

Tarea VI

1966

Arleth Susana Orozco Paredes

9941-24-18927

Universidad Mariano Gálvez de Guatemala

Facultad de Ingeniería en Sistemas de la Información y Ciencias de la Computación

Programación II

Ingeniero David Álvarez

Guatemala, 27 de septiembre de 2025

1. List_ArrayList

Código proporcionado:

```
import java.util.ArrayList;
import java.util.List;

public class Main {
    public static void main(String[] args) {
        List<String> fruits = new ArrayList<>();
        fruits.add("Manzana");
        fruits.add("Banano");
        fruits.add("Naranja");

        System.out.println("Frutas: " + fruits);

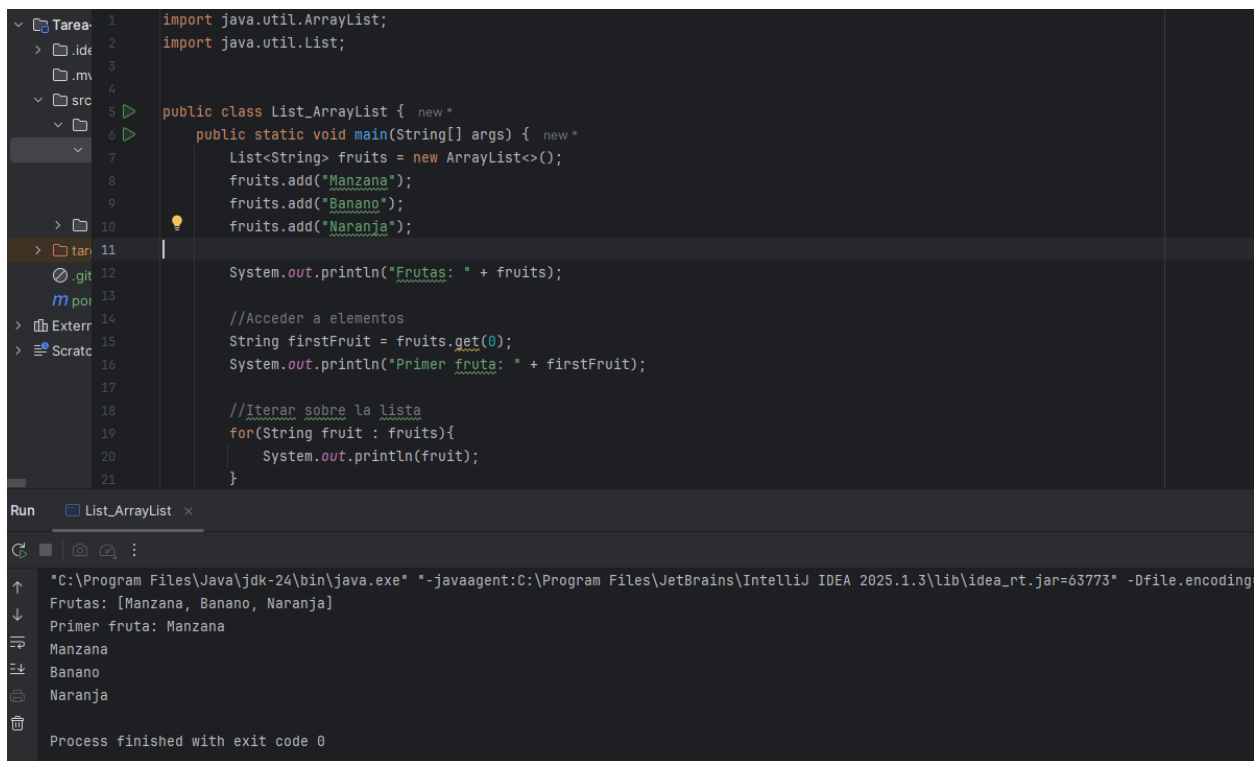
        // Acceder a elementos
        String firstFruit = fruits.get(0);
        System.out.println("Primera fruta: " + firstFruit);

        // Iterar sobre la lista
        for (String fruit : fruits) {
            System.out.println(fruit);
        }
    }
}
```

Breve descripción del código:

- ArrayList, es una colección ordenada, donde cada elemento tiene un índice y permite elementos duplicados.
- En el código primero se crea una lista dinámica que guarda cadenas (String)
- Con el add() se agregan las frutas (Manzana, Banano, Naranja)
- get(0) es para poder acceder al primer elemento de la cadena
- Luego se recorre la lista con un for-each, donde recorre cada elemento de la lista fruits, en cada vuelta la variable fruit toma el valor siguiente del elemento y es mas simple que un for tradicional ya que no se necesitan índices.
- Al ejecutarlo primero se imprimirá la lista completa, luego la primera fruta y después cada fruta en una línea.

Ejecución del Código:



The screenshot shows the IntelliJ IDEA IDE with a Java file named `List_ArrayList.java`. The code defines a `List_ArrayList` class with a `main` method. The `main` method creates an `ArrayList` of `String` objects, adds "Manzana", "Banano", and "Naranja", prints the entire list, prints the first element using `get(0)`, and then iterates over the list using a `for` loop to print each fruit on a new line.

```
1 import java.util.ArrayList;
2 import java.util.List;
3
4
5 public class List_ArrayList { new *
6     public static void main(String[] args) { new *
7         List<String> fruits = new ArrayList<>();
8         fruits.add("Manzana");
9         fruits.add("Banano");
10        fruits.add("Naranja");
11
12        System.out.println("Frutas: " + fruits);
13
14        //Acceder a elementos
15        String firstFruit = fruits.get(0);
16        System.out.println("Primer fruta: " + firstFruit);
17
18        //Iterar sobre la lista
19        for(String fruit : fruits){
20            System.out.println(fruit);
21        }
22    }
23 }
```

The Run window shows the execution output:

```
"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.1.3\lib\idea_rt.jar=63773" -Dfile.encoding=UTF-8
Frutas: [Manzana, Banano, Naranja]
Primer fruta: Manzana
Manzana
Banano
Naranja
Process finished with exit code 0
```

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

Process finished with exit code 0
```

2. HashSet

Código proporcionado:

```
import java.util.HashSet;
import java.util.Set;

public class Main {
    public static void main(String[] args) {
        Set<String> uniqueNames = new HashSet<>();
        uniqueNames.add("David");
        uniqueNames.add("Juan");
        uniqueNames.add("David"); // No se añadirá ya que es duplicado

        System.out.println("Unique Names: " + uniqueNames);

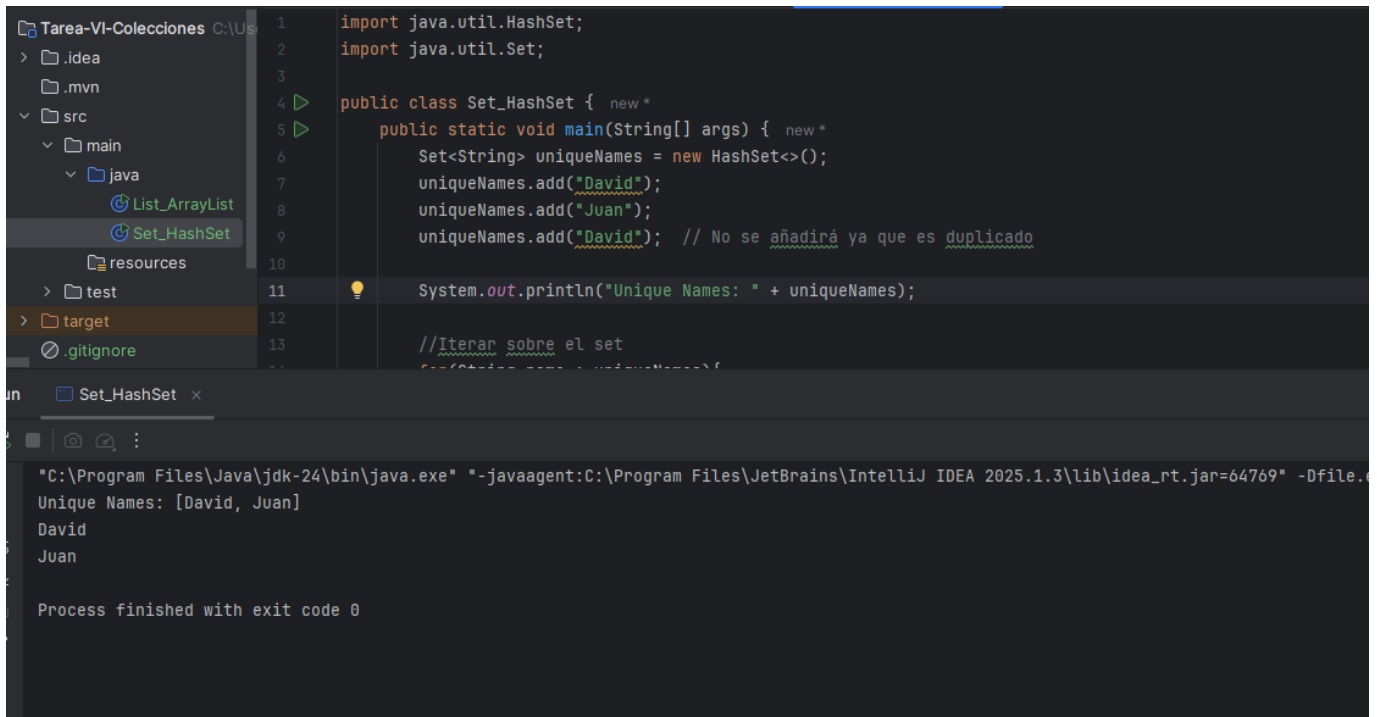
        // Iterar sobre el set
        for (String name : uniqueNames) {
            System.out.println(name);
        }
    }
}
```

Breve Descripción de Código:

- Se crea un HashSet de cadenas llamado uniqueNames.
- Se agregan los nombres “David”, “Juan” y nuevamente “David”.
- Como un Set no admite duplicados solo quedaran dos elementos: “David” y “Juan”.
- System.out.println("Unique Names: " + uniqueNames); muestra el conjunto entre corchetes, pero el orden puede variar.

- El for-each recorre cada elemento y lo imprime.
- Lo que mostrara al ejecutarlo serán los nombres únicos, sin repetidos, y primero mostrará en una sola línea: “David”, “Juan” y nuevamente “David”, “Juan” en cada línea.

Ejecución del Código:



The screenshot shows the IntelliJ IDEA IDE with a project named "Tarea-VI-Colecciones". The file "Set_HashSet.java" is open in the editor. The code defines a public class "Set_HashSet" with a static method "main". Inside "main", a "HashSet" named "uniqueNames" is created, and the strings "David" and "Juan" are added. A comment indicates that "David" is added again but is not duplicated. The output is printed using "System.out.println". Below the code, the "Run" tab shows the execution command and the output: "Unique Names: [David, Juan]", "David", "Juan", and "Process finished with exit code 0".

```
1 import java.util.HashSet;
2 import java.util.Set;
3
4 public class Set_HashSet {
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     }
15 }
```

Run Set_HashSet x

"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.1.3\lib\idea_rt.jar=64769" -Dfile.encoding=UTF-8

Unique Names: [David, Juan]

David

Juan

Process finished with exit code 0

```
"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.1.3\lib\idea_rt.jar=64769" -Dfile.encoding=UTF-8
Unique Names: [David, Juan]
David
Juan

Process finished with exit code 0
```

3. HashMap

Código proporcionado:

```
import java.util.HashMap;
import java.util.Map;

public class Main {
    public static void main(String[] args) {
        Map<String, Integer> ageMap = new HashMap<>();
        ageMap.put("David", 30);
        ageMap.put("José", 25);
        ageMap.put("Juan", 35);

        System.out.println("Age Map: " + ageMap);

        // Acceder a un valor
        int aliceAge = ageMap.get("David");
        System.out.println("Edad de David: " + aliceAge);

        // Iterar sobre el mapa
        for (Map.Entry<String, Integer> entry : ageMap.entrySet()) {
            System.out.println(entry.getKey() + ": " + entry.getValue());
        }
        //ageMap.entrySet().stream().forEach(System.out::println);
    }
}
```

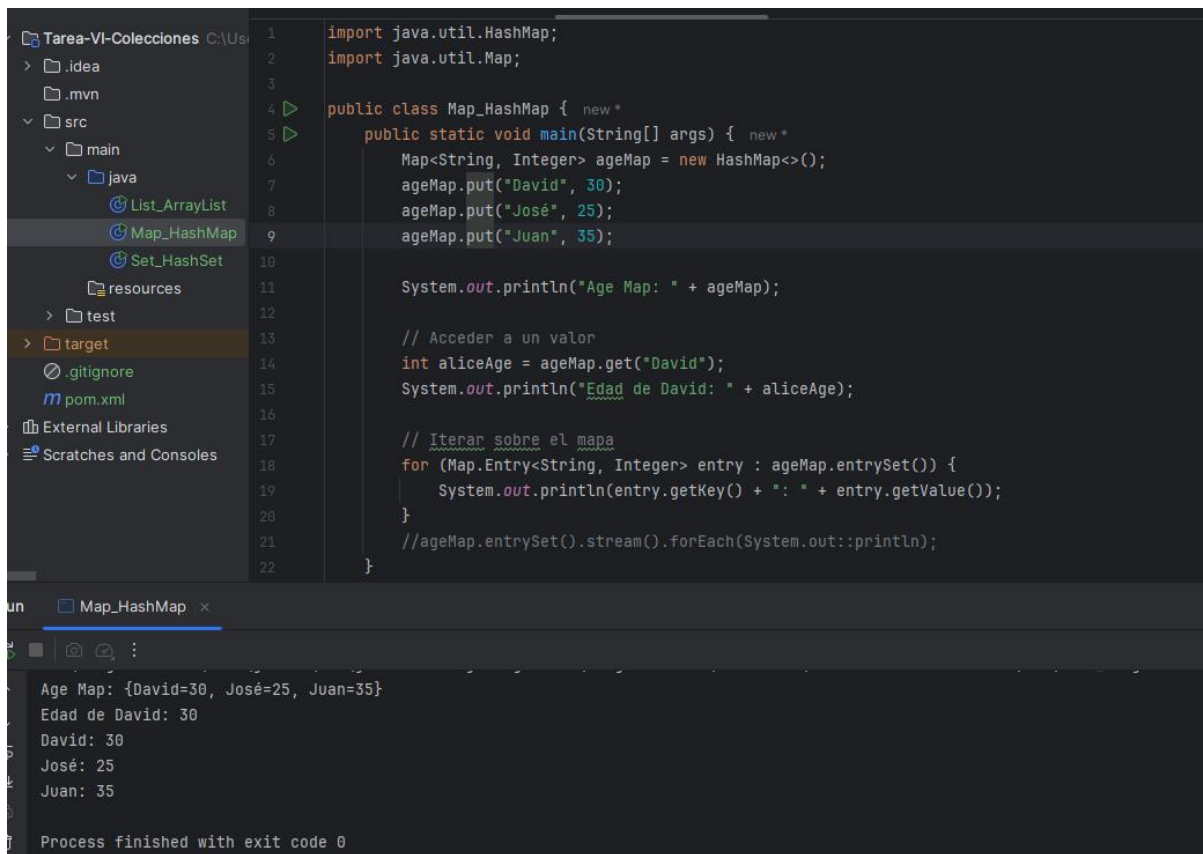
Breve descripción del código:

- Se crea un HashMap llamado ageMap que relaciona un nombre(clave) con una edad (valor).
- Se agregan tres entradas: “David”, 30 , “José”, 25, "Juan" , 35.
- Se imprime todo el mapa con System.out.println("Age Map: " + ageMap);.
- Luego se accede a un valor usando ageMap.get(“David”).
- El for-each recorre cada entrada y las imprime una por una.
- La salida del código, de primero se vera el mapa completo en formato {clave = valor} en una sola linea , luego la edad de “David: y por último cada persona con su edad en cada línea, por ejemplo:

David: 30

Jose: 25

Ejecución del Código:



The screenshot shows an IDE with a project named "Tarea-VI-Colecciones". The file explorer on the left shows the project structure, including a "src/main/java" directory containing "List_ArrayList", "Map_HashMap", and "Set_HashSet". The "Map_HashMap" file is selected, and its code is displayed in the editor. The code defines a class "Map_HashMap" with a "main" method that creates a "HashMap" and adds three entries: "David" with age 30, "José" with age 25, and "Juan" with age 35. The "main" method also prints the map, retrieves the age of "David", and iterates over the map entries. The output window at the bottom shows the execution results, which match the code's output.

```
1 import java.util.HashMap;
2 import java.util.Map;
3
4 public class Map_HashMap {
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 }
```

Age Map: {David=30, José=25, Juan=35}
Edad de David: 30
David: 30
José: 25
Juan: 35
Process finished with exit code 0

```
"C:\Program Files\Java\jdk-24\bin\java.exe" "-j
Age Map: {David=30, José=25, Juan=35}
Edad de David: 30
David: 30
José: 25
Juan: 35

Process finished with exit code 0
```

4. Box

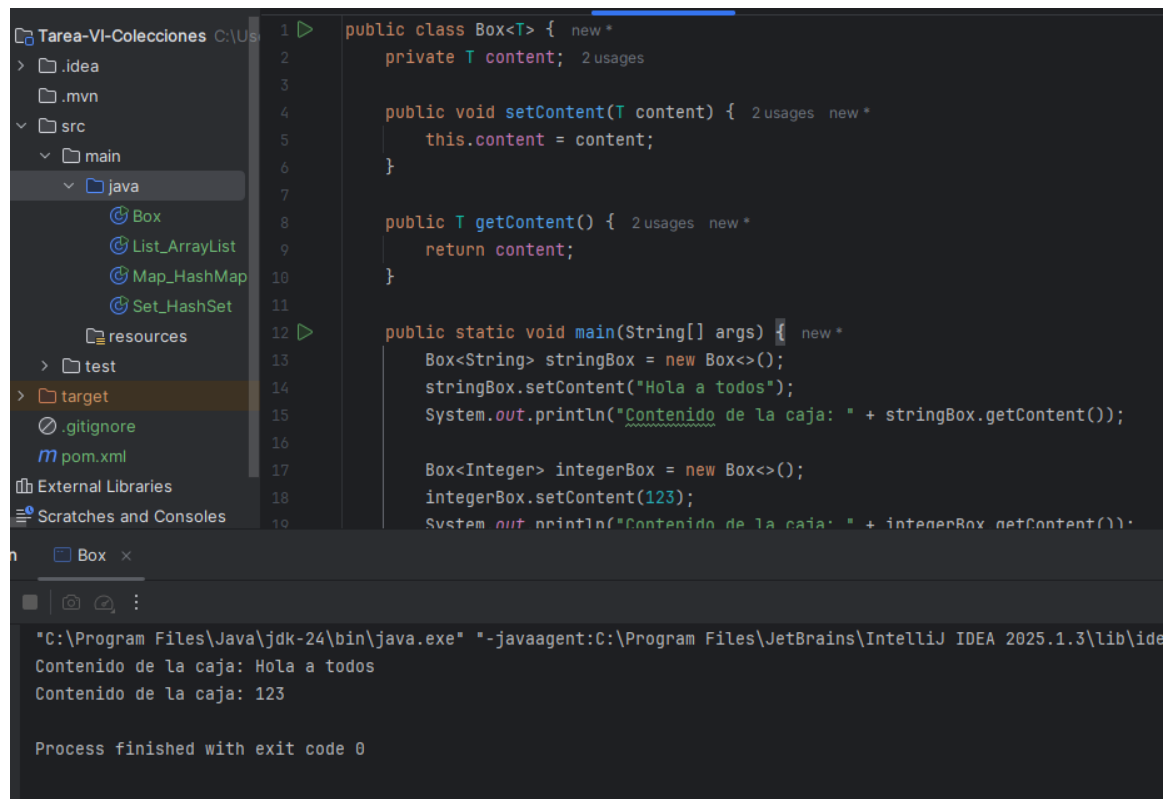
Código proporcionado:

```
public class Box<T> {  
    private T content;  
  
    public void setContent(T content) {  
        this.content = content;  
    }  
  
    public T getContent() {  
        return content;  
    }  
  
    public static void main(String[] args) {  
        Box<String> stringBox = new Box<>();  
        stringBox.setContent("Hola a todos");  
        System.out.println("Contenido de la caja: " + stringBox.getContent());  
  
        Box<Integer> integerBox = new Box<>();  
        integerBox.setContent(123);  
        System.out.println("Contenido de la caja: " + integerBox.getContent());  
    }  
}
```


Breve descripción del código:

- Se define la clase `Box<T>` (Type) que tiene un atributo `content` del tipo genérico `T`.
- Se crea una caja para `String` y se le asigna "Hola a todos".
- Se imprime su contenido.
- Se crea otra caja para `Integer` (tipo de envoltorio para el dato primitivo `int`) y se le asigna el número 123.
- Se imprime su contenido, se mostrará un mensaje con el texto y otro con el número que guardan las cajas.

Ejecución del Código:



The screenshot shows the IntelliJ IDEA IDE with a project named "Tarea-VI-Colecciones". The left sidebar displays the project structure, including the "src/main/java" directory. The main editor window shows the code for the `Box<T>` class and its `main` method. The code defines a generic class `Box<T>` with a private `content` attribute, `setContent` and `getContent` methods, and a `main` method that creates and prints the contents of two boxes: one for `String` and one for `Integer`.

```
1 public class Box<T> { new *
2     private T content; 2 usages
3
4     public void setContent(T content) { 2 usages new *
5         this.content = content;
6     }
7
8     public T getContent() { 2 usages new *
9         return content;
10    }
11
12    public static void main(String[] args) { new *
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        System.out.println("Contenido de la caja: " + integerBox.getContent());
20    }
```

The bottom panel shows the execution output:

```
"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.1.3\lib\ide
Contenido de la caja: Hola a todos
Contenido de la caja: 123

Process finished with exit code 0
```

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

Process finished with exit code 0
|
```

5. GenericMethod

Código proporcionado:

```
public class GenericMethod {

    public static <T> void printArray(T[] array) {
        for (T element : array) {
            System.out.println(element);
        }
    }

    public static void main(String[] args) {
        String[] stringArray = {"Java", "Generics", "Ejemplo"};
        Integer[] intArray = {1, 2, 3, 4, 5};

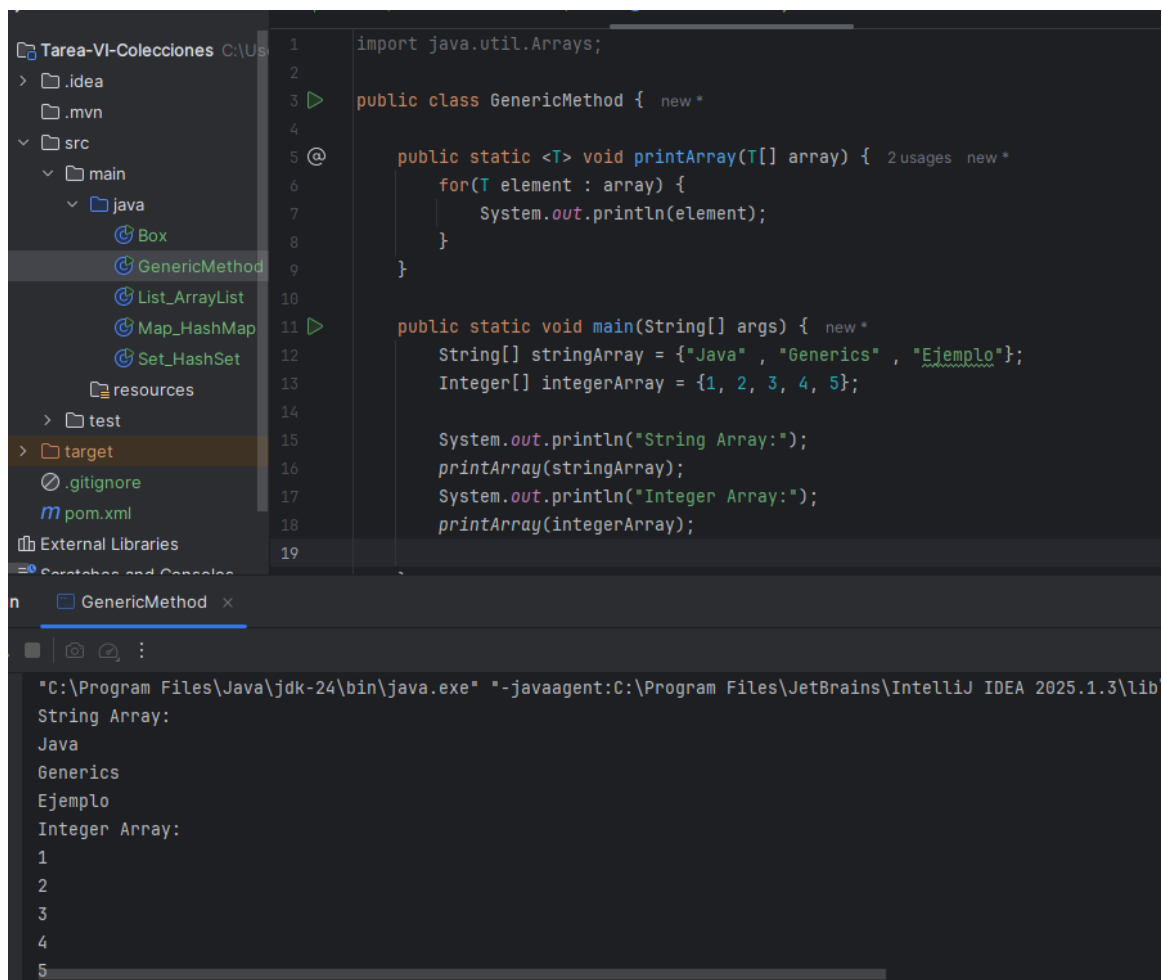
        System.out.println("String Array:");
        printArray(stringArray);

        System.out.println("Integer Array:");
        printArray(intArray);
    }
}
```

Breve descripción del código:

- Se define un método `printArray` que recibe un arreglo de cualquier tipo (`T[]`).
- Se crea un arreglo de cadenas: `{"Java", "Generics", "Ejemplo"}`.
- Se crea un arreglo de enteros: `{1, 2, 3, 4, 5}`.
- Se llama al método para cada arreglo.
- Dentro del método, un `for-each` recorre el arreglo e imprime cada elemento.
- La salida del código: primero se mostrarán los elementos del arreglo de cadenas. Y luego los números del arreglo de enteros, cada uno en una línea.

Ejecución del Código:



The screenshot displays the IntelliJ IDEA IDE. On the left, the Project Explorer shows the file structure of a project named 'Tarea-VI-Colecciones'. The 'src/main/java' directory is expanded, showing files like 'Box', 'GenericMethod', 'List_ArrayList', 'Map_HashMap', and 'Set_HashSet'. The 'GenericMethod' file is selected. The main editor window shows the following Java code:

```
1 import java.util.Arrays;
2
3 public class GenericMethod {
4
5     @
6     public static <T> void printArray(T[] array) {
7         for(T element : array) {
8             System.out.println(element);
9         }
10    }
11
12    public static void main(String[] args) {
13        String[] stringArray = {"Java", "Generics", "Ejemplo"};
14        Integer[] integerArray = {1, 2, 3, 4, 5};
15
16        System.out.println("String Array:");
17        printArray(stringArray);
18        System.out.println("Integer Array:");
19        printArray(integerArray);
20    }
21 }
```

Below the code editor, the 'Run' tab is active, showing the command used to execute the program: `"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.1.3\lib`. The output of the program is displayed in the console:

```
String Array:
Java
Generics
Ejemplo
Integer Array:
1
2
3
4
5
```

```
"C:\Program Files\Java\jdk-24\bin\java.exe" "-Xmx1G" "-Xms1G"
String Array:
Java
Generics
Ejemplo
Integer Array:
1
2
3
4
5

Process finished with exit code 0
```

6. Try Catch

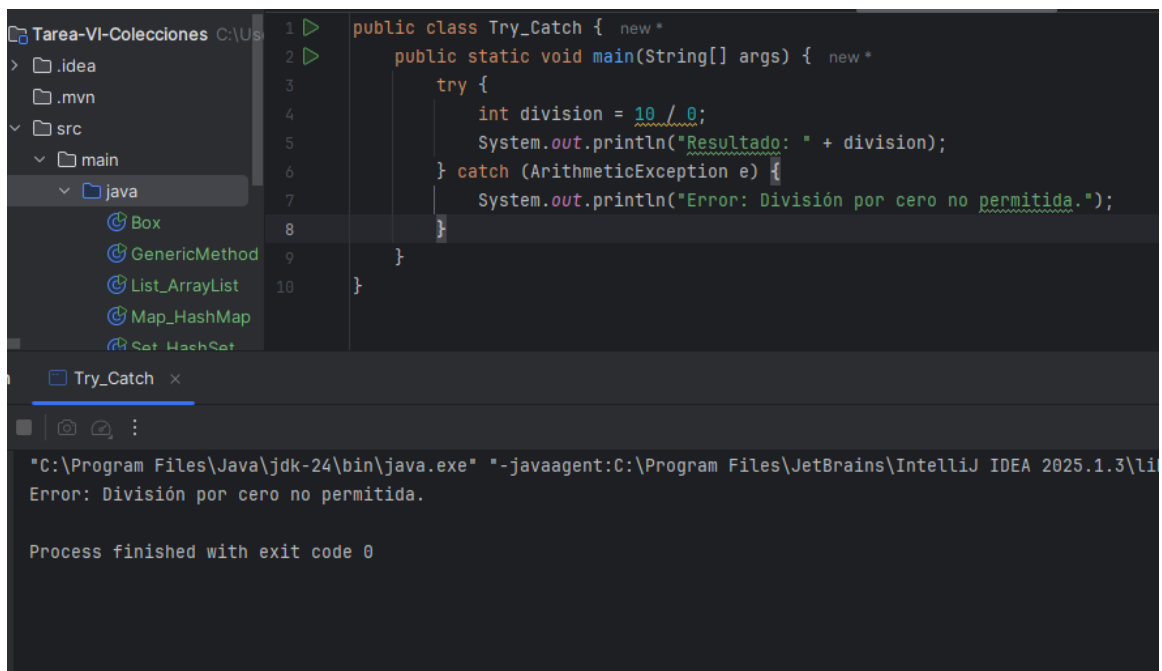
Código proporcionado:

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

Breve descripción del código:

- Se intenta hacer $10 / 0$, lo cual provoca un error matemático (división por cero).
- Como está dentro de un try, el programa no se detiene.
- El catch captura la excepción y muestra un mensaje indicando el error.
- La ejecución del código mostrara un mensaje que indica que la división por cero no esta permitida.

Ejecución del Código:

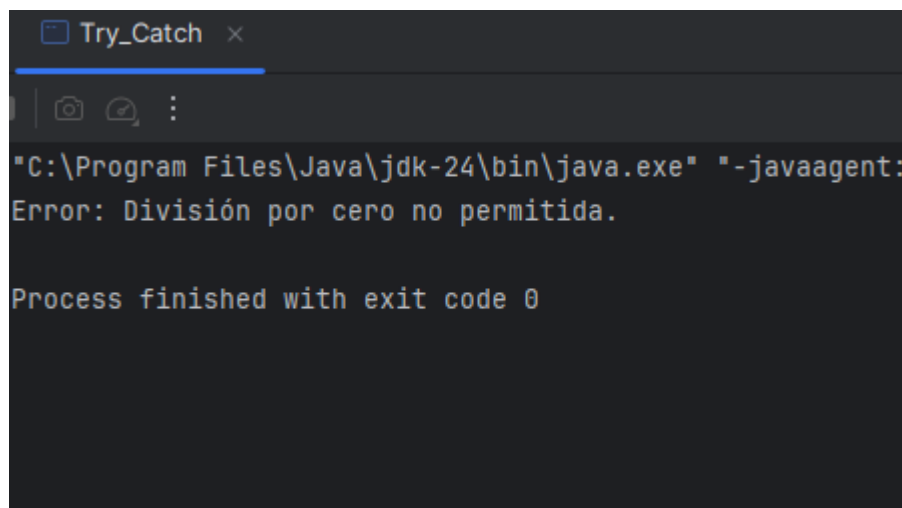


The screenshot shows the IntelliJ IDEA IDE. On the left, the Project Explorer displays the file structure: Tarea-VI-Colecciones > src > main > java. The main file is Try_Catch.java. The code editor shows the following Java code:

```
1 public class Try_Catch {  
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 }
```

Below the code editor, the Run tab shows the execution output:

```
"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.1.3\lib\idea_rt.jar" -Dfile.encoding=UTF-8  
Error: División por cero no permitida.  
  
Process finished with exit code 0
```



This is a close-up of the Run tab output from the previous screenshot. It shows the command executed and the resulting error message:

```
"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.1.3\lib\idea_rt.jar" -Dfile.encoding=UTF-8  
Error: División por cero no permitida.  
  
Process finished with exit code 0
```

7. Try-Catch-Finally

Código proporcionado:

```
import java.io.FileInputStream;
import java.io.IOException;

public class Main {
    public static void main(String[] args) {
        FileInputStream fis = null;
        try {
            fis = new FileInputStream("archivo.txt");
            System.out.println("Archivo abierto correctamente.");
            int data;
            while ((data = fis.read()) != -1) {
                System.out.print((char) data);
            }
        } catch (IOException e) {
            System.out.println("Error al leer el archivo: " + e.getMessage());
        } finally {
            System.out.println("Sección finally");
            try {
                if (fis != null) {
                    fis.close();
                    System.out.println("Archivo cerrado correctamente.");
                }
            } catch (IOException e) {
                System.out.println("Error al cerrar el archivo: " + e.getMessage());
            }
        }
    }
}
```

```

    }
}
}

```

Breve descripción del código:

- Se intenta abrir un archivo llamado archivo.txt con FileInputStream.
- Si el archivo existe:
- Se imprime un mensaje indicando que se abrió correctamente.
- Luego, se lee el archivo carácter por carácter hasta el final.
- Si no existe, el catch muestra un mensaje de error.
- El bloque finally siempre se ejecuta, cerrando el archivo si estaba abierto.
- La salida esperada:
- Si el archivo existe: un mensaje de apertura, seguido de su contenido, luego un mensaje indicando que se cerró correctamente.
- Si no existe: un mensaje de error, seguido de "Sección finally".

Ejecución del Código:

```

1  import java.io.FileInputStream;
2  import java.io.IOException;
3
4  public class Bloque_Finally {
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             }
24         }
25     }
26 }

```

"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.1.3\lib\idea_
 Error al leer el archivo: archivo.txt (El sistema no puede encontrar el archivo especificado)
 Sección finally
 Process finished with exit code 0

```

"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA
Error al leer el archivo: archivo.txt (El sistema no puede encontrar el archivo especificado)
Sección finally

Process finished with exit code 0

```

8. Throw

Código proporcionado:

```

public class Main {

    public static void main(String[] args) {

        try {

            validateAge(15);

        } catch (Exception e) {

            System.out.println("Error: " + e.getMessage());

        }

    }

}

public static void validateAge(int age) throws Exception {

    if (age < 18) {

        throw new Exception("La edad debe ser mayor o igual a 18.");

    } else {

        System.out.println("Edad válida: " + age);

    }

}

```



```

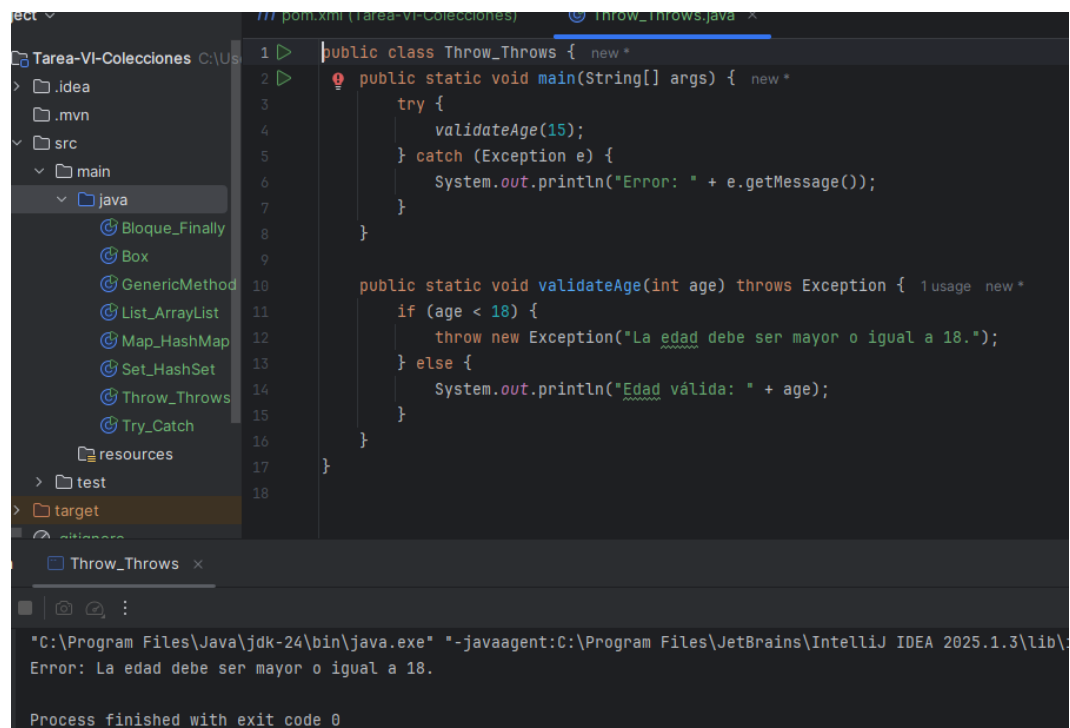
    }
}
}

```

Breve descripción del código:

- El método `validateAge` recibe una edad
- Si la edad es menor que 18, lanza una excepción con un mensaje de error.
- Si la edad es 18 o más, imprime que la edad es válida.
- En el `main`, se llama con la edad 15, por lo que se captura la excepción y se imprime el mensaje de error.
- La salida será: mostrará un mensaje de error que indicar que la edad debe ser mayor o igual a 18.

Ejecución del Código:



```

1 public class Throw_Throws {
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 }
18

```

Process finished with exit code 0

```
"C:\Program Files\Java\jdk-24\bin\java.exe" "-java  
Error: La edad debe ser mayor o igual a 18.  
  
Process finished with exit code 0  
|
```

9. Excepciones_Personalizadas

Código Proporcionado:

```
class InvalidAgeException extends Exception {  
    public InvalidAgeException(String message) {  
        super("Código de error: 1234 - " + message);  
    }  
}  
  
public class Main {  
    public static void main(String[] args) {  
        try {  
            checkAge(15);  
        } catch (InvalidAgeException e) {  
            System.out.println("Error: " + e.getMessage());  
        }  
        try{  
            checkAgeDefault(15);  
        }catch (Exception e) {  
            System.out.println("Error: " + e.getMessage());  
        }  
    }  
  
    public static void checkAgeDefault(int age) throws Exception {  
        if (age < 18) {  
            throw new Exception("La edad debe ser mayor o igual a 18.");  
        } else {  
            System.out.println("Edad válida: " + age);  
        }  
    }  
}
```

```
}

public static void checkAge(int age) throws InvalidAgeException {
    if (age < 18) {

        throw new InvalidAgeException("La edad debe ser mayor o igual a 18.");
    } else {
        System.out.println("Edad válida: " + age);
    }
}
}
```

Breve explicación del Código:

- Se define una clase InvalidAgeException que extiende de Exception.
- Esta excepción agrega un código de error al mensaje.
- En el main, se prueban dos métodos de validación: checkAge(15) que lanza la excepción personalizada.
- checkAgeDefault(15) que lanza una excepción normal.
- Ambos son capturados y muestran sus mensajes.
- Al ejecutar el código mostrara dos mensajes de error, uno con el código personalizado y otro con el mensaje normal.

Ejecución del Código:

The screenshot shows the IntelliJ IDEA IDE with the following code:

```

1  class InvalidAgeException extends Exception {
2      public InvalidAgeException(String message) {
3          super("Código de error: 1234 - " + message);
4      }
5  }
6
7  public class Excepciones_Personalizadas {
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 {

```

The output window shows the following execution results:

```

"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.1.3\lib\id
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

```

```

"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaagent:C:\Program Fi
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

```