Control de excepciones

La clase Throwable

Es la clase que lanza “bengalas SOS” cuando la JVM no es capaz de gestionar algún problema surgido durante la ejecución de un programa.

De esta clase heredan:

- Error: Se genera cuando se produce algún tipo de error grave no gestionable por el programador (no hay que “ocuparse” de ellas, ya que poca solución tienen).

- Exception: Se genera cuando algo sale mal en la ejecución del programa porque no es posible ejecutar alguna sentencia del código y el programa “pide ayuda”. Es responsabilidad del programador controlar esas situaciones que pueden presentarse en tiempo de ejecución.

- Hay excepciones comprobadas por el compilador (no podremos compilar si no hemos escrito el código que las controla) y no controladas (el compilador no puede saber que van a producirse, por lo que es el programador quien tiene que detectar su existencia y peligrosidad). Estas últimas se producen en tiempo de ejecución (heredan de RunTimeException).

Métodos del control de excepciones

La estrategia de control de las excepciones consiste en “perimetrar con un código de seguridad la ejecución de un código que podría dar problemas y no llevarse a cabo”.

Dicho de otra forma, “suavizamos el tono imperativo”. En vez de decirle a la JVM “haz esto” (si no es capaz de hacerlo el programa finalizará con un mensaje de error), por “intenta esto”, y si no puedes haz esto otro.

Las palabras para eso son

```
try{ bloque de código a intentar ejecutar}

catch(TipoExcepcion e){ bloque de código a ejecutar si el del try no ha podido completarse}

finally{código a ejecutar SIEMPRE, tanto si el try sale bien como si no}
```

La existencia de finally es optativa.

throw

Una excepción puede ser lanzada automáticamente por la JVM, pero también puede ser lanzada de forma “manual” durante la ejecución si el programador así lo ha previsto para alguna situación determinada.

Además, es posible crear nuestra propia excepción, si lo consideramos necesario. Para ello, tendremos que crear una clase que herede de Throwable (o de alguna de sus subclases).

Cuando queramos lanzar una excepción, bastará con crear una instancia de esa clase y decirle al código que la lance. Podemos lanzar estas excepciones (o cualquiera de las existentes en la API) utilizando el comando throw de la siguiente forma

throw (new miExcepcion());

pudiendo variar según nuestras necesidades el tipo de clase o los argumentos que pasamos al constructor.

throws

Hay que tener en cuenta que cuando se lanza una excepción, esta se propaga “jerárquicamente” sí no se controla. Es decir, si una sentencia produce una excepción en un método y este método no la controla, se propaga al método que llamó a este hasta que se encuentra dónde controlarla.

Por tanto, si un método puede lanzar una excepción (comprobada), deberemos “avisar del peligro” indicando en la definición del método que puede lanzar excepciones y de qué tipo son. Esto se hace mediante throws en la cabecera del método. Por ejemplo:

```
public void miMetodoExplosivo() throws miExcepcion{código del método}
```

Tipos de excepciones (clases y subclases)

El control de excepciones puede hacerse de forma general o específica según nos interese.

Pueden introducirse tantas cláusulas catch como nos interese, capturando cada una de ellas el tipo de excepción que indique, ejecutándose únicamente ese bloque de código.

Hay que tener en cuenta que la primera expresión que “encaje” con la excepción lanzada será el único que se ejecute, por lo que hay que tener cuidado con el orden en el que se escriben, poniendo las más específicas primero y las más generales después.

La API de java contiene toda la información necesaria para poder implementar el código adecuadamente. Un ejemplo sería:

```
try{código que nos puede generar excepciones}

catch(IndexOutOfBoundsException i){código a ejecutar si hay un error “saliéndose” de un array}

catch(ArithmeticException a){código a ejecutar si hay un error aritmético}

catch(Exception e){código a ejecutar si hay una excepción general}
```