Il linguaggio Java
I flussi

Programmi d'esempio

File (caratteri)

```
import java.io.*;
public class CopyChars {
  public static void main(String[] args)
                        throws IOException {
    String path = "/* un path */";
    String src = path + "pippo.txt";
    String dst = path + "pluto.txt";

    FileReader in = new FileReader(src);
    FileWriter out = new FileWriter(dst);
    int c;

    while ((c = in.read()) != -1)
      out.write(c);
    in.close();
    out.close();
  }
}
```

File (byte)

```
import java.io.*;
public class CopyBytes {
 public static void main(String[] args)
                     throws IOException {
   String path = "/* un path */";
   String src = path + "pippo.txt";
   String dst = path + "paperino.txt";

   FileInputStream in =
                 new FileInputStream(src);
   FileOutputStream out =
               new FileOutputStream(dst);
   int c;

   while ((c = in.read()) != -1)
     out.write(c);

   in.close();
   out.close();
 }
}
```


BufferedCopy

```
import java.io.*;

public class BufferedCopy {
 public static void main(String[] args)
                      throws IOException {
   String path = "/* un path */";
   String src = path + "pippo.txt";
   String dst = path + "topolino.txt";

   FileReader fin = new FileReader(src);
   // buffin wraps fin
   BufferedReader buffin =
                 new BufferedReader(fin);
   FileWriter fout = new FileWriter(dst);
   // printout wraps fout
   PrintWriter printout =
                   new PrintWriter(fout);
   String currline;

   while ((currline = buffin.readLine())
                                    != null)
     printout.println(currline);
   buffin.close();
   // indispensabile per il flush dei dati
   printout.close();
 }
}
```

BufferedCopy (autoflush)

```
import java.io.*;

public class BufferedCopy {
  public static void main(String[] args)
                        throws IOException {
    String path = "/* un path */";
    String src = path + "pippo.txt";
    String dst = path + "topolino.txt";

    FileReader fin = new FileReader(src);
    // buffin wraps fin
    BufferedReader buffin =
                   new BufferedReader(fin);
    FileWriter fout = new FileWriter(dst);
    // printout wraps fout with autoflush
    PrintWriter printout =
              new PrintWriter(fout, true);
    String currline;

    while ((currline = buffin.readLine())
                                    != null)
      // autoflush
      printout.println(currline);
    buffin.close();
    printout.close();
  }
}
```


Utilizzare i flussi filtro (I) [1]

```
import java.io.*;
public class IOFormattato {
   private final static String cammino = /* un path */;

  public static void main(String[] args) {
       String nomeFile = cammino + "mio.dat";
       DataOutputStream dout = null;
       DataInputStream din = null;
       double d = -2.5;
       int i = 3;
       String s = "ciao!";

  try{
       FileOutputStream fout =
                              new FileOutputStream(nomeFile);
       dout = new DataOutputStream(fout);
       dout.writeDouble(d);
       dout.writeInt(i);
       dout.writeUTF(s);
  } catch (IOException e) {
       System.out.println(e);
  }
  finally {
       if ( dout != null )
            try {
                 dout.close();
            } catch (IOException e) {
                 System.out.println("close failed: " + e);
            }
  } // continua ...
```

Utilizzare i flussi filtro (I) [2]

```
    i = 0;
    d = 0.0;
    s = null;
    try{
        FileInputStream fin =
                              new FileInputStream(nomeFile);
        din = new DataInputStream(fin);
        d = din.readDouble();
        i = din.readInt();
        s = din.readUTF();
        din.close();
        System.out.println("d = " + d +
                                        " i = " + i + " s = " + s);
     } catch (IOException e) {
            System.out.println(e);
     }
    finally {
        if ( din != null )
            try {
                din.close();
        } catch (IOException e) {
            System.out.println("close failed: " + e);
        }
    }
   } // main
} // class
```


Utilizzare i flussi filtro (II) [1]

```
import java.io.*;
public class DataIODemo {
public static void main(String[] args)
              throws IOException {

  String filename = "un path";
  DataOutputStream out =
     new DataOutputStream(
          new FileOutputStream(filename));

    double[] prices = { 19.99, 9.99, 15.99,
                        3.99, 4.99 };
    int[] units = { 12, 8, 13, 29, 50 };
    String[] descs = {"Java T-shirt",
                      "Java Mug",
                      "Duke Juggling Dolls",
                      "Java Pin",
                      "Java Key Chain"};
    for (int i = 0; i < prices.length; i ++) {
      out.writeDouble(prices[i]);
      out.writeChar('\t');
      out.writeInt(units[i]);
      out.writeChar('\t');
      out.writeChars(descs[i]);
      out.writeChar('\n');
    }
    out.close();

    DataInputStream in = new
            DataInputStream(new
              FileInputStream(filename));
    double price;
    int unit;
    StringBuffer desc;
    double total = 0.0;

    // continua
```

Utilizzare i flussi filtro (II) [2]

```
    try {
      while (true) {
      price = in.readDouble();
      in.readChar();        // throws out the tab
      unit = in.readInt();
      in.readChar();        // throws out the tab
      char chr;
      desc = new StringBuffer(20);
      char lineSep =

  System.getProperty("line.separator").charAt(1);
      while ((chr = in.readChar()) != lineSep)
        desc.append(chr);
      System.out.println("You've ordered " +
                  unit + " units of " +
                  desc + " at $" + price);
                  total = total + unit * price;
      }
    } catch (EOFException e) {}
    System.out.println("For a TOTAL of: $" +
                                        total);
    in.close();
  } // main
} // class

//You've ordered 12 units of Java T-shirt at $19.99
//You've ordered 8 units of Java Mug at $9.99
//You've ordered 13 units of Duke Juggling Dolls at $15.99
//You've ordered 29 units of Java Pin at $3.99
//You've ordered 50 units of Java Key Chain at $4.99
//For a TOTAL of: $892.8800000000001
```


Serializzazione (I)

```
import java.io.*;
import java.util.*;

class Employee implements Serializable {
  public Employee() {}
  public Employee(String n, double s, int year, int month,
                                                  int day) {
    name = n;
    salary = s;
    GregorianCalendar calendar
          = new GregorianCalendar(year, month - 1, day);
      // GregorianCalendar uses 0 for January
    hireDay = calendar.getTime();
  }

  public String getName() { return name;  }
  public double getSalary() {  return salary; }
  public Date getHireDay() { return hireDay; }
  public void raiseSalary(double byPercent)  {
    double raise = salary * byPercent / 100;
    salary += raise;
  }
  public String toString()  {
    return getClass().getName() + "[name=" + name
      + ",salary=" + salary + ",hireDay=" + hireDay  + "]";
  }
  private String name;
  private double salary;
  private Date hireDay;
}
```

Serializzazione (II)

```
class Manager extends Employee {
   public Manager(String n, double s, int year, int month,
                                                    int day) {
      super(n, s, year, month, day);
      bonus = 0;
   }
   public double getSalary() {
      double baseSalary = super.getSalary();
      return baseSalary + bonus;
   }
   public void setBonus(double b) {
      bonus = b;
   }
   public String toString() {
      return super.toString() + "[bonus=" + bonus + "]";
   }
   private double bonus;
}
```


Serializzazione (III)

```
import java.io.*;
import java.util.*;

class ObjectFileTest {
   public static void main(String[] args)    {
      Manager boss = new Manager("Carl Cracker", 80000,
                                                    1987, 12, 15);
      boss.setBonus(5000);
      Employee[] staff = new Employee[3];
      staff[0] = boss;
      staff[1] = new Employee("Harry Hacker", 50000,1989, 10, 1);
      staff[2] = new Employee("Tony Tester", 40000, 1990, 3, 15);
      try {
        // salva i record
        ObjectOutputStream out = new ObjectOutputStream(new
                                    FileOutputStream("employee.dat"));
        out.writeObject(staff);
        out.close();

        // rilege i record in un nuovo array
        ObjectInputStream in =  new ObjectInputStream(new
                                    FileInputStream("employee.dat"));
        Employee[] newStaff = (Employee[])in.readObject();
        in.close();

       for (int i = 0; i < newStaff.length; i++)
          System.out.println(newStaff[i]);
      }
      catch (Exception e)
      {
        e.printStackTrace();
      }
   }
}
```

Externalizable (I)[1]

```
import java.io.*;
import java.util.*;
////////////////////////////////
class Blip1 implements Externalizable {
  public Blip1() {
    System.out.println("Blip1.Blip1");
  }
  public void writeExternal(ObjectOutput out)
                            throws IOException {
    System.out.println("Blip1.writeExt");

  }
  public void readExternal(ObjectInput in)
                            throws IOException,
                        ClassNotFoundException {
    System.out.println("Blip1.readExt");

  }
}
////////////////////////////////

class Blip2 implements Externalizable {
  public Blip2() {
    System.out.println("Blip2.Blip2");
  }
  public void writeExternal(ObjectOutput out)
                            throws IOException {
    System.out.println("Blip2.writeExt");

  }
  public void readExternal(ObjectInput in)
                            throws IOException,
                        ClassNotFoundException {
    System.out.println("Blip2.readExt");

  }
} // continua
```


Externalizable (I)[2]

```
// continua

public class Blips {
  public static void main(String args[])
                            throws IOException,
                       ClassNotFoundException {
    Blip1 b1 = new Blip1();
    Blip2 b2 = new Blip2();
    ObjectOutputStream o =
      new ObjectOutputStream(
        new FileOutputStream("Blips.out"));
    o.writeObject(b1);
    o.writeObject(b2);
    o.close();
    ObjectInputStream in =
      new ObjectInputStream(
        new FileInputStream("Blips.out"));
    b1 = (Blip1)in.readObject();
    b2 = (Blip2)in.readObject();
  }
}

// -- output --
// Blip1.Blip1
// Blip2.Blip2
// Blip1.writeExt
// Blip2.writeExt
// Blip1.Blip1
// Blip1.readExt
// Blip2.Blip2
// Blip2.readExt
```

Externalizable (II)[1]

```
import java.io.*;
import java.util.*;
class Blip3 implements Externalizable {
  int i;
  String s;
  public Blip3() {
    System.out.println("Blip3.Blip3()");
  }
  public Blip3(String x, int a) {
    System.out.println("Blip3.Blip3
                              (String, int)");
    s = x;
    i = a;
  }
  public String toString() {
    return "Blip3[i = " + i + "; s = "
                        + s + "]";
  }
  public void writeExternal(ObjectOutput
                                          out)
                            throws IOException {
    System.out.println("Blip3.writeExt");
    out.writeObject(s);
    out.writeInt(i);

  }
  public void readExternal(ObjectInput in)
                             throws IOException,
                      ClassNotFoundException {
    System.out.println("Blip3.readExt");
    s = (String)in.readObject();
    i = in.readInt();
  }
// continua
```


Externalizable (II)[2]

```
// continua
public static void main(String args[])
                          throws IOException,
                    ClassNotFoundException {
   Blip3 b3 = new Blip3("ciao", 1);
   ObjectOutputStream o =
     new ObjectOutputStream(
       new FileOutputStream("Blip3.out"));
   o.writeObject(b3);
   o.close();
   ObjectInputStream in =
     new ObjectInputStream(
       new FileInputStream("Blip3.out"));
   b3 = (Blip3)in.readObject();
   System.out.println(b3);
  }
} // class Blip3

// output
// Blip3.Blip3 (String, int)
// Blip3.writeExt
// Blip3.Blip3()
// Blip3.readExt
// Blip3[i = 1; s = ciao]
```

Flussi ad accesso casuale [1]

```
class Prodotto {
   private String nome;
   private double prezzo;
   private int score;

   public Prodotto(String unNome,
                   double unPrezzo, int unScore) {
       nome = unNome;
       prezzo = unPrezzo;
       score = unScore;
   }

   public Prodotto() {}

   public String toString() {
       return "Prodotto[nome = " + nome +
                                  "; prezzo = " + prezzo +
                                 "; score = " + score + "]";
   }

   String ritornaNome(){ return nome; }

   double ritornaPrezzo() { return prezzo; }

   int ritornaScore() {return score; }
} // Prodotto
```


Flussi ad accesso casuale [2]

```
public class RAStream {
   static final int LUNGHEZZASTRINGA = 20;
   static final int LUNGHEZZARECORD =
                        LUNGHEZZASTRINGA * 2 + 4 + 8;
   private final static String cammino = /* un path */ ;

   public static void main(String[] args)
   {
       Prodotto[] p = new Prodotto[] {
            new Prodotto("pippo", 2.5, 1),
            new Prodotto("pluto", 4.2, 3),
            new Prodotto("minnie", 5.7, 2)};
       String nomeFile = cammino + "Prodotti.dat";
       RandomAccessFile raf = null;

       try {
          raf = new RandomAccessFile(nomeFile,
                                                  "rw");
       } catch (FileNotFoundException e) {
          System.out.println(e);
       }

// continua
```

Flussi ad accesso casuale [3]

```
try {
    for (int i = 0; i < p.length; i++) {
        scriviStringaFissa(raf, p[i].ritornaNome(),
                           LUNGHEZZASTRINGA);
        raf.writeDouble(p[i].ritornaPrezzo());
        raf.writeInt(p[i].ritornaScore());
    }

    int record = 2; // posizione del  record
    long position = (record - 1) *
                            LUNGHEZZARECORD;
    raf.seek(position);

    String n = leggiStringaFissa(raf,
                           LUNGHEZZASTRINGA);
    double d = raf.readDouble();
    int i = raf.readInt();
    System.out.println(new Prodotto(n, d, i));
} catch(IOException e) {
    System.out.println(e);
}

// continua
```


Flussi ad accesso casuale [4]

```
        finally {
            if ( raf != null )
                try {
                    raf.close();
                } catch (IOException e) {
                    System.out.println("close failed: " +
                                                    e);
                }
        }

} // main
```

Flussi ad accesso casuale (5)

```
static void
  scriviStringaFissa(RandomAccessFile f,
                                    String s, int n)
      throws IOException
{
  if ( s.length() < n ) {
     f.writeChars(s);
     for ( int i = s.length(); i < n; i++ )
        f.writeChar(' ');
  }
  else f.writeChars(s.substring(0, n));
}

static String
leggiStringaFissa(RandomAccessFile f, int n)
      throws IOException
{
  String b = "";
  for ( int i = 0; i < n; i++ ) b += f.readChar();
  return b.trim();
}

} // class
```


Lettura formattata da tastiera (1)

```
import java.io.*;
import java.util.StringTokenizer;

public class LettoreTastiera extends BufferedReader
{
   private StringTokenizer stok = null;

   public LettoreTastiera(InputStream in)
   {
      super(new InputStreamReader(in));
   }

   private String prendiElemento()
            throws IOException {
      if ( (stok == null) || (!stok.hasMoreElements()) ) {
         String linea = readLine();
         stok = new StringTokenizer(linea, " ");
      }
      return (String)stok.nextElement();
   }

   double leggiDouble() throws IOException
   {
      return Double.parseDouble(prendiElemento());
   }
// continua
```

Lettura formattata da tastiera (2)

```
 int leggiInt() throws IOException
 {
    return Integer.parseInt(prendiElemento());
 }


} // class
```