Esame di Programmazione III, Progr. in Rete e Lab., Ist. di Progr. in Rete e Lab., A.A. 2022/23 -- 13 settembre 2023

Esercizio 1 (10 punti)

Si consideri il codice riportato sotto. Si modifichi il codice di Database per far sì che:

- Quando un thread scrittore (Writer) accede al Database per scrivere, nessun altro thread (Reader o Writer) possa accedervi né in scrittura, né in lettura.
- Quando nessuno scrittore accede al Database per scrivere, i thread lettori (Reader) possano accedervi in parallelo per leggere.

NB: i lettori possono leggere anche prima che gli scrittori scrivano sul Database.

```
class Esercizio {
        public static void main(String[] args) {
                int numReaders = 3; int numWriters = 2;
                Database db = new Database();
                for (int i = 0; i < numWriters; i++)   new Writer(db);
                for (int i = 0; i < numReaders; i++) new Reader(db);
        }
}

class Database {
        private int information = 0;
        public int read() {
                int result = information; return information;
        }
        public void write(int value) {
                information = value;
  }
}

class Reader implements Runnable {
        private Database db;
        private Random r;

        public Reader(Database d) {
                db = d;
                r = new Random();
                new Thread(this).start();
        }

        public void run() {
                try {Thread.sleep(r.nextInt(400));} catch(InterruptedException e) {}
                System.out.println(Thread.currentThread().getName() + ": " + db.read());
        }
}


class Writer implements Runnable {
        private Database db;
        private Random r;
```

```
        public Writer(Database d) {
                db = d;
                r = new Random();
                new Thread(this).start();
        }

        public void run() {
                try {Thread.sleep(r.nextInt(200));} catch(InterruptedException e) {}
                int x = r.nextInt(500);
                db.write(x);
        }
}
```

Esercizio 2 (10 punti)

Si consideri il codice riportato sotto:

1. Spiegare che operazione svolge l'oggetto "mio".
2. Modificare il codice per definire la classe MioAdapter come classe innestata anonima. *Non è necessario riscrivere tutto il codice, basta riportare la parte che viene modificata.*
3. Considerando che la classe WindowAdapter offre in tutto 10 metodi che prendono come unico parametro un WindowEvent (per esempio, void windowActivated(WindowEvent e), void windowClosed(WindowEvent e), etc.), è possibile definire la classe MioAdapter anche come lambda expression?

```
public class Esame extends JFrame {

  private Esame() {
    MioAdapter mio = new MioAdapter();
    addWindowListener(mio);
    setVisible(true);
  }

  public static void main(String[] args) {
    Esame es = new Esame();
  }
}

class MioAdapter extends WindowAdapter {
        public void windowClosing(WindowEvent e) {
                System.exit(0);
        }
}
```

Esercizio 3 (10 punti)

Si sviluppino i seguenti punti:

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

POSSIBILI SOLUZIONI

Esercizio 1

Soluzione 1

```
// Implementa il database per sincronizzare i lettori-scrittori usando il ReadWriteLock.
// I lettori possono leggere il contenuto del database anche prima che gli scrittori inizino a scrivere.

class Database {
        private int information = 0;
        private ReadWriteLock rwl = new ReentrantReadWriteLock();
        private Lock rl = rwl.readLock();
        private Lock wl = rwl.writeLock();

        public int read() {
                rl.lock();
                int result = information;
                rl.unlock();
                return information;
        }

  public void write(int value) {
        wl.lock();
        information = value;
        wl.unlock();
  }
}
```

Soluzione 2

```
/*
  Spezziamo read in due parti per permettere a più thread di lavorare in parallelo.
  La write no perché deve essere eseguita in mutua esclusione.
  */

class Database {
        private int numReaders = 0;

        public synchronized void startRead(int i) {
                numReaders++;
                System.out.println("Start reading " + i + "; ci sono " + numReaders + " lettori attivi");
        }

// quando l'ultimo lettore attivo finisce di leggere notifica i thread in stato di wait (scrittori).
        public synchronized void endRead(int i) {
                numReaders--;
                System.out.println("End reading " + i + "; ci sono " + numReaders + " lettori attivi");
                if(numReaders == 0)
```

```
                    /* NB: fondamentale fare notifyAll e non notify per prevenire deadlock! */
                notifyAll();
        }


// quando uno scrittore vede che ci sono lettori attivi si mette in stato di wait.
// quando ha finito di scrivere notifica i thread in stato di wait (scrittori).
// L'operazione di scrittura è gestita in sezione critica
        public synchronized void write(int i) {
                while(numReaders > 0)
                        try{
                                wait();
                        } catch(InterruptedException e){}
                System.out.println("Start writing " + i);
                try {
                        Thread.sleep((int)(Math.random() * 1000));
                         } catch(InterruptedException e) {}
                System.out.println("End writing " + i);
                        /* NB: fondamentale fare notifyAll e non notify per prevenire deadlock! */
                notifyAll();
        }
}
```

Esercizio 2

1) L'oggetto "mio" permette di chiudere la finestra (e terminare il programma) quando si clicca la X nella barra della finestra.
2)
```
public class EsameSvolto1 extends JFrame {

  private EsameSvolto1() {

        addWindowListener(new WindowAdapter() {

                public void windowClosing(WindowEvent e) {     System.exit(0); }

         } );

        setVisible(true);

   }

  public static void main(String[] args) {

        EsameSvolto1 es = new EsameSvolto1();  }

  }
```
3) Non si può definire la classe come lambda expression a causa della molteplicità di metodi che prendono la stessa lista di parametri.