Lucrare de laborator nr. 1
Clase elementare (I)

1. Scopul lucrării

În această lucrare se studiază clase elementare, ce conţin doar metode (nu au şi variabile de instanţă). Primele aplicaţii sunt clase ce au doar o singură metoda (metoda *main*)). Sunt apoi exemplificate şi clase ce conţin mai multe metode. Se explică şi se exemplifică noţiunea de polimorfism parametric. Se studiază de asemenea citirea numerelor întregi de la tastatură, folosind clasa Scanner.

2. Breviar teroretic

O clasă în general conţine atât variabile de instanţă (care modelează proprietăţile obiectului) cât şi metode (funcţii, ce modelează comportamentul obiectului). Variabilele de instanţă şi metodele sunt membrii clasei. În particular o clasă poate să conţină numai variabile de instanţă (fără metode) sau numai metode (fără variabile de instanţă). În această lucrare de laborator vom studia clase ce au doar metode (o singură metodă sau mai multe metode).

O aplicaţie java constă din una sau mai multe clase scrise de programator. Pentru a putea executa o aplicaţie java, trebuie să existe o singură clasă în aplicaţie, care să definească metoda *main().*
Antetul acestei metode este:
public static void main(String args[])
Dacă o clasă nu defineşte metoda *main,* ea nu poate fi executată. Poate fi doar compilată, în urma compilării se va obţine fişierul binar de octeţi ,ce are extensia .*class* .

În cazul în care trebuie să apelăm dintr-o metodă declarată cu specificatorul *static* (de exemplu din metoda *main*) o altă metodă definită în aceeaşi clasă , această metodă trebuie şi ea să fie declarată cu specificatorul *static*.

Clasele sunt organizate în pachete de clase. S-a adoptat soluţia cu organizarea claselor în pachete de clase în primul rând pentru

evitarea conflictului de nume de clase. Numele complet al unei clase este dat de *numePachet.numeClasă.*

Exemple de pachete de clase:

- javax.swing (clasele pentru grafica)
- java.lang (clasele de bază ale limbajului: String, Integer, System)
- java.util (clasa Random)
- java.io (pentru fişiere: FileOutputStream)
- java.sql (pentru baze de date)
- java.net (pentru clase de comunicaţii în reţea)

Într-o aplicaţie trebuie folosit numele complet al clasei, adică *Pachet.nume.*

Exemplu:

java.util.Random r = new java.util.Random();

Pentru a evita folosirea numelui complet al clasei, într-o aplicaţie, trebuie ca pachetul din care face parte clasa respectivă să fie în mod explicit *importat* la începutul programului:

import java.util. ;*

sau se poate importa în mod explicit doar clasa respectivă:

import java.util.Random;

În acestă situaţie folosim doar numele clasei , fără să-l mai prefixăm cu numele pachetului.

Exemplu:

Random r=new Random();

Pachetul *java.lang* este în mod implicit importat în orice aplicaţie java.

Polimorfismul parametric este un polimorfism la compilare. Acesta se referă la faptul că într-o clasă putem avea mai multe metode ce au acelaşi nume, dar definiţii (implementări) diferite. Metodele care au acelaşi nume diferă între ele prin lista de parametrii (şi evident şi prin implementare). Compilatorul ştie încă din faza de compilare ce versiune de metodă să apeleze, pe baza numărului diferit de parametrii sau pe baza tipurilor diferite de parametrii. Acest mecanism se mai cheamă *overloading* (supraîncărcare).

Pentru a citi numere întregi de la tastatură, folosim în această lucrare de laborator, metoda *nextInt()* a clasei de bibliotecă *Scanner,* clasă ce face parte din pachetul *java.util* . Mai întâi se instanțiază un obiect *Scanner*, ca în exemplul de mai jos:

Scanner sc=new Scanner(System.in);

Apoi se apelează metoda *nextInt( )* , astfel:

int nr=sc.nextInt();

3. Probleme rezolvate

Problema 1

Să se scrie o clasă Java ce conţine doar metoda *main()*, în care se va afişa pe ecran urmatorul text (din Biblie):

”Căutaţi mai întâi Împărăţia lui Dumnezeu!”.

```
class MaiIntai
{
  public static void main(String args[] )
  {
     System.out.println(”Cautati mai intai Imparatia lui Dumnezeu!”);
  }
}
```

Problema 2

Să se calculeze valoare constantei **pi**, pe baza formulei:

pi/4=1-1/3+1/5-1/7+1/9-...

Se vor lua N=3000 de termeni în această sumă. Se va afişa apoi şi valoarea mai exactă acestei constante (constanta definită în clasaMath: Math.PI).

```
class Pi
{
 public static void main(String args[])
 {
  final int N=3000;//nr. de termeni ce se aduna
  double pi=0;
   for(int i=0;i<N;i++)
     if(i%2==0)pi=pi+4.0/(2*i+1);
     else pi=pi-4.0/(2*i+1);
   System.out.println("PI calculat ca suma de "+N+" termeni: "+pi);
   System.out.println("Constanta PI din clasa Math: "+Math.PI);
  }
 }
```

Problema 3

Se citeşte un număr natural de la tastatură. Să se afişeze câţi divizori are.

```
import java.util.*;
class NrDivizori
{
 public static void main(String args[])
 {
  int a;
  Scanner sc=new Scanner(System.in);
  System.out.print("a=");
  a=sc.nextInt();
  int contor;//numarul de divizori
  contor=2; //orice nr natural are cel putin 2 divizori:
        // pe 1 si pe el insusi
  int i;
  for(i=2;i<=a/2;i++)
        if(a%i==0)contor++;
  System.out.println("nr divizori = "+contor);
 }
}
```

Problema 4

Se citeşte un număr natural a. Să se afişeze dacă este pătrat perfect sau nu.

```
import java.util.*;
class PatratPerfect
{
 public static void main(String args[])
 {
  int a;
  Scanner sc=new Scanner(System.in);
  System.out.print("a=");
  a=sc.nextInt();
  double radical=Math.sqrt(a);
  if((int)radical*(int)radical==a)
        System.out.println("este");
  else System.out.println("nu este");
 }
}
```

Problema 5

Se citesc de la tastatură coordonatele x si y pentru două puncte din plan (x1,y1,x2,y2). Să se calculeze distanţa dintre ele.

Se va folosi o metodă separată, apelată din metoda *main(),* ce are ca parametrii patru numere întregi x1, y1, x2, y2: coordonatele celor două puncte. Metoda returnează distanţa dintre ele.

```
import java.util.*;
class DistantaDouaPuncte
{
 public static void main(String args[])
 {
  int x1,y1;//coordonatele primului punct
  int x2, y2;
  //citirea de la tastatura:
  Scanner sc=new Scanner(System.in);
  System.out.print("x1: ");
  x1=sc.nextInt();
  System.out.print("y1: ");
  y1=sc.nextInt();
  System.out.print("x2: ");
  x2=sc.nextInt();
  System.out.print("y2: ");
  y2=sc.nextInt();
  double rezultat=distanta(x1,y1,x2,y2);
  System.out.println("distanta="+rezultat);
 }
 private static double distanta(int x1,int y1, int x2, int y2)
 {
  int dx=x1-x2;
  int dy=y1-y2;
  return Math.sqrt(dx*dx+dy*dy);
 }
}
```

Problema 6

Se citește de la tastatură un număr întreg N. Se citesc apoi N numere întregi. Să se calculeze maximul dintre ele.

```
import java.util.*;
```

```
class Maxim
{
 public static void main(String args[])
 {
  int N;
  Scanner sc=new Scanner(System.in);
  System.out.print("N=");
  N=sc.nextInt();
  int nr;//numarul curent citit
  System.out.print("nr=");
  nr=sc.nextInt();
  int max; //maximul cautat
  //Initializam maximul cu primul numar citit:
  max=nr;
  //Citim restul numerelor:
  int i;
  for(i=2;i<=N;i++){
        System.out.print("nr=");
    nr=sc.nextInt();
    if(nr>max)max=nr;
  }
  System.out.println("maxim = "+max);
 }
}
```

Problema 7

Se citesc de la tastatură trei numere întregi a, b şi c. Să se caculeze maximul dintre ele.

Pentru a ilustra noţiunea de polimorfism parmetric, se vor scrie două metode ce au acelaşi nume *maxim()*, dar care diferă între ele prin numărul de parametrii. Prima returnează maximul dintre două numere întregi, a doua returnează maximul dintre trei numere întregi.

```
import java.util.*;
class Maxim3
{
 public static void main(String args[])
 {
  Scanner sc=new Scanner(System.in);
  int a,b,c;
```

```
 System.out.print("a=");
 a=sc.nextInt();
 System.out.print("b=");
 b=sc.nextInt();
 System.out.print("c=");
 c=sc.nextInt();
 int max=maxim(a,b,c);
 System.out.println("maxim = "+max);
}
private static int maxim(int x, int y)
{
        if(x>=y)return x;
        else return y;
}
private static int maxim(int x, int y, int z)
{
        //calculam maximul dintre x si y:
        int max1=maxim(x,y);
        //Maximul cerut este maximul dintre max1 si z:
        return maxim(max1,z);
}
}
```

Problema 8

Se citesc de la tastatură 2 numere întregi, a şi b, unde a >= 2. Să se calculeze câţi termeni din şirul lui Fibonacci sunt în [a,b].

Se va scrie o metodă ce are ca parametrii de intrare două numere întregi a şi b, capetele intervalului [a,b]. Funcţia returnează numărul de termeni din şirul lui Fibonacci care sunt în [a,b].

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

```
 b=sc.nextInt();
 System.out.println("Numar termeni="+nrTermeniFib(a,b));
 }
 private static int nrTermeniFib(int a, int b)
 {
 int ante=1;//termenul antepenultimul din sir
 int pen=1; //penultimul
 int crt=ante+pen;
 while(crt<a){
  //calculeaza urmatorul termen Fibonacci:
  ante=pen;
  pen=crt;
  crt=ante+pen;
 }
 int contor=0;
 while(crt<=b){
  contor++;
  ante=pen;
  pen=crt;
  crt=ante+pen;
 }
 return contor;
 }
}
```

Problema 9

Se citesc de la tastatură două numere întregi, a şi b, unde a >= 2. Să se calculeze care este cel mai mare număr prim din intervalul [a,b]. Dacă nu există, se va afişa că nu există.

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

```
 b=sc.nextInt();
 int rezultat=maxPrim(a,b);
 if(rezultat==-1)System.out.println("nu exista !");
 else System.out.println("Cel mai mare numar prim = "+rezultat);
}
private static int maxPrim(int a, int b)
{
 int i;
 for(i=b;i>=a;i--)
  if(estePrim(i))return i;
 return -1;//nu exista nici un numar prim in [a,b]
}
private static boolean estePrim(int nr)
{
 int i;
 for(i=2;i<=Math.sqrt(nr);i++)
  if(nr%i==0)return false;
 return true;
}
}
```

4. Probleme propuse

Problema 1

Se citesc de la tastatură doi timpi (pentru fiecare timp se citesc trei valori: numărul de ore, numărul de minute şi numărul de secunde). Se va afişa care timp este mai mare.

Pe lângă metoda *main()* se va scrie o metodă separată ce are ca parametrii şase numere întregi: h1, min1, sec1, h2, min2 şi sec2, unde h1 reprezintă numărul de ore pentru primul timp, min1 reprezintă numărul de minute, etc. Metoda returnează valoarea 1 dacă primul timp este mai mare, 0 dacă cei doi timpi sunt egali şi 2 dacă al doilea timp este mai mare.

Problema 2

Se citeşte de la tastatură un număr natural N. Se citesc apoi N numere întregi. Să se calculeze media lor aritmetică.

Problema 3

Se citeşte de la tastatură un număr natural N. Se citesc apoi N numere întregi. Să se calculeze şi afişeze dacă sunt toate egale între ele sau nu.

Problema 4

Se citesc de la tastatură două numere întregi, a şi b. Să se calculeze câte perechi de numere prime între ele, sunt în intervalul [a,b]. Se va scrie o metodă separată ce are ca parametrii de intrare două numere întregi x şi y. Metoda returnează *true* dacă x şi y sunt prime între ele (nu au decăt pe 1 ca divizor comun).

Problema 5

Se citeşte un număr întreg de la tastatură. Să se afişeze dacă este termen în şirul lui Fibonacci sau nu.

Problema 6

Se citesc de la tastatură trei numere. Pentru a ilustra noţiunea de polimorfism parametric, calculaţi **cmmdc** al primelor două numere şi **cmmdc** pentru toate trei, scriind două metode care au acelaşi nume: *cmmdc()* , dar care diferă prin numărul de parametrii: prima are doi, a doua are trei parametrii.