[Scrieţi pe foile Dvs. Pe fiecare foaie **scrieţi-vă numele, prenumele, grupa şi (B), pe faţa foii**. Lăsaţi o margine de 2cm, la stânga pe faţa,respectiv la dreapta pe verso, pe fiecare foaie pentru capsare.
Cu documentaţie. Timp: 90 minute]

Pe o tablă de şah se aşează trei figuri: un cal de culoare albă şi două turnuri de culoare neagră. Poziţiile lor iniţiale sunt data în fişierul `cal_turn.txt`. Scopul jocului este ca jucătorul, prin mutări ale calului, să evite pentru cât mai mult timp ca acesta să fie capturat de unul dintre cele două turnuri. 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 `cal_turn.txt`.

La fiecare pas, jucătorul uman trebuie să mute calul. Calul poate fi mutat în orice direcţie, în forma literei L (două poziţii pe orizontală + una pe verticală sau două pe verticală + una pe orizontală) – vezi figura 2. Dacă poate ajunge pe poziţia unuia dintre turnuri, el capturează turnul respectiv. După mutarea calului, programul mută turnurile aleator, potrivit regulii turnurilor. Dacă în urma mutării, unul dintre turnuri ajunge pe poziţia curentă a calului, jocul se termină, cu mesajul "**`Calul capturat după X paşi`**", unde x este numărul de mutări efectuate de cal. Jocul se termină şi dacă piesa cal a capturat *unul* dintre turnuri. Mesajul este "**`Bravo calule!`**".

După citirea poziţiei iniţiale (un exemplu este în figura 1) şi după fiecare pas se va afişa situaţia de pe tablă *în mod text*, marcând poziţia turnurilor cu '**`T`**', a calului cu '**`C`**' şi căsuţele negre cu '*' aşa cum se vede în figurile 3 şi 4.

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

- metoda statică numită **`citesteComanda()`**, care returnează un tablou de 4 valori întregi reprezentând coordonate pe tablă. Prima valoare este numărul liniei calului, a doua coloana, a treia numărul liniei unde se va muta, iar a patra numărul coloanei unde se va muta. Astfel, pentru mutarea **`c3 a2`**, metoda **`citesteComanda`** va returna tabloul cu valorile 5, 2, 6, 0 (ceea ce corespunde numerotării de la 0 la 7 a liniilor şi coloanelor tablei, începând din colţul din stânga sus)
- metoda statică numită **`citestePozitie(String s)`**, care returnează un tablou de 6 valori întregi reprezentând coordonate în aceeaşi numerotare. Astfel, pentru şirul din fişierul de intrare din figura 1, valorile returnate în tablou vor fi: 6, 1, 4, 0, 0, 3.

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

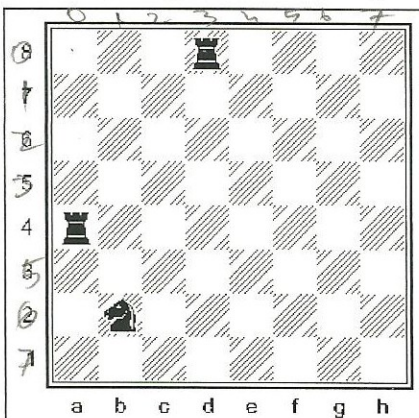

Fig. 1. Configuraţie iniţială
Conţinutul fişierului este:
`Cb2 Ta4 Td8`

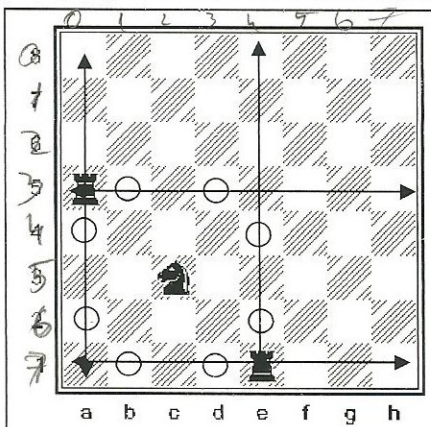

Fig.2. Posibilităţile de mutare ale calului şi turnurilor pentru o poziţie dată. Turnurile se pot muta doar pe lini sau coloane. Calul din poziţia **`c3`** se poate muta în oricare dintre pătratele marcate cu cercuri.

```
8   *   T   *   *
7 *   *   *   *
6   *   *   *   *
5 *   *   *   *
4 T *   *   *   *
3 *   *   *   *
2   C   *   *   *
1 *   *   *   *
  a b c d e f g h
```

Fig. 3. Tipărirea configuraţiei iniţiale

```
8   *   *   *   *
7 *   *   *   *
6   *   *   *   *
5 *   *   *   * T
4   *   *   *   *
3 *   *   *   *
2 C *   *   *   *
1 * T *   *   *
  a b c d e f g h
```

Fig. 4. Tipărirea configuraţiei rezultate după mutarea calului la `a2`, a turnului de la `e1` la `b1` şi a turnului de la `a5` la `h5`.

```
/**
 * Write a description of class Joc here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Joc
{
    public static void main(){
    Tabla sah=new Tabla();
    while(!sah.gameOver()){
    sah.view();
    sah.moveHorse();
    if(!sah.gameOver())sah.moveT();
    }
    sah.view();
    }
}



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

import java.util.*;
public class Tabla extends Pozitie
{
    private boolean gameO=false;
    public  int mutariCal=0;
    public Tabla(){
        super();
        gameO=false;
    }

    public void setPozitii(int []x)
    {
        super.pozitii=x;
    }

    public boolean validMove(int[] x){
        boolean ok=false;
        try{
            ok=true;
            for(int i=0;i<4;i++)
                if((x[i]<0)||(x[i]>7)) ok=false;
        }
        catch(ArrayIndexOutOfBoundsException e){System.out.println("Error array index!");}
       System.out.println(ok);
```

```
    return ok;
}

public boolean moveHorse(){
    boolean ok=true;
    mutariCal++;
    int[] ret=citesteComanda();
    ok=validMove(ret);
    if(ok){
        System.out.println("Mutam calul de la " +(ret[0])+" "+(ret[1])+" la "+(ret[2])+" "+(ret[3])+"!");
        int cx=ret[2];
        int cy=ret[3];
        int [] aux=citestePozitie();
        if(((cx==aux[2])&&(cy==aux[3]))||((cx==aux[4])&&(cy==aux[5]))){
            System.out.println("Bravo calutule ai castigat dupa "+mutariCal+" mutari!");
            gameO=true;
        }
        aux[0]=cx;
        aux[1]=cy;
        setPozitii(aux);
    }

    return ok;
}

public void moveT()
{
    Random r=new Random();
    int[] aux;
    //prima tura
    do{
        int c=r.nextInt(2);//sa vedem cum mutam tura, pe linie -0 sau coloana -1
        int d=r.nextInt(8);
        aux=new int[]{pozitii[2], pozitii[3],pozitii[2],pozitii[3]};
        aux[2+c]=d;
    }while (!validMove(aux));
    pozitii[2]=aux[2];
    pozitii[3]=aux[3];
    if((pozitii[0]==pozitii[2])&&(pozitii[1]==pozitii[3])){
        System.out.println("Calul a murit dupa "+mutariCal+" pasi!");
        gameO=true;
        pozitii[0]=9;
        pozitii[1]=9;
    }
    //a doua tura
    do{
        int c=r.nextInt(2);//sa vedem cum mutam tura, pe linie -0 sau coloana -1
        int d=r.nextInt(8);
        aux=new int[]{pozitii[4], pozitii[5],pozitii[4],pozitii[5]};
        aux[2+c]=d;
    }while (!validMove(aux));
    pozitii[4]=aux[2];
    pozitii[5]=aux[3];
    if((pozitii[0]==pozitii[4])&&(pozitii[1]==pozitii[5])){
```

```
            System.out.println("Calul a murit dupa "+mutariCal+" pasi!");
            gameO=true;
            pozitii[0]=9;
            pozitii[1]=9;
        }
    }

    public void view()
    {
        boolean ok;
        for(int i=0;i<8;i++){
            for(int j=0;j<8;j++)
            {
                ok=true;
                for(int k=0;k<6;k=k+2)
                    if((i==pozitii[k])&&(j==pozitii[k+1])){
                        if(ok){
                            if(k==0) System.out.print("L");
                            else System.out.print("T");
                        }
                        ok=false;
                }
                if((ok)&&((i+j)%2==0))System.out.print("#");
                else System.out.print(' ');
            }
            System.out.println();
        }

    }


    public boolean gameOver(){
        return gameO;
    }
}
```

```
/**
 * Write a description of class Pozitie here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
import java.util.*;

public class Pozitie
{
    protected int[] pozitii=new int[6];
    public Pozitie()
    {
```

```
        pozitii=IOClass.citireFisier();
    }



    public int[] citestePozitie(){
        return pozitii;
    }


    public int[] citesteComanda()
    {
        int ret[]=new int[4];
        String line="";
        Scanner in=new Scanner(System.in);
        System.out.println("Dati numarul mautarii dupa cele posibile pe tabla de sah, de la 1 la 8!");
        int move=in.nextInt();
        int x=0, y=0;
        switch(move){
            case 1: x=-1; y=-2;
            break;
            case 2: x=-2; y=-1;
            break;
            case 3: x=-2; y=1;
            break;
            case 4: x=-1; y=2;
            break;
            case 5: x=1; y=2;
            break;
            case 6: x=2; y=1;
            break;
            case 7: x=2; y=-1;
            break;
            case 8: x=1; y=-2;
            break;
            case 9: System.exit(0);
            break;
            default: x=0; y=0;
            break;
        }
        ret[0]=pozitii[0];
        ret[1]=pozitii[1];
        ret[2]=ret[0]+x;
        ret[3]=ret[1]+y;
        return ret;
    }
}




/**
```

```
 * Write a description of class IOClass here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
import java.io.*;
public class IOClass
{
   public static int[] citireFisier(){
      BufferedReader in=null;
      int[] ret=new int[6];
      String line="Ca1 Ca1 Ca1";
      try{
         in=new BufferedReader(new FileReader("cal_tura.txt"));

         line=in.readLine();
      }catch(IOException e){System.out.println("File not open for input!");}
      finally{
         try{
            if(in!=null)in.close();
         }catch(IOException e2){System.out.println("Error closing file!");}
      }
      ret[1]=line.charAt(1)-'a';
      ret[0]=line.charAt(2)-'1';
      ret[3]=line.charAt(5)-'a';
      ret[2]=line.charAt(6)-'1';
      ret[5]=line.charAt(9)-'a';
      ret[4]=line.charAt(10)-'1';
      for(int i=0;i<6;i++)
      System.out.println(ret[i]);
      return ret;
   }
   public static void main()
   {
    citireFisier();
   }
}
```