Il linguaggio Java
Le eccezioni

Programmi d'esempio

Lettura di un file (I)

```
// Pseudo-codice
// La funzione legge un file e lo
// copia in memoria

readFile {
    open the file;
    determine its size;
    allocate that much memory;
    read the file into memory;
    close the file;
}


// cosa succede se:
// l'operazione di apertura fallisce
// la lunghezza dl file non può essere
//    determinata
// non può essere allocata memoria in
//     quantità sufficiente
// l'operazione di lettura fallisce
// l'operazione di chiusura fallisce
```

Lettura di un file (II)

```
// spaghetti code

errorCodeType readFile {
  initialize errorCode = 0;
  open the file;
  if (theFileIsOpen) {
     determine the length of the file;
     if (gotTheFileLength) {
        allocate that much memory;
        if (gotEnoughMemory) {
           read the file into memory;
           if (readFailed) { errorCode = -1; }
        }
        else { errorCode = -2; }
     }
     else { errorCode = -3; }
     close the file;
     if (theFileDidntClose && errorCode == 0) {
        errorCode = -4;}
     else { errorCode = errorCode and -4;}
  }
  else { errorCode = -5; }
  return errorCode;
}
```


Lettura di un file (III)

```
readFile {
  try {    // codice "normale"
       open the file;
       determine its size;
       allocate that much memory;
       read the file into memory;
       close the file;
  }    catch (fileOpenFailed) { // gestione errore
       doSomething;
  } catch (sizeDeterminationFailed) {
       doSomething;
  } catch (memoryAllocationFailed) {
       doSomething;
  } catch (readFailed) {
       doSomething;
  } catch (fileCloseFailed) {
       doSomething;
  }
}
```

Lettura di un file (IV)

```
// method1 è l'unico metodo interessato agli
// errori in readFile

method1 {
   call method2;
   resto del corpo 1;
}

method2 {
  call method3;
    resto del corpo 2;
}

method3 {
  call readFile;
    resto del corpo 3;
}
```


Lettura di un file (V)

```
method1 {
  errorCodeType error;
  error = call method2;
  if (error)
       doErrorProcessing;
 else
        resto del corpo 1;
}

errorCodeType method2 {
  errorCodeType error;
  error = call method3;
  if (error)
       return error;
  else
        resto del corpo 2;
}

errorCodeType method3 {
  errorCodeType error;
  error = call readFile;
  if (error)
       return error;
  else
        resto del corpo 3;
}
```

Lettura di un file (VI)

```
method1 {   // interessato agli errori in
            // readFile
  try {
      call method2;
  } catch (exception) {
      doErrorProcessing;
  }
}

method2 throws exception {
   call method3;
}

method3 throws exception {
   call readFile;
}
```


ListOfNumbers (vers. 1)

```
// questo programma non compila
// volutamente

import java.io.*;
import java.util.Vector;

public class ListOfNumbers {
 private Vector victor;
 private static final int size = 10;

 public ListOfNumbers () {
 victor = new Vector(size);
 for (int i = 0; i < size; i++)
    victor.addElement(
       new Integer(i));
 }

 public void writeList() {
    PrintWriter out = new PrintWriter(
         new FileWriter("OutFile.txt"));
    for (int i = 0; i < size; i++)
       out.println("Value at: " + i
                                  + " = "
                 + victor.elementAt(i));
    out.close();
 }
}
```

Il metodo writeList (vers. 2)

```
// exception are thrown

public void writeList() throws
                                IOException,
          ArrayIndexOutOfBoundsException {
 PrintWriter out =
   new PrintWriter(
    new FileWriter("OutFile.txt"));

 for (int i = 0; i < size; i++)
   out.println("Value at: " + i + " = " +
                     victor.elementAt(i));
   out.close(); // attenzione il file può
                // rimanere aperto
}
```


il metodo writeList (vers. 3)

```
// exceptions are caught

public void writeList() {
  PrintWriter out = null;
  try {
    System.out.println("Entering try
                                        statement");
    out = new PrintWriter(
                 new FileWriter("OutFile.txt"));
    for (int i = 0; i < size; i++)
      out.println("Value at: " + i
                 + " = " + victor.elementAt(i));
  } catch (ArrayIndexOutOfBoundsException e) {
    System.err.println("Caught
            ArrayIndexOutOfBoundsException: " +
                                    e.getMessage());
  } catch (IOException e) {
    System.err.println("Caught IOException: " +
                                    e.getMessage());
  } // to be completed
    // file may remain open
}
```

il metodo writeList (vers. 4)

```
// a complete version closing the file

public void writeList() {
  PrintWriter out = null;
  try {
    System.out.println("Entering try
                                    statement");
    out = new PrintWriter(
                 new FileWriter("OutFile.txt"));
    for (int i = 0; i < size; i++)
      out.println("Value at: " + i
                + " = " + victor.elementAt(i));
  } catch (ArrayIndexOutOfBoundsException e) {
    System.err.println("Caught
            ArrayIndexOutOfBoundsException: " +
                                e.getMessage());
  } catch (IOException e) {
    System.err.println("Caught IOException: " +
                                e.getMessage());
  } finally {
    if (out != null) {
      System.out.println("Closing PrintWriter");
      out.close();
    } else {
      System.out.println("PrintWriter
                                    not open");
    }
  }
}
```


Crearsi le proprie eccezioni (I)

```
class SimpleException extends Exception {}

public class SimpleExceptionDemo {

  public void f() throws SimpleException {
    System.out.println("Throwing
                SimpleException from f()");
    throw new SimpleException();
  }

  public static void main(String[] args) {
    SimpleExceptionDemo sed = new
    SimpleExceptionDemo();
    try {
      sed.f();
    } catch(SimpleException e) {
      System.err.println("Caught it!");
    }
  }
}

// output
// Throwing SimpleException from f()
// Caught it!
```

Crearsi le proprie eccezioni (II)

```
class SimpleException extends Exception {
    public SimpleException() {}
    public SimpleException(String message) {
        super(message);
    }
}
public class FullConstructors {
  public void f() throws SimpleException {
   System.out.println("Throwing SimpleException
                                        from f()");
    throw new SimpleException();
  }
  public void g() throws SimpleException {
    System.out.println("Throwing SimpleException
                                        from g()");
    throw new SimpleException("Originated in
                                        g()");
  }
  public static void main(String[] args) {
    FullConstructors fc =
                        new FullConstructors();
    try {
      fc.f();
    } catch(SimpleException e) {
      e.printStackTrace();
    }
    try {
      fc.g();
    } catch(SimpleException e) {
      e.printStackTrace();
    }
  }
}
```


Eccezioni padre e figlio

```
import java.io.*;
public class FatherException
                        extends IOException {
  public FatherException() { }
  public FatherException(String messaggio)
{
     super(messaggio);
  }
}
//////////////////////////////////////////
//////////////////////////////////////
public class SonException
                    extends FatherException {
  public SonException() { }
    public SonException(String messaggio) {
         super(messaggio);
  }
}
//////////////////////////////////////////
//////////////////////////////////////
public class Intermedia {
  void f(int option)
                     throws FatherException {
     if ( option > 0 )
        throw new FatherException("Padre");
     if ( option < 0 )
        throw new SonException("Figlio");
  }
}
```

La classe pilota

```
public class Eccezioni1 {
  public static void main(String[] args) {
     Intermedia i = new Intermedia();
     int opt = 1;
     try {
        System.out.println("f: inizio");
        i.f(opt);
        System.out.println("f: fine");
     }  catch (FatherException e) {
        System.out.println(e.getMessage());
     }
     System.out.println("main: fine");
   }
}

// opt = 1
// f: inizio
// Padre
// main: fine

// opt = 0
// f: inizio
// f: fine
// main: fine

// opt = -1
// f: inizio
// Figlio
// main: fine
```


La classe pilota

```
public class Eccezioni2 {
  public static void main(String[] args) {
    Intermedia i = new Intermedia();
    int opt = 1;
    try {
     i.f(opt);
    }  catch (FatherException e) {
     System.out.println(e.getMessage());
    } catch (SonException e) {
     System.out.println(e.getMessage());
    }
    System.out.println("main: fine");
   }
}

// Eccezioni1.java [XX:XX] exception
//  SonException
// has already been caught
```

Esempio: la classe pilota

```
public class Eccezioni3 {
  public static void main(String[] args) {
    Intermedia i = new Intermedia();
    int opt = 1;
    try {
      i.f(opt);
    } catch (SonException e) {
      System.out.println(e.getMessage());
    }
    System.out.println("main: fine");
   }
}

// Eccezioni3.java [24:1] unreported
// exception
// FatherException; must be caught or
// declared to be thrown
```