# Esame di Programmazione III, Progr. in Rete e Lab., Ist. di Progr. in Rete e Lab., A.A. 2022/23 -- 19 luglio 2023

**Esercizio 1 (10 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 numEven;
   private int numOdd;

   public Buffer(int n) {
         buffer = new int[n];
         numEven = 0;  // indica il numero di numeri pari inseriti nel buffer;
         numOdd = 0;   // indica il numero di numeri dispari inseriti nel buffer;
   }

   public synchronized boolean full() {
            return(numOdd + numeEven >= buffer.length);
   }

   ... int get() {...}

   ... void putEven(int x) {...}

  //  Assumiamo che esista anche un metodo putOdd(int x) che inserisce i numeri dispari nel buffer.
}
```

**Definire** (solo) **i metodi**

- **int get()** che preleva un elemento dal buffer, aggiornando i contatori dei numeri pari o dispari a seconda dell'elemento estratto;
- **void putEven(int x)** che inserisce l'elemento x nel buffer solo se il buffer non è pieno e x è pari;

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 putEven(int) solo quando ci sono posti disponibili nel buffer, altrimenti vengono messi in attesa che la condizione diventi vera;
- i thread di tipo consumatore devono poter eseguire il metodo get() solo quando ci sono elementi da prelevare nel buffer, altrimenti vengono messi in attesa che la condizione diventi vera.

**Esercizio 2 (14 punti)**

Sia dato il seguente estratto di codice:

```
interface I { public void metodoI(A arg); }

interface J extends I {
        public void metodoI(B arg);
        public void metodoJ(B arg);
        public static void metodoStatico() {
                System.out.println("CIAO!");
        }
}

class A implements I {
        public void metodoI(A arg) {
                System.out.println("metodo I_A: " +arg);
        }
        public String toString () {
                return "Sono un A!";
        }
}

class B extends A implements J {
        public void metodoI(B arg) {
                System.out.println("metodo I_B: " + arg);
        }
        public void metodoJ(B arg) {
                System.out.println("metodo J_B: " +arg);
        }
        public String toString() {
                return "Sono un B!";
        }
}
```

**Rispondere alle seguenti domande:**

1. Si consideri la sequenza di istruzioni:
   **A a = new A(); B b = new B(); a.metodoI(b); b.metodoI(a); b.metodoI(b);**
   Compila? Se no, perché? Se si, cosa stampa e perché?

2. Si consideri la sequenza di istruzioni:
   **A a = new A(); B b = new B(); a.metodoJ(b); b.metodoJ(a); J.metodoI(b);**
   Compila? Se no, perché? Se si, cosa stampa e perché?

3. Si consideri la sequenza di istruzioni:
   **B b = new B(); J.metodoStatico(); b.metodoJ(b);**
   Compila? Se no, perché? Se si, cosa stampa e perché?

**Esercizio 3 (6 punti)**

Si sviluppino i seguenti punti:

1. Definire cosa si intende per Thread o Lightweight process.
2. Definire il ciclo di vita di un Thread in java e le sue transizioni di stato (eventualmente con l'aiuto di un disegno).

**POSSIBILI SOLUZIONI**

**Esercizio 1**

```
  public synchronized int get() throws InterruptedException {
        while (numOdd+numEven <= 0)
              wait();
        int result = buffer[numOdd+numEven-1];
        if (result%2==0)
              numEven--;
        else numOdd--;
        notifyAll( );
        return result;
  }

  public synchronized void putEven (int value) throws InterruptedException {
        if (value%2==0) {
              while (full())
                    wait();
              buffer[numEven+numOdd] = value;
              numEven++;
          }
        notifyAll();
  }
}
```

**Esercizio 2**

1. Si, compila e stampa:
   metodo I_A: Sono un B!
   metodo I_A:  Sono un A!
   metodo I_B: Sono un  B!

2. Non compila perché contiene i seguenti errori:
   a.metodoJ(b); la classe A non offre il metodo metodoJ(B arg)
   b.metodoJ(a); metodoJ(B arg) richiede parametro di tipo B
   J.metodoI(b); J è un'interface per cui su di essa si possono solo invocare metodi statici e metodoI(B arg) non è statico.

3. Si, compila e stampa:
   CIAO!
   Metodo J_B: Sono un B!