Programación de Sistemas

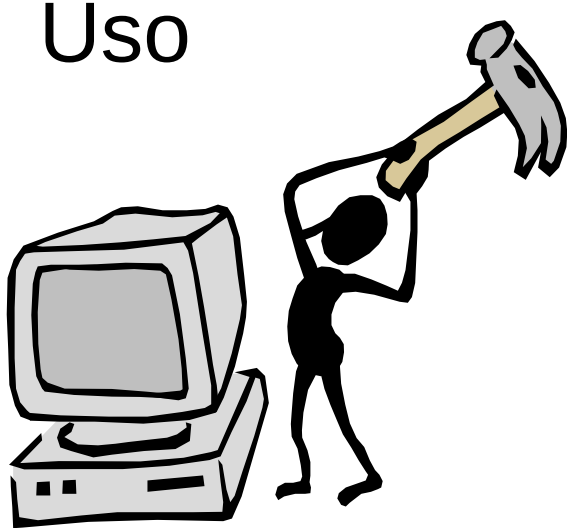
Excepciones

Julio Villena Román
<jvillena@it.uc3m.es>

Excepciones

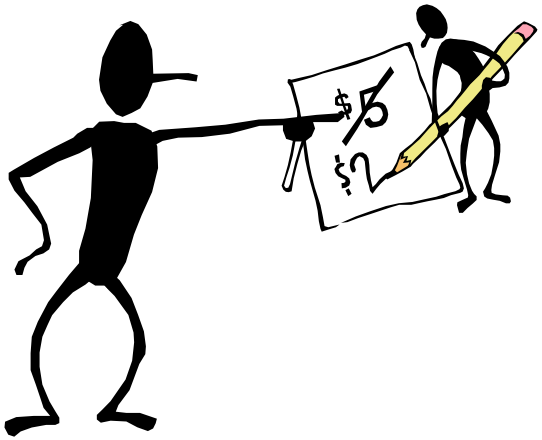
- Qué son
- Propósito
- Tipos
- Uso

Exception



Ignorar ➡ Fin

Catch



Throw

Excepciones: Qué son

- **Eventos** que afectan a la ejecución normal del programa (habitualmente errores)
- Cuando ocurre una excepción, se crea un **objeto Exception** y se pasa al sistema de control de ejecución de Java
- El sistema de control de ejecución:
  - **Busca** un fragmento de código que maneje dicha excepción recibida
  - Si no hay ningún código manejador, el programa **termina** con mensaje en consola

Excepciones: Propósito

- Para **separar** el código de gestión de errores (`try-catch`) del resto del código
- Para **propagar** erroress en la pila de llamadas (`throws`)
- Para **agrupar** y diferenciar entre sí distintos tipos de errores (como las excepciones son objetos, pueden agruparse en clases)
- Cada método en Java debe:
  - O `capturar` (catch)
  - O `lanzar` (throws)

  cualquier excepción que pueda producirse durante la ejecución

Excepciones: Tipos

- Dos tipos principales:
  - Excepciones en tiempo de ejecución (`RuntimeException`)
    - No se comprueban en compilación
    - Ej: `NullPointerException`, `ArithmeticException`, `NumberFormatException`, `IndexOutOfBoundException`…
  - Excepciones comprobadas en compilación
    - Ej: Excepciones de entrada/salida (`IOException`, `FileNotFoundException`, `EOFException`)
    - **Definidas por el usuario** (`MyException`)
- En compilación, se comprueba que toda excepción (excepto excepciones runtime) :
  - Se **capture**
  - O se **declare** que el método puede lanzarla

Excepciones: Uso

- Las excepciones ocurren:
  - Implícitamente (cuando hay un error)
  - Explícitamente: `throw new MyException(message)`
- Qué hacer:
  - **Manejar la excepción**:
    - Incluir en un bloque `try{}` las sentencias que puedan generar excepciones
    - Incluir en un bloque `catch(MyException e){}` las sentencas a ejecutar para manejar la excepción
  - **Lanzar la excepción** (con declaración en el método)**:**
    - `public void myMethod throws MyException`
- El bloque `finally{}` incluye el código que se ejecuta siempre para cualquier excepción

Excepciones: Ejemplos

```
try {
  // Se ejecuta algo que puede producir una excepción
} catch (IOException e) {
  // manejo de una excepción específica
} catch (Exception e) {
  // manejo de una excepción cualquiera
} finally {
  // código a ejecutar haya o no excepción
}
```

```
public class MyException extends Exception {
  public MyException (String msg) {
    super(msg);
  }
}
```

```
public int myMethod(int param) throws MyException {
  if(param<0) throw new MyException("Número negativo");
  int res = 10/param;
  return res;
}
```

```
public int myMethod(int param) throws MyException {
  try {
    int res = 10/param;
    return res;
  } catch(ArithmeticException e) {
    throw new MyException("Número negativo");
  }
}
```