UNICAL - Esame di **Reti di Calcolatori** - 9 Gennaio 2003

Esercizio 1

Si vuole realizzare un sistema distribuito su rete, per lo svolgimento di gare d'appalto per l'esecuzione di opere pubbliche. Il sistema è costituito da:

- **1** applicazione *Giudice* di gara;
- **n** applicazioni *Partecipanti* (identificate da un intero *id* univoco).

Un ente che intende realizzare un'opera pubblica invia al Giudice di gara una *Richiesta*, contenente la descrizione dell'opera da realizzare e l'importo massimo disponibile per la sua realizzazione. Il Giudice invia gli estremi della Richiesta a tutti i Partecipanti. Ciascun Partecipante risponde inviando al Giudice una *Offerta*, contenente l'*id* del Partecipante ed l'importo da esso richiesto per la realizzazione dell'opera. Il Giudice di gara seleziona quindi l'Offerta con l'importo richiesto inferiore (a parità di prezzo richiesto, seleziona l'Offerta con *id* del Partecipante inferiore). Questa Offerta, ritenuta vincitrice della gara, viene inviata dal Giudice di gara all'ente richiedente, e, per notifica, a tutti i Partecipanti.
Si richiede di realizzare in Java l'applicazione *Giudice*.

L'applicazione ***Giudice*** è basata sulle seguenti operazioni:

- Usa la porta TCP 2000 per la connessione di **una** applicazione remota (che rappresenta l'ente che intende realizzare l'opera).
- L'applicazione connessa invia un oggetto istanza della classe *Richiesta*. La classe *Richiesta*, che implementa l'interfaccia Serializable, memorizza la descrizione dell'opera e l'importo massimo disponibile per la sua realizzazione.
- Comunica a tutti i Partecipanti una stringa con la descrizione dell'opera e l'importo massimo. Per esempio, la stringa "Stadio olimpico di Arcavacata – 1000". La stringa viene inviata ai Partecipanti in broadcast (si usi la porta 3000 e l'indirizzo 230.0.0.1).
- Usa la porta TCP 4000 per ricevere **una sequenza di n** connessioni da parte delle applicazioni Partecipanti (**non** è richiesto di gestire connessioni multiple).
- Riceve un oggetto istanza della classe *Offerta* da ciascuna applicazione Partecipante connessa. La classe Offerta, che implementa Serializable, memorizza un intero che rappresenta l'id del Partecipante ed un intero che rappresenta l'importo richiesto per la realizzazione dell'opera.
- Seleziona l'Offerta vincitrice. L'Offerta selezionata viene inviata all'applicazione remota (che rappresenta l'ente richiedente). A tutti i Partecipanti, inoltre, invia in broadcast una stringa contenente l'id del vincitore e l'importo da esso richiesto (ad esempio, la stringa "12 – 950" indica che il Partecipante con id 12 ha vinto la gara con un importo richiesto di 950).

La costante **n** è nota all'applicazione Giudice. Si richiede di implementare anche le classi *Richiesta* e *Offerta*.

Proposta di soluzione per l'Esercizio 1

```
/* Richiesta.java */

import java.io.*;

public class Richiesta implements Serializable
{
  private String descrizioneOpera;
  private int importoMassimo;

  public Richiesta (String descrizioneOpera, int importoMassimo)
  {
    this.descrizioneOpera = descrizioneOpera;
    this.importoMassimo = importoMassimo;
  }
  public String getDescrizioneOpera ()
  {
    return descrizioneOpera;
  }
  public int getImportoMassimo ()
  {
    return importoMassimo;
  }
}
```

/* Offerta.java */

```
import java.io.*;

public class Offerta implements Serializable
{
  private int id;
  private int importoRichiesto;

  public Offerta (int id, int importoRichiesto)
  {
    this.id = id;
    this.importoRichiesto = importoRichiesto;
  }
  public int getId ()
  {
    return id;
  }
  public int getImportoRichiesto ()
  {
    return importoRichiesto;
  }
}
```

/* Giudice.java */

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

public class Giudice
{
  private static final int n = 10;

  public static void main (String args[])
  {
    try
    {
      // accetta la connessione dell'ente richiedente
      ServerSocket server1 = new ServerSocket(2000);
      Socket ente = server1.accept();

      // riceve una Richiesta
      ObjectInputStream input1 = new ObjectInputStream (ente.getInputStream());
      Richiesta richiesta = (Richiesta)input1.readObject();

      // invia la stringa contenente la Richiesta ai Partecipanti
      String r = richiesta.getDescrizioneOpera()+" – "+
                 richiesta.getImportoMassimo();
      MulticastSocket msocket = new MulticastSocket(3000);
      byte buf[] = new byte[512];
      buf = r.getBytes();
      InetAddress group = InetAddress.getByName("230.0.0.1");
      DatagramPacket packet = new DatagramPacket(buf, buf.length, group, 3000);
      msocket.send(packet);

      // accetta n connessioni dai Partecipanti, riceve n Offerte
      // e seleziona l'Offerta migliore
      Offerta offertaMigliore = null;
      ServerSocket server2 = new ServerSocket(4000);
      for (int i = 0; i < n; i++)
      {
        Socket partecipante = server2.accept();
```

```
        // riceve una Offerta
        ObjectInputStream input2 = new ObjectInputStream
                                     (partecipante.getInputStream());
        Offerta offerta = (Offerta)input2.readObject();

        // verifica se l'Offerta corrente è la migliore corrente
        if ((offertaMigliore == null)||
           (offerta.getImportoRichiesto()<offertaMigliore.getImportoRichiesto())||
           ((offerta.getImportoRichiesto()==offertaMigliore.getImportoRichiesto())&&
            (offerta.getId() < offertaMigliore.getId())))
          offertaMigliore = offerta;
        partecipante.close();
      }

      // invia l'Offerta migliore all'ente richiedente
      ObjectOutputStream output1 = new ObjectOutputStream (ente.getOutputStream());
      output1.writeObject(offertaMigliore);
      ente.close();

      // invia la stringa contenente l'offerta Migliore ai Partecipanti
        String o = offertaMigliore.getId()+" - "+
                   offertaMigliore.getImportoRichiesto();
        buf = o.getBytes();
        packet = new DatagramPacket(buf, buf.length, group, 3000);
        msocket.send(packet);
        msocket.close();
    }
    catch (Exception e)
    {
      System.out.println ("Error: "+e);
    }
  }
}
```