8.A. Excepciones.

1. Excepciones.

1.1. Capturar una excepción.

Para poder capturar excepciones, emplearemos la estructura de captura de excepciones `try-catch-finally`.

Básicamente, para capturar una excepción lo que haremos será declarar bloques de código donde es posible que ocurra una excepción. Esto lo haremos mediante un bloque try (intentar). Si ocurre una excepción dentro de estos bloques, se lanza una excepción. Estas excepciones lanzadas se pueden capturar por medio de bloques `catch`. Será dentro de este tipo de bloques donde se hará el manejo de las excepciones.

Su sintaxis es:

```
try {
código que puede generar excepciones;
} catch (Tipo_excepcion_1 objeto_excepcion) {
Manejo de excepción de Tipo_excepcion_1;
} catch (Tipo_excepcion_2 objeto_excepcion) {
Manejo de excepción de Tipo_excepcion_2;
}
...
finally {
instrucciones que se ejecutan siempre
}
```

En esta estructura, la parte `catch` puede repetirse tantas veces como excepciones diferentes se deseen capturar. La parte `finally` es opcional y, si aparece, solo podrá hacerlo una sola vez.

Cada `catch` maneja un tipo de excepción. Cuando se produce una excepción, se busca el `catch` que posea el manejador de excepción adecuado, será el que utilice el mismo tipo de excepción que se ha producido. Esto puede causar problemas si no se tiene cuidado, ya que la clase Exception es la superclase de todas las demás. Por lo que si se produjo, por ejemplo, una `excepción` de tipo `AritmethicException` y el primer `catch` captura el tipo genérico Exception, será ese `catch` el que se ejecute y no los demás.

Por eso el último `catch` debe ser el que capture excepciones genéricas y los primeros deben ser los más específicos. Lógicamente si vamos a tratar a todas las excepciones (sean del tipo que sean) igual, entonces basta con un solo `catch` que capture objetos Exception.

Ejercicio resuelto

Realiza un programa en Java en el que se solicite al usuario la introducción de un número por teclado comprendido entre el 0 y el 100. Utilizando manejo de excepciones, debes controlar la entrada de dicho número y volver a solicitarlo en caso de que ésta sea incorrecta.

Solución:

```
/*
 * Ejercicio resuelto sobre manejo de excepciones.
 * El programa solicita que el usuario introduzca por teclado
 * un número entre 0 y 100, debiendo gestionarse la entrada
 * por medio de excepciones.
 */

import java.io.*;
```

```
public class ejercicio_resuelto_excepciones {
    public static void main(String[] args){
        int numero=-1;
        int intentos=0;
        String linea;

        BufferedReader teclado = new BufferedReader(new InputStreamReader(System.in));

        do{
            try{
                System.out.print("Introduzca un número entre 0 y 100: ");
                linea = teclado.readLine();
                numero = Integer.parseInt(linea);

            }catch(IOException e){
                System.out.println("Error al leer del teclado.");

            }catch(NumberFormatException e){
                System.out.println("Debe introducir un número entre 0 y 100.");

            }finally{
                intentos++;
            }
        }while (numero < 0 || numero > 100);

        System.out.println("El número introducido es: " + numero);
        System.out.println("Número de intentos: " + intentos);

    }
}
```

Se han utilizado estructuras try-catch-finally. En este programa se solicita repetidamente un número utilizando una estructura do-while, mientras el número introducido sea menor que 0 y mayor que 100. Como al solicitar el número pueden producirse los errores siguientes:

- De entrada de información a través de la excepción IOException generada por el método readLine() de la clase BufferedReader.

- De conversión de tipos a través de la excepción NumberFormatException generada por el método parseInt().

Entonces se hace necesaria la utilización de bloques catch que gestionen cada una de las excepciones que puedan producirse. Cuando se produce una excepción, se compara si coincide con la excepción del primer catch. Si no coincide, se compara con la del segundo catch y así sucesivamente. Si se encuentra un catch que coincide con la excepción a gestionar, se ejecutará el bloque de sentencias asociado a éste.

Si ningún bloque catch coincide con la excepción lanzada, dicha excepción se lanzará fuera de la estructura try-catch-finally.

El bloque finally, se ejecutará tanto si try terminó correctamente, como si se capturó una excepción en algún bloque catch. Por tanto, si existe bloque finally éste se ejecutará siempre.

Autoevaluación

Si en un programa no capturamos una excepción, será la máquina virtual de Java la que lo hará por nosotros, pero inmediatamente detendrá la ejecución del programa y mostrará una traza y un mensaje de error. Siendo una traza, la forma de localizar dónde se han producido errores. ¿Verdadero o Falso?

- ○ Verdadero
- ○ Falso