Esercizio E6.1

Impostazione:

1. Quali processi?

Nave (che può essere di tre tipi: passeggeri, merci e petroliere)

2. Quale struttura per i processi

```
Nave:
  public void run()
  {
      CanSuez.richiestaAccesso(TipoNave);
      /* Passaggio della nave nel canale */
      CanSuez.uscita(TipoNave);
   }
```

3. Definizione della classe monitor

```
Dati:
 public Canale()
   {
     inTransito = new int [3]; /* Navi in transito nel canale */
     inAttesa = new int [3]; /* Navi in attesa di entrare nel canale */
   }
```

Operazioni:

`richiestaAccesso`: metodo eseguito dal thread `Nave` per richiedere l'accesso al `canale`.

`uscita`: metodo eseguito dal thread `Nave` per uscire dal `canale`.

`Condizione`: metodo invocato all'interno del metodo `richiestaAccesso` per verificare le condizioni di accesso del thread `Nave`.

Soluzione:

```
import java.io.*;
import java.lang.*;
import java.util.*;
import java.math.*;

public class Suez{
/* Il main è la classe di lancio del programma */

  public static void main(String[] args) {

    int NumPasseggeri,NumPetroliere,NumMerci,NumNaviTot,i;
    Canale CanaleSuez = new Canale();

    /* Il numero di ciascuna tipologia di nave, viene passato come argomento al main */
    NumPasseggeri = Integer.parseInt(args[0]);
    NumPetroliere = Integer.parseInt(args[1]);
    NumMerci = Integer.parseInt(args[2]);
    NumNaviTot = NumPasseggeri+NumPetroliere+NumMerci;
```

```
    Nave NaveVett [] = new Nave [NumNaviTot];

    /* Tramite l'invocazione del metodo start, viene messo in esecuzione il thread corrispondente a ciascuna nave.*/
     for(i=0;i<NumPasseggeri;i++){
        NaveVett[i]= new Nave(CanaleSuez,0);
        NaveVett[i].start();
      }
     for(i=NumPasseggeri;i<NumPasseggeri+NumPetroliere;i++)
      {
        NaveVett[i] = new Nave(CanaleSuez, 1);
        NaveVett[i].start();
      }
    for(i=NumPasseggeri+NumPetroliere;i<NumNaviTot;i++)
      {
        NaveVett[i] = new Nave(CanaleSuez, 2);
        NaveVett[i].start();
      }
     System.err.println("NumPasseggeri "+NumPasseggeri+"; NumPetroliere
     "+NumPetroliere+"; NumMerci "+NumMerci);
  }
}

/* La classe Nave è il thread che rappresenta la Nave*/
class Nave extends Thread {
  static boolean flag = true;
  Canale CanSuez;
  int TipoNave;

   /* Il costruttore inizializza la risorsa condivisa che è il canale e il tipo di nave */
   Nave (Canale c, int tipo)
  {
      CanSuez = c;     TipoNave = tipo;
  }

/* Il metodo run contiene il codice eseguito dal thread quando questo viene messo in
   esecuzione. Il metodo run viene invocato automaticamente all'esecuzione del metodo
   start.*/
   public void run(){
      try
      {
        CanSuez.richiestaAccesso(TipoNave);
      }
      catch(InterruptedException e){}

      /* Passaggio della nave nel canale */
      try
      {
        CanSuez.uscita(TipoNave);
      }
      catch(InterruptedException e){}
    }
  }
}
```

```
/* Canale è la classe monitor, che rappresenta la risorsa condivisa da sincronizzare */
class Canale {
  private  int inTransito [];
  private  int inAttesa [];
  /* Costruttore per inizializzare il monitor */
  public Canale()
  {
      int i;
      inTransito = new int [3];
      inAttesa = new int [3];
      for (i=0; i<3; i++) inTransito[i] = 0;
      for (i=0; i<3; i++) inAttesa[i] = 0;
   }
/* Il metodo è dichiarato synchronized perché richiede l'accesso mutuamente esclusivo alle variabili condivise
   del monitor. Inoltre contiene sia l'istruzione wait sia notifyAll che devono essere sempre inserite all'interno di
   una sezione synchronized */

   public synchronized void richiestaAccesso(int tipo)throws
       InterruptedException {
       /* La seguente istruzione verifica le condizioni di sospensione  delle navi (thread) */
       while (Condizione(tipo))
       {
         inAttesa[tipo]++;
           /* wait è l'istruzione che sospende il thread (la nave) che la invoca e libera il lock
associato all'istanza del monitor. */
         wait();
         inAttesa[tipo]--;
        }
      inTransito[tipo]++;

       /* notifyAll è l'istruzione che risveglia tutte le altre navi (thread) in attesa di poter passare nel canale */
       notifyAll();
  }

  public synchronized void uscita(int tipo) throws InterruptedException {
       inTransito[tipo]--;
       show();
      if(inTransito[tipo]==0) {notifyAll();}
  }

/* Il metodo Condizione restituisce un boolean che rappresenta il diritto di passaggio per il tipo di nave passato
come argomento */
  private boolean Condizione (int tipo)
  {
      if ((tipo == 0 && inTransito[1]!= 0) ||
          (tipo == 1 && inTransito[0]!= 0) ||
          (tipo == 1 && inTransito[2]!= 0) ||
          (tipo == 1 && inAttesa[0]!=0) ||
          (tipo == 2 && inTransito[1]!= 0) ||
          (tipo == 2 && inAttesa[1]!=0))
            {
              return true;
            }else
```

```
            {
              return false;
            }
  }
}
```