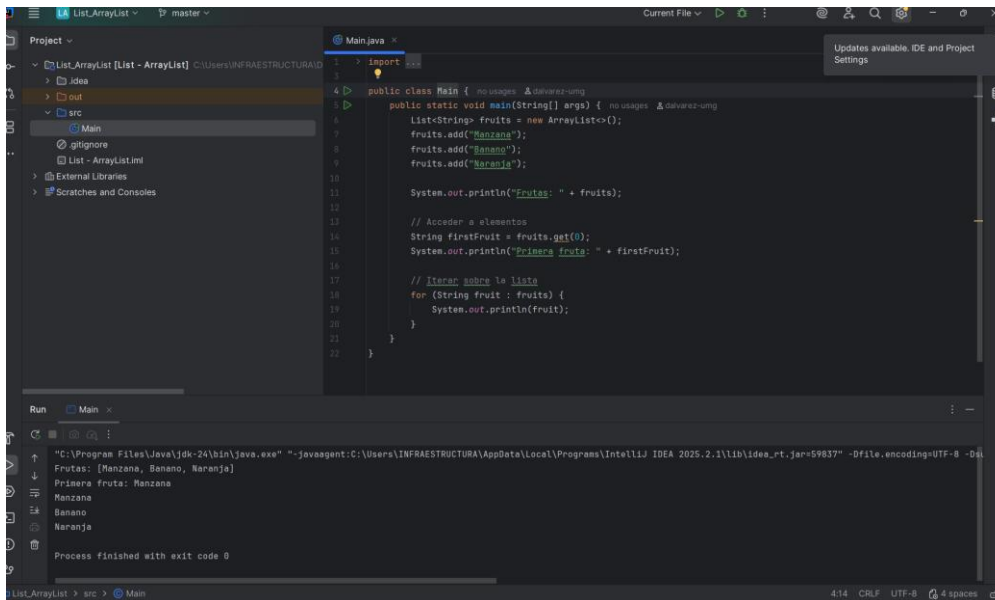


LIST ARRAYLIST



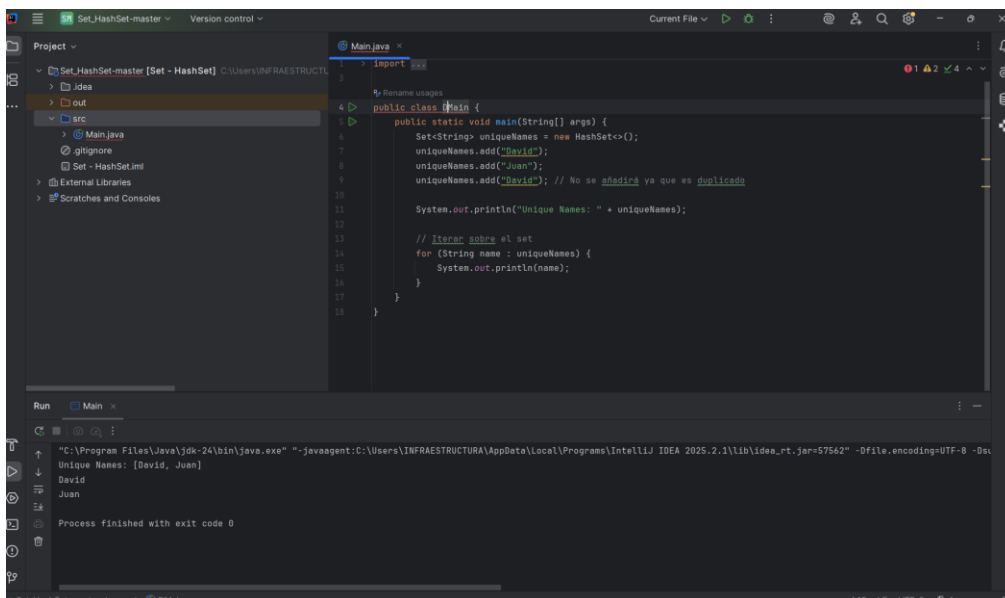
```
1  import java.util.*;
2
3  public class Main {
4      public static void main(String[] args) {
5          List<String> fruits = new ArrayList<>();
6          fruits.add("Manzana");
7          fruits.add("Banana");
8          fruits.add("Naranja");
9
10         System.out.println("Frutas: " + fruits);
11
12         // Acceder a elementos
13         String firstFruit = fruits.get(0);
14         System.out.println("Primera fruta: " + firstFruit);
15
16         // Iterar sobre la lista
17         for (String fruit : fruits) {
18             System.out.println(fruit);
19         }
20     }
21 }
22
```

Run Main

Process finished with exit code 0

- Permite elementos duplicados (hay dos "Hansana")
- Mantiene el orden de inserción
- Proporciona acceso por índice
- Fácil de itera

HASHSET



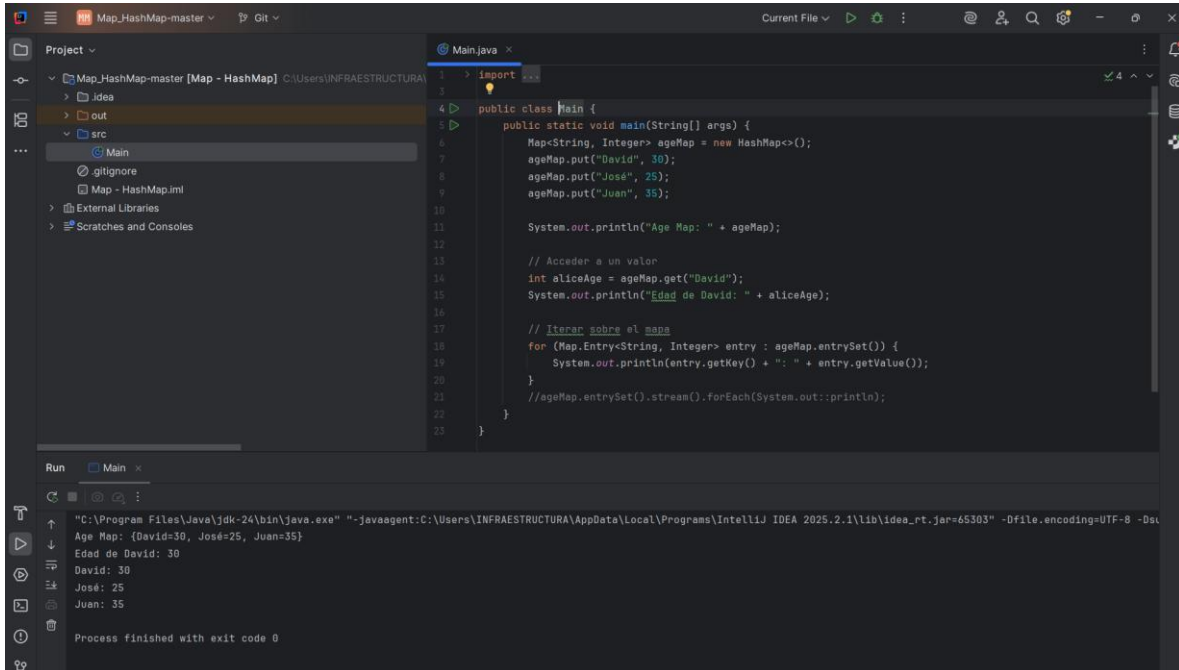
```
1  import java.util.*;
2
3  public class Main {
4      public static void main(String[] args) {
5          Set<String> uniqueNames = new HashSet<>();
6          uniqueNames.add("David");
7          uniqueNames.add("Juan");
8          uniqueNames.add("David"); // No se añade ya que es duplicado
9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

Run Main

Process finished with exit code 0

Almacenar nombres únicos en una colección que automáticamente elimine duplicados.

HASHMAP



```
1 import java.util.*;
2
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()) {
19             System.out.println(entry.getKey() + ": " + entry.getValue());
20         }
21         //ageMap.entrySet().stream().forEach(System.out::println);
22     }
23 }
```

Run Main x

"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Users\INFRAESTRUCTURA\AppData\Local\Programs\IntelliJ IDEA 2025.2.1\lib\idea_rt.jar=65303" -Dfile.encoding=UTF-8 -D...

Age Map: {David=30, José=25, Juan=35}

Edad de David: 30

David: 30

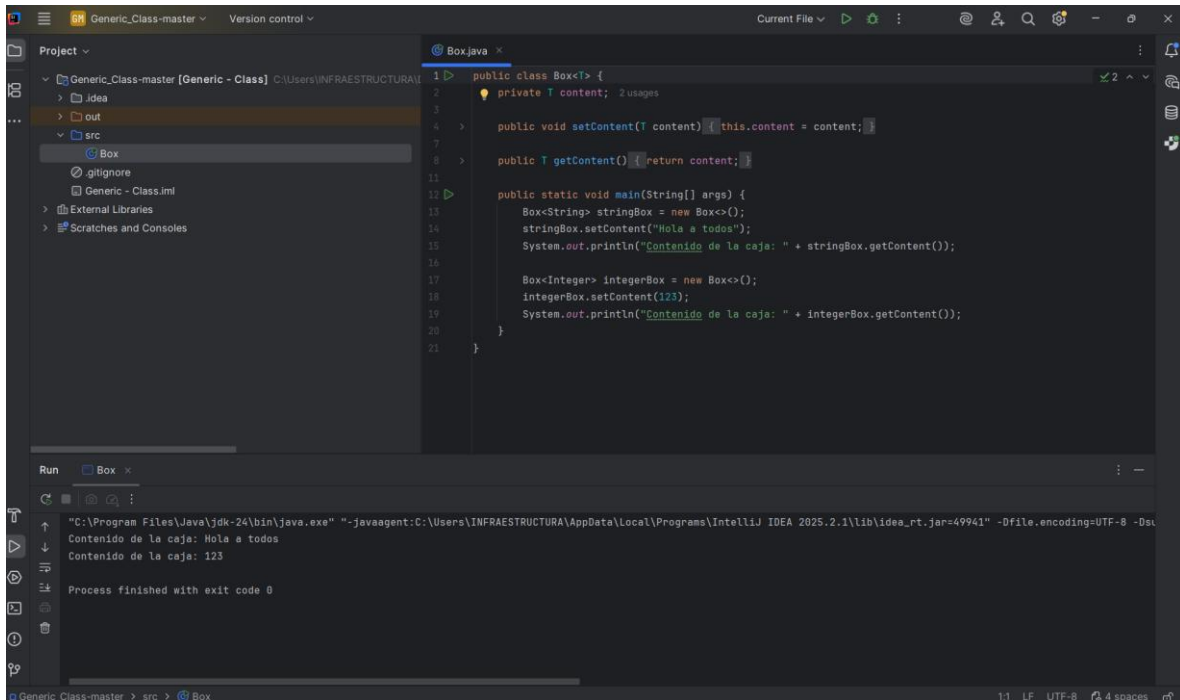
José: 25

Juan: 35

Process finished with exit code 0

- Almacenamiento clave-valor: Cada elemento tiene una clave única y un valor asociado
- Acceso rápido: Búsqueda eficiente por clave
- No garantiza orden: Los elementos pueden no mostrarse en el orden de inserción
- Claves únicas: No permite claves duplicadas

GENERIC CLASS



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

Run Box x

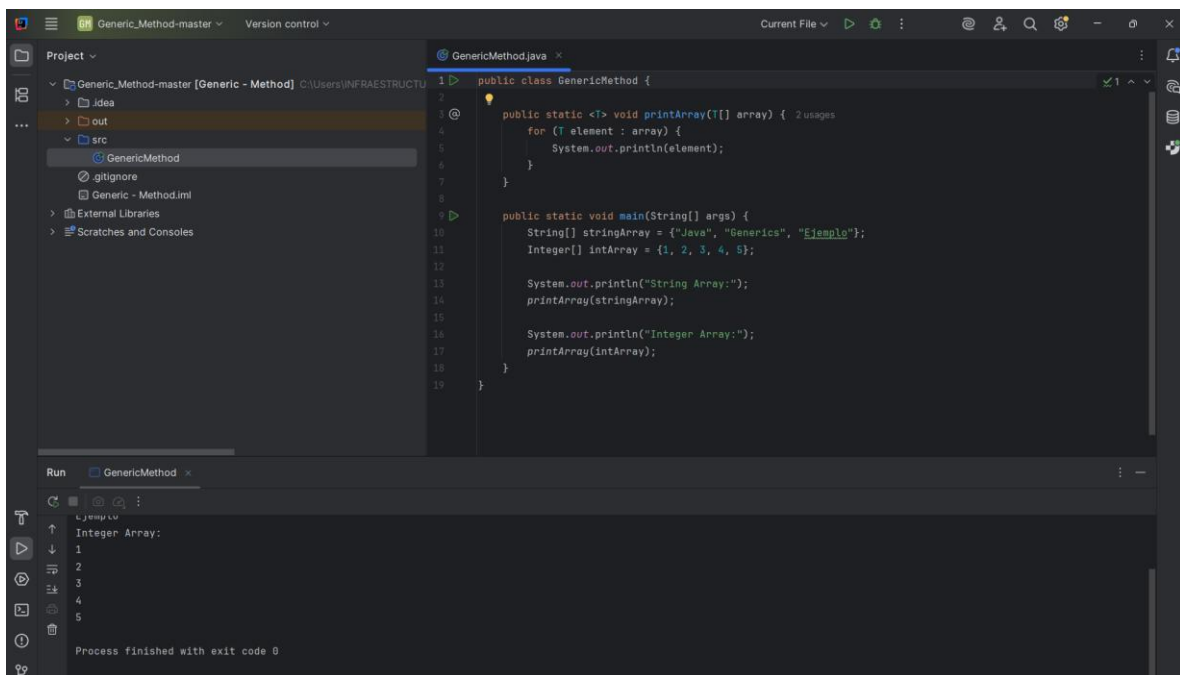
"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Users\INFRAESTRUCTURA\AppData\Local\Programs\IntelliJ IDEA 2025.2.1\lib\idea_rt.jar=49941" -Dfile.encoding=UTF-8 -D...

Contenido de la caja: Hola a todos
Contenido de la caja: 123

Process finished with exit code 0

Este programa demuestra una Clase Genérica en Java llamada `Box<T>` que funciona como un contenedor reusable para cualquier tipo de dato.

GENERIC METHOD



```
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 }
```

Run GenericMethod x

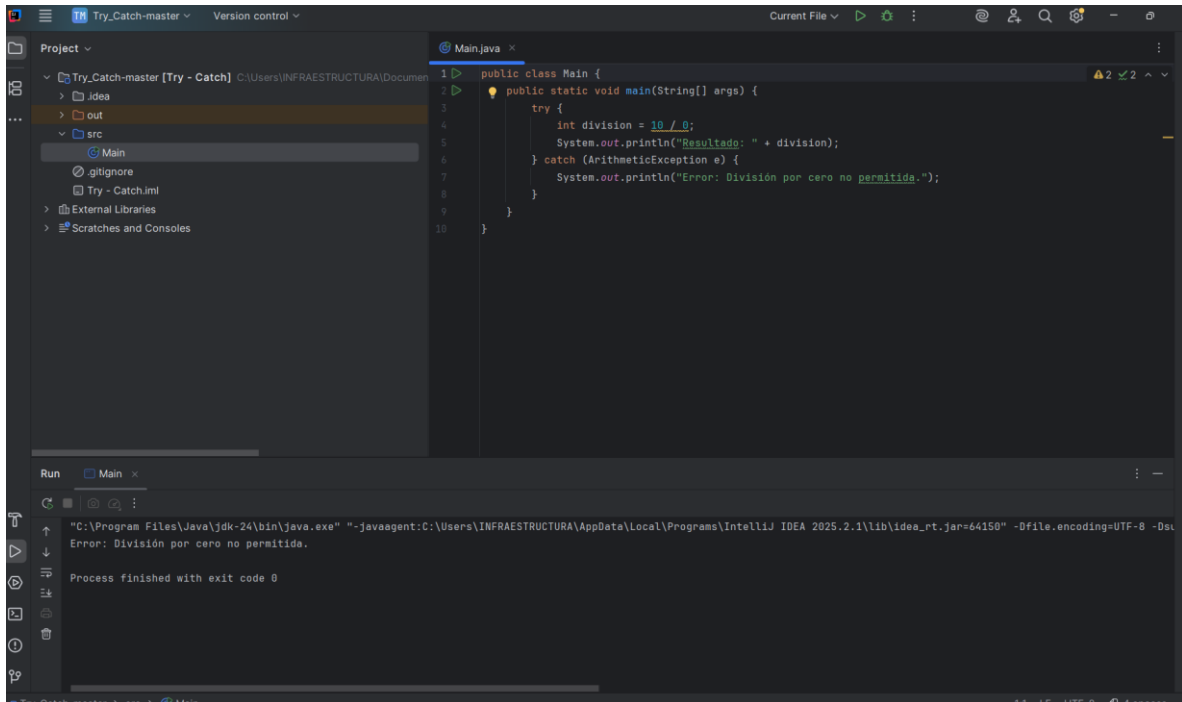
Ejemplo

Integer Array:
1
2
3
4
5

Process finished with exit code 0

demuestra un Método Genérico en Java que puede imprimir arrays de cualquier tipo de dato.

TRY CATCH



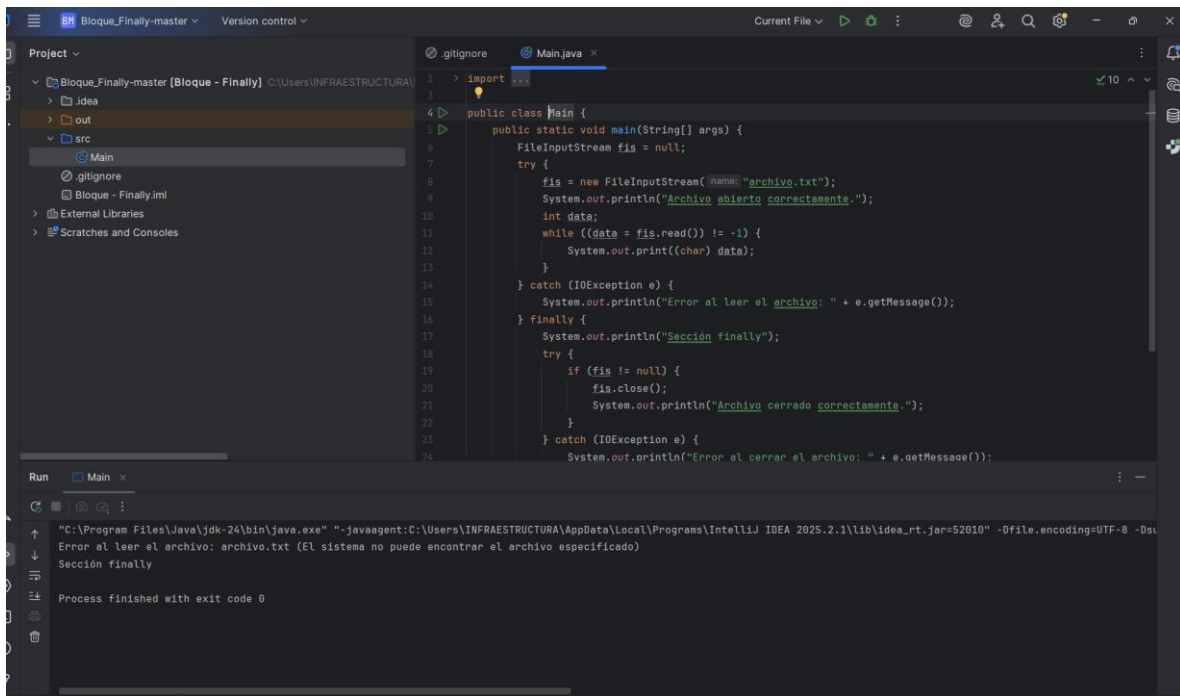
The screenshot shows an IDE window titled 'Try_Catch-master'. The 'Project' view on the left shows a project structure with 'src' containing 'Main.java'. The 'Main.java' file is open in the editor, showing the following code:

```
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' view at the bottom shows the command used to execute the program: `"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Users\INFRAESTRUCTURA\AppData\Local\Programs\IntelliJ IDEA 2025.2.1\lib\idea_rt.jar=64150" -Dfile.encoding=UTF-8 -Dsrc`. The output of the program is: `Error: División por cero no permitida.` and `Process finished with exit code 0`.

demuestra el uso de bloques try-catch para manejar excepciones y prevenir que el programa se detenga abruptamente.

BLOQUE FINALLY



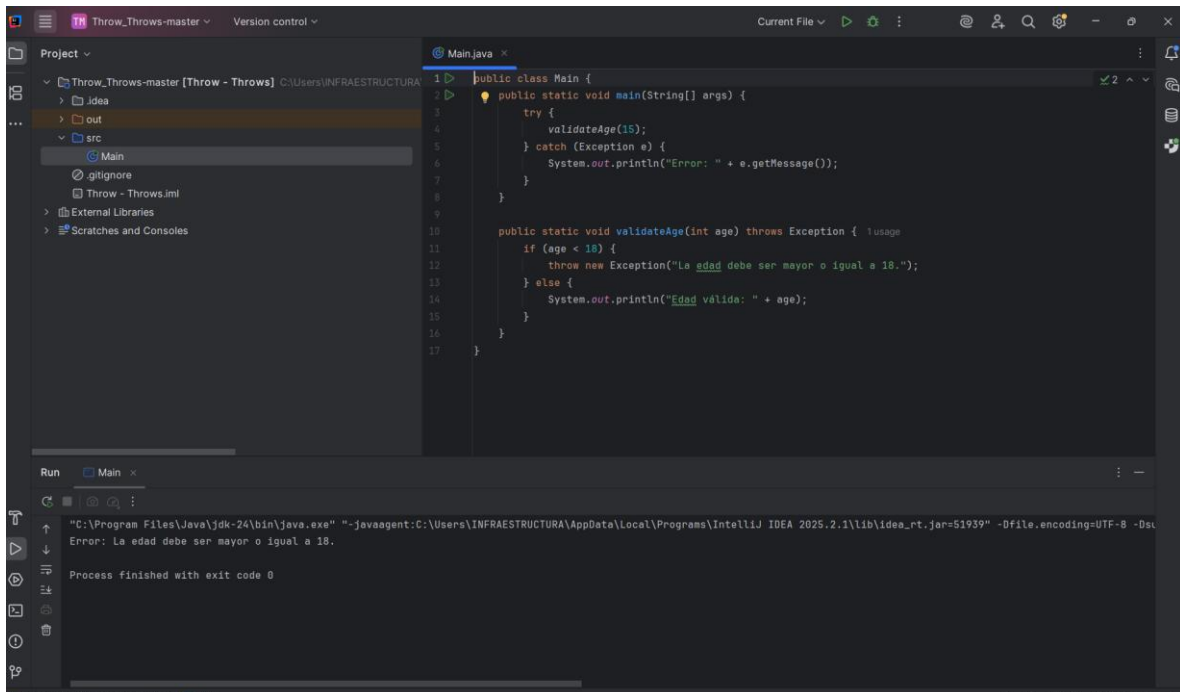
The screenshot shows the IntelliJ IDEA interface with a project named 'Bloque_Finally-master'. The 'Main.java' file is open, displaying a Java class with a `main` method. The code uses a `try-finally` block to read from a file named 'archivo.txt'. The `try` block contains the file opening and reading logic, while the `finally` block ensures the file is closed. The console output shows an error message: 'Error al leer el archivo: archivo.txt (El sistema no puede encontrar el archivo especificado)' followed by 'Sección finally' and 'Process finished with exit code 0'.

```
1 import java.io.*;
2
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            try {
19                if (fis != null) {
20                    fis.close();
21                    System.out.println("Archivo cerrado correctamente.");
22                }
23            } catch (IOException e) {
24                System.out.println("Error al cerrar el archivo: " + e.getMessage());
25            }
26        }
27    }
28 }
```

Run: "C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Users\INFRAESTRUCTURA\AppData\Local\Programs\IntelliJ IDEA 2025.2.1\lib\idea_rt.jar=52010 -Dfile.encoding=UTF-8 -Dsu
Error al leer el archivo: archivo.txt (El sistema no puede encontrar el archivo especificado)
Sección finally
Process finished with exit code 0

el uso del bloque finally para garantizar que los recursos se liberen correctamente, incluso si ocurren errores.

THROW THROWS



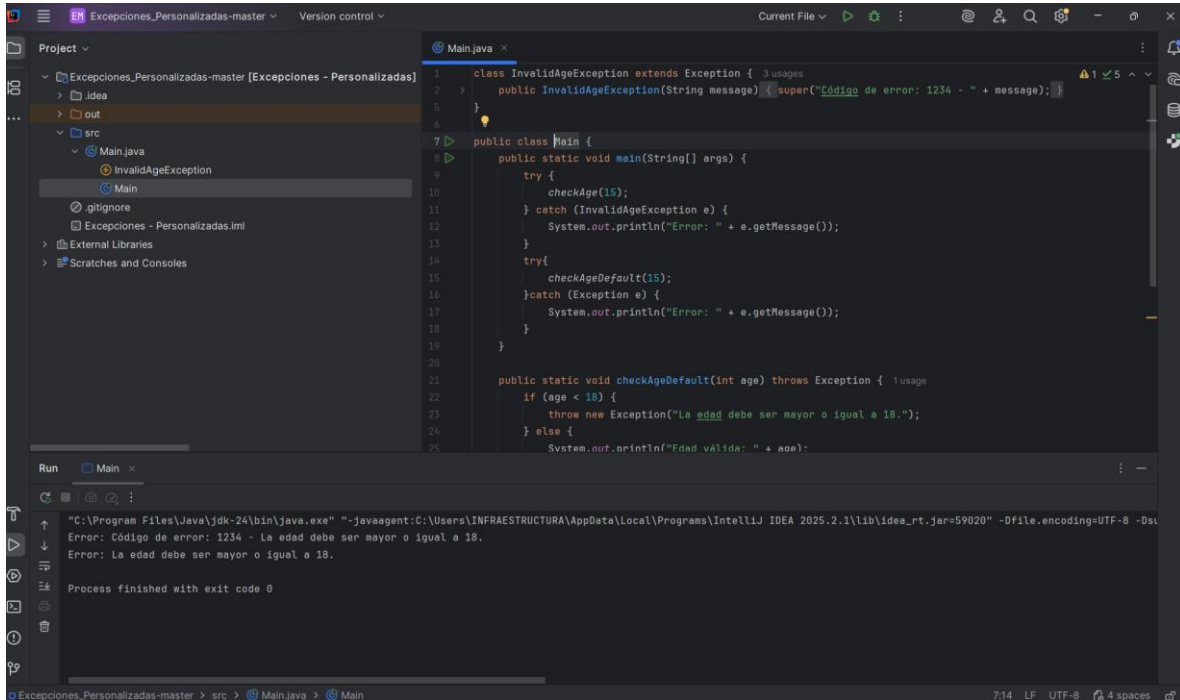
The screenshot shows the IntelliJ IDEA interface with a project named 'Throw_Throws-master'. The 'Main.java' file is open, displaying a Java class with a `main` method and a `validateAge` method. The `main` method calls `validateAge(15)` and catches any `Exception`. The `validateAge` method is declared with `throws Exception` and contains logic to check if the age is less than 18, throwing an `Exception` if so. The console output shows an error message: 'Error: La edad debe ser mayor o igual a 18.' followed by 'Process finished with exit code 0'.

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

Run: "C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Users\INFRAESTRUCTURA\AppData\Local\Programs\IntelliJ IDEA 2025.2.1\lib\idea_rt.jar=51939 -Dfile.encoding=UTF-8 -Dsu
Error: La edad debe ser mayor o igual a 18.
Process finished with exit code 0

El programa demuestra cómo lanzar y propagar excepciones personalizadas en Java

EXCEPCIONES PERSONALIZADAS



The screenshot shows an IDE window titled "Excepciones_Personalizadas-master". The left sidebar displays the project structure with files like "Main.java", "InvalidAgeException", and "Main". The main editor shows the code for "Main.java".

```
1 class InvalidAgeException extends Exception { 3 usages
2     public InvalidAgeException(String message) { super("Código de error: 1234 - " + message); }
3 }
4
5
6
7 public class Main {
8     public static void main(String[] args) {
9         try {
10             checkAge(15);
11         } catch (InvalidAgeException e) {
12             System.out.println("Error: " + e.getMessage());
13         }
14         try {
15             checkAgeDefault(15);
16         } catch (Exception e) {
17             System.out.println("Error: " + e.getMessage());
18         }
19     }
20
21     public static void checkAgeDefault(int age) throws Exception { 1 usage
22         if (age < 18) {
23             throw new Exception("La edad debe ser mayor o igual a 18.");
24         } else {
25             System.out.println("Edad valida: " + age);
26         }
27     }
28 }
```

The bottom panel shows the output of the program:

```
"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Users\INFRAESTRUCTURA\AppData\Local\Programs\IntelliJ IDEA 2025.2.1\lib\idea_rt.jar=59020" -Dfile.encoding=UTF-8 -Dsrc
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
```

El programa demuestra cómo crear y utilizar excepciones personalizadas en Java para manejar errores específicos del dominio.