Java Avanzado

Manejo de Excepciones

Copyright

Gestión de errores en C

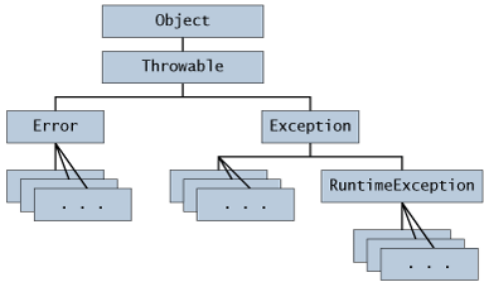
- Los errores en C se pueden:
  - Ignorarlos.
  - Usar una variable global errno.
  - Devolver un código de error: rc = function();
  - Pasar una estructura de error como parámetro:

```
ErrorStruct err;
function(x, y, &err);
```

Gestión de errores en Java

- Utiliza las Excepciones Java.
- Una excepción es una condición anormal que se produce en una porción de código durante su ejecución.
- La existencia de las excepciones permite:
  - Encapsular en clases los errores.
  - Separar el flujo de ejecución normal del tratamiento de errores.
- Las excepciones se pueden: o tratar o relanzar para que sean tratadas por otro método del stack.

Jerarquía de clases



Jerarquía de clases

- La clase java.lang.Throwable describe cualquier clase que pueda ser lanzada como excepción.
- Existen dos tipos de clases Throwable:
  - java.lang.Error representa errores de compilación y errores del sistema.
  - java.lang.Exception representa las excepciones generadas por la aplicación.
- Existe un tipo especial de clases Exception:
  - java.lang.RuntimeException representa excepciones generadas por la aplicación cuya gestión no es obligatoria.

Excepciones verificadas vs. no verificadas

- Verificadas (checked):
  - Su tratamiento es obligatorio y el compilador así lo chequea.
  - Todas aquellas clases hijas de java.lang.Exception que no lo sean de java.lang.RuntimeException
- No verificadas (unchecked):
  - Su tratamiento no es obligatorio y el compilador no lo chequea.
  - Todas aquellas clases hijas de java.lang.Error o de java.lang.RuntimeException

Manejo de Excepciones

- Se manejan mediante bloques try & catch:

```
try
{
   // Código susceptible de lanzar una excepción.
}
catch(tipo-excepción ex)
{
   // Código de tratamiento de la excepción.
}
catch(tipo-excepción ex)
{
   // Código de tratamiento de la excepción.
}
finally
{
   // Código que se ejecuta siempre.
}
```

Manejo de Excepciones

- El código susceptible de lanzar o producir una excepción debe estar incluido en un bloque *try*.

```
try
{
   // Código susceptible de lanzar una excepción.
}
```

- El código a ejecutar para tratar una excepción concreta debe estar incluido en un bloque *catch*.

```
catch(Exception ex)
{
   // Código de tratamiento de la excepción.
}
```

Manejo de Excepciones

- El código a ejecutar independientemente de que se produzca o no una excepción debe estar incluido en un bloque *finally*.

```
finally
{
   // Código que se ejecuta siempre.
}
```

- Atención con las variables locales definidas en cualquiera de los bloques porque no son accesibles desde fuera.

Ejemplo

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;

public class ExceptionTest
{
   public static void main(String[] args)
   {
      File f = new File("c:\\test.txt)");
      FileInputStream fis = null;

      try
      {
         fis = new FileInputStream(f);
      }
      catch(FileNotFoundException ex)
      {
         System.out.println("Fichero no encontrado.");
      }
      finally
      {
         try
         {
            fis.close();
         }
         catch(IOException ex)
         {
            System.out.println("Error al cerrar el fichero.");
         }
      }
   }
}
```

Los bloques try & catch pueden estar anidados en cualquiera de las estructuras.

Lanzamiento de excepciones

- Un método indica que puede lanzar o (relanzar) una excepción mediante la palabra *throws*.
- Un método crea y lanza una excepción mediante la palabra *throw*.

```
public class FileInputStream extends Inputstream
{
   public FileInputStream(File aFile) throws IOException
   {
      if(...)
      {
         throw new IOException("No existe el fichero.");
      }
   }
}
```

Lanzamiento de excepciones

- Si una excepción es relanzada mediante *throws* consecutivamente y llega a la JVM, esta se detendrá mostrando la excepción.

```
Exception occurred during event dispatching:
java.lang.NumberFormatException: test
at java.lang.FloatingDecimal.readJavaFormatString(FloatingDecimal.java:1194)
at java.lang.Double.parseDouble(Double.java:198)
at edu.upco.einf.practica12.EuroConversor.actionPerformed(EuroConversor.java:65)
at javax.swing.AbstractButton.fireActionPerformed(AbstractButton.java:1461)
at javax.swing.AbstractButton$ForwardActionEvents.actionPerformed(AbstractButton.java:1515)
at javax.swing.DefaultButtonModel.fireActionPerformed(DefaultButtonModel.java:392)
at javax.swing.DefaultButtonModel.setPressed(DefaultButtonModel.java:264)
at javax.swing.plaf.basic.BasicButtonListener.mouseReleased(BasicButtonListener.java:254)
at java.awt.Component.processMouseEvent(Component.java:3799)
at java.awt.Component.processEvent(Component.java:3628)
at java.awt.Container.processEvent(Container.java:1202)
at java.awt.Component.dispatchEventImpl(Component.java:2678)
at java.awt.Container.dispatchEventImpl(Container.java:1251)
at java.awt.Component.dispatchEvent(Component.java:2581)
at java.awt.LightweightDispatcher.retargetMouseEvent(Container.java:2496)
at java.awt.LightweightDispatcher.processMouseEvent(Container.java:2261)
at java.awt.LightweightDispatcher.dispatchEvent(Container.java:2170)
at java.awt.Container.dispatchEventImpl(Container.java:1238)
at java.awt.Window.dispatchEventImpl(Window.java:964)
at java.awt.Component.dispatchEvent(Component.java:2581)
at java.awt.EventQueue.dispatchEvent(EventQueue.java:434)
at java.awt.EventDispatchThread.pumpOneEventForHierarchy(EventDispatchThread.java:234)
at java.awt.EventDispatchThread.pumpEventsForHierarchy(EventDispatchThread.java:141)
at java.awt.EventDispatchThread.pumpEvents(EventDispatchThread.java:132)
at java.awt.EventDispatchThread.run(EventDispatchThread.java:124)
```

Ejemplo de definición

```
public class FactorNegativoException extends Exception
{
   public FactorNegativoException(double param)
   {
      super("Has introducido un valor erroneo: " + param);
   }
}
```

Nota: java.lang.Exception tiene varios constructores, pero el más usado es el que recibe un String con el motivo de la excepción. Este String será accesible vía el método getMessage().

Ejemplo de lanzamiento

```
.....

public escalar(double factor) throws FactorNegativoException
{
   if (factor < 0)
      throw new FactorNegativoException(factor);
   else
      radio *= factor;
}

.....
```

Ejemplos de tratamiento

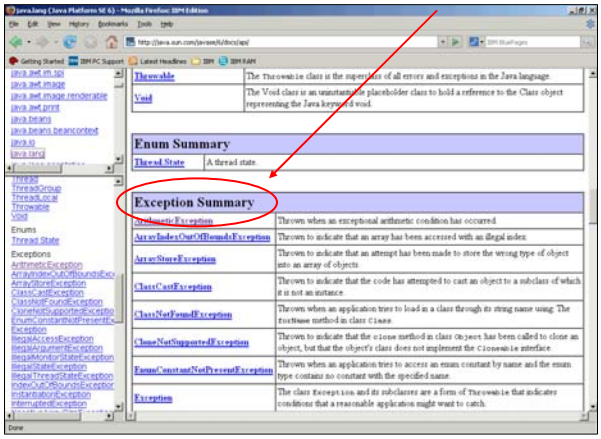
```
public Circulo unMetodo(double r, double f) throws FactorNegativoException
{
   Circulo c = new Circulo(r);          // Ejemplo relanzando la excepción.
   c.escalar(f);
   return c;
}

public Circulo unMetodo(double r, double f)
{
   Circulo c = new Circulo(r);          // Ejemplo tratando la excepción.
   try
   {
      c.escalar(f);
      return c;
   }
   catch (FactorNegativoException ex)
   {
      // Tratamiento de la excepción.
   }
}
```

Algunos métodos útiles

- public void printStackTrace();
  - Imprime por la salida estándar la pila (stack) de llamadas incluyendo los números de línea y ficheros donde se ha producido la excepción.
- public String getMessage();
  - Devuelve una cadena de caracteres con la descripción de la excepción.
- public String toString();
  - Devuelve la representación en cadena de caracteres de la excepción.

Algunas excepciones/errores

- Todas las excepciones y errores Java tienen un un apartado específico dentro del API.



Algunas excepciones/errores

- java.lang.NoClassDefFoundError (no verificada)
  - Lanzada cuando se intenta instanciar una clase y no se encuentra.
- java.lang.IllegalArgumentException (no verificada)
  - Lanzada cuando se llama a un método con un parámetro erróneo.
- java.lang.IndexOutOfBounds (no verificada)
  - Lanzada cuando se intenta acceder a una posición inexistente en una colección (array, vector, etc.....).
- java.lang.InterruptedException (verificada)
  - Lanzada cuando se despierta un thread que estaba *sleep*.

Algunas excepciones/errores

- java.lang.NoSuchFieldError (no verificada)
  - Lanzada cuando se intenta acceder a un atributo inexistente.
- java.lang.NoSuchMethodError (no verificada)
  - Lanzada cuando se intenta acceder a un método inexistente.
- java.lang.NullPointerException (no verificada)
  - Lanzada cuando se accede a un objeto cuyo valor es null.
- java.lang.NumberFormatException (no verificada)
  - Lanzada cuando se intenta convertir una cadena de caracteres a un número y tiene caracteres no numéricos.

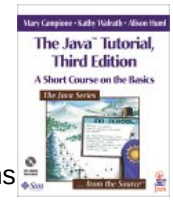
Algunas excepciones/errores

- java.lang.OutOfMemoryError (no verificada)
  - Lanzado cuando la JVM no puede instanciar un objeto por falta de memoria física.
- java.lang.UnsatisfiedLinkError (no verificada)
  - Lanzada cuando se intenta acceder a una DLL del sistema inexistente.
- java.lang.ClassCastException (no verificada)
  - Lanzada cuando se realiza un casting de un objeto mediante una clase de la que no es instancia.
- java.lang.ArithmeticException (no verificada)
  - Lanzada cuando se produce un situación anómala en una operación matemática (por ejemplo divisiones por 0).

Algunas excepciones/errores

- java.io.IOException (verificada)
  - Lanzado cuando ocurre un problema de I/O genérico.
- java.io.EOFException (verificada)
  - Lanzada cuando se detecta el final de un stream de manera inesperada.
- java.io.FileNotFoundException (verificada)
  - Lanzada cuando se intenta acceder a un fichero en modo lectura y no se encuentra.
- javax.swing.UnsupportedLookAndFeelException (verificada)
  - Lanzada cuando se intenta utilizar un Look & Feel inexistente en el sistema.

Bibliografía



- Learning Java (3rd edition)
  Patrick Niemever y Jonathan Knudsen.
  O'Reilly.
- Java Cookbook (2nd edition)
  Ian F. Darwin.
  O'Reilly.
- Java in a Nutshell (5th edition)
  David Flanagan.
  O'Reilly.
- The Java tutorial (3rd edition)
  Mary Campione, Kathy Walrath y Alison Huml.
  Addison-Wesley.
- The Java tutorial (on-line)
  http://java.sun.com/docs/books/tutorial/essential/exceptions