

Soluzione Esercizio 1

```
1 // Asta.java
2 package es1;
3
4 public class Asta {
5
6     // Descrizione del bene messo all'asta
7     protected String descrizione;
8     // Durata dell'asta in secondi
9     protected int durata;
10    // Offerta piu' alta ricevuta
11    protected float offMax;
12    // Partecipante che ha fatto l'offerta piu' alta
13    protected Partecipante agg;
14    // Indica se l'asta e' attiva o meno
15    protected boolean attiva;
16    // Indica se l'asta e' terminata
17    protected boolean finita;
18    // Istante di inizio dell'asta
19    protected long tempoInizio;
20    // Aggiudica il bene allo scadere del tempo
21    protected Sveglia sveglia;
22
23    // Crea un oggetto asta
24
25    public Asta(String descrizione, int durata) {
26        this.descrizione = descrizione;
27        this.durata = durata;
28        this.sveglia = new Sveglia();
29    }
30
31    /**
32     * Restituisce la descrizione del bene
33     */
34    public String getDescrizione() {
35        return descrizione;
36    }
37
38    // L'asta ha inizio
39    public synchronized void attiva() {
40        attiva = true;
41        // Prende il tempo attuale come istante di inizio
42        tempoInizio = System.currentTimeMillis();
43        // Fa partire il thread ausiliario sveglia
44        sveglia.start();
45    }
46
47
48    /**
49     * Termina prematuramente l'asta, aggiudicando il bene al migliore offerente
50     */
51    public synchronized void aggiudicaAdesso() throws AstaNonAttivaException {
52        if(!attiva)
53            throw new AstaNonAttivaException();
54        // Interrompe il thread sveglia
55        sveglia.interrupt();
56    }
57
58    // Offerta del partecipante c
59    // Restituisce true se l'offerta fatta e' la piu' alta, false altrimenti
60}
```

```

61     public synchronized boolean offerta(float quanto, Partecipante c) throws
AstaNonAttivaException {
62         if(!attiva)
63             throw new AstaNonAttivaException("Asta non ancora attiva");
64         // Controlla se l'offerta attuale e' la piu' alta
65         if(quanto > offMax) {
66             offMax = quanto;
67             agg = c;
68             return true;
69         }
70         return false;
71     }
72
73     // Metodo chiamato dai partecipanti, permette loro di capire se hanno vinto
74
75     public synchronized boolean hoVinto(Partecipante c) throws InterruptedException {
76         // Attende la fine dell'asta
77         while(!finita)
78             wait();
79         return c == agg;
80     }
81
82     // Aggiudica il bene al migliore offerente
83
84     private synchronized void aggiudica() {
85         System.out.println("Tempo scaduto");
86         finita = true;
87         notifyAll();
88     }
89
90     class Sveglia extends Thread {
91
92         public void run(){
93             try {
94                 // Se non l'asta non e' arrivata a scadenza o se il thread non e' stato interrotto
95                 // allora dorme
96                 while(System.currentTimeMillis() < tempoInizio + 1000*durata && !isInterrupted())
97                 {
98                     sleep(tempoInizio + 1000*durata - System.currentTimeMillis());
99                 }
100                 // Se arriva un interrupt esce dal ciclo
101             } catch (InterruptedException ie) {}
102             aggiudica();
103         }
104     }
105 }
1
// AstaNonAttivaException
2 package es1;
3
4 public class AstaNonAttivaException extends Exception {
5     public AstaNonAttivaException() {
6         super();
7     }
8
9     public AstaNonAttivaException(String s){
10         super(s);
11     }
12 }
13
1
// Partecipante.java
2 package es1;
3

```

```

4 public class Partecipante extends Thread {
5
6     private Asta asta;
7
8     public Partecipante(Asta asta) {
9         this.asta = asta;
10    }
11
12    public void run() {
13        try {
14            boolean r = asta.offerta(100, this);
15            System.out.println("Risultato dell'offerta: " + r);
16            r = asta.hoVinto(this);
17            System.out.println("Ho vinto: " + r);
18        } catch (Exception e) {
19            System.err.println("Eccezione: " + e.getMessage());
20        }
21    }
22 }

```

```

1 // Prova.java
2 package es1;
3
4 public class Prova {
5     public static void main(String[] args) {
6         try {
7             Asta a = new Asta("Guernica", 5);
8             a.attiva();
9             Partecipante p1 = new Partecipante(a);
10            Partecipante p2 = new Partecipante(a);
11            p1.start();
12            p2.start();
13            a.aggiudicaAdesso();
14        } catch (AstaNonAttivaException e){
15            System.err.println("Asta non attiva");
16        }
17    }
18 }

```

Soluzione Esercizio 2

```

1 // PCStatement.java
2 public class PCStatement {
3     public final int senderPort;
4     public final String statement;
5     public final double item;
6
7     public PCStatement(int sp, String st, double it) {
8         senderPort = sp; statement = st; item = it;
9     }
10 }
11
12

```

```

1 // PCList.java
2 import java.net.*;
3 import java.io.*;
4 import java.util.*;
5 import com.thoughtworks.xstream.*;
6 import java.sql.*;
7
8 public class PCList {
9     private static int receivePort, sendPort;
10
11     public static void configure(int rp, int sp) {

```

```

12     receivePort = rp; sendPort = sp;
13 }
14
15 public static void send(PCStatement o) {
16     try (Socket sock = new Socket("localhost", sendPort);
17         DataOutputStream dos =
18             new DataOutputStream(sock.getOutputStream());
19     ) { dos.writeUTF((new XStream()).toXML(o));
20     } catch (IOException e) { e.printStackTrace();}
21     System.out.println("- invio a " + sendPort);
22 }
23
24 public static PCStatement receive() {
25     PCStatement o = null;
26     try (
27         ServerSocket servsock = new ServerSocket(receivePort);
28         Socket sock = servsock.accept();
29         DataInputStream dis =
30             new DataInputStream(sock.getInputStream());
31     ) { o = (PCStatement)(new XStream()).fromXML(dis.readUTF());
32     } catch (IOException e) { e.printStackTrace();}
33     System.out.println("- ricevo");
34     return o;
35 }
36
37 private static ArrayList read() {
38     try {
39         Connection co = DriverManager.getConnection("jdbc:mysql://localhost:3306/pclist",
"root","");
40         PreparedStatement ps = co.prepareStatement("SELECT xmlArrayList FROM pclist WHERE
receivePort = ?");
41         ps.setInt(1, receivePort);
42         ResultSet rs = ps.executeQuery();
43         if (rs.next()) {
44             String x = rs.getString("xmlarraylist");
45             return (ArrayList) (new XStream()).fromXML(x);
46         }
47     } catch (SQLException e) { System.err.println(e.getMessage());}
48     return null;
49 }
50
51 private static void write(ArrayList a) {
52     String x = (new XStream()).toXML(a);
53     try (
54         Connection co = DriverManager.getConnection("jdbc:mysql://localhost:3306/pclist",
"root","");
55         PreparedStatement ps = co.prepareStatement("UPDATE pclist SET xmlarraylist = ? WHERE
receiveport = ?");
56     ) {
57         ps.setInt(2, receivePort); ps.setString(1, x);
58         System.out.println("rows affected: " + ps.executeUpdate());
59     } catch (SQLException e) { System.err.println(e.getMessage());}
60 }
61
62 public static void create(PCStatement o) {
63     write(new ArrayList());
64     System.out.println((new XStream()).toXML(o) + "\nCreato:\n" + read());
65     send(o);
66     if (o.senderPort == receivePort)
67         receive(); //08
68 }
69
70 public static void add(PCStatement o) {
71     ArrayList al = read();

```

```

72     int dimensioneDesiderata = 0;
73     while(al.size() > dimensioneDesiderata) { //01
74         send(o); //02
75         o = receive(); //03
76         dimensioneDesiderata++; //04
77     } //05
78     al.add(o.item);
79     write(al);
80     System.out.println((new XStream()).toXML(o) + "\nInserito:\n" + read());
81 }
82
83 public static void remove(PCStatement o) {
84     ArrayList al = read();
85     if (!al.remove(o.item)) { //06
86         send(o); //07
87         if (o.senderPort == receivePort)
88             receive(); //08
89     }
90     else {
91         write(al); //09
92         System.out.println((new XStream()).toXML(o) + "\nEstratto:\n" + read());
93     }
94 }
95
96 public static void main(String[] args) {
97     System.out.println("- sono " + args[0]);
98     configure(Integer.parseInt(args[0]), Integer.parseInt(args[1]));
99     PCStatement o;
100     if (args.length > 2) //10
101         o = new PCStatement (Integer.parseInt(args[0]), args[2],
Double.parseDouble(args[3]));
102     else //11
103         o = receive();
104     switch(o.statement) {
105         case "create": create(o); break;
106         case "add": add(o); break;
107         case "remove": remove(o); break;
108     }
109 }
110 }
111
112 /* Note
113 01 finchè non trova la dimensione desiderata
114 02 invia al successivo
115 03 attende nel caso non si trovi la dimensione desiderata
116 04 incrementa la dimensione desiderata
117 05 se e' stata trovata la dimensione desiderata l'operazione non sara' propagata
118 06 se il valore non viene estratto
119 07 invia al prossimo se giro non completo
120 08 se richiedente rimani in attesa dell'ultimo
121 09 se viene estratto archivia e stampa
122 10 e' il richiedente
123 11 non e' il richiedente
124 */

```