2.2. Gestión de excepciones

Acceso a Datos

Alejandro Roig Aguilar
alejandro.roig@iesalvarofalomir.org

IES Álvaro Falomir
Curso 2023-2024

¿Qué es una excepción?

Una excepción es un evento anormal que ocurre durante la ejecución de un programa.

Pueden ser causadas por situaciones imprevistas como divisiones por cero, acceso a índices fuera de rango, archivos que no se pueden encontrar o errores de red, entre otros. Por ejemplo:

```
public static void main(String[] args) {
    int a = 5, b = 0;
    System.out.println(a + “/” + b + “=” + a/b);
}
```

Al ejecutar este programa se obtendrá algo similar a lo siguiente:

Exception in thread “main” java.lang.ArithmeticException: / by zero

Gestión de excepciones

La gestión de excepciones implica detectar, manejar y recuperarse de errores en el código. Tenemos dos opciones:

1. Capturar Excepciones (try-catch)
    - El bloque try contiene el código que podría arrojar una excepción.
    - El bloque catch especifica cómo manejar la excepción si ocurre.

Estructura try-catch

```
Inicialización y asignación de recursos
try {
        Cuerpo
} catch (Excepcion_Tipo_1 e1) {
        Gestión de excepción tipo 1
} catch (Excepcion_Tipo_2 e2) {
        Gestión de excepción tipo 2
} catch (Excepcion e) {
        Gestión del resto de tipos de excepciones
}
finally { // Bloque a ejecutar antes de abandonar la ejecución ante error
        Finalización y liberación de recursos
}
```

Ejemplo de try-catch

```
try {
        BufferedWriter bw = new BufferedWriter(new FileWriter(nomFich));
        bw.write(str);
} catch (IOException e) {
        // gestión de la excepción
} finally {
        try {
                if (bw != null)
                        bw.close();
        } catch (IOException e) {
                // gestión de la excepción
        }
}
```

Gestión de excepciones

2. Lanzar Excepciones (throw):
    - Podemos lanzar excepciones manualmente con throw.
    - Es útil para indicar una excepción personalizada o cuando una condición específica debe generar una excepción.
    - Si un método puede generar una excepción que no desea manejar, puede declarar la excepción con la cláusula throws en la firma del método.
    - Esto pasa la responsabilidad de manejar la excepción al código que llama al método. No tratamos la excepción sino que delegamos su tratamiento.

Ejemplo de throw/throws

```
public void dividir(int valor1, int valor2) throws Exception {
        // Código que podría lanzar Exception
        if (valor2 == 0)
                throw new Exception("No se puede dividir por cero");
        int division = valor1/valor2;
        System.out.println("La división es " + division);
}
```

Gestión de excepciones

Desde Java 7 tenemos disponible una alternativa al try-catch clásico para leer/escribir un archivo u otros recursos: el try-with-resources.

try-with-resources es útil porque se asegura de que cada recurso que se abre se cierre automáticamente al final de la declaración, lo que puede ayudar a prevenir errores y fugas de recursos.

Estructura try-with-resources

```
Inicialización y asignación de recursos
try (recursos) {
        Cuerpo
} catch (Excepcion_Tipo_1 e1) {
        Gestión de excepción tipo 1
} catch (Excepcion_Tipo_2 e2) {
        Gestión de excepción tipo 2
} catch (Excepcion e) {
        Gestión del resto de tipos de excepciones
}
```

Ejemplo de try-with-resources

```
try (BufferedWriter bw = new BufferedWriter(new FileWriter(nomFich))) {
        bw.write(str);
} catch (IOException e) {
        // gestión de la excepción
}
```

```
BufferedWriter bw = new BufferedWriter(new FileWriter(nomFich));
try (bw) {
        bw.write(str);
} catch (IOException e) {
        // gestión de la excepción
}
```