II Quadrimestre – Verifica di TPSIT
valida per lo scritto – a.s. 2022/2023
Classe 4I-C – 25 Maggio 2023

Tempo: 50 min.

1. Si gestisca il seguente problema di programmazione concorrente in Java con la definizione di thread.

   Il parlamento di Canberra (Australia) è periodicamente aperto al pubblico, possono accedere al massimo n persone alla volta, di queste persone n/2 possono prima visitare la camera dei rappresentanti mentre gli altri n/2 possono prima visitare la camera del senato. La visita può durare al massimo un tempo t per ogni zona del percorso, una volta che un gruppo ha visitato una camera passerà alla visita dell'altra, comunque ogni camera può essere visitata da un solo gruppo alla volta.
   Si gestisca la coda di m persone (da identificarsi con nome e cognome) che vogliono visitare il parlamento, tenendo presente che alcune hanno tempi contingentati di visita della città, per cui se l'attesa per entrare in parlamento eccede un tempo massimo tmax il turista rinuncia alla visita ed esce dalla coda di attesa. [7]

2. Date le classi indicate di seguito, stabilire il risultato dell'esecuzione della classe **Task2**, come stringa o comunque sequenza di caratteri stampati. [2]

```
class Adder extends Thread {

  private int loops;
  private Vector<Integer> buffer;  // Vector rappresenta
                                   // sequenze di interi

  public Adder( int loops, Vector<Integer> buffer ) {

    this.loops = loops;
    this.buffer = buffer;
  }

  public void run() {

    for ( int i=0; i<loops; i=i+1 ) {

      synchronized ( buffer ) {
        while ( buffer.size() < 2 ) {
          try {
            buffer.wait();
          } catch ( Exception e ) {}
        }
        // n: somma i primi elementi di buffer
        int n = buffer.get(0) + buffer.get(1);
        buffer.clear();  // buffer ritorna vuoto
        System.out.print( ""+n );
        buffer.notify();
    }}
    System.out.println();
  }

}  // Adder
```

```
class Provider extends Thread {

  private int[] stream;
  private Vector<Integer> buffer;

  public Provider( int[] stream, Vector<Integer> buffer ) {

    this.stream = stream;
    this.buffer = buffer;
  }

  public void run() {

    for ( int i=0; i<stream.length; i=i+1 ) {

      int x = stream[i];
      synchronized ( buffer ) {
        while ( buffer.size() == 2 ) {
          try {
            buffer.wait();
          } catch ( Exception e ) {}
        }
        System.out.print( "P" );
        buffer.add(x);  // aggiunge un elemento a buffer
        buffer.notify();
      }
    }
  }

}  // Provider
```

```
public class Task2 {

  public static void main( String[] args ) {

    int[] stream = new int[] { 1, 2, 3, 4, 3, 2 };

    // buffer inizialmente vuoto:
    Vector<Integer> buffer = new Vector<Integer>();
    Adder adder = new Adder( 3, buffer );
    Provider provider = new Provider( stream, buffer );

    adder.start();
    provider.start();
  }

}  // Task2
```

Griglia di valutazione

Elemento	Punteggio massimo	Punteggio attribuito
Class Camera (se utile)		
Gestione dati e costruttore	1	
Class Persona		
Gestione dati e costruttore – con passaggio della risorsa condivisa come parametro	0.5	
Metodo RUN – gestione acquisizione e rilascio risorsa e tempo di utilizzo risorsa	2	
Gestione lista persone/turisti in attesa di accedere	1,5	
Class Parlamento		
Gestione dati e costruttore	0,5	
Metodo synchronized di acquisizione risorsa con opportuni controlli	1	
Metodo synchronized di rilascio risorsa	0,5	
Task 2	2	
		+1
Voto proposto (espresso in decimi /10)		