Esercizio 6.4

Impostazione:

1. Quali processi?

Giocatore

2. *Quale struttura per i processi*

Giocatore:

```
public void run()
{
    gc.noleggiaPalline(numPalline,nome);
    /*Simulazione del tempo di gioco*/
    gc.restituisciPalline(numPalline,nome);
}
```

3. Definizione della classe monitor

Dati:
GolfClubGiusto

Operazioni:

`void synchronized` **`noleggiaPalline`**`(int n, String nome)` /*metodo eseguito dal thread
Giocatore per noleggiare un numero precisato di palline*/

`void synchronized` **`restituisciPalline`**`(int n, String nome)` /*metodo eseguito dal thread
Giocatore per rilasciare le palline precedentemente noleggiate*/

Soluzione:


```
import java.util.Random;

public class Main {

  public static void main (String args[]){

      GolfClubGiusto gc;
      gc = new GolfClubGiusto();
      String nomi = "abcdefghijklnopqrstuvxyz";
      int nextname = 0;
      int numDaNoleggiare = 0;

      for(int i=0; i<14; i++)
      {
        Random x = new Random();
//Numero di palline (random) che il giocatore corrente desidera noleggiare
        numDaNoleggiare = x.nextInt(gc.NMAX-1);

        Thread t= new

      Giocatore(gc,numDaNoleggiare+1,nomi.substring(nextname,nextname+1));
```

/*Tramite l'invocazione del metodo start, viene messo in esecuzione il thread corrispondente a ciascun giocatore.*/

```
        t.start();
        nextname = (nextname+1)%nomi.length();
      }
  }
}
```

// La classe GolfClubGiusto è il thread che rappresenta la classe monitor da sincronizzare

```
public class GolfClubGiusto {

  final static int tempoDiGioco = 1000;
  final static int NMAX = 14;
  private  int disponibili = NMAX;
  private long turno = 0;
  private long prossimo = 0;

  void synchronized noleggiaPalline(int n, String nome) throws
InterruptedException
  {
      long mioTurno = turno;
      ++turno;
```

/* La seguente istruzione verifica le condizioni di sospensione per i giocatori(thread). Se si stanno richiedendo più palline di quelle disponibili oppure non è il turno del giocatore corrente, allora il giocatore corrente (istanza di thread) si sospende.*/

```
      while (n>disponibili || mioTurno != prossimo)
      {
        wait();
      }
      ++prossimo;
      disponibili -= n;
```

/*La NotifyAll cerca di risvegliare i giocatori per cui è arrivato il turno e per cui ci sono abbastanza palline disponibili. Questa notifyAll aumenta la concorrenza. */

```
      notifyAll();
  }

  void synchronized restituisciPalline(int n, String nome) throws
InterruptedException
  {
      disponibili += n;
```

/*Dopo aver restituito le palline, il giocatore risveglia gli altri thread che possono verificare il proprio turno e se ci sono abbastanza palline disponibili*/

```
      notifyAll();
  }
}

class Giocatore extends Thread
{
    GolfClub gc;
    String nome;
    int numPalline;

    Giocatore(GolfClub g, int n, String s)
    {
        gc = g;
        numPalline = n;
```

```
        nome = s;
    }

    public void run()
    {
      try
      {
        gc.noleggiaPalline(numPalline,nome);
//Simulazione del tempo di gioco
        Thread.sleep(gc.tempoDiGioco);
        gc.restituisciPalline(numPalline,nome);
      }
      catch (InterruptedException e){}
    }
}
```