Programación Orientada a Objetos

Excepciones

Dr. Said Polanco Martagón[1]

Universidad Politécnica de Victoria

October 20, 2015

[1]E-mail address: spolancom@upv.edu.mx

Outline

Declaración de la excepción

Los métodos que pueden provocar excepciones marcadas, deben declarar éstas en la definición del método.
Para declarar una excepción se utiliza la palabra *throws*, seguida de la lista de excepciones que el método puede provocar:

Example

```
public String readLine() throws IOException
public void service(...) throws
                    ServletException, IOException
```

Así, que siempre que vayamos a utilizar algún método que tenga declaras excepciones, hemos de tener presente que estamos obligados a capturar dichas excepciones.

Excepciones no marcadas

Pertenecen a este grupo todas las excepciones de tiempo de ejecución, es decir, *RuntimeException* y todas sus subclases.

Nota

No es obligatorio capturar dentro de un programa Java una excepción no marcada, el motivo es que gran parte de ellas se producen como consecuencia de una mala programación.

Nota

Sólo las excepciones de tipo *ArithmeticException* es recomendable capturarlas.

nota

Si durante la ejecución de un programa Java se produce una excepción y ésta no es capturada, la máquina virtual provoca la finalización inmediata del mismo, enviando a la consola el volcado de la pila con los datos de la excepción.

Example

```
public class Division{
    public static void main(Sring[] args) {
        int k = 4/0;
    }
}
```

Outline

Captura de excepciones

El mecanismo de captura de excepciones de Java, permite "atrapar" el objeto de excepción lanzado por la instrucción e indicar las diferentes acciones a realizar según la clase de excepción producida.
A diferencia de las excepciones, los errores representan fallos de sistema de los cuales el programa no se puede recuperar.

Outline

Manejo de excepciones

Las instrucciones *try, catch y finally*, proporcionan una forma elegante y estructurada de capturar excepciones dentro de un programa Java, evitando la utilización de instrucciones de control que dificultarían la lectura del código y lo harían más propenso a errores.

Sintaxis

La sintaxis para la utilización de estas instrucciones se indica a continuación

Example

```
try
{
    //instrucciones donde se pueden producir excepciones
}
catch(TipoExcepcion1 arg)
{
    /tratamiento excepción1
}
catch(TipoExcepcion2 arg)
{
    //tratamiento excepción2
}
finally
{
    //instrucciones de última ejecución
}
```

TRY

El bloque *try* delimita aquella o aquellas instrucciones donde se puede producir una excepción. Cuando esto sucede, el control del programa se transfiere al bloque *catch* definido para el tipo de excepción que se ha producido, pasándole como parámetro la excepción lanzada. Opcionalmente, se puede disponer de un bloque *finally* en el que definir un grupo de instrucciones que obliga ejecución.

CATCH

Un bloque *catch* define las instrucciones que deberán ejecutarse en caso de que se produzca un determinado tipo de excepción.
Sobre la utilización de los bloques *catch*, se debe tener en cuenta lo siguiente:

- Se puede definir tanto bloques *catch* como se considere necesario. Cada bloque *catch* servirá para tratar un determinado tipo de excepción, **no pudiendo haber dos o más catch que tengan declarada la misma clase de excepción.**

- Un bloque *catch* sirve para capturar cualquier excepción que se corresponda con el tipo declarado o cualquiera de sus subclases. Por ejemplo, un *catch* como el siguiente:

Example

```
catch(RuntimeException e) {
    :
}
```