U.D.6

Programación orientada a objetos. Objetos

(3ª parte: Control y manejo de excepciones)

PRO – IES EL MAJUELO

ÍNDICE

1. Concepto de excepción

- Son los posibles errores que se pueden dar durante la ejecución del programa, debido a situaciones extraordinarias que pudieran ocurrir:
  - El programa pide un número y el usuario inserta letras.
  - Se quiere abrir un fichero que no existe.
  - Se pretende hacer una división y el denominador vale cero.
  - Se quiere usar un objeto que no apunta a ningún espacio de memoria.
  - Etc.
- Es importante que el programa se realice teniendo en cuenta estas situaciones.
  - Por ejemplo, si el usuario inserta letras cuando se le ha pedido un número se podría plantear:
    - cerrar el programa con un mensaje apropiado.
    - dar la oportunidad al usuario de que vuelva a intentar introducirlo.

2. Jerarquía de excepciones

Todas las excepciones que se pueden producir en un programa están representadas mediante clases que derivan de la clase *Throwable*. Dicha clase se divide en dos grupos:

- Clase *Error*. Errores internos, anormales, del sistema, de difícil recuperación.
- Clase *Exception*. Errores de ejecución por situaciones anormales que se pueden dar. Clases derivadas:
  - *RuntimeException*. No es obligatorio controlar este tipo de errores.
    - *NullPointerException*, *ArithmeticException*, *IlegalArgumentException* (*NumberFormatException*), …
  - Resto de clases. JAVA obliga a controlar estos errores.
    - *ClassNotFoundException*, *InterruptedException*, *FontFormatException*, *IOException* (*FileNotFoundException*), …

3. Manejo de excepciones

a. Captura

```
try
{
  /* Instrucciones que pueden provocar algún
  error, si se dan ciertas situaciones. */
}
catch(TipoError e)
{
  /* Instrucciones que se ejecutan en el caso
  de que se produjera un error del tipo
  indicado. */
}
```

Captura de excepciones

- Si se produce el error, dejarían de ejecutarse las instrucciones que hay en el bloque *try* para ejecutar las instrucciones que hay en el bloque *catch*.
- Puede haber más de un bloque *catch*, cada uno de ellos se encargará de capturar un tipo de error.
  - Cuando el programa ejecute un bloque *catch* no ejecutará nada de los siguientes (por ello, el orden es importante).
- El bloque *finally* es opcional y contiene instrucciones que se ejecutan siempre, haya o no haya ocurrido un error. Tiene que estar al final de todos los bloques *catch*.

3. Manejo de excepciones
b. Propagación

- Cuando se produce un error dentro de un método, se puede capturar dentro de él y tratarlo allí mismo. Si no se captura dicho error, JAVA lo propaga al método que lo invocó para que sea éste el que lo capture. El proceso se repite sucesivamente hasta el método *main*.
- Sólo se propagan los errores que se derivan de la clase *RuntimeException*.
  - Los errores de la clase *Exception* que no pertenezcan a *RuntimeException* no se propagan y es obligatorio capturarlos o lanzarlos.

3. Manejo de excepciones
c. Lanzamiento

Dos situaciones:

- Dentro de un método donde se pueden producir uno de los errores de los que JAVA obliga a controlar. Dos opciones:
  - Capturar el error con try-catch.
  - Lanzar el error, poniendo una cabecera al método.

```
tipoDevuelto nombreMetodo(argumentos) throws
        NombreClaseException1, ...
```

- Dentro de un método podemos querer controlar un error pero puede suceder que también queramos controlarlo fuera de dicho método.

```
throw (nombre_objeto.tipo_Exception);
```

3. Manejo de excepciones
d. Clases de excepciones

Podemos crear nuestras clases de excepciones si las que propone JAVA no nos sirven para lo que queremos hacer.

```
class <nombre_clase> extends
   <nombre_de_una_clase_existente_de_tipo_Exception>
```