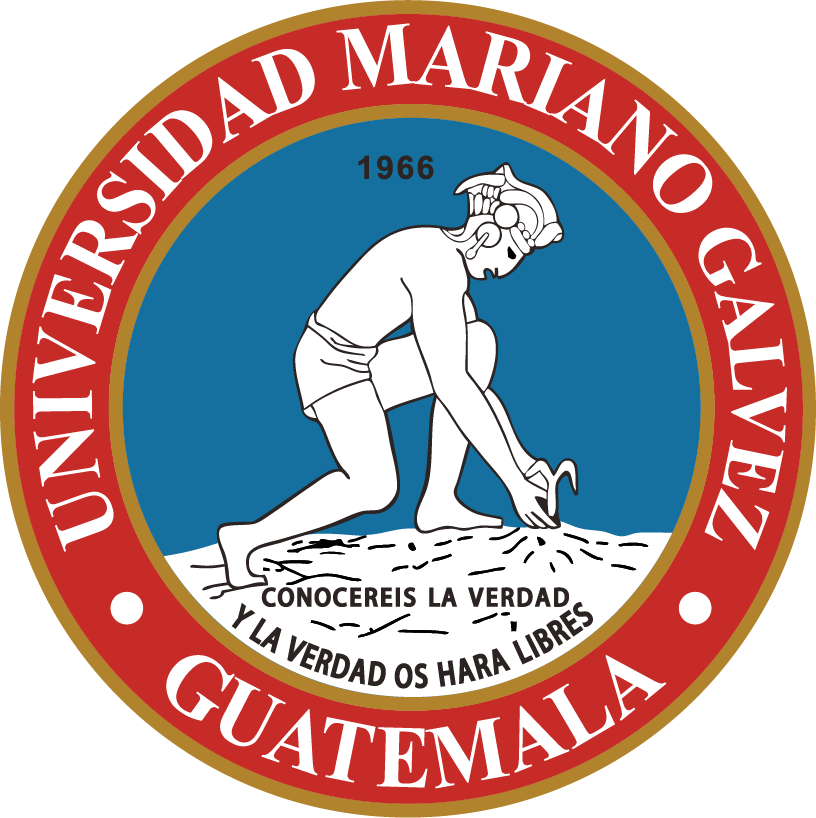
**Universidad Mariano Gálvez de Guatemala**

**Facultad de Ingeniería en Sistemas**

**Sede Portales**

****

**Curso**

**Programación II**

**Catedrático**

**Ing. MBA David Alvarez**

**Tarea No. 6**

**Ejemplos de Excepciones**

**Guatemala, 28 de septiembre de 2025**

Ejemplo 1: **Array List**

Este código demuestra el uso básico de un ArrayList en Java. Crea una lista dinámica de frutas, agrega elementos, muestra la lista completa, accede al primer elemento usando el índice 0, y finalmente recorre todos los elementos con un ciclo for-each para imprimirlos individualmente.

**Texto

El contenido generado por IA puede ser incorrecto.**

Ejemplo 2: **Has Shet**

Este código demuestra el uso de un HashSet en Java, que es una colección que no permite elementos duplicados. Crea un conjunto de nombres únicos, intenta agregar "David" dos veces pero el HashSet automáticamente ignora el duplicado, dejando solo valores únicos. Luego imprime el conjunto completo y recorre cada nombre individualmente con un ciclo for-each. La característica principal es que garantiza que no haya repeticiones en la colección.

**Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.**

Ejemplo 3: **Hash Map**

Este código demuestra el uso de un HashMap en Java, que es una estructura de datos que almacena pares clave-valor. Crea un mapa que asocia nombres de personas con sus edades, agrega tres entradas (David con 30 años, José con 25, y Juan con 35), imprime el mapa completo, accede a la edad de David usando su nombre como clave, y finalmente recorre todas las entradas del mapa mostrando cada par nombre-edad. La ventaja del HashMap es que permite acceder rápidamente a valores usando sus claves únicas.

Texto

El contenido generado por IA puede ser incorrecto.

Ejemplo 4: **Generic Class**

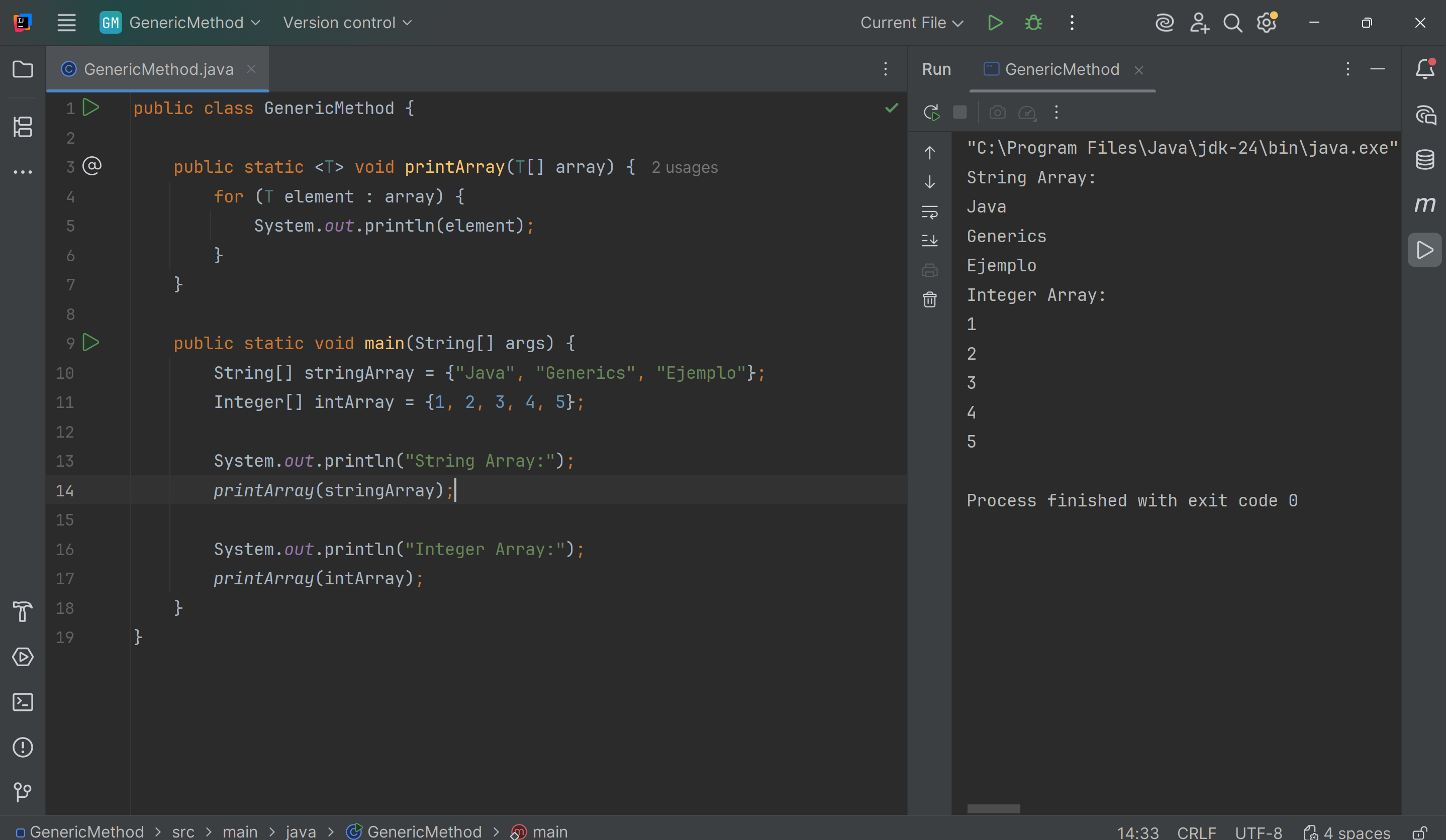
Este código demuestra el uso de genéricos en Java mediante una clase Box que puede almacenar cualquier tipo de dato. La clase usa un parámetro de tipo genérico <T> que actúa como un contenedor flexible, permitiendo guardar y recuperar un valor de cualquier tipo especificado al crear la instancia. Demostrando cómo la misma clase puede trabajar con distintos tipos de datos de forma segura y reutilizable sin necesidad de crear múltiples versiones de la clase.

Texto

El contenido generado por IA puede ser incorrecto.

Ejemplo 5: **Generic Method**

Este código usa genéricos para crear un método que imprime arrays. El método printArray puede recibir arrays de cualquier tipo (String, Integer, etc.) y los recorre para mostrar cada elemento. En el ejemplo se imprimen un array de palabras y otro de números usando el mismo método.



Ejemplo 6: **Try Catch**

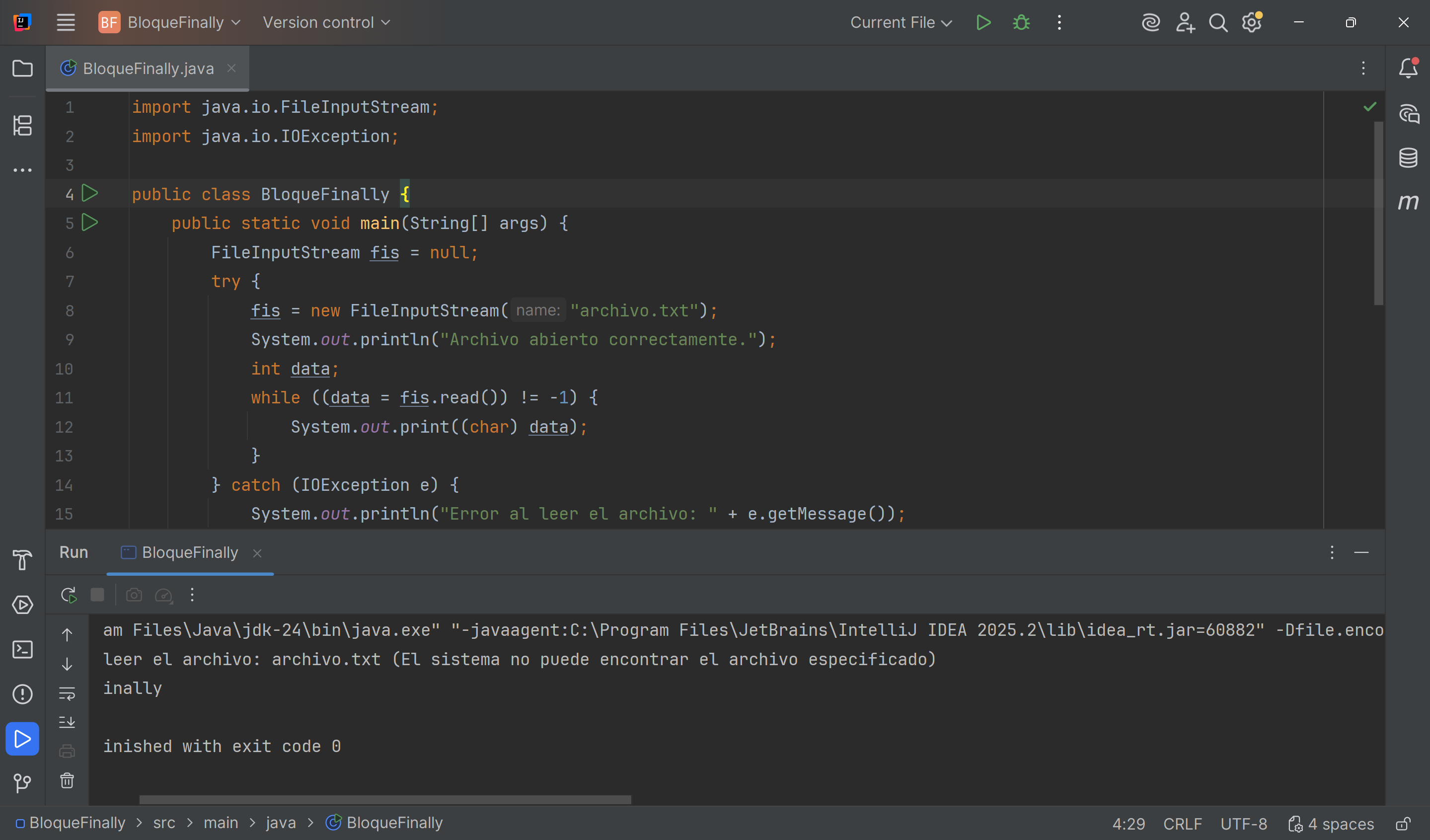
Este código muestra cómo manejar excepciones en Java usando try-catch. Intenta dividir 10 entre 0, lo cual genera un error. El bloque catch captura la excepción ArithmeticException y en lugar de que el programa se detenga, muestra un mensaje de error amigable indicando que no se puede dividir por cero.

Texto

El contenido generado por IA puede ser incorrecto.

Ejemplo 7: **Bloque Finally**

Este código demuestra el manejo de archivos con try-catch-finally. Intenta abrir y leer el archivo "archivo.txt" mostrando su contenido carácter por carácter. Si hay algún error al leer, el catch lo captura y muestra el mensaje de error. El bloque finally garantiza que el archivo siempre se cierre correctamente al final, incluso si ocurrió un error, liberando así los recursos del sistema.



Ejemplo 8: **Throw Throws**

Este código muestra cómo lanzar excepciones personalizadas en Java. El método validateAge verifica si una edad es mayor o igual a 18, y si no lo es, lanza una excepción con un mensaje de error. En el main se llama al método con la edad 15, lo que genera la excepción que es capturada por el catch y muestra el mensaje de error indicando que la edad debe ser mayor o igual a 18.

Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

Ejemplo 9: **Excepciones Personalizadas**

Este código muestra cómo crear y usar excepciones personalizadas en Java. Se define una clase InvalidAgeException que extiende de Exception y agrega un código de error al mensaje. Luego se crean dos métodos: checkAge que usa la excepción personalizada y checkAgeDefault que usa la excepción genérica de Java. Ambos validan que la edad sea mayor o igual a 18, pero la excepción personalizada permite tener mensajes más específicos y estructurados con códigos de error.

