Lucrare de laborator nr. 4
Tablouri unidimensionale (Vectori)

1. Scopul lucrării

În această lucrare se studiază vectorii intrinseci, modul de instanţiere, modul de transmitere al acestor vectori ca parametrii într-o metodă şi se dau ca exemple atât vectori de numere de tip *int* (tip de date primitiv) cât şi vectori de obiecte.

2. Breviar teroretic

Vectorii intrinseci sunt tablouri unidimensionale.

Spre deosebire de limbajul C, în limbajul Java, vectorii intrinseci sunt obiecte, instanţe ale unei clase intrinsecă limbajului.

Fiind obiecte, vectorii intrinseci (pe scurt ii vom numi vectori) se instanţiază cu ajutorul operatorului *new*.

Numele vectorului, ca şi în limbajul C, este o referinţă către zona de memorie unde se memorează componentele vectorului.

Componentele vectorului se accesează prin indexare, ca şi în C.

Un vector, cu ajutorul operatorului *new*, se instanţiază conform sintaxei:

tip numeVector[ ]=new tip[dimensiune];

Exemplu:

int a[ ]=new int[10];

Componentele vectorului sunt:

a[0], a[1], ... ,a[9]

Un vector poate fi creat si prin listarea directa a valorilor lui initiale.

Exemplu:

```
int b[]={1,5,-1};
```

Aceasta instructiune, putea fi realizata echivalent, astfel:

```
int b[]=new int[3];
b[0]=1;
b[1]=5;
```

b[2]=-1;

OBSERVAȚII:

1. Spre deosebire de limbajul C, în Java, se verifică indexarea în afara graniţelor semnalându-se eroare de sintaxă.

2. Spre deosebire de limbajul C, dimensiunea vectorului poate să fie şi o variabilă, ce a fost iniţializată.

Exemplu:

int dim=10;

int a[ ]=new int[dim];

3. Atenţie la diferenţa între declararea unui vector şi crearea sau instanţierea lui.

Exemplu:

double b[ ]; // declarare vector

b=new double [10]; // Crearea obiectului vector:

Vectorii pot conține ca elemente , nu numai date primitive, ci și obiecte. Exemplu:

Integer v1[]=new Integer[3];

Sau:

```
Integer v2[]={new Integer(1),new Integer(3),
              new Integer(4)};
```

În clasa din care se instanţiază vectorii, este definită variabila de instanţă publică ***length*** de tip read-only (poate fi doar citită, nu şi modificată) în care, în mod automat, la instanţierea vectorului, se memorează dimensiunea vectorului.

Spre deosebire de limbajul C, unde într-o funcţie ce avea ca parametru un vector, trebuia dat ca parametru şi dimensiunea vectorului, în Java, dimensiunea vectorului nu mai este necesară ca parametru, deoarece dimensiunea vectorului este memorată în variabila publică ***length***.

3. Probleme rezolvate

Problema 1

Citim un număr natural N. Citim N numere întregi într-un vector a. Să calculăm şi afişăm maximul din vector.

```
import java.util.*;
class CalculMaxim
{
 public static void main(String args[ ])
```

```
{
 Scanner sc=new Scanner(System.in);
 System.out.print("N=");
 int N=sc.nextInt():
 int a[ ]=new int[N];
 int i;
 for (i=0; i<N; i++){
 System.out.print("nr=");
  a[i]=sc.nextInt();}
 int max=a[0];
 for (i=1; i<N; i++)
  if (a[i]>max) max=a[i];
    System.out.println(max);
 }
}
```

Problema 2

Se citeşte de la tastatură un număr natural N; se instanţiază un vector de N numere întregi. Să se completeze acest vector cu numere aleatoare în gama 0..N-1, cu condiţia ca fiecare număr din această gamă să apară o singură dată.

Vom folosi următorul algoritm: vom iniţializa vectorul cu numerele 0,1, .., N-1, date în această ordine. Apoi, aceste numere iniţiale, le vom comuta, poziţiile de comutare fiind generate aleator.

```
import java.util.*;
class InitNumAleatoare
{
 public static void main(String args[])
 {
  Scanner sc=new Scanner(System.in);
  System.out.print("N=");
  int N=sc.nextInt():
  int a[] = new int [N];
  int i;
  //se initializeaza vectorul cu numerele 0,1,...,N-1, in aceasta ordine:
   for(i=0;i<a.length;i++)
     a[i] = i;
  Random r = new Random();
  //se repeta de N ori:
```

```
 for(i=0;i<N;i++){
   //se genereaza doua numere aleatoare n1 si n2:
    int n1 = r.nextInt(N);
    int n2 = r.nextInt(N);
    //se comuta variabilele a[n1] si a[n2]:
    int aux = a[n1];
    a[n1] = a[n2];
    a[n2] = aux;
 }
 //Afisare vector generat:
 for (i=0;i<N;i++)
 System.out.println(a[i]);
}
}
```

Problema 3

Se citesc de la tastatură doi vectori a şi b, ce au aceeaşi dimensiune (dimensiunea lor comună, N, este cunoscută). Să se afişeze dacă vectorul b este o permutare a vectorului a.

Vom folosi următorul algoritm: vom sorta crescător cei doi vectori a şi b. apoi vom compara elementele de pe aceaşi poziţie din cei doi vectori.

Pentru a sorta crescător un vector vom folosi metoda statică *sort()* din clasa *Arrays* .

```
import javax.swing.*;
import java.util.*;
class SuntPermutari
{
 public static void main(String args[])
 {
  int N;
  Scanner sc=new Scanner(System.in);
  System.out.print("N=");
  int N=sc.nextInt():
  int a[] = new int [N];
  int b[] = new int [N];
  int i;
  System.out.println("Citire vector a: ");
  for(i=0;i<N;i++){
```

```
  System.out.print(”nr=”);
  a[i]=sc.nextInt();}
 System.out.println(”Citire vector b: ”);
 for(i=0;i<N;i++){
  System.out.print(”nr=”);
  a[i]=sc.nextInt();}
 //sortam cei doi vectori:
 Arrays.sort(a);
 Arrays.sort(b);
 //comparam cei doi vectori:
 boolean suntPerm=true;
 for(i=0;i<N;i++)
  if(a[i]!=b[i]){
     suntPerm=false;
     break;
  }
 if(suntPerm)System.out.println("sunt");
 else System.out.println("nu sunt");
}
}
```

Problema 4

Pentru un vector de numere întregi dat, să i se inverseze elementele, astfel: primul devine ultimul, al doilea devine penultimul, etc.

Exemplu: a={1,-6,8,3};

După inversare: a={3,8,-6,1}

Se va scrie o metodă separată ce are ca parametru vectorul şi în care acesta se inversează.

```
import java.util.*;
class Inversare
{
 public static void main(String args[])
 {
        Scanner sc=new Scanner(System.in);
        int N;//dimensiune vector
        System.out.print("N=");
        N=sc.nextInt();
        int a[]=new int[N];
```

```
        int i;
        //citire vector:
        for(i=0;i<N;i++){
                System.out.print("nr=");
                a[i]=sc.nextInt();
        }
        inversareVector(a);
        //afisare vector inversat:
        for(i=0;i<N;i++)
                System.out.print(a[i]+" ");
}

private static void inversareVector(int a[])
{
 //parcurgem vectorul a[] pana la mijlocul lui:
 int i;
 for(i=0;i<a.length/2;i++){
  //comutam pe a[i] cu simetricul lui fata de mijlocul vectorului:
  int aux=a[i];
  a[i]=a[a.length-i-1];
  a[a.length-i-1]=aux;
 }
}
}
```

Problema 5

Scrieţi o aplicaţie în care se implementează algoritmul de căutare binară a prezenţei unui număr x citit de la tastatură într-un vector sortat crescător. Vectorul se citeşte de la tastatură dar avem grijă ca să fie dat crescător.

```
import java.util.*;
class CautareBinara
{
 public static void main(String args[])
 {
  Scanner sc=new Scanner(System.in);
  int N;//dimensiune vector
  System.out.print("N=");
  N=sc.nextInt();
```

```
 int a[]=new int[N];
 int i;
 //citire vector (se introduce crecator!):
 for(i=0;i<N;i++){
  System.out.print("nr=");
    a[i]=sc.nextInt();
 }
  System.out.print("Dati numarul pe care il cautam in vector x=");
  int x=sc.nextInt();
 //Este prezent x in vectorul a[] ?
 if(estePrezent(x,a,0,N-1))System.out.println("Este prezent.");
 else System.out.println("Nu este prezent.");
}

private static boolean estePrezent(int x,int a[],int s,int d)
{
 //cazuri limita:
 if(s>d)return false;
 if(s==d){
   if(x==a[s])return true;else return false;}
 int m;
 m=(s+d)/2;
 if(x==a[m])return true;
 else if(x<a[m])return estePrezent(x,a,s,m-1);
 else return estePrezent(x,a,m+1,d);
}//estePrezent
}
```

Problema 6

Se citesc de la tastatură un număr natural N şi un număr natural p. Se citeşte o mulţime de N numere întregi într-un vector a. Să se afişeze toate grupele de p elemente luate din cele N, care conţin un singur număr par. În grupă nu contează ordinea elementelor. Se va folosi metoda backtracking, varianta recursivă.

```
import java.util.*;
class GenerareGrupeUnNumarPar
{
public static void main(String args[])
```

```
{
    int N,p;
    Scanner sc=new Scanner(System.in);
    System.out.print("N=");
    N=sc.nextInt();
    System.out.print("p=");
    p=sc.nextInt();
    int a[]=new int[N];
    int i;
    //citire multime:  (trebuie date valori diferite)
    for(i=0;i<N;i++){
        System.out.print("nr=");
        a[i]=sc.nextInt();
    }
    int X[]=new int[p];//vectorul solutie
    back(a,N,p,X,0);//genereaza pe X[0], recursiv
}
private static void back(int a[],int N, int p, int X[], int k)
{
        //da urmatoarea valoare disponibila lui X[k]:
        int i;
        if(k==p)verificareSiAfisare(X);
        else
          for(i=0;i<N;i++){
                X[k]=a[i];
                if(valid(X,k))back(a,N,p,X,k+1);
           }
}
private static boolean valid(int X[],int k)
{
        //componentele in grupa se iau in ordine crescatoare
        if(k==0)return true;
        if(X[k]>X[k-1])return true;
        else return false;
}
```

```
 private static void verificareSiAfisare(int X[])
 {
        //calculam cate numere pare sunt in grupa:
        int contor=0;
        int i;
        for(i=0;i<X.length;i++)
          if(X[i]%2==0)contor++;
        if(contor!=1)return;//nu e o grupa buna
        //afisare grupa:
        for(i=0;i<X.length;i++)
                System.out.print(X[i]+" ");
        System.out.println();
 }
}
```

Problema 7

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 *calculArie()*,ce returnează aria cercului;
- metoda *esteEgal()* ce compară cercul curent cu un alt cerc dat ca parametru. Returnează *true* dacă cele două obiecte *Cerc* comparate au aceeaşi rază.

Să scrie apoi o clasă de test în care citim N cercuri de la tastatură într-un vector de cercuri şi apoi afişăm dacă toate cercurile sunt diferite între ele sau nu,

```
import java.util.*;
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 calculArie()
 {
  return Math.PI*raza*raza;
 }
 public boolean esteEgal(Cerc c)
 {
        if(raza==c.raza)return true;
        else return false;
 }
}
class TestVectorCercuri
{
  public static void main (String args[])
  {
   Scanner sc=new Scanner(System.in);
   int N;//numarurl de cercuri
   System.out.print("N=");
   N=sc.nextInt();
   Cerc c[]=new Cerc[N];
   int i;
   //citire N cercuri in vector c[]:
   for(i=0;i<N;i++){
        System.out.print("raza cercului nr. "+(i+1)+" = ");
        int r=sc.nextInt();
        c[i]=new Cerc(r);
   }
   //Verificam daca toate cercurile sunt diferite intre ele:
   boolean rezultat=suntDiferite(c);
   if(rezultat==true)
    System.out.println("sunt toate diferite");
   else System.out.println("nu sunt toate diferite");
  }
```

```
 private static boolean suntDiferite(Cerc c[])
 {
        int i, j;
        for(i=0;i<c.length-1;i++)
          for(j=0;j<c.length;j++)
                //sunt cercurile c[i] si c[j] egale?
                if(c[i].esteEgal(c[j])==true)return false;
        return true;
 }
}
```

3. Probleme propuse

Problema 1

Citim un număr natural N. Citim N numere întregi într-un vector a. Să calculăm şi afişăm (folosind cinci metode separate):

- suma elementelor vectorului
- dacă toate numerele din vector sunt egale între ele
- dacă toate numerele din vector sunt diferite între ele
- câte numere pare sunt în vector
- dacă un număr x, citit de la tastatură, este prezent în vector (căutare liniară)

Problema 2

Citim un număr natural N. Citim N numere întregi într-un vector a. Să calculăm care este al doilea număr ca mărime din vector. Exemplu: dacă a={1,7, 8, 4, 6, 8}, atunci al doilea număr ca mărime din vector este 7.

Problema 3

Citim un număr natural N. Citim N numere întregi într-un vector a. Să calculăm şi afişăm toate perechile de numere prime între ele, din vector.

Problema 4

Se citeşte un număr natural N. Se citesc N numere naturale. Care cifră apare de cele mai multe ori în scrierea celor N numere ?
Exemplu:

Dacă N=3 şi numerele sunt : 77, 4234, 3337 , atunci cifra 3 apare de cele mai multe ori (de 4 ori).

Indicaţie: Se va folosi un vector auxiliar *nrAparitiiCifra[ ]* , ce are 10 componente. Astfel, în *nrAparitiiCifra[0]* se va memora numărul total de apariţii al cifrei 0, etc.

Problema 5

Citim un număr natural N. Citim N numere întregi într-un vector a. Să calculăm şi afişăm care este numărul par ce apare de cele mai multe ori în vector.

Exemplu:

Dacă a={1,4,5,1,4,1,6,8}, numărul par ce apare de cele mai multe ori este 4.

Indicaţie: vom crea un vector contor[ ] ce are N elemente, contor[i] memorează de câte ori apare elementul de pe poziţia i din vectorul a, dacă a[i] este număr par..

Problema 6

Să se construiască clasa *Complex* ce modelează un număr complex. Clasa va conţine şi metoda *modul( )* ce returnează modulul numărului complex. Să se scrie apoi o clasă separată în a cărei metodă *main( )* citim un vector de N obiecte *Complex* şi în care afişăm numărul complex ce are cel mai mare modul.