Excepciones y manejo de archivos

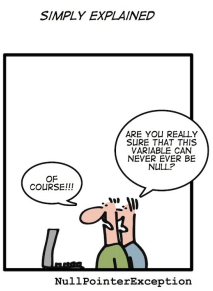
Facultad de Estudios Superiores Acatlán
Programa de Matemáticas Aplicadas y Computación

2

Excepciones

- Están asociadas a ”errores”
  - Ocurre durante la ejecución de un programa.
    - Interrumpe el flujo de control normal de éste
- Es un objeto de la clase `Exception`
  - Métodos
    - `getMessage()`
    - `printStackTrace()`
- Son generados por
  - Métodos de clases Java
  - El programador

```
throw new Exception(mensaje_error)
```



- El error no se evita, se maneja
  - Se usan las instrucciones `try`, `catch` y opcionalmente `finally`

Facultad de Estudios Superiores Acatlán
Programa de Matemáticas Aplicadas y Computación

4

Instrucciones manejo de errores.

```
try {
       // Conjunto de instrucciones que normalmente
       // se ejecuta sin problemas, pero puede
fallar
 } catch ( Clase_TipoExcepción1  objeto) {
      // Ejecuta código si se presentó
TipoExcepción1
 } catch ( Clase_TipoExcepción2  objeto) {
      // Ejecuta código si se presentó
TipoExcepción2
...
 } finally {
      // Siempre ejecuta este código al salir del
      // try. Se suele usar para "limpieza", como
      // cerrar archivos y liberar recursos
}
```


6

Ejemplo: Excepciones del programador

```
import java.util.Scanner;
public class MiExcepcion {
  public static void main(String[] args) {
    Scanner escaner = new Scanner(System.in);
    int calif;
    try {
       calif = escaner.nextInt();
       if (calif > 100 || calif < 0)
          throw new Exception("Fuera del límite 0-100");
       System.out.println("Calificación: "+(float)calif/10);
    }catch(Exception mi_error){
       System.err.println("Error: "+mi_error.getMessage());
    }finally{
            System.out.println("FIN");
   }
  }
}
```

```
run:
89
Calificación: 8.9
FIN
BUILD SUCCESSFUL (total time: 14 seconds)
```

```
run
121
FIN
Error: Fuera del límite 0-100
BUILD SUCCESSFUL (total time: 7 seconds)
```


8

Ejemplos de excepciones generadas por Java

```
public static void main(String[] args) {
   int a = 4, b=0;
   System.out.println(a/b);
}
```

provoca:

```
Exception in thread "main" java.lang.ArithmeticException: / by zero
```

```
public static void main(String[] args) {
      int [] array = new int [5];
      array[5] = 1;
}
```

provoca:

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 5
```

```
public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Introduce un número entero: ");
      int n = sc.nextInt();
      System.out.print("Número introducido: " + n);
  }
```

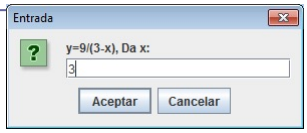
Se espera un int. Si por ejemplo se lee 4,5 provoca:

```
Exception in thread "main" java.util.InputMismatchException
    at java.util.Scanner.throwFor(Scanner.java:909)
    at java.util.Scanner.next(Scanner.java:1530)
    at java.util.Scanner.nextInt(Scanner.java:2160)
    at java.util.Scanner.nextInt(Scanner.java:2119)
    at excepciones1.Excepciones1.main(Excepciones1.java:7)
```

Tomado de: http://puntocomnoesunlenguaje.blogspot.mx/2014/04/java-excepciones.html

10

Ejemplo: error desde un método



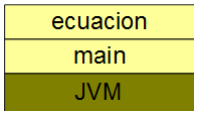

```
import javax.swing.JOptionPane;
public class Errores {
  public static int ecuacion(int x){
    return 9/(3-x);
  }
  public static void main(String[] args) {
    String valor;
    valor=JOptionPane.showInputDialog("y=9/(3-x),Da x:");
    int x=Integer.parseInt(valor);
    JOptionPane.showMessageDialog(null,"y="+ecuacion(x));
  }
}
```

```
Exception in thread "main" java.lang.ArithmeticException: / by zero
    at presenta.Errores.ecuacion(Errores.java:14)
    at presenta.Errores.main(Errores.java:20)
Java Result: 1
```

Mensaje error

Pila de métodos: Stack Trace

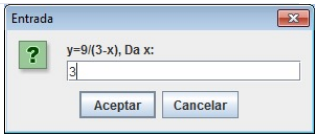
Si se presenta una excepción el método lanza la excepción, fuera de él, traspasando la responsabilidad al método que lo invocó o si es el último a la JVM

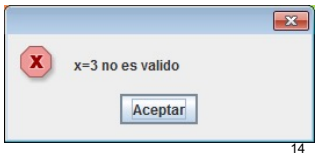


12

Manejando el error aritmético

```
import javax.swing.JOptionPane;
public class ErroresM {
  public static int ecuacion(int x){
    return 9/(3-x);
  }
  public static void main(String[] args) {
    String valor;
    valor=JOptionPane.showInputDialog("y=9/(3-x), Da x: ");
    int x=Integer.parseInt(valor);
    try{
        JOptionPane.showMessageDialog(null,"y="+ecuacion(x));
    }catch(ArithmeticException error){
        JOptionPane.showMessageDialog(null,"x="+x+" no es
                        valido","",JOptionPane.ERROR_MESSAGE);
    }
  }
}
```





14

Ejemplo: error generado por la clase Scanner

```
import java.util.*;
public class ErrorLectura {
  public static void main(String[] args) {
    Scanner escaner = new Scanner(System.in);
    try {
       System.out.print("Apellido? ");
       String apellido=escaner.next();
       System.out.print("Calificación? ");
       float calif=escaner.nextFloat();
    }catch(InputMismatchException e){
       System.err.print("Error en tipo de dato: ");
       System.err.print("Usar primer palabra apellido");
    }
  }
}
```

```
run:

Error en tipo de dato: Usar primer palabra apellido
BUILD SUCCESSFUL (total time: 6 seconds)
```


16

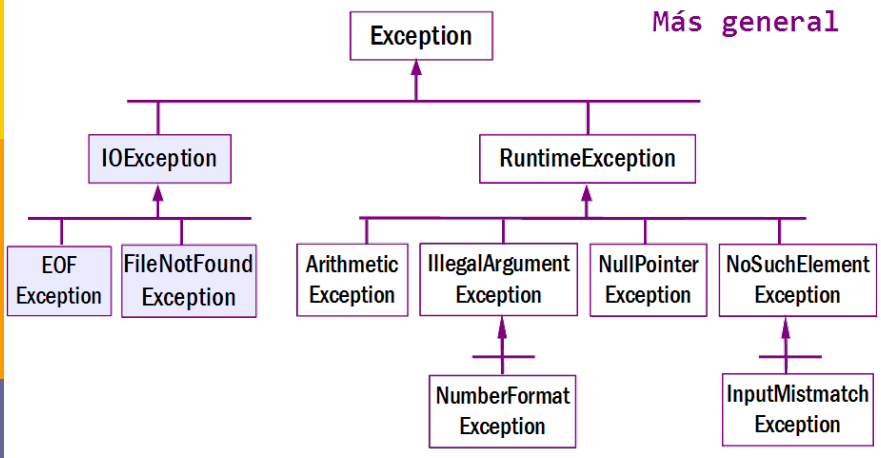
Excepciones más comunes

Tipo Excepción	Significado
IOException	Incluye muchas excepciones de E/S
FileNotFoundException	Archivo no encontrado
EOFException	Final de archivo antes de lo esperado
RuntimeException	Incluye muchas excepciones de ejecución
IndexOutOfBoundsException	Acceso a un elemento inexistente de un vector o de un String
NullPointerException	Intento de uso de una referencia nula
ArithmeticException	Desbordamiento o división entera por cero
IllegalArgumentException	Se envía un parámetro de tipo diferente al esperado
NumberFormatException	Conversión ilegal de un String a un tipo numérico
InputMismatchException	Error en tipo de dato en la clase Scanner

Excepciones marcadas — Los métodos que las generen, siempre van dentro de un try

18

Parte de la jerarquía de excepciones



Nota: Colocar los bloques catch de la clase más específica a la más general

20

Creando varias excepciones

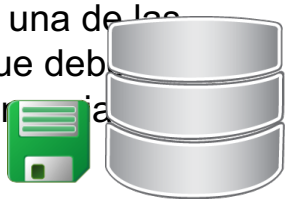
```
public class PruebaTiempo {
    public static void main(String[] args) {
        Reloj digital = new Reloj();
        Scanner lee = new Scanner(System.in);
        System.out.print("Introduce la hora hh:mm:ss  ");
        String[] hora = lee.next().split(":");
        try{
            switch (hora.length) {
                case 1: digital = new Reloj(Integer.parseInt(hora[0])); break;
                case 2: digital = new Reloj(Integer.parseInt(hora[0]),
                                            Integer.parseInt(hora[1])); break;
                case 3: digital = new Reloj(Integer.parseInt(hora[0]),
                                            Integer.parseInt(hora[1]),
                                            Integer.parseInt(hora[2])); break;
                default: throw new Exception ("Formato incorrecto");
            }
            if (digital.validarHora())digital.imprime();
            else throw new Exception ("Error en rango de datos !!!");
        } catch (NumberFormatException e){
            System.err.println("Error de datos !!!..."+e.getMessage());
        }catch (Exception mi_error){
            System.err.println(mi_error.getMessage());
        }
    }
}
```

22

22

Archivos

- Las computadoras utilizan archivos para la retención a largo plazo de datos
- Los archivos se almacenan en dispositivos secundarios
  - Discos magnéticos
  - Discos ópticos
  - Cintas magnéticas
- El procesamiento de archivos en una de las herramientas más importantes que debe un lenguaje para aplicaciones co



Acatlán

Carmen Villar / René Martínez

24

24

Archivos en Java

- Existen más de 20 clases para manejar archivos
- Se dividen en dos tipos principales
- Flujo (stream)
  - Una secuencia de datos, procesados uno después de otro
- Acceso aleatorio (random)
  - Accesar los datos directamente, es decir, saltarse los datos anteriores y llegar a una posición exacta (un byte específico) en el archivo
  - Se suele usar en accesos a bases de datos

Clase File

- Define métodos para conocer propiedades de archivos
  - Puede ser un directorio o un archivo
- Constructores
  - `File(String nombre_completo)`
    - `File miArchivo= new File("C:\Libro\LaOdisea.dat")`
  - `File(String directorio, String nombre`
    - `File receta= new File("Cocina","Pasta.dat")`
  - `File(File ruta, String nombre)`
    - `File dir=new File("C:\Java\Practicas")`
    - `File archivo= new File(dir,"Datos.dat")`

Clase File (cont)

- `File(URI uri)`
  - `File archivo= new File(file//C:/datos.txt)`

30

Información de un archivo

- Métodos de File
  - `boolean exists()`
  - `boolean canWrite()`
  - `boolean canRead()`
  - `boolean isFile()`
  - `boolean isDirectory()`
  - `boolean isAbsolute()`
  - `long length()`
  - `String getAbsolutePath()`
  - `String getName()`
  - `String getParent()`

Es una buena práctica crear objetos File, para hacer controles previos al archivo, en lugar de sólo enviar el nombre del archivo a los constructores de flujo.

32

Flujos en Java

- Un flujo (stream) es una abstracción que se refiere a un flujo o corriente de datos
  - En Java, un archivo es un flujo secuencial de bytes
    - Contiene un marcador como fin de archivo
- Cuando se ejecuta un programa, siempre abre 3 objetos flujo relacionados con dispositivos
  - `System.in`
  - `System.out`
  - `System.err`
  - Se pueden redireccionar
    - `setIn, setOut, setErr`

Clases de entrada y salida

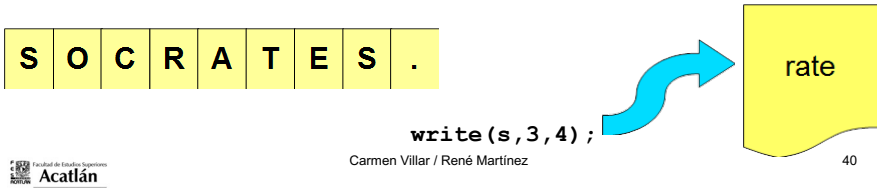
- Los flujos de datos se clasifican en
  - Flujos de entrada en bytes
    - `InputStream` (clase abstracta)
      - `FileInputStream, ByteArrayInputStream, DataInputStream ...`
  - Flujo de salida de bytes
    - `OutputStream`
      - `FileOutputStream, ByteArrayOutputStream, DataOutputStream`
  - Flujo de entrada y salida basados en caracteres
    - `Reader` y `Writer` (clases abstractas)
      - `BufferedReader, BufferedWriter, CharArrayReader, CharArrayWriter…`

FileOutputStream

- Escribe bytes en un flujo asociado a un archivo
- Métodos más usados
  - Constructores
    - `FileOutputStream(String nombre) throws IOException`
      - Crea un objeto inicializado con el nombre del archivo
  - `FileOutputStream(String nombre, boolean s) throws IOException`
    - Crea un objeto inicializado con el nombre del archivo.
    - Si `s=true` los bytes escritos se añaden al final
  - `FileOutputStream(File nombre) throws IOException`
    - Crea un objeto inicializado con el objeto archivo

FileOutputStream (cont.)

- `int write(byte a) throws IOException`
  - Escribe el byte “a” en el flujo asociado
- `int write(byte[ ] s) throws IOException`
  - Escribe el arreglo de bytes
  - La clase `String` tiene un método `getBytes()`, que permite convertir un `String` en un arreglo de bytes
- `int write(byte[ ] s, int pos, int long) throws IOException`
- Escribe el arreglo “s” de bytes, desde la posición “pos” y un máximo de “long” bytes



FileInputStream

- Lee bytes desde un archivo
- Métodos más usados
  - `FileInputStream(String nombre) throws FileNotFoundException`
    - Crea un objeto inicializado con el nombre del archivo
  - `FileInputStream(File nombre) throws FileNotFoundException`
    - Crea un objeto inicializado con el objeto archivo
  - `int read() throws IOException`
    - Lee un byte. Devuelve -1 si alcanza el fin de archivo

FileInputStream (cont.)

- `int read(byte[ ] s) throws IOException`
  - Lee una secuencia de bytes y los almacena en un arreglo “s”. Devuelve el número de bytes leídos o -1 si es fin de archivo
- `int read(byte[ ] s, int pos, int long) throws IOException`
  - Almacena en “s”, a partir de “pos”, un máximo de “long” bytes

Ejemplo lectura/escritura

```
run:
Leyendo el archivo
AATTCGGCGTAGCTTATAGACCCGA
BUILD SUCCESSFUL (total time
```

```
import java.io.*;
public class ArchivoGenoma {
    public static void main(String[] args) {
        double aleatorio;
        try{
            FileOutputStream archivo=new FileOutputStream("genoma.txt");
            for(int i=0; i<25; i++){
                aleatorio=Math.random();
                if(aleatorio<0.25) { archivo.write((byte)'A'); }
                else if (aleatorio<0.5) { archivo.write((byte)'G'); }
                else if(aleatorio<0.75) { archivo.write((byte)'C'); }
                else { archivo.write((byte)'T'); }
            }
            System.out.println("Leyendo el archivo");
            FileInputStream arch=new FileInputStream("genoma.txt");
            int c;
            while((c=arch.read())!=-1) // mientras no sea fin de archivo
                System.out.print((char)c);
            System.out.println();
        }
        catch(IOException e){
            System.err.print("Problemas con el archivo");
        }
    }
}
```

genoma 05/04/2016 11:54 a... Documento de tex... 1 KB

Escribiendo un archivo a partir de un String

```
import java.io.*;
public class ArchivoStream {
  public static void main(String[] args) {
    String frase="Solo se que no se nada";
    byte [] s;
    try{
       FileOutputStream arch=new
                                FileOutputStream("sal.txt");
       s=frase.getBytes();
       arch.write(s);
       arch.write((byte)'\n');
       arch.close();
    }
    catch(IOException e){
        System.out.print("Problemas con archivo");
    }
  }
}
```

sal.txt 29/07/2014 04:53 ... Documento de tex... 1 KB

Agregando un archivo

```
run:
¿Quien dijo Solo sé que no sé nada?
Socrates
BUILD SUCCESSFUL (total time: 8 seconds)
```

```
import java.io.*;
import java.util.Scanner;
public class Agrega {
  public static void main(String[] args) {
    Scanner lector = new Scanner(System.in);
    try{
        FileOutputStream arch=new
                        FileOutputStream("sal.txt",true);
        System.out.println("¿Quien dijo Solo sé que no sé
                            nada? ");
        String autor=lector.next();
        arch.write(autor.getBytes());
        arch.write((byte)'\n');
        arch.close();
    }catch(IOException e){
       System.out.print("Problemas con el archivo");
    }
  }
}
```

Leyendo un archivo

```
import java.io.*;
public class LeeArchivo {
  public static void main(String[] args) {
    int c;
    try{
       FileInputStream arch=new FileInputStream("sal.txt");
       //mientras no sea fin de archivo
       while((c=arch.read())!=-1)
           System.out.print((char)c);
     }catch(IOException e){
           System.out.print("Problemas con el archivo");
     }
  }
}
```

```
run:
Solo se que no se nada
Socrates

BUILD SUCCESSFUL (total time: 0 seconds)
```

Try con recursos

```
import java.io.*;
public class ArchivoStream {
    public static void main(String[] args) {
        String frase="Solo se que no se nada";
        byte [] s;
        FileOutputStream arch=new FileOutputStream("sal.txt");
        s=frase.getBytes();
        arch.write(s);
        arch.write((byte)'\n');
        arch.close();
    } catch(IOException e){
        System.out.print("Problemas con archivo");
    }
  }
}
```

Convert to try-with-resources

(Alt-Enter shows hints)

```
import java.io.*;
public class ArchivoStreamTryR {
    public static void main(String[] args) {
        String frase="Solo se que no se nada";
        byte [] s;
        try (FileOutputStream arch = new FileOutputStream("sal.txt")) {
            s=frase.getBytes();
            arch.write(s);
            arch.write((byte)'\n');
        } catch(IOException e){
            System.out.print("Problemas con archivo");
        }
    }
}
```

The try-with-resources statement is a try statement that declares one or more resources. A resource is an object that must be closed after the program is finished with it. The try-with-resources statement ensures that each resource is closed at the end of the statement.

Facultad de Estudios Superiores Acatlán

54

Uso clase Scanner para archivos

- Importar paquete `java.util`
- Declarar un objeto de tipo `Scanner`
- Crear objeto con parámetro un objeto de tipo `File`
  - Nombre del archivo
- Usar un método `nexttipo` para leer el dato
  - `next()`
  - `nextLine()`
  - `nextFloat()`
  - `next Double`
  - `nextByte()`
  - `nextInt()`
  - `nextLong()`
  - `nextShort()`

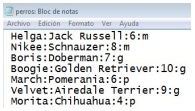
56

Uso clase Scanner para archivos

- Leer hasta fin de archivo
  - `hasNexttipo`
- Excepción
  - `FileNotFoundException`

Lecturas usando clase Scanner

```
import java.util.Scanner;
import java.io.*;
public class ConScanner {
   public static void main(String[] args) {
      File archivo = new File("perros.csv");
      String perro,raza;
      int edad, numero=0;
      char clase;
      try{
          Scanner lector = new Scanner(archivo);
          lector.useDelimiter(":|\n");
          System.out.println("Lista participantes");
          while(lector.hasNext){
             perro = lector.next();
             raza = lector.next();
             edad = lector.nextInt();
             clase = lector.next().charAt(0);
             numero++;
             System.out.println(numero+"."+perro+"("+raza+")");
          }
      }catch(FileNotFoundException e){
          System.out.println("No se encontró archivo");
      }
   }
}
```




```
run:
Lista participantes
1. Helga(Jack Russell)
2. Nikee(Schnauzer)
3. Boris(Doberman)
4. Boogie(Golden Retriever)
5. March(Pomerania)
6. Velvet(Airedale Terrier)
7. Morita(Chihuahua)
BUILD SUCCESSFUL (total time:
```

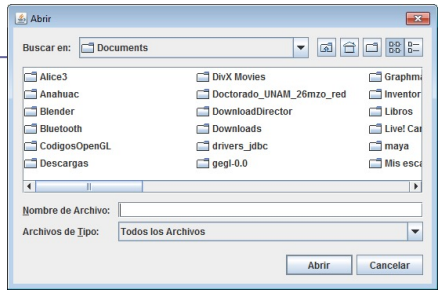
JFileChooser

- Permite abrir y guardar archivos

```
java.lang.Object
  └java.awt.Component
      └java.awt.Container
          └javax.swing.JComponent
              └javax.swing.JFileChooser
```

- Importar
  - `javax.swing.JFileChooser;`
- Constructores
  - `JFileChooser()`
  - `JFileChooser(String ruta)`
  - `JFileChooser(File ruta)`

Constantes y métodos



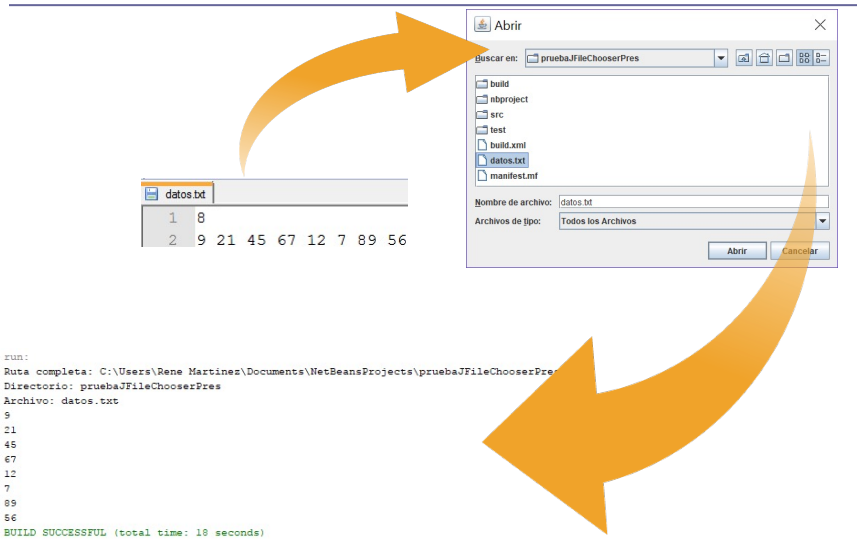
- Constantes
  - `APPROVE_OPTION`
  - `CANCEL_OPTION`
  - `ERROR_OPTION`
- Métodos
  - `File getSelectedFile()`
  - `File[] getSelectedFiles()`
  - `File getCurrentDirectory()`
  - `showOpenDialog(Component padre)`
  - `showSaveDialog(Component padre)`
  - `setDialogTitle(String titulo)`

Ejemplo JFileChooser

```
public class PruebaJFileChooserPres {
  public static void main(String[] args) {
    JFileChooser selector = new JFileChooser();
    File archivo, directorio;
    int estado = selector.showOpenDialog(null);
    if (estado == JFileChooser.APPROVE_OPTION) {
      try{
          archivo = selector.getSelectedFile();
          directorio = selector.getCurrentDirectory();
          System.out.println("Ruta completa: "+archivo.getParent());
          System.out.println("Directorio: "+directorio.getName());
          System.out.println("Archivo: "+archivo.getName());
          Scanner escaner = new Scanner(archivo);
          int num = escaner.nextInt();
          for(int i=0; i<num; i++)
             System.out.println(escaner.nextInt());
       }catch (FileNotFoundException e){
          System.out.println("Error en la lectura");
       }catch (NoSuchElementException e){
          System.out.println("Error en formato o tipo de archivo");
       }
    } else System.out.println("Se cancelo abrir archivo");
  }
}
```


66

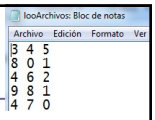
Ejecución. PruebaJFile



68

Ejercicio

- Edita un archivo con una lista de 5 elementos, con 3 coordenadas enteras, separados por espacios
- Modifica `PruebaJFileChooser` pa
  - Abrir con `JFileChoser` el archivo
  - Escribir si es un archivo
  - Escribir el nombre del archivo
  - Leer los datos y almacenarlos:
    - En una matriz
    - En un arreglo de tipo Vector3D
  - Imprimir la matriz y los vectores



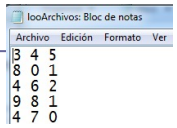

```
run:
Es un archivo: true
Archivo: looArchivos.txt
3 4 5
8 0 1
4 6 2
9 8 1
4 7 0
[3,4,5]
[8,0,1]
[4,6,2]
[9,8,1]
[4,7,0]
BUILD SUCCESSFUL (total t
```

```
run:
Es un archivo: true
Archivo: looArchivos.txt
Valor negativo en datos
BUILD SUCCESSFUL (total time
```

```
run:
Es un archivo: true
No se puede escribir
BUILD SUCCESSFUL (total
```


Ejercicio (cont)

- Modifica el programa para generar las siguientes excepciones creadas por ti
  - Existe un número negativo en la matriz
  - El archivo no se puede escribir




```
run:
Es un archivo: true
Archivo: looArchivos.txt
3 4 5
8 0 1
4 6 2
9 8 1
4 7 0
[3,4,5]
[8,0,1]
[4,6,2]
[9,8,1]
[4,7,0]
BUILD SUCCESSFUL (total t
```

```
run:
Es un archivo: true
Archivo: looArchivos.txt
Valor negativo en datos
BUILD SUCCESSFUL (total time
```

```
run:
Es un archivo: true
No se puede escribir
BUILD SUCCESSFUL (total
```