca parametru un pătrat p şi scoate ca rezultat *true* dacă pătratul curent (cel pentru care se apelează metoda) este egal cu pătratul p.

Scrieţi o clasă de test pentru clasa *Patrat*.

Problema 4

Să se dezvolte clasa *Timp* ce are ca variabile de instanţă trei numere întregi: h, min, sec (ore, minute, secunde). Ca metode:

- constructorul
- *conversieInSecunde()* ce returnează timpul curent, exprimat în secunde;

- *compara()*, ce are ca parametru un Timp t, şi care returnează 1 dacă timpul current este mai mare ca t, 0 dacă cei doi timpi sunt egali , şi -1 în caz contrar.

Scrieţi şi o clasă de test pentru clasa Timp.