Soluzione Esercizio 2

```java
// Operazione.java
public class Operazione {
  public final int portaRichiedente;
  public final String operazione;
  public final double valore;

  public Operazione(int mit, String op, double val) {
    portaRichiedente = mit;
    operazione = op;
    valore = val;
  }
}
```

```java
// ArchivioDistribuito.java
import java.net.*;
import java.io.*;
import java.util.*;
import com.thoughtworks.xstream.*;
import java.nio.file.*;

public class ArchivioDistribuito {
   private static int portaRicevi, portaInvia;

public static void inizializzaArchivio(int pR, int pI) {
   portaRicevi = pR; portaInvia = pI;
  }

 public static void inviaOperazione(Operazione o) {
   try (Socket sock = new Socket("localhost", portaInvia);
     DataOutputStream dos =
       new DataOutputStream(sock.getOutputStream());
   ) {  dos.writeUTF((new XStream()).toXML(o));
   } catch (IOException e) { e.printStackTrace();}
   System.out.println("- invio a " + portaInvia);
  }

 public static Operazione riceviOperazione() {
   Operazione o = null;
   try (
     ServerSocket servsock = new ServerSocket(portaRicevi);
     Socket sock = servsock.accept();
     DataInputStream dis =
        new DataInputStream(sock.getInputStream());
   ) { o = (Operazione)(new XStream()).fromXML(dis.readUTF());
   } catch (IOException e) { e.printStackTrace();}
   System.out.println("- ricevo");
   return o;
  }

 private static ArrayList leggiArchivio() {
   try { String x = new String(Files.readAllBytes(Paths.get(portaRicevi +
"_archivio.xml")));
   return (ArrayList) (new XStream().fromXML(x));
   } catch (IOException e) { e.printStackTrace(); }
   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
```

```
02 invia al successivo
03 attende nel caso non si trovi la dimensione desiderata
04 incrementa la dimensione desiderata
05 se e' stata trovata la dimensione desiderata l'operazione non sara' propagata
06 se il valore non viene estratto
07 invia al prossimo se giro non completo
08 se richiedente rimani in attesa dell'ultimo
09 se viene estratto archivia e stampa
10 e' il richiedente
11 non e' il richiedente
*/
```