Lucrare de laborator nr. 3
Clase elementare (III)

1. Scopul lucrării

În această lucrare se continuă studiul claselor elementare complete, care conţin atât variabile de instanţă cât şi metode. Se studiază de asemenea şi clase care conţin numai date (numai variabile de instanţă). Se prezintă modalitatea de a defini într-o clasă metode care returnează mai multe rezultate (prin formarea unei clase suplimentare ce are ca şi variabile publice de instanţă rezultatele de scos prin metodă). De asemenea se studiază modul de apelare al metodelor publice şi statice din afara clasei şi se arată construcţia unei clase ce conţine doar metode statice.

2. Breviar teroretic

Aşa cum s-a văzut în lucrarea precedentă de laborator, din afara unei clase , membrii publici ai clasei, se accesează in doi paşi:

1. instanţiem un obiect din clasa respectivă:

2. accesăm membrul clasei cu sintaxa generală:

obiect.numeMembruDeAccesat.

În cazul însă al metodelor publice dar care sunt şi statice (declarate cu specificatorul **static**) pentru a le apela din afara clasei în care sunt definite, nu se mai instanţiază obiect din clasa respectivă, ci le apelăm folosind sintaxa:

NumeClasă.numeMetoda( );

Exemplu:

Metoda *parseInt( )* din clasa *Integer*, are semnătura:

public static int parseInt(String s)

Este folosită pentru a converti un şir de caractere în valoarea numerică corespunzătoare (dacă este posibilă conversia). Astfel, pentru a converti şirul "12" în valoarea întreagă 12, metoda statică *parseInt()* se apelează astfel:

int nr=Integer.parseInt("12");

3. Probleme rezolvate

Problema 1

Să se dezvolte clasa *Complex*, ce are variabile de instanţă private două numere întregi *re* şi *im* (partea reală şi partea imaginară a unui număr complex) şi ca metode:

- constructorul ce face iniţializările;
- *modul()*, ce returnează modulul numărului complex;
- *suma()*, ce are ca parametru un număr complex c, prin care la numărul complex curent se adună numărul complex c (rezultatul se depune în numărul curent);
- *produs()*, ce are ca parametru un număr complex c, prin care în numărul complex curent se depune rezultatul înmulţirii dintre numărul complex curent şi numărul complex c;
- *getRe()*, ce returnează partea reală a numărului complex;
- *getIm()*, ce returnează partea imaginară a numărului complex;
- *egale()*, prin care se compară din punct de vedere al conţinutului, două obiecte Complex: obiectul curent şi obiectul dat ca parametru;

Scriem şi o clasă de test pentru clasa Complex.

```
class Complex
{
 private double re;
 private double im;
 public Complex(double x, double y)
 {
  re=x;
  im=y;
 }
 public double getRe()
 {
  return re;
 }
 public double getIm()
 {
  return im;
 }
```

```
 public double modul()
 {
  return Math.sqrt(re*re+im*im);
 }
 //adunarea nr. complex curent, cu un alt nr. complex, cu depunerea
 //rezultatului in numarul complex curent:
 public void suma(Complex c)
 {
  re=re+c.re;
  im=im+c.im;
 }
 //inmultirea nr. complex curent, cu un alt nr. complex, cu depunerea
 //rezultatului in numarul complex curent:
 public void produs(Complex c)
 {
  double re_1=re;  re=re*c.re-im*c.im;
  im=re_1*c.im+im*c.re_1;
 }

public boolean egale(Complex c)
 {
    if((re==c.re)&&(im==c.im))return true;
   return false;
 }

class TestComplex
{
public static void main(String args[])
{
  Complex c1=new Complex(1,1);
  System.out.println("Modulul este= "+c1.modul());
  Complex c2=new Complex(1,1);
  c1.suma(c2);
  System.out.println("partea reala a sumei  = "+c1.getRe());
  System.out.println("partea imaginara a sumei  = "+c1.getIm());
  Complex c3=new Complex(1,1);
  System.out.println("sunt egale: "+c2.egale(c3));
}
}
```

Problema 2

Să se implementeze în limbajul Java o listă simplu înlănţuită de numere întregi. Se vor scrie următoarele trei clase:

a. clasa *Nod* ce implementează un nod al unei liste simplu înlănţuite (conţine variabilele publice de instanţă *nr* ce memorează informaţia utilă din listă şi *next* – adresa următorului nod din listă)
b. clasa *Lista* ce are ca variabilă de instanţă *head*-ul listei, şi ca metode:
   - constructorul listei
   - *inserareLaInceput()* ce inserează un număr dat ca parametru la începutul listei
   - *nrPare()* ce returnează numărul de numere pare din listă
   - *afisare()* ce afişează numerele din listă
c. clasa *TestLista* ce conţine metoda *main()* folosită pentru a testa clasa *Lista* .

```
class Nod
{
        public int nr;
        public Nod next;
}
class Lista
{
 private Nod head;
 public Lista()
 {
   head=null;
 }
 public void inserareLaInceput(int x)
 {
        //este lista vida?
        if(head==null){
                head=new Nod();
                head.nr=x;
                head.next=null;
                return;
        }
        //lista nu este vida.
        Nod nodNou=new Nod();
        nodNou.nr=x;
```

```
        //Noul nod devine capul listei:
        nodNou.next=head;
        head=nodNou;
 }
 public int pare()
 {
    int contor=0;
    //Parcurgem lista de la inceput:
   Nod nodCrt=head;
   while(nodCrt!=null){
        if(nodCrt.nr % 2==0)contor++;
        //avansam in lista:
        nodCrt=nodCrt.next;
   }
   return contor;
 }
 public void afisare()
 {
   if(head==null){
        System.out.println("Lista este vida.");
        return;
   }
   //Parcurgem lista de la inceput:
   Nod nodCrt=head;
   while(nodCrt!=null){
        System.out.print(nodCrt.nr+" ");
        //avansam in lista:
        nodCrt=nodCrt.next;
   }
   System.out.println();
 }
}
class TestLista
{
 public static void main (String args[])
 {
  Lista ls= new Lista();
  for(int i=1;i<=10;i++)
        ls.inserareLaInceput(i);
  ls.afisare();
```

```
  System.out.println("Lista are "+ls.pare()+" numere pare.");
 }
}
```

Problema 3

Să se implementeze clasa *Calcule* ce conţine doar metode statice, şi anume:

- metoda *afisareDivizori()* ce are ca parametru un număr întreg căruia îi afişează toţi divizorii
- metoda *nrDivizori( )* ce are ca parametru un nmăr întreg şi care returnează numărul total de divizori ai acestuia
- metoda *estePrim( )* ce are ca parametru un nmăr întreg şi care returnează *true* dacă numărul este prim.

```
import java.util.*;
class Calcule
{
 public static void afisareDivizori(int nr)
 {
  if(nr==1){
   System.out.println("1");
   return;
  }
  //orice nr diferit de 1, are doi divizori: pe 1 si pe el insusi.
  System.out.print("1 ");
  for(int i=2;i<=nr/2;i++)
   if(nr%i==0)System.out.print(i+" ");
  System.out.println(nr);
 }
 public static int nrDivizori(int nr)
 {
  if(nr==1)return 1;
  //orice nr diferit de 1, are doi divizori: pe 1 si pe el insusi:
  int contor=2;
  for(int i=2;i<=nr/2;i++)
   if(nr%i==0)contor++;
  return contor;
 }
 public static boolean estePrim(int nr)
 {
```

```
  for(int i=2;i<=Math.sqrt(nr);i++)
   if(nr%i==0)return false;
  return true;
 }
}

class TestCalcule
{
 public static void main(String args[])
 {
   Scanner sc=new Scanner(System.in);
   System.out.print("a=");
   int a=sc.nextInt();
   Calcule.afisareDivizori(a);
   System.out.println("are "+Calcule.nrDivizori(a)+" divizori.");
   if(Calcule.estePrim(a)==true)
   System.out.println("este numar prim.");
   else System.out.println("nu este numar prim.");
 }
}
```

Problema 4

Pentru două numere întregi a şi b, să se calculeze minimul, maximul, şi să se afişeze dacă cele două numere sunt prime între ele, folosind o metodă separată ce are ca parametrii două numere întregi şi care returnează trei rezultate: minimul, maximul şi un rezultat de tip *boolean* ce specifică dacă numerele sunt prime între ele. Ca o metodă să poată să returneze N rezultate, trebuie construită o altă clasă ce are N variabile de instanţă (fiecare variabilă corespunde unui rezultat din cele ce trebuie returnate). Metoda ce trebuie să returneze mai multe rezultate de fapt va returna un obiect din clasa nou creată, ale cărui câmpuri au fost corespunzător completate.

```
import java.util.*;
class TreiRezultate
{
 public static void main(String args[])
 {
  Scanner sc=new Scanner(System.in);
  System.out.print("a=");
```

```
 int a=sc.nextInt();
 System.out.print("b=");
 int b=sc.nextInt();
 Triplet t=getRezultate(a,b);
 //afisam rezultatele:
 System.out.println("minim="+t.x);
 System.out.println("maxim="+t.y);
 if(t.z==true)System.out.println("sunt prime intre ele");
 else System.out.println("nu sunt prime intre ele");
 }
 //metoda ce returneaza cele trei rezultate:
 private static Triplet getRezultate(int a, int b)
 {
 int min;
 if(a<=b)min=a;
 else min=b;
 int max;
 if(a>=b)max=a;
 else max=b;
 boolean sunt=true;//presupunem ca sunt prime intre ele
 for(int i=2;i<=a;i++)
        if((a%i==0)&&(b%i==0)){
                sunt=false;
                break;
        }
 //formez un obiect din clasa Triplet:
 Triplet t=new Triplet();
 t.x=min;
 t.y=max;
 t.z=sunt;
 //returnam acest obiect:
 return t;
 }
}
class Triplet
{
 public int x;
 public int y;
 public boolean z;
}
```

Problema 5

Să se scrie clasa *Generare2Aleatoare* ce are metoda *main()* care apelează metoda *getDouaNumereDiferite()* ce are ca parametru un număr natural N și care returnează două numere aleatoare diferite, în gama 0...N-1.

```
âimport java.util.*;
class Generare2Aleatoare
{
 public static void main(String args[])
 {
  final int N=100;
  Dublet d=getDouaNumereDiferite(N);
  //afisam numerele:
  System.out.println(d.x);
  System.out.println(d.y);
 }
 //metoda ce returneaza doua numere aleatoare
//diferite in gama 0..N-1
 private static Dublet getDouaNumereDiferite(int N)
 {
  Random r=new Random();
  int n1=r.nextInt(N);//primul numar
  int n2;
  for(;;){
    n2=r.nextInt(N);
    if(n1!=n2)break;
  }
  //formez un obiect din clasa Dublet:
  Dublet d=new Dublet();
  d.x=n1;
  d.y=n2;
  return d;
 }
}
class Dublet
{
 public int x;
 public int y;
}
```

4. Probleme propuse

Problema 1

Să se adauge în clasa *Complex* metoda *complexConjugat()* ce returnează numărul complex conjugat al numărului complex curent (complex conjugatul numărului complex *a+ib* este *a-ib*).

Problema 2

Să se adauge în clasa Lista următoarele metode:

- metoda *media()* ce returnează media aritmetică a numerelor din lista simplu înlănţuită
- metoda *suntCrescatoare()* care returnează *true* dacă toate numerele din listă sunt în ordine crescătoare
- metoda *suntEgale()* ce returneză *true* dacă toate numerele din listă sunt egale între ele

Problema 3

Să se implementeze o listă dublu înlănţuită de numere întregi.

Problema 4

Să se adauge în clasa *Calcule* şi metodele statice :

- *cmmdc ()* ce are ca parametrii două numere întregi şi care returnează cel mai mare divizor comun al lor
- *suntPrimeIntreEle()* ce are ca parametrii două numere întregi şi care returnează *true* dacă cele două numere sunt prime între ele.

Problema 5

Pentru trei numere întregi a, b şi c, să se calculeze maximul, minimul şi media aritmetică a celor trei numere, folosind o metodă separată ce are ca parametrii trei numere întregi şi care returnează trei rezultate: maximul, minimul şi media aritmetică a celor trei numere.