Università degli Studi dell'Insubria
Dipartimento di Scienze Teoriche e Applicate

Programmazione Concorrente e Distribuita
Client-server con socket

Luigi Lavazza
Dipartimento di Scienze Teoriche e Applicate
luigi.lavazza@uninsubria.it

Esercizio 0

- Si consideri il codice che implementa l'esercizio in cui il server gestisce un segmento.
- Si doti il client di un proxy.

Esercizio 1

- Si modifichi il codice dato, per creare un sistema client-server, in cui client e server comunicano attraverso socket.

class OperationRequest

```
public class OperationRequest {
  int CCnumber;
  int amount;
  String request;
  OperationRequest(int cc, int val, String what){
    CCnumber=cc;
    amount=val;
    request=what;
  }
  public int getCCnumber() {
    return CCnumber;
  }
  public int getAmount() {
    return amount;
  }
  public String getRequest() {
    return request;
  }
}
```

class `Client`

```
import java.util.Random;

public class Client {
  Bank myBank;
  int myAccountNum;
  Client(int accNum, Bank b){
    myBank = b;
    myAccountNum=accNum;
  }
  public static void main(String[] args) {
    Client c=new Client(1, new Bank());
    c.exec();
  }
}
```

class `Client`

```
public void exec() {
  Result r;
  String opReq=null;
  int howMuch=0;
  int times=new Random().nextInt(10);
  for(int i=0; i<times; i++) {
    howMuch=new Random().nextInt(1000);
    if(new Random().nextBoolean()) {
      r=myBank.executeOperation(new OperationRequest(1,
                                        howMuch, "Deposit"));
    } else {
      r=myBank.executeOperation(new OperationRequest(1,
                                        howMuch, "Withdraw"));
    }
    System.out.println(r);
  }
  r=myBank.executeOperation(new OperationRequest(1,2,"Double"));
  System.out.println(r);
}
```

class **Bank**

```
public class Bank {
  int ccAmounts[]={0,0,0};
  public Result executeOperation(OperationRequest op) {
    int ccNum=op.getCCnumber();
    String opType=op.getRequest();
    if(ccNum<0 || ccNum>2) {
      return new Result(-1, 0, opType, false);
    }
    if(opType.equals("Deposit")) {
      ccAmounts[ccNum]+=op.getAmount();
      return new Result(ccNum, ccAmounts[ccNum], opType, true);
    } else if (opType.equals("Withdraw")) {
      if(op.getAmount()>ccAmounts[ccNum]) {
        return new Result(ccNum, 0, opType, false);
      } else {
        ccAmounts[ccNum]-=op.getAmount();
        return new Result(ccNum, ccAmounts[ccNum],opType,true);
      }
    } else { return new Result(ccNum, 0, "unknown", false);
}}}
```

class `Result`

```
public class Result {
  int CCnumber;
  int amount;
  String opType;
  boolean successful;
  Result(int cc, int val, String ot, boolean ok){
    CCnumber=cc;
    amount=val;
    opType=ot;
    successful=ok;
  }
  public int getCCnumber() { return CCnumber; }
  public int getAmount() { return amount; }
  public boolean isSuccessful() { return successful; }
  public String getType() { return opType; }
  public String toString() {
    return "op. "+opType+" on CC num. "+ CCnumber+
      (successful?" OK":" KO")+" resulting amount="+amount;
  }
}
```

Esercizio 2

- Si modifichi il programma sviluppato nell'esercizio 1 per tenere conto dei requisiti seguenti:
  - L'operazione richiesta può richiedere parecchio tempo.
  - Il client, una volta mandata la richiesta alla banca, non resta in attesa della risposta, ma continua ad eseguire le proprie elaborazioni.
  - Quando l'operazione è completata, la banca manda un avviso al client.
    - L'avviso comprende il valore del saldo del CC del client.
  - Quando il client riceve l'avviso e legge il valore del saldo, visualizza l'esito dell'operazione.