Programación en Java – Clase 8

Excepciones

Una excepción en Java es un evento que ocurre durante la ejecución de un programa y que interrumpe el flujo normal de la ejecución. Puede ser causada por una variedad de situaciones, como errores de programación, condiciones inesperadas o problemas con el entorno de ejecución.

En Java, las excepciones se manejan utilizando el mecanismo de "try-catch". A continuación, veremos sus componentes y funcionamiento:

1. **Bloque "try":**
   Dentro del bloque **try**, se coloca el código que potencialmente podría generar una excepción. El código dentro de este bloque se ejecuta de forma normal hasta que se produce una excepción.

```
try
{
        // Código que puede generar una excepción
}
catch (TipoDeExcepcion1 excepcion1)
{
        // Manejo de la excepción 1
}
catch (TipoDeExcepcion2 excepcion2)
{
        // Manejo de la excepción 2
}
finally
{
        // Código a ejecutar siempre, ocurra o no una excepción
}
```

2. **Bloques "catch"**:
   Después del bloque **try**, podemos tener uno o varios bloques **catch** para manejar excepciones específicas. Cada bloque **catch** especifica el tipo de excepción que puede ser capturada y el código que se ejecuta cuando se produce dicha excepción.

3. **Bloque "finally" (opcional):**
   Podemos usar un bloque **finally** en forma opcional que se ejecutará siempre, ocurra o no una excepción. Es útil para realizar tareas de limpieza o liberación de recursos.

En caso de que se produzca una excepción dentro del bloque **try**, el control del programa salta al bloque **catch** correspondiente. Si no se encuentra un bloque **catch** para manejar la excepción específica, la excepción se propaga hasta que se encuentre un bloque **catch** adecuado o el programa termine su ejecución.

Veamos un ejemplo:

```
try
{
        int resultado = 10 / 0; // Genera una excepción ArithmeticException
}
catch (ArithmeticException e)
{
        System.out.println("Se produjo una excepción: " + e.getMessage());
}
finally
{
        System.out.println("El bloque finally se ejecuta siempre.");
}
```

En este ejemplo, la división por cero genera una excepción del tipo **ArithmeticException**. El programa captura la excepción en el bloque **catch** correspondiente y muestra un mensaje de error. Luego, el bloque **finally** se ejecuta independientemente de si se produjo una excepción o no.

El manejo de excepciones en Java es fundamental para controlar situaciones excepcionales y garantizar la robustez y estabilidad de los programas. Permite manejar los errores de manera controlada y tomar acciones adecuadas para recuperarse de ellos o notificarlos al usuario.

Manejo de archivos

En Java podemos trabajar con archivos utilizando la API provista por el lenguaje. La manipulación de archivos se realiza a través de las clases File, FileReader, FileWriter, BufferedReader y BufferedWriter, entre otras.

A continuación, presentamos una serie de pasos para manejar archivos:

1. *Creación de un objeto del tipo* **File**:
   El primer paso es crear un objeto **File** que representa el archivo en el sistema de archivos. Podemos proporcionar la ruta completa del archivo o simplemente el nombre del archivo, en cuyo caso se ubicará en el directorio de trabajo actual.

   ```
   File archivo = new File("ruta_del_archivo/nombre_archivo.txt");
   ```

2. *Lectura de archivos:*
   Para leer el contenido de un archivo, podemos utilizar las clases **FileReader** y **BufferedReader**. Primero, debemos crear un objeto del tipo **FileReader** pasando el objeto **File** como argumento. Luego, crearemos un objeto **BufferedReader** para facilitar la lectura de líneas.

   ```
   try
   {
       FileReader fileReader = new FileReader(archivo);
       BufferedReader bufferedReader = new BufferedReader(fileReader);
       String linea;
       while ((linea = bufferedReader.readLine()) != null) {
           System.out.println(linea);
       }
       bufferedReader.close();
   }
   catch (IOException e)
   {
       e.printStackTrace();
   }
   ```

3. **Escritura en archivos:**
   Para escribir en un archivo, podemos utilizar las clases **FileWriter** y **BufferedWriter**. Similar a la lectura, primero creamos un objeto **FileWriter** pasando el objeto **File** como argumento. Luego, creamos un objeto `BufferedWriter` para facilitar la escritura de líneas.

```
try
{
        FileWriter fileWriter = new FileWriter(archivo);
        BufferedWriter bufferedWriter = new BufferedWriter(fileWriter);

        bufferedWriter.write("Línea 1");
        bufferedWriter.newLine();
        bufferedWriter.write("Línea 2");

        bufferedWriter.close();
}
catch (IOException e)
{
         e.printStackTrace();
}
```

Es importante manejar las excepciones adecuadamente al trabajar con archivos utilizando bloques `try-catch` o propagando las excepciones.