Esame di Programmazione III, Programmazione in Rete e Laboratorio, Ist. di Programmazione in Rete e Laboratorio a.a. 2023/24 - Appello del 2 febbraio 2024

Esercizio 1 (14 punti)

Sia dato il seguente frammento di codice (la classe **Buffer definisce un buffer di interi – non ordinato - realizzato con un array**):

```
public class Buffer {
        private int[] buffer;
        private int numElem;

        public Buffer(int n) {
                buffer = new int[n];
                numElem = 0; // number of elements stored in the buffer
        }

        public synchronized boolean full() {
                return (numElem >= buffer.length);
        }

/*
 * Se il buffer è pieno, il metodo calcola la somma s dei valori memorizzati
 * nel buffer; poi svuota il buffer e restituisce il valore calcolato, s.
 * Altrimenti, il metodo sospende il thread che lo invoca fino a che il buffer diventa pieno.
*/
      … int getSumAndEmptyBuffer() throws InterruptedException { … }

/*
 * Se il buffer non è pieno, il metodo memorizza un nuovo valore nel primo elemento vuoto
 * del buffer.
 * Altrimenti, sospende il thread che lo invoca fino a che il buffer diventa non pieno.
*/
        …  void put (int value) throws InterruptedException { … }
}
```

Definire (solo) i metodi

- **… int getSumAndEmptyBuffer()** e
- **… void put(int x)**

in modo da garantire l'accesso in mutua esclusione a thread che realizzano consumatori e produttori dei valori memorizzati negli oggetti di tipo Buffer. In particolare:

- i thread di tipo produttore devono poter eseguire il metodo put(int) solo quando ci sono posti disponibili nel buffer (not full), altrimenti vengono messi in attesa che la condizione diventi vera;
- i thread di tipo consumatore devono poter eseguire il metodo getSumAndEmptyBuffer() solo quando il buffer è pieno (full), altrimenti vengono messi in attesa che la condizione diventi vera.

Esercizio 2 (6 punti)

Si sviluppino i seguenti punti:

- Si dica cosa si intende per polimorfismo nei linguaggi Object-Oriented.
- Si specifichi come, in Java, si utilizza il polimorfismo e su quale tipo di relazione si basa.
- Si faccia un semplice esempio di programma Java che utilizza il polimorfismo spiegando come il programma funziona.

Esercizio 3 (10 punti)

Si consideri il seguente codice. La classe Point rappresenta punti bidimensionali, con le loro coordinate.

Si sviluppi il metodo **public boolean equals(…)** della classe Point **facendo overriding del metodo omonimo di Object.**

Il metodo equals() deve restituire true se e solo se l'oggetto su cui viene invocato e quello passato per parametro hanno le stesse coordinate, ignorando il loro segno. Per esempio, un punto con coordinate (2, 3) viene considerato uguale sia a un punto con le stesse coordinate che a un punto con coordinate (-2, -3).

HELP: si ricorda che il metodo abs(int i) della classe Math restituisce il valore assoluto del numero intero (positivo o negativo) passato come parametro.

```
public class Point {
        private int x;
        private int y;

        public Point(int x, int y) {
                this.x = x;
                this.y = y;
        }

        public int getX() {
                return x;
        }

        public int getY() {
                return y;
        }
}
```

POSSIBILI SOLUZIONI

Esercizio 1

```
public synchronized int getSumAndEmptyBuffer()
                                        throws InterruptedException {
        while (!full())
                wait();
        int result = 0;
        for (int i=0; i< buffer.length; i++) {
                result = result+buffer[i];
                buffer[i] = 0;
                }
        numElem = 0;
        notifyAll( );
        return result;
}


public synchronized void put (int value) throws InterruptedException {
        while (full())
                wait();
        buffer[numElem] = value;
        numElem++;
        notifyAll();
}
```

Esercizio 3

```
public boolean equals(Object o) {
        if (o != null && this.getClass()==o.getClass()) {
                Point p = (Point)o;  // safe cast
                return (Math.abs(this.x) == Math.abs(p.x) && Math.abs(this.y) == Math.abs(p.y));
        }
        else return false;
        }
```