Gestión de excepciones

¿Qué es una excepción?

Evento que ocurre durante la ejecución de un programa y que interrumpe su curso normal de ejecución. Por ejemplo:

```
public static void main(String[] args) {
        int a = 5, b = 2;
        System.out.println(a + "/" + b + "=" + a/b);
        b = 0;
        System.out.println(a + "/" + b + "=" + a/b);
}
```

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

```
5/2=2
Exception in thread "main" java.lang.ArithmeticException: / by zero
```

¿Qué hacer con las excepciones?

Ante una excepción no gestionada, el programa aborta su ejecución. Por lo tanto, es necesario tratarlas. Tenemos dos opciones:

- Capturar la excepción. La destruimos y mostramos un mensaje controlado y personalizado.
- Lanzar la excepción. En este caso no la tratamos sino que delegamos en que se trate en otro lugar del código.

Capturar excepciones

Cuando un código lanza una excepción, se puede capturar mediante un bloque try...catch:

- Try: el código que podría causar la excepción.
- Catch: el código que responde al error.

```
try {
    int a = 5, b = 0;
    System.out.println(a + "/" + b + "=" + a/b);
} catch (ArithmeticException ex) {
    System.err.println("Error al dividir por 0: " + ex.getMessage());
}
```

Lanzar excepciones

Lanzamos una excepción mediante throw con la idea de que la función main sea quien la trate.

```
public static int dividir(int n1, int n2) {
	if (n2 == 0)
		throw new ArithmeticException("No se puede dividir por 0");
	else
		return n1 / n2;
}
```

Excepciones con recursos

Es muy frecuente que un programa de Java esté estructurado así:

Inicialización y asignación de recursos

Cuerpo

Finalización y liberación de recursos

La primera (inicialización y asignación) y última parte (finalización y liberación) se han de ejecutar siempre... ¿Cómo gestionamos aquí las excepciones?

Excepciones con recursos

```
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-finally

```
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
      }
}
```

Excepciones con recursos AutoCloseable

```
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
}
```

ó

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