[Scrieţi pe foile Dvs. **Pe fiecare foaie scrieţi-vă numele, grupa şi codul variantei de subiect, încercuit, pe faţa foii în colţul din dreapta, sus**. Lăsaţi o margine de 2cm, la stânga pe faţa,respectiv la dreapta pe verso, pe fiecare foaie pentru capsare. Timp: 90 minute]

Jaguarul şi cârtiţele

Jocul descris aici se joacă pe o tablă dreptunghiulară, cu dimensiunile 9 rânduri × 11 coloane. Pe tablă se află un jaguar, figurat prin litera "J" şi trei cârtiţe, figurate prin litere "C". Jaguarul poate fi deplasat de un jucător uman cu unul sau două pătrate pe direcţiile E, SE, S, SW, W, NW, N sau NE prin comenzi date din consolă. Poziţiile iniţiale sunt data în fişierul `jaguar_cartite.txt`. Scopul jocului este ca jucătorul uman, prin mutări ale jaguarului, să-l aducă pe acesta în pătrate care conţin cârtiţe. Atunci când jaguarul ajunge în acelaşi pătrat cu o cârtiţă vizibilă, el o mănâncă, iar cârtiţa dispare din joc. Fiecare pas constă din două mutări: una a jucatorului uman şi una a programului. Mai întâi se execută mutarea jucătorului – adică se mută jaguarul. Apoi, o singură cârtiţă, aleasă la întâmplare, este mutată aleator cu un pătrat în una dintre cele 8 direcţii posibile (E, SE, S, SW, W, NW, N sau NE). Dacă cârtiţa ar ajunge pe un loc ocupat, mutarea nu are loc. Timp de un pas după mutare cârtiţa se află sub pământ (este invizibilă), iar jaguarul nu o vede (cârtiţa nu este în acest timp figurată pe tablă). Apoi ea devine din nou vizibilă. În figura 1 se prezintă o configuraţie iniţială a tablei de joc. Exemplul din textul figurii arată cum se codifică poziţia de pe tablă în fişierul `jaguar_cartite.txt`.
După citirea poziţiei iniţiale (un exemplu este în figura 1), precum şi după fiecare pas se va afişa situaţia de pe tablă *în mod text, pe consolă* aşa cum se vede în figurile 1 şi 2. Spaţiile libere sunt marcate cu '_'.

Mutările jucătorului uman se citesc cu metoda `citesteComanda` prezentată mai jos. Presupuneţi că este deja implementată clasa `Pozitie`, în pachetul `examen.poo`, clasă care conţine:

- metoda statică numită `citesteComanda()`, care returnează un tablou de 2 valori întregi reprezentând coordonate pe tablă. Prima valoare este numărul liniei, iar cea de a doua numărul coloanei unde se va muta. Astfel, pentru mutarea jaguarului aflat în linia 7 şi coloana 0 pe direcţia NE cu două poziţii, metoda `citesteComanda` va returna tabloul cu valorile 7, 2 (ceea ce corespunde numerotării de la 0 la 9 a liniilor şi coloanelor tablei, începând din colţul din stânga sus)
- metoda statică numită `citestePozitie(String s)`, care returnează un tablou de 4 valori întregi reprezentând coordonate în aceeaşi numerotare. Astfel, pentru şirul din fişierul de intrare din figura 1, valorile returnate în tablou vor fi: 7, 0, 7, 5, 1, 3, 2, 9.

*Desenaţi **diagrama de clase** şi scrieţi un program Java de sine stătător care să simuleze acest joc.*

```
_ _ C _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ C _
_ _ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _ _
J _ _ _ _ C _ _ _ _ _
_ _ _ _ _ _ _ _ _ _ _
```

Fig.1 Poziţie iniţială. Fişierul conţine:
`J(7,0) C(7,5) C(1,3) C(2,9)`

```
_ _ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ C _
_ _ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _ _
_ _ J _ _ C _ _ _ _ _
_ _ _ _ _ _ _ _ _ _ _
```

Fig.2 După deplasarea jaguarului în direcţia E cu 2 şi a cârtiţei de la (2,9) aleator (de exemplu spre SE cu 1). Observaţi că respectiva cârtiţă nu este momentan vizibilă.

```
/**
 * Write a description of class Joc here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
import java.awt.Point;
public class Joc
{
   public static void  afisare(Point[] p,int oi){
      Point p2=null;
      boolean ok=true;
      for(int i=0;i<10;i++){
         for(int j=0;j<10;j++)
         {
            p2=new Point(i,j);
            if(p2.equals(p[0]))System.out.print("L");
            else
            {
               ok=true;
               for(int k=1;k<oi+1;k++)
                  if(p2.equals(p[k])){System.out.print("O");
                     ok=false;}
               if(ok) System.out.print("+");
            }
         }
         System.out.println("");
      }
   }


   public static void main(){
      Vanatoare hunt=new Vanatoare();
      afisare(hunt.getPoint(),hunt.getOi());
      int[] aux=new int[2];
      while(!hunt.over()){
         do{
            aux=Pozitie.citesteComanda();
         }
         while(!hunt.validMove(aux));
         afisare(hunt.getPoint(),hunt.getOi());

      }
      System.out.println("Lup satul!");
   }
}




import java.io.FileReader;
import java.io.FileWriter;
```

```
import java.io.BufferedReader;
import java.io.PrintWriter;
import java.io.IOException;
import java.util.Scanner;
import java.util.InputMismatchException;

/**
 * Clasa care are functii de citire din fisier si citire din consola mutare
 */
import java.awt.Point;

public class Pozitie
{

   public Pozitie(){

   }

   public static Point[] citestePozitie(String s){
      Point[] p=new Point[5];
      int k=0;
      int c=0;
      int aux, auy;
      System.out.println("DA"+s);
      FileReader inputStream = null;

      try {
         inputStream = new FileReader(s);

         while(k<5){
            c = inputStream.read();
            if((c=='L')||(c=='O')){
               c = inputStream.read();//citeste (
               aux = inputStream.read()-'0';
               c = inputStream.read();//citeste ,
               auy = inputStream.read()-'0';
               p[k]=new Point(aux,auy);
               System.out.println(aux+" "+auy);
               k++;
               c = inputStream.read();//citeste )
               if(k<5) c = inputStream.read();//citeste spatiu
            }
         }
      }
      catch(IOException e)
      {System.out.println("llkjj6897867gjh");}
      finally {
         try{
            if (inputStream != null) {
               inputStream.close();
            }
         } catch(IOException e){}

         return p;
```

```
    }
}

public static int[] citesteComanda()
{
    int[] p=new int[2];
    System.out.println("Dati directia 1-N,2-NE,3-E,4-SE,5-S,6-Sv,7-V,8-NV si apasati Enter");
    Scanner in=new Scanner(System.in);
    int move=0;
    int dist=0;
    try{
        move=in.nextInt();
        //System.out.println(move);
    }catch(InputMismatchException e)
    {
        System.out.println("Not an int");
    }
    System.out.println("Dati deplasarea 1 sau 2 si apasati Enter");
    //in=new Scanner(System.in);
    try{
        dist=in.nextInt();
       // System.out.println(dist);
    }catch(InputMismatchException e)
    {
        System.out.println("Not an int");
    }
    int x=0;
    int y=0;
    switch(move){
        case 1: x=-dist;
        break;
        case 2: x=-dist;
        y=dist;
        break;
        case 3: y=dist;
        break;
        case 4: x=dist; y=dist;
        break;
        case 5: x=dist;
        break;
        case 6: x=dist; y=-dist;
        break;
        case 7: y=-dist;
        break;
        case 8: x=-dist; y=-dist;
        break;
        default :x=0; y=0;
        break;
    }
    p[0]=x;
    p[1]=y;
    return p;
}
```

```
    public static void main(){
        citestePozitie("Lupoi.txt");

        citesteComanda();
    }

}
```

```
/**
 * Write a description of class Vanatoare here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */

import java.awt.Point;
import java.util.Random;
import java.util.*;
public class Vanatoare
{
    ArrayList<Point> animals=new ArrayList<Point>();
    private Point[] locatii= new Point[5];
    private int nroi=4;
    public int getOi(){
        return nroi;
    }

    public Vanatoare(){
        locatii=Pozitie.citestePozitie("Lupoi.txt");
        for(int i=0;i<5;i++)
            animals.add(locatii[i]);
    }

    /*public void setPoint(Point[] c){
    for(int i=0;i<c.length;i++){
    //locatii[i]=new Point(c.x,c.y);
    }
    }*/

    public Point[]  getPoint(){
        //return locatii;
        locatii=animals.toArray(locatii);
        //return (Point)animals.toArray();
        return locatii;
    }

    public boolean over(){
        boolean ok=false;
        if (nroi==0) ok=true;
```

```
    return ok;
}

private double distanta(int xa, int ya, int xb, int yb){
    double d=0;
    d=(xa-xb)*(xa-xb)+(ya-yb)*(ya-yb);
    d=Math.sqrt(d);
    return d;

}

private void mutaOaie(){
    boolean ok =false;
    double dmin=20;
    int min=0;
    Point oaieApropiata=null;
    int ox=0;
    int oy=0;
    int move=0;
    Random r=new Random();
    Point auxp=null;
    Point auxlup=null;
    double d=0;
    try{
        auxlup=animals.get(0);
    } catch (IndexOutOfBoundsException e) {
        System.out.println("Index ERROR!");
    }
    int lupx=(int)auxlup.getX();
    int lupy=(int)auxlup.getY();

    dmin=20;
    min=0;
    for(int i=1;i<animals.size();i++)
    {
        try{
            auxp=animals.get(i);
        } catch (IndexOutOfBoundsException e) {
            System.out.println("Index ERROR!");
        }
        ox=(int)auxp.getX();
        oy=(int)auxp.getY();
        d=distanta(lupx,lupy,ox,oy);
        if(d<dmin){
            dmin=d;
            min=i;
            oaieApropiata=auxp;
        }
    }
    ok=false;
    int x=0;
    int y=0;
    while(!ok){//cattimp mutarea oii nu e valida
        move=r.nextInt(8);
```

```
        x=0; y=0;
        int dist=1;
        switch(move){
            case 1: x=-dist;
            break;
            case 2: x=-dist; y=dist;
            break;
            case 3: y=dist;
            break;
            case 4: x=dist; y=dist;
            break;
            case 5: x=dist;
            break;
            case 6: x=dist; y=-dist;
            break;
            case 7: y=-dist;
            break;
            case 8: x=-dist; y=-dist;
            break;
            default :x=1; y=1;
            break;
        }
        ox=(int)oaieApropiata.getX();
        oy=(int)oaieApropiata.getY();
        ox=ox+x;
        oy=oy+y;
        ok =true;
        if((ox<0)||(ox>9)||(oy<0)||(oy>9)) ok=false;//verificam daca mutarea oii e valida
    }
    Point oaieMutata=new Point(ox,oy);
    int altaOaie=animals.indexOf(oaieMutata);
    if(altaOaie<0)//nu am gasit oaie pe pozitia asta
    {
        animals.set(min,oaieMutata);
        System.out.println("Am mutat o oaie de la coordonatele "+(ox-x)+" "+(oy-y)+" la coordonatele "+ ox+" "+oy);
    }
    else{
        if(altaOaie==0){
            System.out.println("Am atacat lupul!");
        }
        else
        {
            nroi++;
            move=r.nextInt(100);
            animals.add(new Point(move/10, move%10));
            System.out.println("Am avut o oaie noua la coordonatele :"+ (move/10)+" "+(move%10));
        }
    }
}

public boolean validMove(int[] aux){
    boolean ok=true;
    Point auxp=animals.get(0);
    int lupx=(int)auxp.getX();
```

```
      int lupy=(int)auxp.getY();
      lupx=lupx+aux[0];
      lupy=lupy+aux[1];
      if((lupx<0)||(lupx>9)||(lupy<0)||(lupy>9)) ok=false;
      else{
         Point p2=new Point(lupx,lupy);
         int oaieMancata=animals.indexOf(p2);
         if(oaieMancata>0)//am gasit un element care corespunde
         {
            nroi--;
            animals.remove(oaieMancata);
            System.out.println("Am mancat o oaie!");
         }
         animals.set(0,new Point(lupx,lupy));
         if(animals.size()>1)  mutaOaie();
      }
      return ok;
   }
}
```