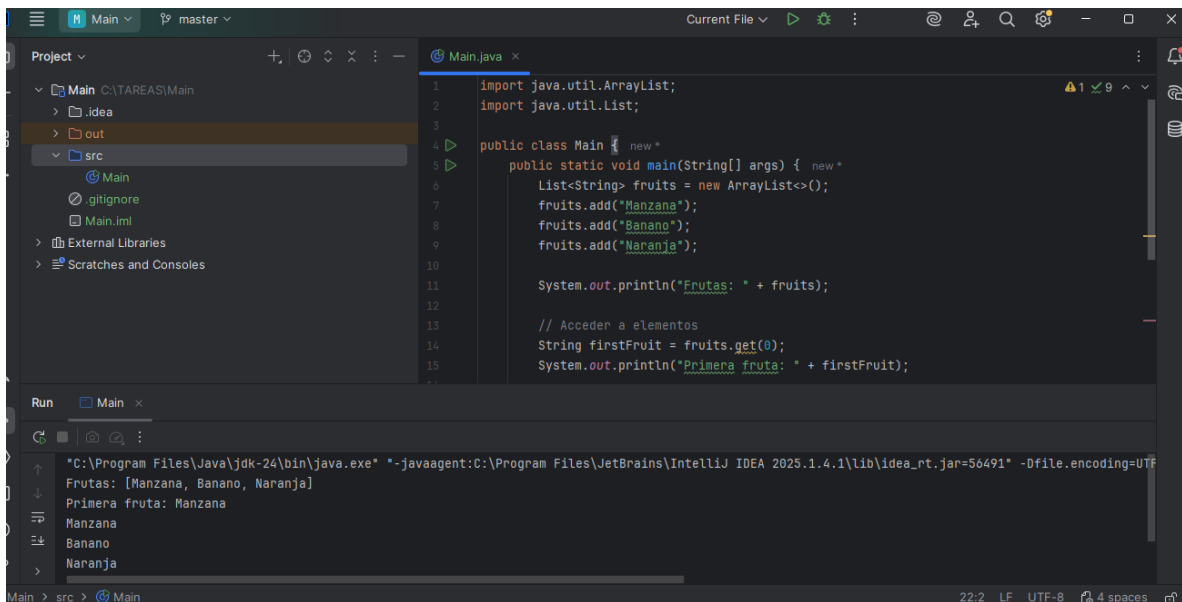


## ARRAYLIST

El objetivo principal del código es demostrar el uso de una lista dinámica con un ArrayList para almacenar y modificar una colección de frutas.

El programa:

- 🚩 Crea una lista de tipo String.
- 🚩 Agrega frutas a la lista.
- 🚩 Imprime toda la lista.
- 🚩 Accede e imprime el primer elemento.
- 🚩 Itera e imprime cada fruta por separado.



The screenshot shows the IntelliJ IDEA IDE with a project named 'Main'. The 'src' directory contains a file named 'Main.java'. The code in 'Main.java' is as follows:

```
1 import java.util.ArrayList;
2 import java.util.List;
3
4 public class Main {
5     public static void main(String[] args) {
6         List<String> fruits = new ArrayList<>();
7         fruits.add("Manzana");
8         fruits.add("Banano");
9         fruits.add("Naranja");
10
11         System.out.println("Frutas: " + fruits);
12
13         // Acceder a elementos
14         String firstFruit = fruits.get(0);
15         System.out.println("Primera fruta: " + firstFruit);
16     }
17 }
```

The 'Run' output window at the bottom shows the execution of the program. The command used is: `"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.1.4.1\lib\idea_rt.jar=56491" -Dfile.encoding=UTF-8`. The output is:

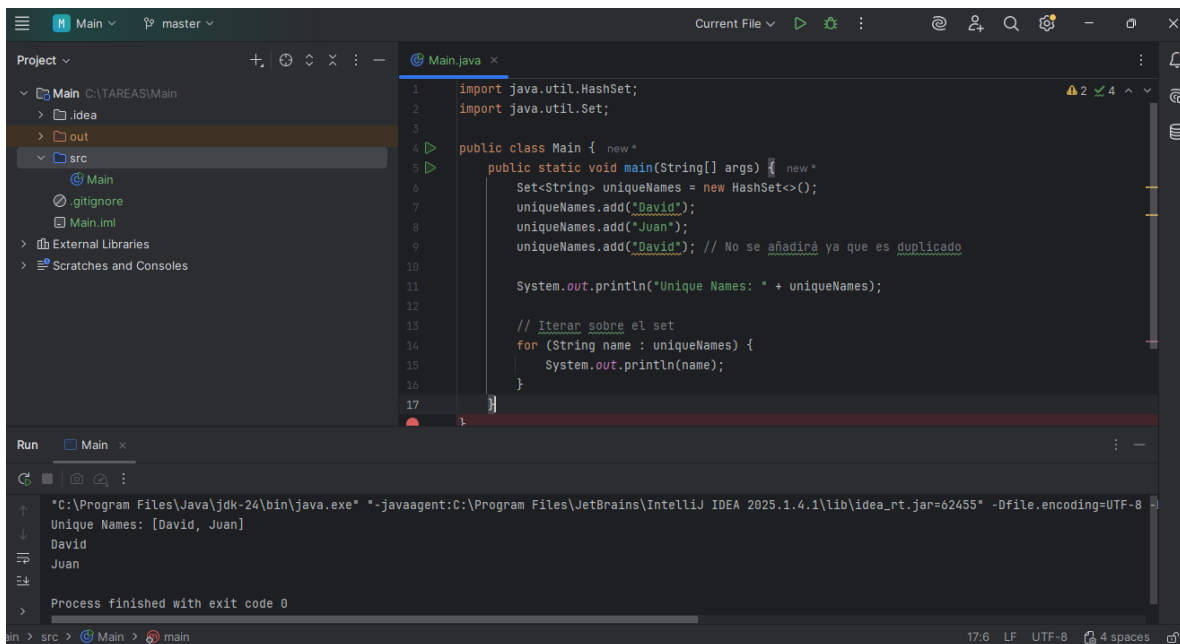
```
Frutas: [Manzana, Banano, Naranja]
Primera fruta: Manzana
Manzana
Banano
Naranja
```

## HASHSET

Este programa en Java demuestra cómo usar un Set (conjunto) para almacenar elementos únicos, evitar elementos duplicados automáticamente y almacenar elementos sin preocuparse del orden.

El programa:

- Crea un HashSet de tipo String llamado uniqueNames.
- Agrega tres nombres, incluyendo un duplicado ("David" dos veces).
- Imprime todos los nombres únicos.
- Recorre el conjunto con un for-each para imprimir cada nombre individualmente.



The screenshot shows an IDE window with a project named 'Main'. The 'src' folder contains a file named 'Main.java'. The code in 'Main.java' is as follows:

```
1 import java.util.HashSet;
2 import java.util.Set;
3
4 public class Main {
5     public static void main(String[] args) {
6         Set<String> uniqueNames = new HashSet<>();
7         uniqueNames.add("David");
8         uniqueNames.add("Juan");
9         uniqueNames.add("David"); // No se añadirá ya que es duplicado
10
11         System.out.println("Unique Names: " + uniqueNames);
12
13         // Iterar sobre el set
14         for (String name : uniqueNames) {
15             System.out.println(name);
16         }
17     }
18 }
```

The 'Run' output window shows the following output:

```
"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.1.4.1\lib\idea_rt.jar=62455" -Dfile.encoding=UTF-8
Unique Names: [David, Juan]
David
Juan
Process finished with exit code 0
```

## HASHMAP

Este programa en Java muestra cómo usar un mapa (Map), específicamente un HashMap, para almacenar pares clave-valor.

El programa:

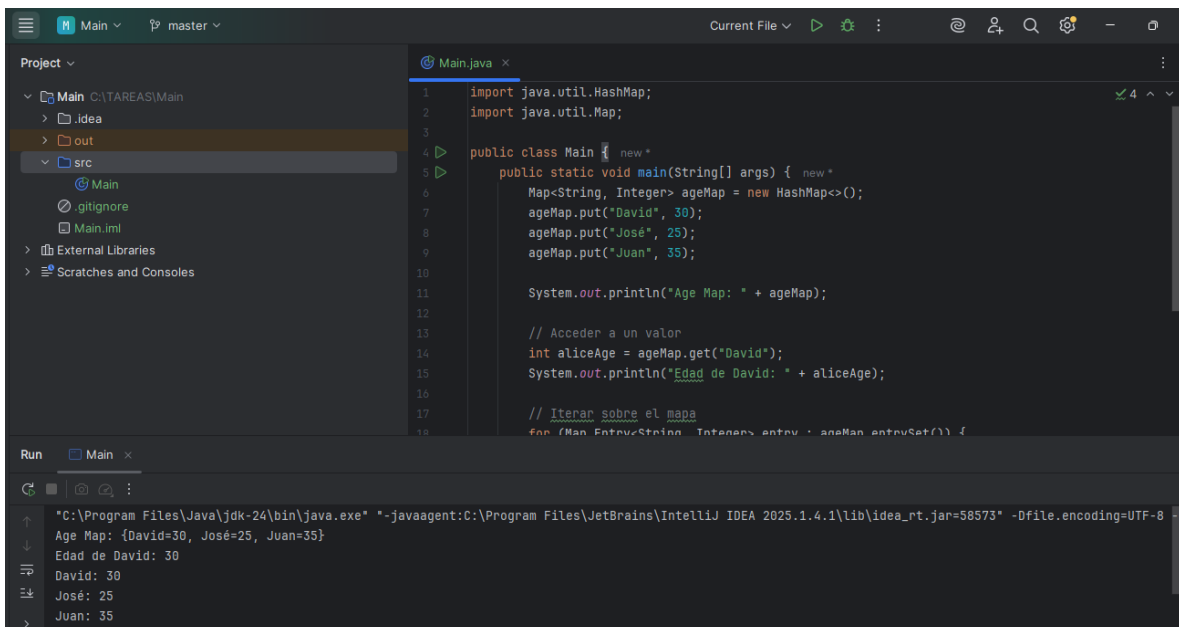
- Crea un HashMap llamado ageMap, que asocia nombres (String) con edades (Integer).
- Agrega tres entradas al mapa:
  - "David" → 30
  - "José" → 25
  - "Juan" → 35
- Imprime el mapa completo con todos los pares clave-valor.
- Accede a la edad de "David" y la imprime.
- Recorre el mapa con un bucle for-each, imprimiendo cada clave y su valor.

### String, Integer:

Indica que las claves serán de tipo String (texto), y los valores de tipo Integer (número entero).

### ¿Por qué Integer y no int?

- int es un tipo primitivo (no se puede usar directamente en estructuras genéricas como Map<K, V>).
- Integer es una clase envolvente (wrapper class) que permite usar números como objetos. **(esto no lo sabía por lo estoy agregando)**



The screenshot shows an IDE with a project named 'Main'. The 'src' directory contains a file named 'Main.java'. The code in 'Main.java' is as follows:

```
1 import java.util.HashMap;
2 import java.util.Map;
3
4 public class Main {
5     public static void main(String[] args) {
6         Map<String, Integer> ageMap = new HashMap<>();
7         ageMap.put("David", 30);
8         ageMap.put("José", 25);
9         ageMap.put("Juan", 35);
10
11         System.out.println("Age Map: " + ageMap);
12
13         // Acceder a un valor
14         int aliceAge = ageMap.get("David");
15         System.out.println("Edad de David: " + aliceAge);
16
17         // Iterar sobre el mapa
18         for (Map.Entry<String, Integer> entry : ageMap.entrySet()) {
```

The Run tab at the bottom shows the output of the program:

```
"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.1.4.1\lib\idea_rt.jar=58573" -Dfile.encoding=UTF-8
Age Map: {David=30, José=25, Juan=35}
Edad de David: 30
David: 30
José: 25
Juan: 35
```

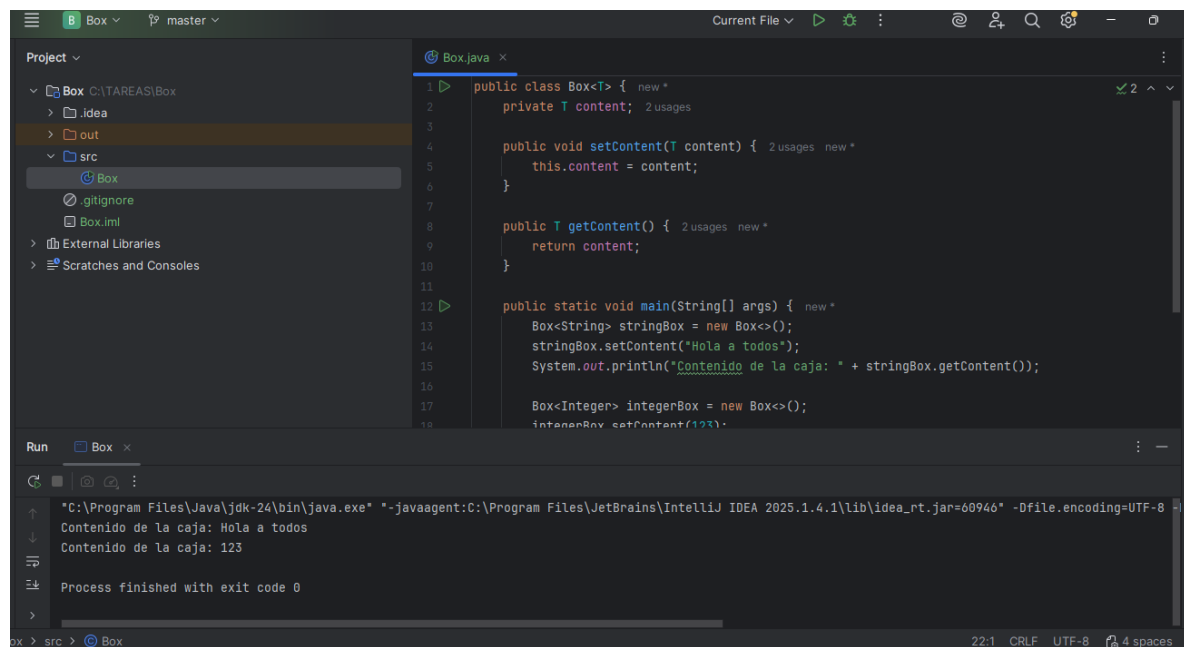
## GENERIC CLASS

El programa define una clase genérica `Box<T>`, que representa una "caja" capaz de almacenar un único objeto de cualquier tipo (especificado al momento de crear la instancia).

Este diseño utiliza Generics en Java, permitiendo reutilizar el mismo código para diferentes tipos de datos, garantizando seguridad de tipo en tiempo de compilación.

El programa:

- Crea una clase genérica llamada `Box<T>`, que permite almacenar un objeto de cualquier tipo (T).
- Define un atributo `content` para guardar ese objeto.
- Incluye dos métodos:
  - `setContent(T content)`: asigna un valor al contenido de la caja.
  - `getContent()`: devuelve el contenido almacenado.
- En el método `main`:
  - Crea una caja de tipo `String` y guarda el texto "Hola a todos".
  - Crea una caja de tipo `Integer` y guarda el número 123.
  - Imprime el contenido de ambas cajas en consola.



The screenshot shows an IDE with a project named 'Box'. The 'src' directory contains a file named 'Box.java'. The code in 'Box.java' defines a generic class `Box<T>` with a private attribute `content`, a `setContent` method, a `getContent` method, and a `main` method. The `main` method creates a `Box<String>` object, sets its content to 'Hola a todos', prints it, then creates a `Box<Integer>` object, sets its content to 123, and prints it. The output window shows the execution of the program, displaying the text 'Contenido de la caja: Hola a todos' and 'Contenido de la caja: 123' on separate lines, followed by 'Process finished with exit code 0'.

```
1 public class Box<T> {  
2     private T content;  
3  
4     public void setContent(T content) {  
5         this.content = content;  
6     }  
7  
8     public T getContent() {  
9         return content;  
10    }  
11  
12    public static void main(String[] args) {  
13        Box<String> stringBox = new Box<>();  
14        stringBox.setContent("Hola a todos");  
15        System.out.println("Contenido de la caja: " + stringBox.getContent());  
16  
17        Box<Integer> integerBox = new Box<>();  
18        integerBox.setContent(123);  
19    }  
20 }
```

Run: Box

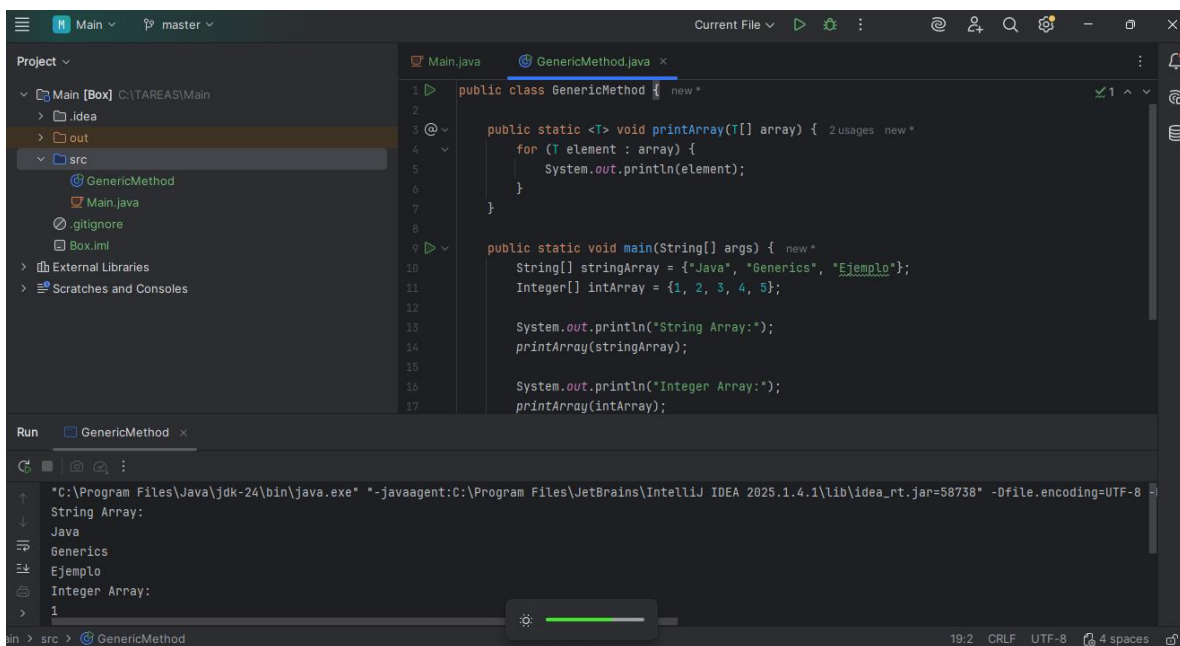
```
"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.1.4.1\lib\idea_rt.jar=60946" -Dfile.encoding=UTF-8  
Contenido de la caja: Hola a todos  
Contenido de la caja: 123  
Process finished with exit code 0
```

## GENERIC METHOD

La función principal del programa es demostrar cómo usar un método genérico en Java para imprimir los elementos de un arreglo, sin importar el tipo de datos que contenga (por ejemplo, String, Integer, etc.).

El programa:

- 🔗 Define un método genérico llamado `printArray(T[] array)`, donde `<T>` indica que puede trabajar con arreglos de cualquier tipo de dato (por ejemplo, String, Integer, etc.)
- 🔗 Este método recorre el arreglo con un bucle `for-each` y muestra cada elemento por consola
- 🔗 Al ser genérico, se puede reutilizar para distintos tipos de arreglos sin repetir código
- 🔗 En el método `main`:
  - Se crea un arreglo de Strings llamado `stringArray` con tres elementos: "Java", "Generics" y "Ejemplo".
  - Se crea un arreglo de enteros (`Integer[]`) llamado `intArray` con cinco números: 1, 2, 3, 4, 5.
  - Se llama al método `printArray` con `stringArray`, lo que imprime:  
Java  
Generics  
Ejemplos
  - 🔗 Luego, se llama a `printArray` con `intArray`, imprimiendo: 1 2 3 4 5



The screenshot shows an IDE with a project named 'Main'. The 'src' directory contains 'GenericMethod.java'. The code in this file is as follows:

```
1 public class GenericMethod {  
2  
3     public static <T> void printArray(T[] array) {  
4         for (T element : array) {  
5             System.out.println(element);  
6         }  
7     }  
8  
9     public static void main(String[] args) {  
10        String[] stringArray = {"Java", "Generics", "Ejemplo"};  
11        Integer[] intArray = {1, 2, 3, 4, 5};  
12  
13        System.out.println("String Array:");  
14        printArray(stringArray);  
15  
16        System.out.println("Integer Array:");  
17        printArray(intArray);  
18    }  
19 }
```

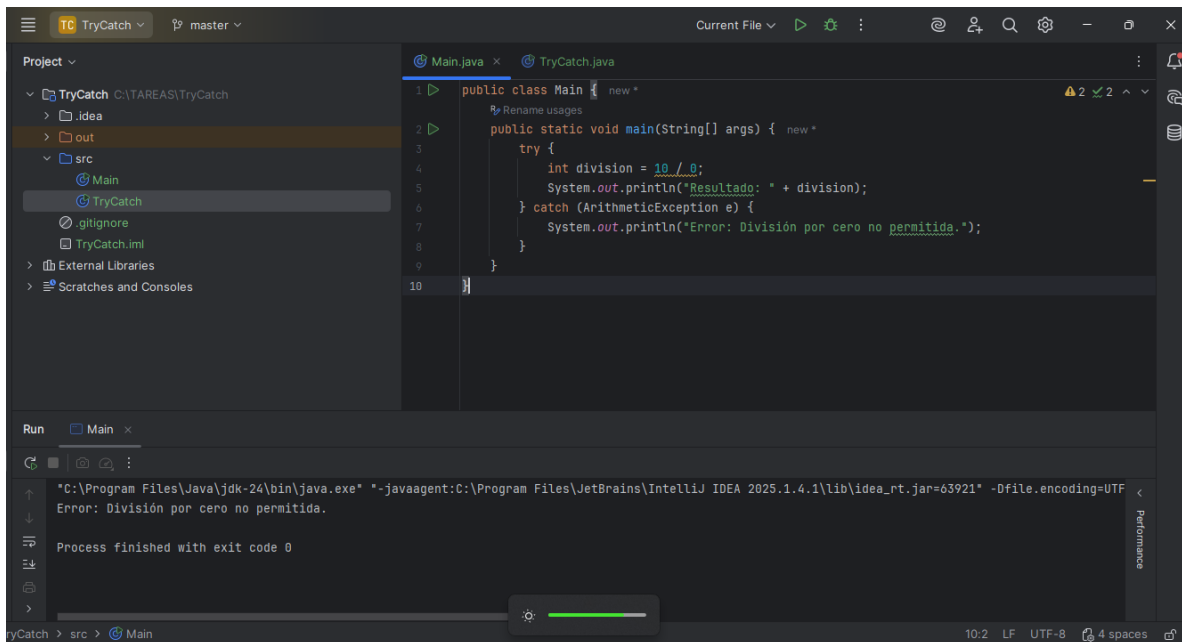
The 'Run' tab at the bottom shows the output of the program:

```
"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.1.4.1\lib\idea_rt.jar=58738" -Dfile.encoding=UTF-8  
String Array:  
Java  
Generics  
Ejemplo  
Integer Array:  
1  
2  
3  
4  
5
```

## TRY CATCH

El programa demuestra el uso del manejo de excepciones en Java, capturando una división por cero utilizando un bloque try-catch.

- El programa intenta dividir 10 entre 0, lo cual genera una excepción
- Usa un bloque try-catch para capturar el error de división por cero (ArithmeticException)
- En lugar de detenerse, el programa muestra un mensaje de error controlado: **Error: División por cero no permitida.**
- Objetivo:** Demostrar cómo manejar errores en tiempo de ejecución usando try-catch



The screenshot shows the IntelliJ IDEA IDE with a project named 'TryCatch'. The 'src' directory contains two files: 'Main.java' and 'TryCatch.java'. The 'Main.java' file is open, showing the following code:

```
1 public class Main {  
2     public static void main(String[] args) {  
3         try {  
4             int division = 10 / 0;  
5             System.out.println("Resultado: " + division);  
6         } catch (ArithmeticException e) {  
7             System.out.println("Error: División por cero no permitida.");  
8         }  
9     }  
10 }
```

The 'Run' tab at the bottom shows the execution output:

```
"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.1.4.1\lib\idea_rt.jar=63921" -Dfile.encoding=UTF-8  
Error: División por cero no permitida.  
Process finished with exit code 0
```

The status bar at the bottom indicates the file is 'Main.java' in the 'src' directory, with a line number of 10:2, using LF line endings, UTF-8 encoding, and 4 spaces for indentation.

## BLOQUE FINALLY

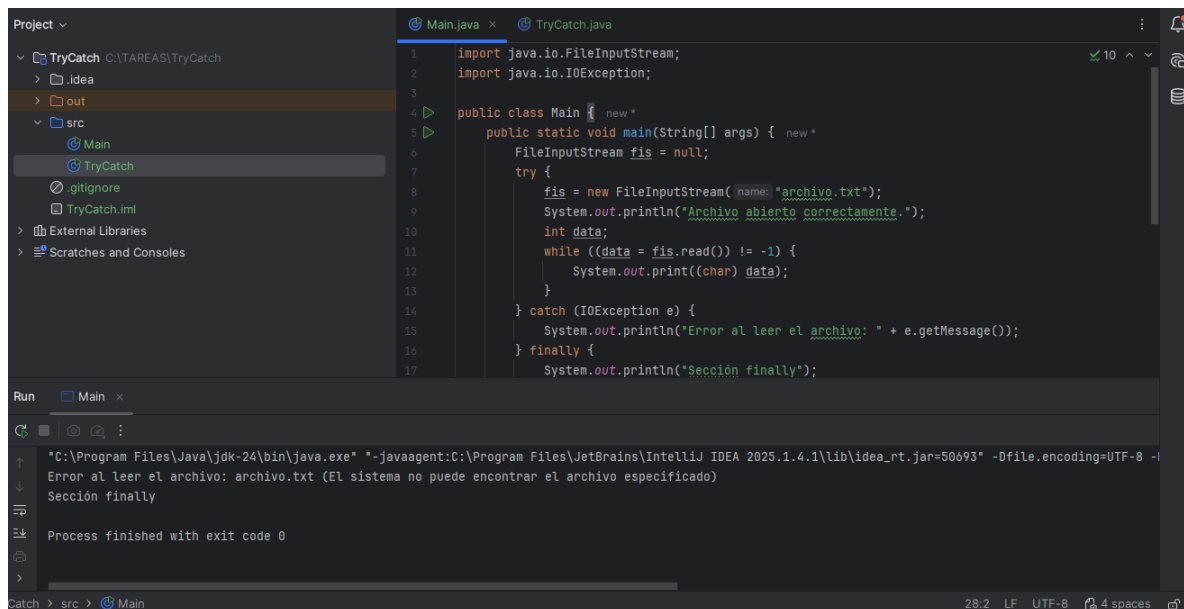
La función principal del programa es:

Leer el contenido de un archivo llamado archivo.txt y mostrarlo por pantalla, manejando correctamente posibles errores de entrada/salida y asegurando el cierre del archivo.

El programa intenta abrir y leer un archivo llamado archivo.txt usando FileInputStream. Utiliza bloques try-catch-finally para manejar errores de lectura y asegurar el cierre del archivo correctamente.

El programa:

- 🚩 **Abre el archivo** usando FileInputStream.
- 🚩 **Lee su contenido** carácter por carácter.
- 🚩 **Maneja errores** si el archivo no existe o ocurre un problema durante la lectura.
- 🚩 **Cierra el archivo** correctamente en el bloque finally, incluso si hubo un error.



The screenshot shows an IDE with a project named 'TryCatch'. The 'src' folder contains 'Main.java' and 'TryCatch.java'. The 'Run' tab is active, showing the output of the program. The code in 'TryCatch.java' is as follows:

```
1 import java.io.FileInputStream;
2 import java.io.IOException;
3
4 public class Main {
5     public static void main(String[] args) {
6         FileInputStream fis = null;
7         try {
8             fis = new FileInputStream("archivo.txt");
9             System.out.println("Archivo abierto correctamente.");
10            int data;
11            while ((data = fis.read()) != -1) {
12                System.out.print((char) data);
13            }
14        } catch (IOException e) {
15            System.out.println("Error al leer el archivo: " + e.getMessage());
16        } finally {
17            System.out.println("Sección finally");
18        }
19    }
20 }
```

The output in the Run tab is:

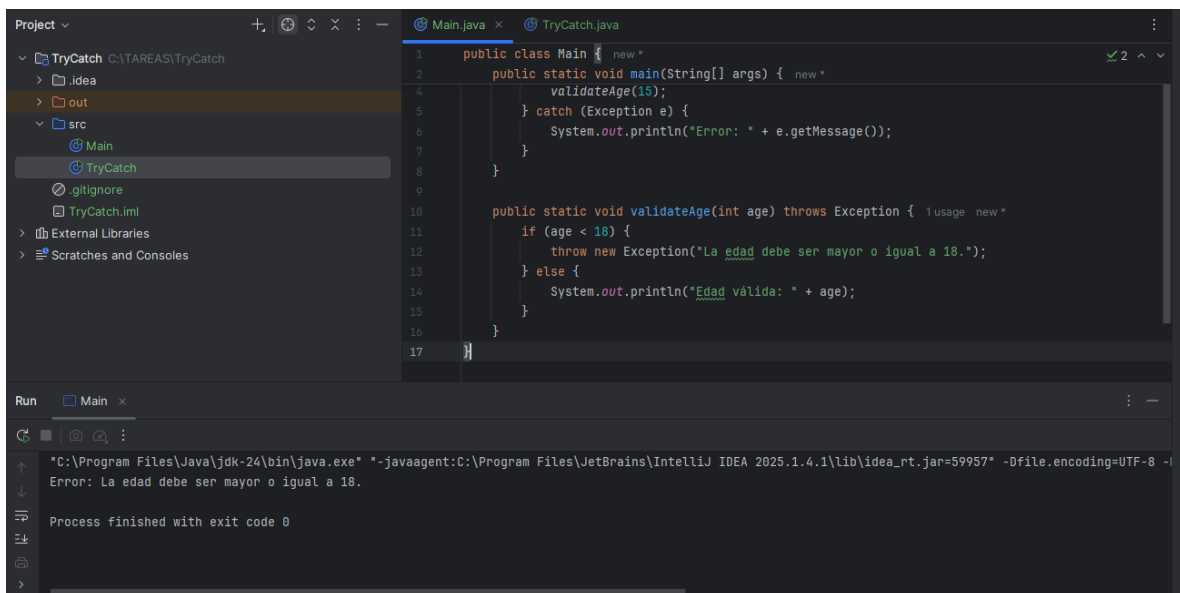
```
"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.1.4.1\lib\idea_rt.jar=50693" -Dfile.encoding=UTF-8 -I
Error al leer el archivo: archivo.txt (El sistema no puede encontrar el archivo especificado)
Sección finally
Process finished with exit code 0
```

## TROW TROWS

Validar si una persona tiene la edad suficiente (al menos 18 años) y, si no la tiene, lanzar y manejar una excepción con un mensaje explicativo.

El programa:

- 🚩 Llama al método `validateAge` con una edad menor a 18.
- 🚩 Lanza una excepción si la edad es menor a 18.
- 🚩 Captura la excepción con `try-catch`.
- 🚩 Imprime un mensaje de error si la edad no es válida.



The screenshot displays an IDE with two files: `Main.java` and `TryCatch.java`. The `TryCatch.java` file contains the following code:

```
1 public class Main {  
2     public static void main(String[] args) {  
3         validateAge(15);  
4     } catch (Exception e) {  
5         System.out.println("Error: " + e.getMessage());  
6     }  
7 }  
8  
9  
10  
11 public static void validateAge(int age) throws Exception {  
12     if (age < 18) {  
13         throw new Exception("La edad debe ser mayor o igual a 18.");  
14     } else {  
15         System.out.println("Edad válida: " + age);  
16     }  
17 }
```

The `Run` tab shows the execution output:

```
"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.1.4.1\lib\idea_rt.jar=59957" -Dfile.encoding=UTF-8 -l  
Error: La edad debe ser mayor o igual a 18.  
  
Process finished with exit code 0
```

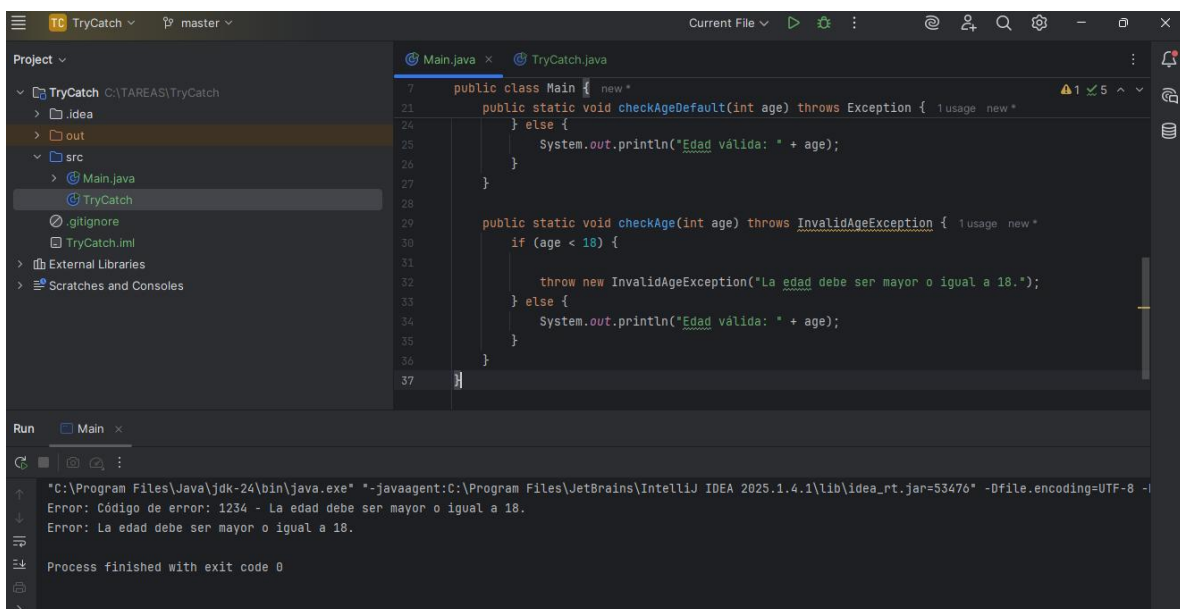


## EXCEPCIONES PERSONALIZADAS

La función principal del programa es validar si una persona tiene al menos 18 años y demostrar cómo manejar errores utilizando tanto excepciones personalizadas como excepciones estándar.

El programa:

- Define una excepción personalizada llamada `InvalidAgeException` con un mensaje de error específico.
- Llama al método `checkAge(15)` que:
- Lanza `InvalidAgeException` si la edad es menor a 18.
- Captura e imprime el mensaje de error.
- Llama al método `checkAgeDefault(15)` que:
- Lanza una excepción estándar (`Exception`) si la edad es menor a 18.
- Captura e imprime el mensaje de error.
- Imprime "Edad válida: [edad]" si la edad es 18 o más.



The screenshot shows an IDE window with a project named 'TryCatch'. The left sidebar displays the project structure with folders 'idea', 'out', and 'src', and files 'Main.java', 'TryCatch', '.gitignore', and 'TryCatch.iml'. The main editor shows the code for 'Main.java' and 'TryCatch.java'. The 'Run' tab at the bottom shows the execution output.

```
7 public class Main {  
21     public static void checkAgeDefault(int age) throws Exception {  
24         } else {  
25             System.out.println("Edad válida: " + age);  
26         }  
27     }  
28  
29     public static void checkAge(int age) throws InvalidAgeException {  
30         if (age < 18) {  
31             throw new InvalidAgeException("La edad debe ser mayor o igual a 18.");  
32         } else {  
33             System.out.println("Edad válida: " + age);  
34         }  
35     }  
36 }  
37 }
```

Run Main

```
"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.1.4.1\lib\idea_rt.jar=53476" -Dfile.encoding=UTF-8 -l  
Error: Código de error: 1234 - La edad debe ser mayor o igual a 18.  
Error: La edad debe ser mayor o igual a 18.  
Process finished with exit code 0
```