Programmazione III, Programmazione in Rete e Laboratorio, Ist. di Programmazione in Rete e Laboratorio, 2023/24 – 18 luglio 2024

Esercizio 1 (14 punti)

Il seguente programma gestisce i conti correnti di una banca.

1. Si dica se il programma gestisce correttamente gli aggiornamenti dei conti correnti.
2. In caso negativo, si spieghi il perché e si modifichi il codice con la sincronizzazione lato client, massimizzando l'esecuzione delle operazioni che possono essere fatte in parallelo. Si riportino sul foglio solo i metodi che vengono modificati.

```
public class Esercizio1 {
        private static int NUM_ACCOUNTS = 5;    private static int INITIAL_BALANCE = 1000;
        private static int AMOUNT_TO_TRANSFER = 100;

        public static void main(String[] args) {
                Bank b = new Bank(NUM_ACCOUNTS, INITIAL_BALANCE);
                b.printTotal(); // prints total amount in bank
                for (int i=0; i < NUM_ACCOUNTS; i++)
                        new TransferThread(b, AMOUNT_TO_TRANSFER);
        }
} // end Esercizio1

class BankAccount {
        private int balance; // amount of money in the bank account

        public void deposit(int amount) { balance = balance + amount; }

        public boolean withdraw(int amount) {
                if (amount > balance)
                        return false;
                balance = balance - amount;
                return true;
        }

        public int getBalance() { return balance; }
} // end BankAccount

class Bank {
        private BankAccount[] accounts;

        Bank(int n, int init) {
           accounts = new BankAccount[n];
           for (int i=0; i<n; i++) {
                accounts[i] = new BankAccount(); accounts[i].deposit(init);
           }
        }

        public void transfer(int from, int to, int amount) {
                boolean done = accounts[from].withdraw(amount);
                if (done) {
                        accounts[to].deposit(amount);
                        System.out.println("transfer from " + from + " to " + to + ": Euros " + amount);
                        printTotal();
                }
        }
```

```
 public int size() { return accounts.length; }

 public void printTotal() {
        int sum = 0;
        for (int i=0; i < accounts.length; i++) {
                sum = sum + accounts[i].getBalance();
        }
        System.out.println("Total amount of money in the bank: " + sum);
   }
} // end Bank

class TransferThread extends Thread {
        private Bank bank;  private int maxAmount;

   public TransferThread(Bank b, int max) { bank = b;    maxAmount = max; start(); }

   public void run() {
       int from = (int)(bank.size() * Math.random());    int to = (int)(bank.size() * Math.random());
       bank.transfer(from,to,maxAmount);
   }
}
```

Esercizio 2 (9 punti)

Si sviluppino i seguenti punti:

- Descrivere in dettaglio il modello di programmazione guidata dagli eventi, spiegando su quali concetti (ed elementi architetturali) si basa e in cosa differisce dalla programmazione sequenziale.
- Descrivere brevemente come tale modello viene implementato nelle GUI Java, facendo un semplice esempio che mostri come si gestiscono gli eventi generati da un bottone (JButton SWING).

Esercizio 3 (7 punti)

Si consideri il seguente codice. La classe **Point** rappresenta punti bidimensionali, con le loro coordinate x e y. Si sviluppi il metodo **clone()** della classe **Point** (il metodo deve restituire una copia dell'oggetto su cui viene invocato) **facendo overriding** del metodo omonimo di **Object**.

```
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

Metodi modificati per sincronizzare lato client: in modifica bisogna sincronizzare rispetto all'oggetto di tipo BankAccount che si vuole aggiornare. Nella printTotal() bisogna sincronizzare l'accesso al singolo oggetto BankAccount di cui si vuole conoscere il balance.

```
public void transfer(int from, int to, int amount) {
                boolean done = false;
                synchronized(accounts[from]) {
                        done = accounts[from].withdraw(amount);
                }
        if (done) {
                        synchronized(accounts[to]) {
                        accounts[to].deposit(amount);
                        }
                        System.out.println("transfer from " + from + " to " + to + ": Euros " + amount);
                printTotal();
                }
  }

public void printTotal() {
                int sum = 0;
        for (int i=0; i < accounts.length; i++) {
                        synchronized(accounts[i]) {
                sum = sum + accounts[i].getBalance();
                        }
                }
        System.out.println("Total amount of money in the bank: " + sum);
  }
```

Esercizio 3

```
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; }
        public String toString() { return x + "," + y;}
        public Object clone() {
                return new Point(x, y);
        }
}
```