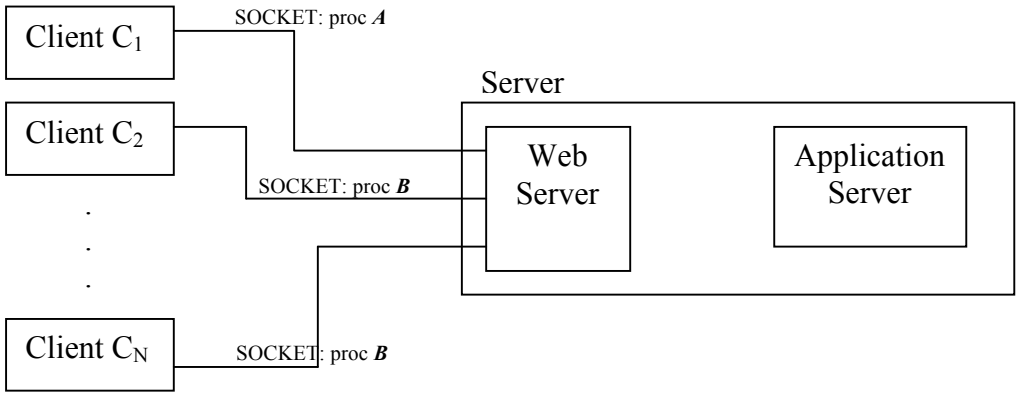
LAUREA SPECIALISTICA IN INGEGNERIA INFORMATICA PER LA GESTIONE D'AZIENDA

TECNOLOGIE INFORMATICHE PER LA GESTIONE AZIENDALE

Prova scritta del 14 Luglio 2006

Nome e cognome_____ **Numero di matricola** _____

Si consideri il sistema rappresentato in figura costituito da N client e da un server. Un client si limita a inoltrare una semplice richiesta di esecuzione della procedura *A* o *B*. La comunicazione tra server e client avviene tramite una connessione sulla porta 8080. Il server è costituito a sua volta da due thread: il primo, di classe *WebServer*, si occupa di accettare le richieste provenienti dai client e il secondo, di classe *ApplicationServer*, si occupa di eseguire la procedura richiesta *A* o *B*. In particolare, il thread ApplicationServer deve essere bloccato (**NO attese attive**) quando non ci sono richieste pendenti e nel caso di più richieste deve dare precedenza all'esecuzione delle procedure di tipo A rispetto a quelle di tipo B. Per far ciò i due thread, WebServer e ApplicationServer, condividono un oggetto di tipo *Gestore* che ha il compito di di sincronizzarli.



Progettare e realizzare le classi che realizzano il *WebServer* e il *Gestore*. L'elaborato si *deve* articolare sui files WebServer.java, **ApplicationServer.java, Gestore.java** e Main.java che devono contenere *tutte* le classi necessarie alla realizzazione del sistema multithread.

TRACCIA PROPOSTA AL CANDIDATO:

```
//Main.java
import java.net.*;
import java.io.*;

class Client extends Thread {
      private String richiesta = null;
      final private String INDIRIZZO_IP = "127.0.0.1";
      final private int PORTA = 8080;
      final private int NUM_DATI = 5;

    public Client(String nome, String r) {
        super(nome);
        richiesta = r;
    }
    public void run() {
        InetAddress ind = null;
        Socket socket = null;
```

```
        ObjectOutputStream oout = null;
        for (int i=0; i<NUM_DATI; i++) {
            try {
                ind = InetAddress.getByName(INDIRIZZO_IP);
                socket = new Socket(ind, PORTA);
                oout = new ObjectOutputStream (socket.getOutputStream());
                System.out.println(getName()+":invio della richiesta "+richiesta+"..\n");
                oout.writeObject(richiesta);
            }catch (Exception e) { e.printStackTrace(); }
            finally {
                try {
                        oout.close();
                        socket.close();
                }catch (IOException e) { e.printStackTrace(); }
               }
           try {              //attende un tempo random compreso fra 200 e 500 msec
               java.lang.Thread.sleep((long)(200.0 + Math.random()*300.0));
           }catch (Exception e) { e.printStackTrace(); }
        }// fine for
    System.out.println(getName() + " terminato.");
    }// fine run
}

public class Main {
    public Main() {}
    public static void main(String[] args) {
       Gestore g = new Gestore();                  //creo il gestore
       new WebServer("WebS", g).start();    //WebServer accetta le richieste
       new ApplicationServer("AppS", g).start();          //ApplicationServer esegue le richieste
                                                          dando precedenza a quelle di tipo "A"
                                                          rispetto a "B"
       new Client("C1", "A").start();              //creo 3 client che effettuano
       new Client("C2", "B").start();              //richiesta di tipo "A" o "B"
       new Client("C3", "B").start();
    }

}
```

//ApplicationServer.java

```
class ApplicationServer extends Thread {
    private Gestore ges= null;
    public ApplicationServer(String nome, Gestore r) {
        super(nome);
        ges = r;
    }

    public void run() {
           while (true) {
               System.out.println(getName() +": Eseguo richiesta diprocedura"+
this.ges.eseguiRichiesta() +".\n");
                try {
                      java.lang.Thread.sleep(300);
               }catch (Exception e) { e.printStackTrace(); }
           }
    }  // fine run
}
```

//Gestore.java

```
class Gestore{
       //..
       public Gestore(){//..}
       public synchronized void aggiungiNuovaRichiesta(String ric){//..}
       public synchronized String eseguiRichiesta(){//..}
}
```

//WebServer.java

```
class WebServer extends Thread {
       //..
       public WebServer(String nome, Gestore r) {//..}
       public void run() { //..}
}
```