

MECI FOTBAL - LOVITURI DE DEPARTAJARE(PENALTY-URI)**1 Introducere - prezentarea problemei**

Se consideră un meci de fotbal aflându-se la momentul loviturilor de departajare, astfel că doar un jucător va putea executa o lovitură de departajare într-un moment de timp. Ceilalți vor aștepta la rând până când jucătorul curent va marca sau va rata lovitură de departajare. Un jucător poate urma la executarea unei lovituri de departajare numai în momentul în care cel dinaintea sa a executat lovitură.

1.1 Pas 1. Definire problemă

Condițiile sunt următoarele:

- Fiecare jucător își va aștepta rândul pentru a executa lovitură de departajare.
- Un jucător poate marca sau rata o lovitură de departajare.
- Un alt jucător poate executa o lovitură de departajare doar în momentul în care cel dinaintea sa a terminat.

2 Analiza problemei**2.1 Pas 2. Analiza cerințelor**

Fiecare jucător este un fir de execuție diferit.

Secvențe posibile:

- Primul jucător așteaptă începerea loviturilor.
Al doilea jucător așteaptă începerea loviturilor
Primul jucător execută lovitură de departajare.
Primul jucător marchează sau ratează lovitură.
Al doilea jucător execută lovitură de departajare.
Al doilea jucător marchează sau ratează lovitură.
- Primul jucător așteaptă începerea loviturilor.
Primul jucător execută lovitură de departajare.
Al doilea jucător așteaptă începerea loviturilor
Primul jucător marchează sau ratează lovitură.
Al doilea jucător execută lovitură de departajare.
Al doilea jucător marchează sau ratează lovitură.

Secvențe imposibile:

- Primul jucător așteaptă începerea loviturilor.
Al doilea jucător așteaptă începerea loviturilor
Primul jucător execută lovitura de departajare.
Al doilea jucător execută lovitura de departajare.. - nu îndeplinește condițiile 2 și 3
- Primul jucător execută lovitura de departajare.
Primul jucător marchează sau ratează lovitura.
Al doilea jucător execută lovitura de departajare.
Al doilea jucător marchează sau ratează lovitura. - nu îndeplinește condiția 1

3 Definirea structurii aplicației

Fiecare jucător reprezintă un fir de execuție diferit, care accesează o resursă comună, loviturile de departajare ale unui meci de fotbal. Am utilizat 2 clase, clasa Player, în care sunt caracterizați jucătorii ce vor executa loviturile de departajare (penalty-urile), și clasa Main, care simulează executarea celor n penalty-uri.

4 Definirea soluției în vederea implementării

4.1 Pas 4. Soluție de implementare

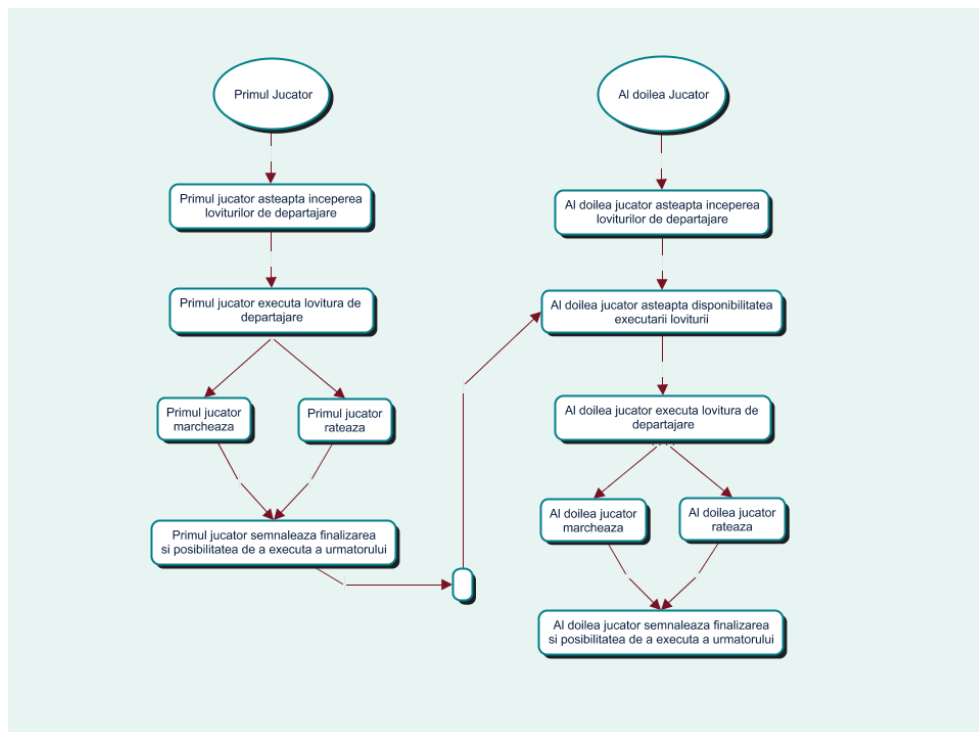


Figure 1: Soluție implementare - organigrame taskuri

Aleg mecanismele de sincronizare și comunicare între taskuri, utilizez lock din bibliotecă ”java.util.concurrent.locks.*”, prezentând organigramele taskurilor (e.g. în Fig. 1).

5 Implementarea soluției

5.1 Cod program

Inserare cod program: Clasa Player

```
package proiect;
import java.util.concurrent.locks.Lock;
import java.util.Random;

public class Player extends Thread {
    private Lock SoccerGame;
    private String nume;
    // GET si SET
    public String getNume(){
        return nume;
    }
    public void setNume(String nume){
        this.nume=nume;
    }

    // Constructor
    public Player(Lock SoccerGame, String nume){
        this.setNume(nume);
        this.SoccerGame=SoccerGame;
        this.start();
    }

    public void run(){

        try{
            System.out.println(getNume());
            SoccerGame.lock();
            System.out.println("\n");
            System.out.println(getNume() + " pune mingea pe punctul cu var pentru a executa. ");
            Thread.sleep(1500); // Timpul necesar executiei unei lovituri de departajare.
        } catch (InterruptedException e){
            System.err.println(e);
        }
        finally{
            Random rand = new Random();
            int x=rand.nextInt(2);
            String a=" ";

            if(x==1) a=" a marcat";
            else if(x==0) a=" a ratat";

            System.out.println(getNume() +a+".\n");
            SoccerGame.unlock();
        }
    }
}
```

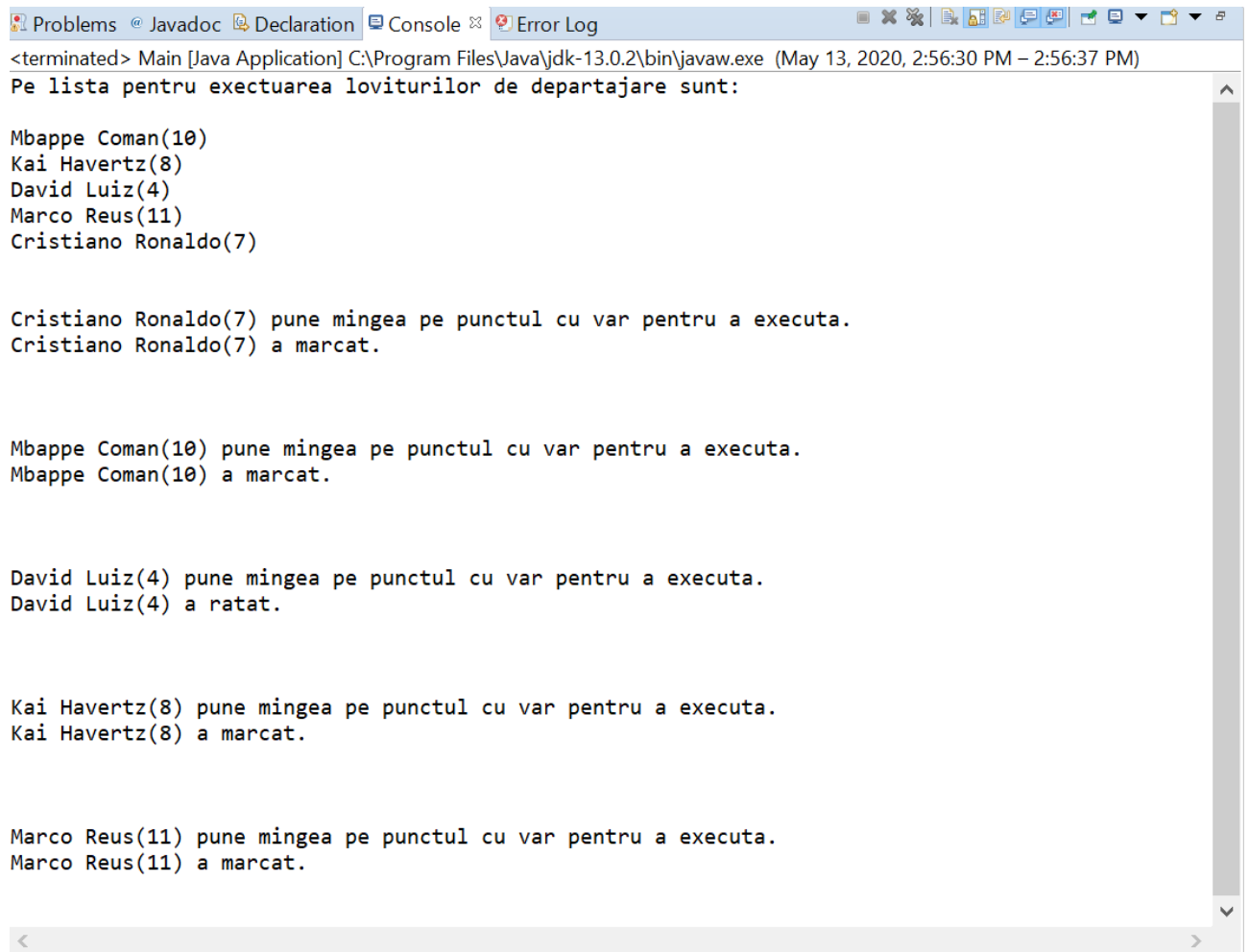
```
}  
}
```

Inserare cod program: Clasa Main

```
package proiect;  
import java.util.concurrent.locks.Lock;  
import java.util.concurrent.locks.ReentrantLock;  
  
public class Main {  
    public static void main(String []args){  
        Lock SoccerGame = new ReentrantLock(); // In cazul unui jucator ce va executa inca o data.  
  
        // Lista cu cei ce vor executa loviturile de departajare  
        System.out.println("Pe lista pentru exectuarea loviturilor de departajare sunt: \n");  
        new Player(SoccerGame,"Mbappe Coman(10)");  
        new Player(SoccerGame,"Kai Havertz(8)");  
        new Player(SoccerGame,"David Luiz(4)");  
        new Player(SoccerGame,"Marco Reus(11)");  
        new Player(SoccerGame,"Cristiano Ronaldo(7)");  
    }  
}
```

6 Testarea aplicației si validarea soluției propuse

Din rezultatul execuției se poate observa clar o afișare corespunzătoare pașilor precedenți , dar și a cerințelor aferente. Rând pe rând , fiecare jucător așteaptă începerea loviturilor de departajare, dar și finalizarea execuției jucătorului anterior pentru a înainta în vederea executării și marcării/ratării loviturii. Doar un jucător poate executa o lovitură într-un moment de timp deoarece vor exista mai mulți executanți, dar o singură poartă și un singur goalkeeper. (e.g. în Fig. 2)



```
<terminated> Main [Java Application] C:\Program Files\Java\jdk-13.0.2\bin\javaw.exe (May 13, 2020, 2:56:30 PM – 2:56:37 PM)
Pe lista pentru exectuarea loviturilor de departajare sunt:

Mbappe Coman(10)
Kai Havertz(8)
David Luiz(4)
Marco Reus(11)
Cristiano Ronaldo(7)

Cristiano Ronaldo(7) pune mingea pe punctul cu var pentru a executa.
Cristiano Ronaldo(7) a marcat.

Mbappe Coman(10) pune mingea pe punctul cu var pentru a executa.
Mbappe Coman(10) a marcat.

David Luiz(4) pune mingea pe punctul cu var pentru a executa.
David Luiz(4) a ratat.

Kai Havertz(8) pune mingea pe punctul cu var pentru a executa.
Kai Havertz(8) a marcat.

Marco Reus(11) pune mingea pe punctul cu var pentru a executa.
Marco Reus(11) a marcat.
```

Figure 2: Rezultatul a 5 lovituri de departajare