Introducción a Ficheros

Conociendo el control de errores.

Para introducirnos en esta parte lo primero hay que conocer el concepto de control de error. No el error en sí, sino su control, ya que hasta el momento hemos tenido la oportunidad de ver como se producen los errores y como "rompen" la ejecución del programa sin que podamos hacer nada. El control de errores pretende evitar que esto suceda, pero no que no suceda el error, si no que podamos continuar con la ejecución del programa como si el error no se hubiera producido aunque lo haya hecho.

Cada lenguaje tiene una forma de controlar los errores y en nuestro caso concreto al estar utilizando java es necesario conocer la estructura try-catch-finally. Es una estructura muy sencilla que lo que permite es asegurar bloques de código en los que, susceptiblemente nos puede saltar un error.

De momento, tenemos que saber que hay dos tipos de errores:

- En tiempo de ejecución: Son los que se producen cuando se ejecuta el programa y el IDE no puede preveer antes de que ocurran. Ejemplo, el desbordamiento de un array, si yo tengo un array de diez posiciones pero escribo esta sintaxis arr[10], en el IDE sintácticamente es correcto, pero a la hora de ejecutarse va a "romperse" la ejecución. Conclusión, el IDE no nos ayuda a detectar estos errores con lo que habrá que ir descubriendo cuando y bajo que circunstancias se producen.
- En tiempo de diseño o compilación: Estos errores se producen cuando estamos escribiendo el código, si no tuviéramos un IDE con las herramientas adecuadas que nos marcan nuestros errores, los descubriríamos al momento de compilar, que es el proceso que se hace antes de ejecutar el programa. El compilador es una herramienta que comprueba la sintaxis del código escrito, la valida y si es correcta hace la conversión al código ejecutable y finalmente se ejecuta. Si escribimos el código en un bloc de notas sufriremos muy a menudo errores de sintáxis que pasaríamos por alto.
  Además, en este caso, el IDE nos indica bloques que obligatoriamente tienen que ser protegidos, suministrándonos él mismo el código como veremos en el ejemplo.

Como es su sintaxis.

La palabra reservada try abre el bloque, en él escribimos nuestro código.

Luego tenemos la palabra reservada catch, que es el nombre de una función y que tiene como parámetro el tipo de error y un nombre para él.

Por último, tenemos el bloque finally, que **siempre** se ejecutura independientemente del error que ocurra. Sirve para cerrar flujos de ficheros, cerrar conexiones de bases de datos, parar hilos… Todo lo necesario que haya que hacer para garantizar el buen tratamiento de los elementos del programa.

```
try {
    // Código que pueda generar Errores ("Exception's")
     } catch(Tipo1 id1) {
       // Manejar "Exception's" para la Clase Tipo1
     } catch(Tipo2 id2) {
       // Manejar "Exception's" para la Clase Tipo2
     } finally {
       // Actividades que siempre ocurren
     }
```