

Soluzione Esercizio 2

```
1 // Operazione.java
2 public class Operazione {
3     public final int portaRichiedente;
4     public final String operazione;
5     public final double valore;
6
7     public Operazione(int mit, String op, double val) {
8         portaRichiedente = mit;
9         operazione = op;
10        valore = val;
11    }
12 }

1 // ArchivioDistribuito.java
2 import java.net.*;
3 import java.io.*;
4 import java.util.*;
5 import com.thoughtworks.xstream.*;
6 import java.nio.file.*;
7
8 public class ArchivioDistribuito {
9     private static int portaRicevi, portaInvia;
10
11    public static void inizializzaArchivio(int pR, int pI) {
12        portaRicevi = pR; portaInvia = pI;
13    }
14
15    public static void inviaOperazione(Operazione o) {
16        try (Socket sock = new Socket("localhost", portaInvia);
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 " + portaInvia);
22    }
23
24    public static Operazione riceviOperazione() {
25        Operazione o = null;
26        try (
27            ServerSocket servsock = new ServerSocket(portaRicevi);
28            Socket sock = servsock.accept();
29            DataInputStream dis =
30                new DataInputStream(sock.getInputStream());
31        ) { o = (Operazione) (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 leggiArchivio() {
```

```

38     try { String x = new String(Files.readAllBytes(Paths.get(portaRicevi +
"_archivio.xml")));
39     return (ArrayList) (new XStream().fromXML(x));
40     } catch (IOException e) { e.printStackTrace(); }
41     return null;
42 }
43
44 private static void scriviArchivio(ArrayList a) {
45     String x = (new XStream()).toXML(a);
46     try { Files.write(Paths.get(portaRicevi + "_archivio.xml"), x.getBytes());
47     } catch (IOException e) { e.printStackTrace(); }
48 }
49
50 public static void crea(Operazione o) {
51     scriviArchivio(new ArrayList());
52     System.out.println((new XStream()).toXML(o) + "\nCreato:\n" + leggiArchivio());
53     inviaOperazione(o);
54     if (o.portaRichiedente == portaRicevi)
55         riceviOperazione(); //08
56 }
57
58 public static void inserisci(Operazione o) {
59     ArrayList al = leggiArchivio();
60     int dimensioneDesiderata = 0;
61     while(al.size() > dimensioneDesiderata) { //01
62         inviaOperazione(o); //02
63         o = riceviOperazione(); //03
64         dimensioneDesiderata++; //04
65     } //05
66     al.add(o.valore);
67     scriviArchivio(al);
68     System.out.println((new XStream()).toXML(o) + "\nInserito:\n" + leggiArchivio());
69 }
70
71 public static void estrai(Operazione o) {
72     ArrayList al = leggiArchivio();
73     if (!al.remove(o.valore)) { //06
74         inviaOperazione(o); //07
75         if (o.portaRichiedente == portaRicevi)
76             riceviOperazione(); //08
77     }
78     else {
79         scriviArchivio(al); //09
80         System.out.println((new XStream()).toXML(o) + "\nEstratto:\n" + leggiArchivio());
81     }
82 }
83
84 public static void main(String[] args) {
85     System.out.println("- sono " + args[0]);
86     inizializzaArchivio(Integer.parseInt(args[0]), Integer.parseInt(args[1]));
87     Operazione o;
88     if (args.length > 2) //10
89         o = new Operazione (Integer.parseInt(args[0]), args[2], Double.parseDouble(args[3]));
90     else //11
91         o = riceviOperazione();

```

```
92     switch(o.operazione) {
93         case "crea": crea(o); break;
94         case "inserisci": inserisci(o); break;
95         case "estrai": estrai(o); break;
96     }
97 }
98 }
99
100 /* Note
101 01 finchè non trova la dimensione desiderata
102 02 invia al successivo
103 03 attende nel caso non si trovi la dimensione desiderata
104 04 incrementa la dimensione desiderata
105 05 se e' stata trovata la dimensione desiderata l'operazione non sara' propagata
106 06 se il valore non viene estratto
107 07 invia al prossimo se giro non completo
108 08 se richiedente rimani in attesa dell'ultimo
109 09 se viene estratto archivia e stampa
110 10 e' il richiedente
111 11 non e' il richiedente
112 */
```