J2SE
Excepciones en Java

Antonio Espín Herranz

Control de Errores: try catch

- Las excepciones de producen cuando:
  - El fichero que se quiere abrir no existe.
  - La conexión de red se ha perdido.
  - Los operandos que se manejan se salen de rango.
  - El fichero de clase que se quiere cargar no existe.

- Un error de clase define unas condiciones de error muy serias.

Excepciones

Ejemplo: Podemos anidar varios **catch** y los **try** también.

```
try {
    // Bloque de código que se monitorizan los errores.
} catch (TipoExcepcion1 exOb){
    // Gestión de la excepción
} catch (TipoExcepcion2 exOb2){
    // Gestión de la excepción.
}
```

Excepciones mas frecuentes

- ArithmeticException: Error al realizar una operación aritmética, por ejemplo, una división por cero.
- NullPointerException: Uso incorrecto de una referencia a null.
- NegativeArraySizeException: El array tiene un tamaño negativo.
- ArrayIndexOutOfBoundsException: Nos hemos salido fuera de los límites del array.
- SecurityException: Intento de violación de seguridad.
- ClassNotFoundException: No se ha encontrado la clase.

Excepciones

- La clase **Exception** (hereda de la clase Throwable → Lanzable) define las condiciones de error con los que los programas se pueden encontrar.

- La gestión de excepciones en Java se llevan a cabo mediante 5 palabras clave: **try**, **catch**, **throw**, **throws** y **finally**.

Excepciones

Forma de uso:

```
try {
   // Operaciones que pueden dar lugar a
   // Excepciones.
} catch (Exception e){
   System.out.println("Se produjo el error: " +
   e.getMessage());
}
```

Excepciones

- **finally**: No es obligatorio, pero si forma parte de la Excepción se ejecuta siempre.

```
try {
    // Bloque de código que se monitorizan los errores.
} catch (TipoExcepcion1 exOb){
    // Gestión de la excepción
} catch (TipoExcepcion2 exOb2){
    // Gestión de la excepción.
} finally {
    // Este bloque se ejecuta siempre.
}
```

También podemos anidar los try.

Excepciones

- **throw** instancia
  - Nuestro programa puede lanzar excepciones.
- Ejemplo:
  - throw new NullPointerException(“demo”);
  - Incluso nos podemos diseñar una clase de Excepciones propia y lanzarlas.
- **throws** lista de excepciones: Indicamos que un método lanza excepciones. Cuando un método da lugar a una excepción que no es capaz de gestionar él mismo, lo que hacemos es propagar la excepción hacia arriba
- Ejemplo:
  - void miMetodo() throws MiExcepcion { … }
- **Ver Ejemplo1**

Excepciones

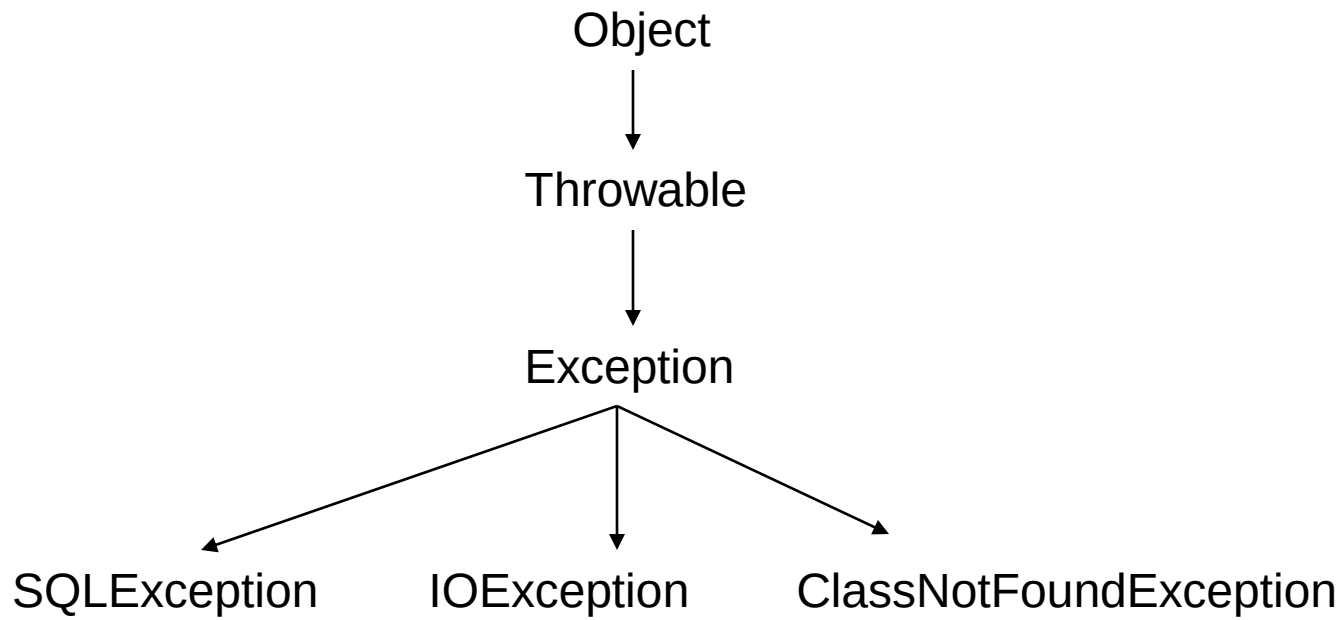
- Ejemplo de relanzar una Excepción.

```
class Ejemplo1 {
  static void demoproc() {
    try {
      throw new NullPointerException("demo");
    } catch(NullPointerException e) {
      System.out.println("Capturada dentro de demoproc.");
      throw e; // Se relanza la excepción.
    }
  }

  public static void main(String args[]) {
    try {
      demoproc();
    } catch(NullPointerException e) {
      System.out.println("Recaught: " + e);
    }
  }
}
```

Excepciones

- Jerarquía de clases de Exception



Excepciones

- **OJO: cuando coloquemos los catch anidados tenemos que tener la precaución de primero capturar las excepciones de las subclases.**

- **Si capturamos por ejemplo Exception y por debajo tenemos ArithmeticException nunca será capturada cuando se produzca un error.**

- **OJO con los ámbitos de las variables.**

Creación de nuestras propias Excepciones

- Java permite la creación de nuestras propias excepciones. Para crear una excepción lo que haremos es heredar de la clase **Exception**.

- class MiException extends **Exception**

Ejemplo

```
class MyException extends Exception {
  private int detail;

  MyException(int a) {
    detail = a;
  }

  public String toString() {
    return "MyException[" + detail + "]";
  }
}
```

Ejemplo (continuación)

```
public class ExceptionDemo {
  static void compute(int a) throws MyException {
    System.out.println("Called compute(" + a + ")");
    if(a > 10)
      throw new MyException(a);
    System.out.println("Normal exit");
  }

  public static void main(String args[]) {
    try {
      compute(1);
      compute(20);
    } catch (MyException e) {
      System.out.println("Caught " + e.toString());
    }
  }
}
```

Prácticas: Excepciones

Multicatch

- A partir de Java 1.7 las excepciones se pueden agrupar en el mismo catch

```
try {
    // instruccion1
    // instruccion2
    // …
    // instrucción N

} catch (exception1 | exception2 | … | exceptionN ex) {
    // Tratamiento del error.
}
```