



# Universidad Mariano Gálvez de Guatemala

## Facultad de Ingeniería en Sistemas

Nombre	Josué David Barrera Santos	Carné	9989-24-1944
--------	----------------------------	-------	--------------

Carrera	Ingeniería en Sistemas	Código de Carrera	9941
Asignatura	Programación II	Código de Curso	017
Ciclo	Cuarto Ciclo	Jornada	Domingos
Catedrático	Ing. MBA David Alvarez	Fecha	28/09/2025
Semestre	Segundo Semestre	Sección	D

## Objetivo

Evaluar el funcionamiento de los distintos tipos de excepciones, expresiones lambda, streams, colecciones y otros conceptos aplicados en Java.

## Requerimientos

- Clonar y analizar cada uno de los siguientes repositorios de GitHub:
  - [https://github.com/dalvarez-umg/List\\_ArrayList](https://github.com/dalvarez-umg/List_ArrayList)
  - [https://github.com/dalvarez-umg/Set\\_HashSet](https://github.com/dalvarez-umg/Set_HashSet)
  - [https://github.com/dalvarez-umg/Map\\_HashMap](https://github.com/dalvarez-umg/Map_HashMap)
  - [https://github.com/dalvarez-umg/Generic\\_Class](https://github.com/dalvarez-umg/Generic_Class)
  - [https://github.com/dalvarez-umg/Generic\\_Method](https://github.com/dalvarez-umg/Generic_Method)
  - [https://github.com/dalvarez-umg/Try\\_Catch](https://github.com/dalvarez-umg/Try_Catch)
  - [https://github.com/dalvarez-umg/Bloque\\_Finally](https://github.com/dalvarez-umg/Bloque_Finally)
  - [https://github.com/dalvarez-umg/Throw\\_Throws](https://github.com/dalvarez-umg/Throw_Throws)
  - [https://github.com/dalvarez-umg/Excepciones\\_Personalizadas](https://github.com/dalvarez-umg/Excepciones_Personalizadas)
- Ejecutar y documentar pruebas de funcionamiento de cada repositorio (capturas de pantalla con explicación breve de qué hace el programa).
- Redactar un documento con carátula oficial que incluya:
- Breve descripción de cada ejemplo.
- Evidencia (capturas de ejecución).

## Entrega

Enlace a tu repositorio de GitHub con los 9 ejemplos.  
Documento de pruebas en PDF.

## Rúbrica de Evaluación – Control de notas de estudiantes

Criterio	Descripción detallada	Punteo
Repositorio en GitHub	Contiene los 9 ejemplos organizados y ejecutables	1pt
Evidencia de pruebas	Todas las pruebas se documentan con capturas claras y explicación breve	1pt
Análisis individual	Cada ejemplo incluye descripción clara de su funcionamiento	1pt
Conclusiones	Reflexión personal clara sobre lo aprendido en excepciones, colecciones, streams y lambda	1pt

```
1 package org.example;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 public class Main {
7     public static void main(String[] args) {
8         List<String> fruits = new ArrayList<>();
9         fruits.add("Manzana");
10        fruits.add("Banano");
11        fruits.add("Naranja");
12
13        System.out.println("Frutas: " + fruits);
14
15        // Acceder a elementos
16        String firstFruit = fruits.get(0);
17        System.out.println("Primera fruta: " + firstFruit);
18
19        // Iterar sobre la lista
20        for (String fruit : fruits) {
21            System.out.println(fruit);
22        }
23    }
24 }
```

Run Main

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

Process finished with exit code 0

**ArrayList:** este programa crea una lista de frutas, muestra toda la lista, accede a la primera fruta y luego imprime cada fruta en una línea por separado.

```
1 package org.example;
2
3 import java.util.HashSet;
4 import java.util.Set;
5
6 public class Main {
7     public static void main(String[] args) {
8         Set<String> uniqueNames = new HashSet<>();
9         uniqueNames.add("David");
10        uniqueNames.add("Juan");
11        uniqueNames.add("David"); // No se añadirá ya que es duplicado
12
13        System.out.println("Unique Names: " + uniqueNames);
14
15        // Iterar sobre el set
16        for (String name : uniqueNames) {
17            System.out.println(name);
18        }
19    }
20 }
```

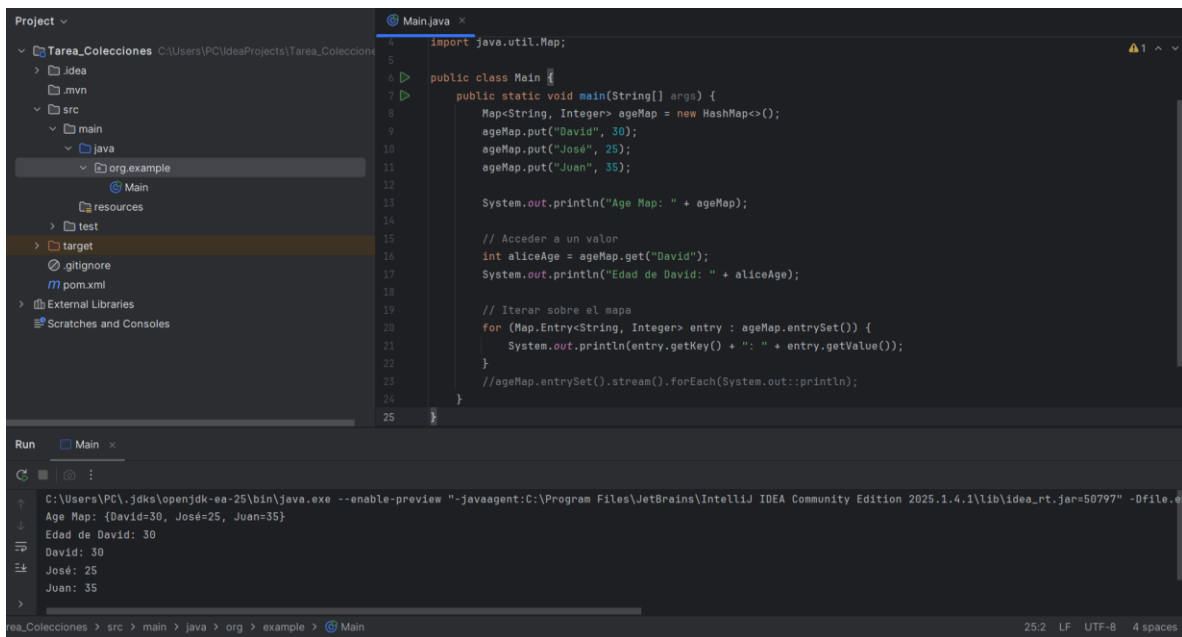
Run Main

C:\Users\PC\.jdk\openjdk-ea-25\bin\java.exe --enable-preview --javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2025.1.4.1\lib\idea\_rt.jar=50773 -Dfile.encoding=UTF-8

Unique Names: [David, Juan]  
David  
Juan

Process finished with exit code 0

**HashSet:** este programa guarda nombres en un hashset, y evita que se dupliquen automáticamente luego imprime todos los datos sin repetir.



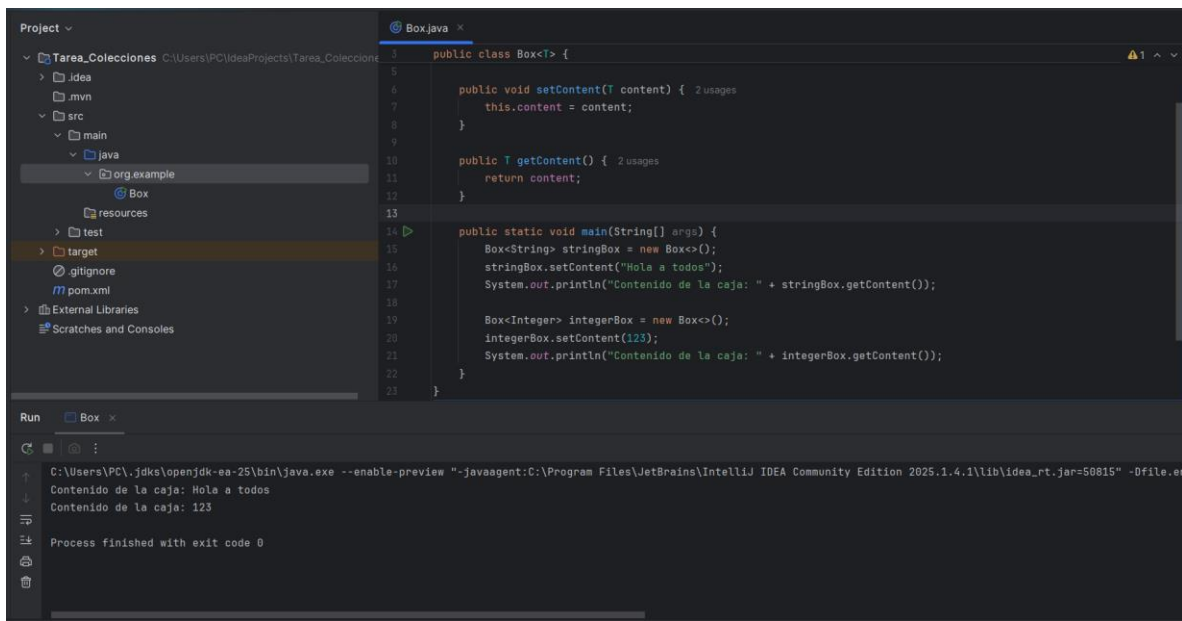
The screenshot shows the IntelliJ IDEA interface. On the left, the Project view displays the file structure of 'Tarea\_Colecciones', including 'src/main/java/org/example/Main'. The main editor shows the code for 'Main.java':

```
1 import java.util.Map;
2
3 public class Main {
4     public static void main(String[] args) {
5         Map<String, Integer> ageMap = new HashMap<>();
6         ageMap.put("David", 30);
7         ageMap.put("José", 25);
8         ageMap.put("Juan", 35);
9
10        System.out.println("Age Map: " + ageMap);
11
12        // Acceder a un valor
13        int aliceAge = ageMap.get("David");
14        System.out.println("Edad de David: " + aliceAge);
15
16        // Iterar sobre el mapa
17        for (Map.Entry<String, Integer> entry : ageMap.entrySet()) {
18            System.out.println(entry.getKey() + ": " + entry.getValue());
19        }
20        //ageMap.entrySet().stream().forEach(System.out::println);
21    }
22 }
```

The Run window at the bottom shows the execution output:

```
C:\Users\PC\jdk\openjdk-ea-25\bin\java.exe --enable-preview --javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2025.1.4.1\lib\idea_rt.jar=50797 --Dfile.encoding=UTF-8
Age Map: {David=30, José=25, Juan=35}
Edad de David: 30
David: 30
José: 25
Juan: 35
```

**HashMap:** este programa guarda nombres con sus edades, muestra todo el mapa, accede a la edad de un nombre específico y luego imprime todos los pares.



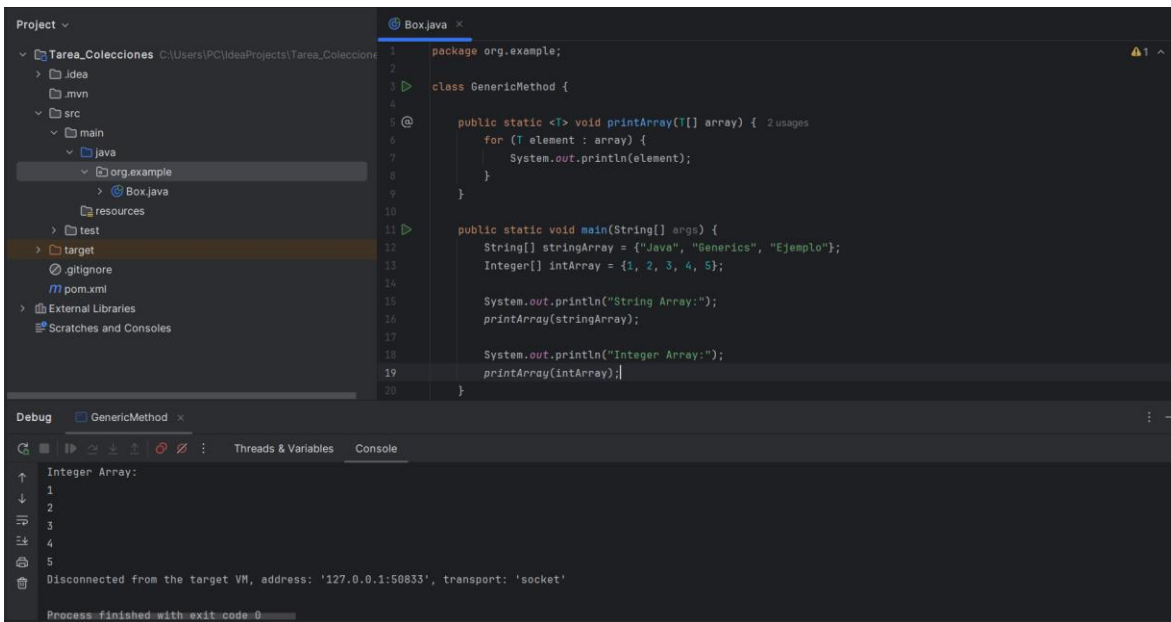
The screenshot shows the IntelliJ IDEA interface. On the left, the Project view displays the file structure of 'Tarea\_Colecciones', including 'src/main/java/org/example/Box'. The main editor shows the code for 'Box.java':

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

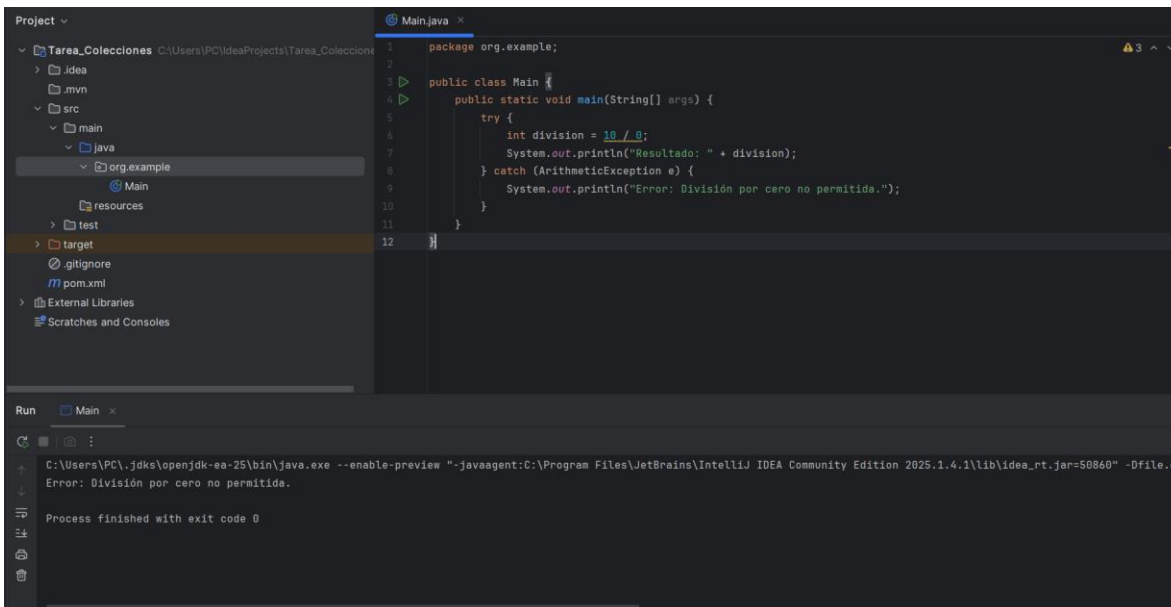
The Run window at the bottom shows the execution output:

```
C:\Users\PC\jdk\openjdk-ea-25\bin\java.exe --enable-preview --javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2025.1.4.1\lib\idea_rt.jar=50815 --Dfile.encoding=UTF-8
Contenido de la caja: Hola a todos
Contenido de la caja: 123
Process finished with exit code 0
```

**BOX:** este programa define una clase genérica (Box) que puede guardar un valor de cualquier tipo. Luego se crean dos cajas: una para String y otra para Integer, demostrando cómo los genéricos permiten reutilizar la misma clase con diferentes tipos de datos sin tener que escribir clases separadas.



**printArray:** este programa define un método genérico (`printArray`) que puede imprimir arreglos de cualquier tipo de dato sin necesidad de duplicar código para cada tipo.



**Try\_Catch:** este programa intenta hacer una división por cero, lo que normalmente generaría un error y detendría el programa.

Gracias a `try-catch`, el error se controla y se muestra un mensaje en lugar de un fallo.

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

```
Run: Main
C:\Users\PC\.jdk\openjdk-ea-25\bin\java.exe --enable-preview --javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2025.1.4.1\lib\idea_rt.jar=50966 -Dfile.encoding=UTF-8
Error al leer el archivo: archivo.txt (El sistema no puede encontrar el archivo especificado)
Sección finally
Process finished with exit code 0
```

**Finally:** Este programa intenta abrir un archivo .txt y leerlo, si lo encuentra lo imprime, si no lo encuentra muestra un error igualmente si lo encuentra o no se asegura de cerrar el archivo automáticamente.

```
Project: Tarea_Colecciones
src/main/java/org/example/Main.java
1 package org.example;
2
3 public class Main {
4     public static void main(String[] args) {
5         try {
6             validateAge(15);
7         } catch (Exception e) {
8             System.out.println("Error: " + e.getMessage());
9         }
10    }
11
12    public static void validateAge(int age) throws Exception {
13        if (age < 18) {
14            throw new Exception("La edad debe ser mayor o igual a 18.");
15        } else {
16            System.out.println("Edad válida: " + age);
17        }
18    }
19 }
```

```
Run: Main
C:\Users\PC\.jdk\openjdk-ea-25\bin\java.exe --enable-preview --javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2025.1.4.1\lib\idea_rt.jar=50991 -Dfile.encoding=UTF-8
Error: La edad debe ser mayor o igual a 18.
Process finished with exit code 0
```

**Throw\_Throws:** este programa asegura en comprobar la edad si es menor muestra un error y si es mayor imprime que es edad valida.

The screenshot displays the IntelliJ IDEA IDE interface. The top-left pane shows the project structure for 'Tarea\_Colecciones', with the file 'Main.java' selected under 'src/main/java/org/example'. The top-right pane shows the code of 'Main.java', which includes a 'main' method and two static methods: 'checkAgeDefault' and 'checkAge'. The 'main' method calls 'checkAgeDefault(15)' and catches any 'Exception'. The 'checkAgeDefault' method throws an 'Exception' if the age is less than 18. The 'checkAge' method throws an 'InvalidAgeException' if the age is less than 18. The bottom pane shows the 'Run' output, which includes the command to run the program, the error message 'Error: Código de error: 1234 - La edad debe ser mayor o igual a 18.', and the message 'Process finished with exit code 0'.

```
Project: Tarea_Colecciones
src/main/java/org/example/Main.java

public class Main {
    public static void main(String[] args) {
        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.");
        }
    }
}
```

Run: Main

C:\Users\PC\jdk\openjdk-25\bin\java.exe --enable-preview "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2025.1.4.1\lib\idea\_rt.jar=51014" -Dfile.encoding=UTF-8

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

**Excepción:** Este programa compara dos formas de validar edad: con excepción personalizada vs excepción genérica.