



Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

Ingeniería en computación

Laboratorios de computación salas A y B

Profesor: _____ Ing. Rene Adrian Davila Perez

Asignatura: _____ Programación Orientada a Objetos

Grupo: _____ 1

No. de práctica: _____ Practica 9 y 10

Integrante: _____ 424121462

Semestre: _____ 2025-1

Fecha de entrega: _____ 25 de octubre de 2024

Observaciones: _____

CALIFICACIÓN: _____

Índice

1. Introducción (2)
<hr/>	
2. Marco Teórico (3)
<hr/>	
3. Desarrollo (4-6)
<hr/>	
4. Resultados (6-8)
<hr/>	
5. Conclusiones (8)
<hr/>	
6. Referencias (9)
<hr/>	

Introducción:

El proyecto se centra en la creación de excepciones personalizadas en Java para gestionar y controlar errores que puedan surgir en diversas situaciones comunes de programación. Se plantea la siguiente estructura:

- **Ejercicio 1:** Definir una excepción usando un bloque try-catch para capturar una división entre cero.
- **Ejercicio 2:** Crear una excepción personalizada que capture una raíz cuadrada cuando se intente calcular con un valor negativo, empleando throws de las excepciones.
- **Ejercicio 3:** Definir una excepción personalizada para leer una lista de enteros introducida por el usuario y lanzar una si hay números duplicados.
- **Ejercicio 4:** Escribir un programa en Java que defina un método que reciba una cadena como entrada y lance una excepción si no contiene vocales.

Adicionalmente, los ejercicios deberán cumplir con los siguientes requisitos: encapsulación, definición en un paquete, generación de archivos jar y javadoc, así como la inclusión de diagramas UML estáticos y dinámicos.

Es esencial dar solución a este problema debido a que el manejo correcto de excepciones en programación es clave para asegurar que los sistemas sean eficientes, manejando errores de manera controlada y evitando que el programa falle. Además, la creación de excepciones personalizadas permite capturar errores específicos, estos errores siendo clave para la correcta resolución de los programas, lo que proporciona una mayor gestión de errores.

Adicionalmente se tienen como objetivos:

- Aplicar los conceptos de manejo de excepciones en Java mediante try-catch y throws.
- Implementar excepciones personalizadas para capturar errores en situaciones concretas.
- Cumplir con los principios de encapsulamiento y organización vistos en clase mediante paquetes en Java.
- Generar archivos jar y documentación utilizando javadoc para asegurar la correcta entrega del proyecto.
- Producir diagramas UML estáticos y dinámicos que reflejen el diseño y comportamiento del sistema.

Se espera que estos ejercicios permitan una mejor comprensión y manejo del control de errores en aplicaciones Java, promoviendo buenas prácticas y garantizando la estabilidad del código frente a situaciones inesperadas.

Marco Teórico:

Las excepciones en Java son mecanismos que permiten gestionar situaciones inesperadas o errores que ocurren durante la ejecución de un programa. Su función principal es separar el código normal del código que maneja las excepciones.

Las excepciones son eventos que alteran el flujo normal de ejecución de un programa. En Java, las excepciones son objetos derivados de la clase base **Throwable**, la cual tiene dos subclases principales: **Error** y **Exception**. Mientras que los errores (**Error**) generalmente son problemas graves relacionados con el entorno de ejecución, las excepciones (**Exception**) son problemas que el programa puede manejar y que no alteran de sobremanera su funcionamiento.

Las excepciones son clasificadas en dos tipos [1]:

- Checked Exceptions: Son las que se deben manejar explícitamente en tiempo de compilación, utilizando bloques try-catch o la cláusula throws.
- Unchecked Exceptions: Son subclases de **RuntimeException**, no es obligatorio manejarlas en tiempo de compilación, pero pueden ser atrapadas en tiempo de ejecución.

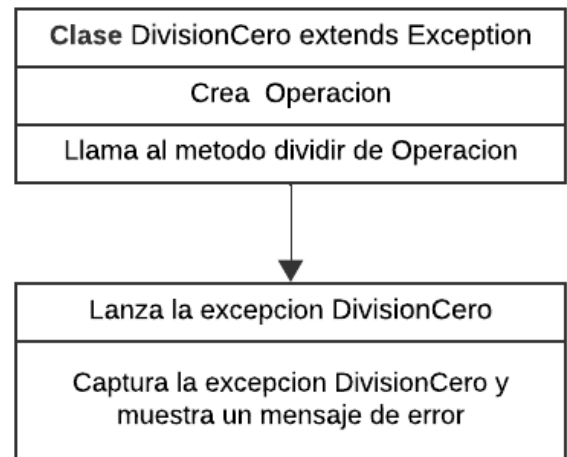
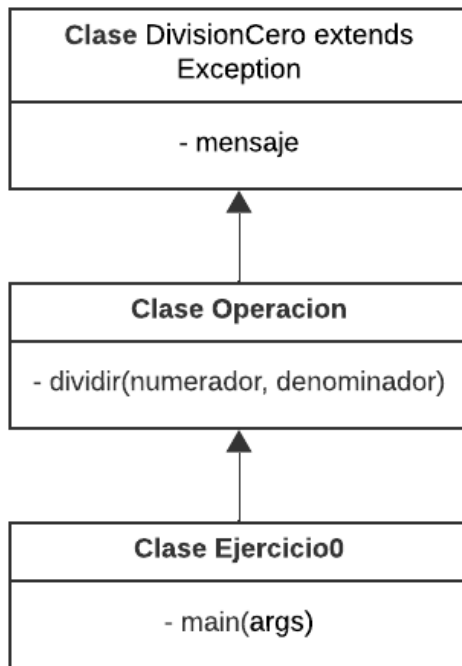
El manejo de excepciones permite identificar y reaccionar ante situaciones de error. Un bloque try encierra el código que puede lanzar una excepción, mientras que un bloque catch captura y gestiona la excepción correspondiente.

En Java, es posible crear excepciones personalizadas heredadas de las clases **Exception** o **RuntimeException**. Esto permite definir reglas de error propias, adaptadas a las necesidades específicas del proyecto o programa en el cual se esté trabajando.

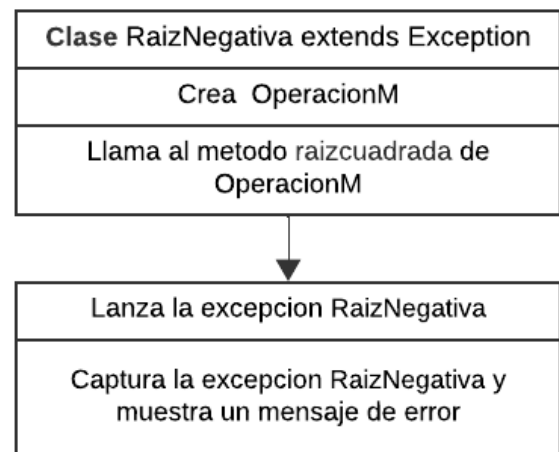
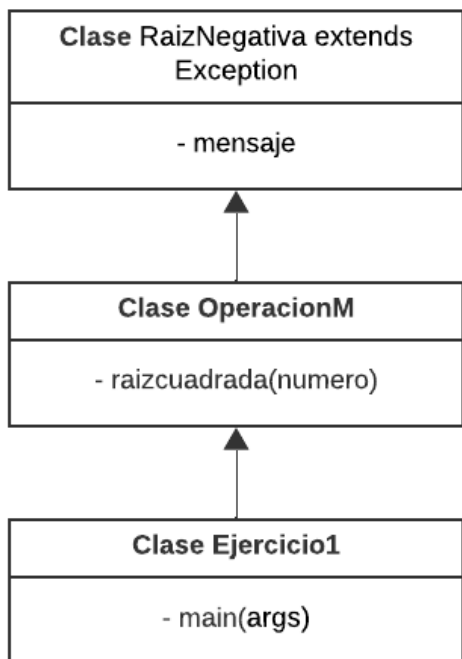
Los throws se utilizan para propagar una excepción desde un método hacia donde sea llamado, permitiendo que este último decida cómo manejarla. Este concepto es útil cuando no se desea o no es posible manejar la excepción dentro del mismo método en el que se origina.

En resumen, el manejo adecuado de excepciones es fundamental para construir proyectos de manera robusta y confiable, haciendo que el manejo de los errores sea bueno, permitiendo encontrarlos y resolverlos de mejor manera. Las prácticas que se desarrollarán aplican estos conceptos para gestionar de manera efectiva los errores, utilizando las herramientas proporcionadas por Java como excepciones personalizadas, bloques try-catch, y los throws. [2]

Desarrollo:

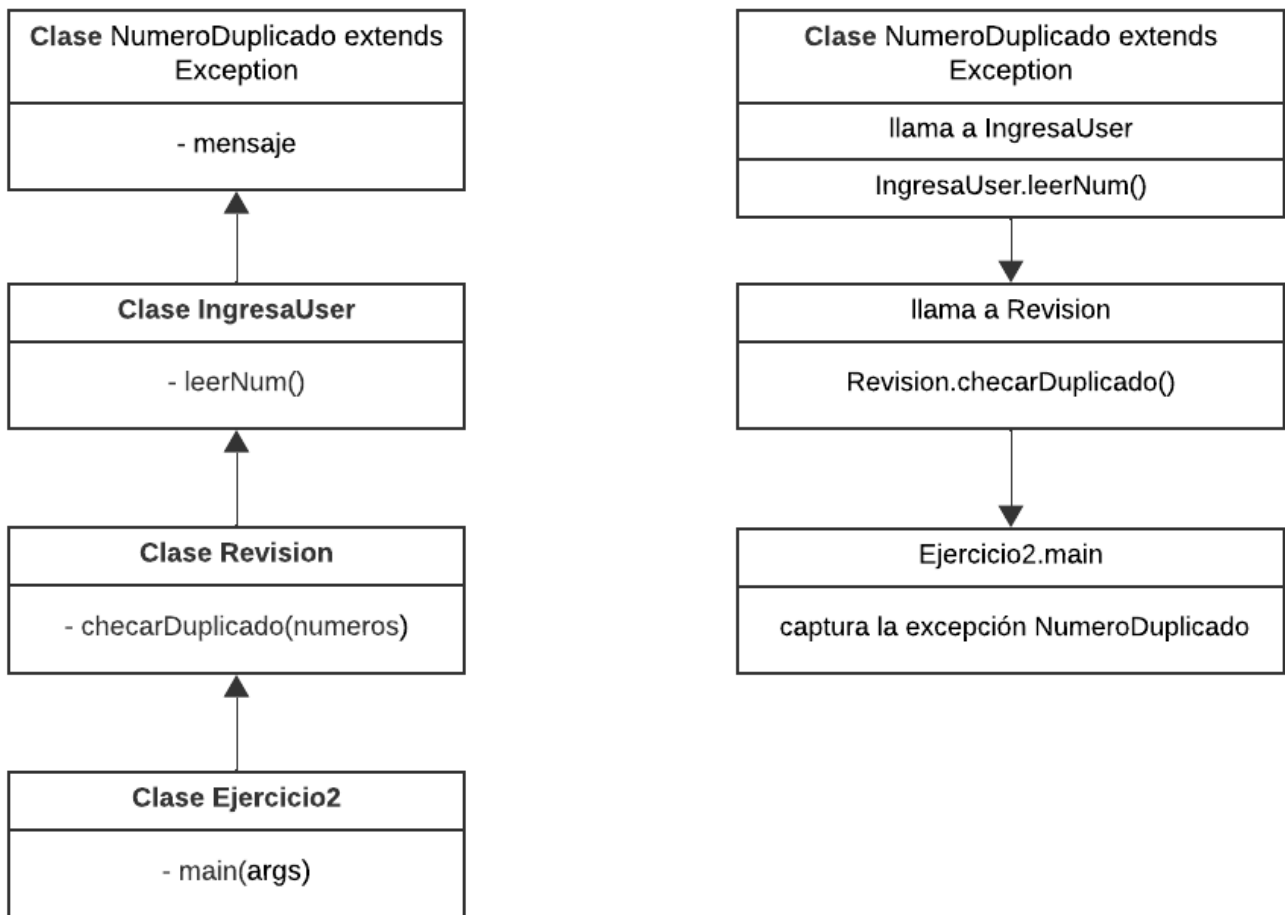


Para el primer ejercicio se ingresaron los valores de 10 y 0 para el numerador y denominador respectivamente, por lo que se espera que se arroje la excepción personalizada que se creo, una vez ejecutado el programa se puede observar en pantalla que si se arrojó la excepción dando como resultado lo esperado.

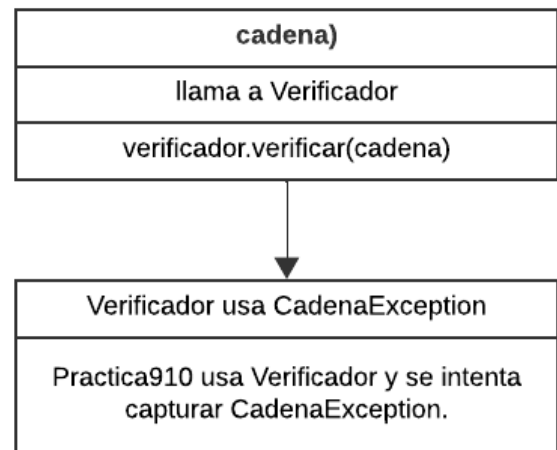
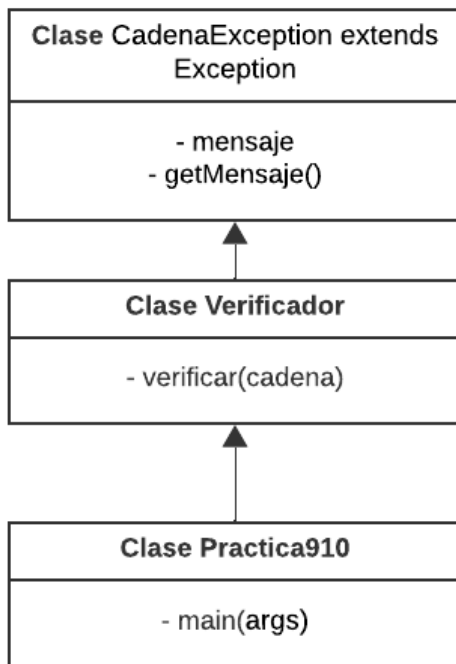


Para el segundo ejercicio se ingresó como número el -9.0, esperando como resultado que se arroje la excepción personalizada que no permite la raíz cuadrada de un número negativo, a la

hora de ejecutarse pudimos observar que esto fue lo que dio como resultado por lo que el código planteado no contaba con errores visibles.



Para el tercer ejercicio se ejecutó dos veces el mismo código, en ambas se ingresó como cantidad de números 5, para así probar las dos opciones, una donde no se repitieran números y otra donde sí, esperando que de manera adecuada y se lea de manera sencilla.



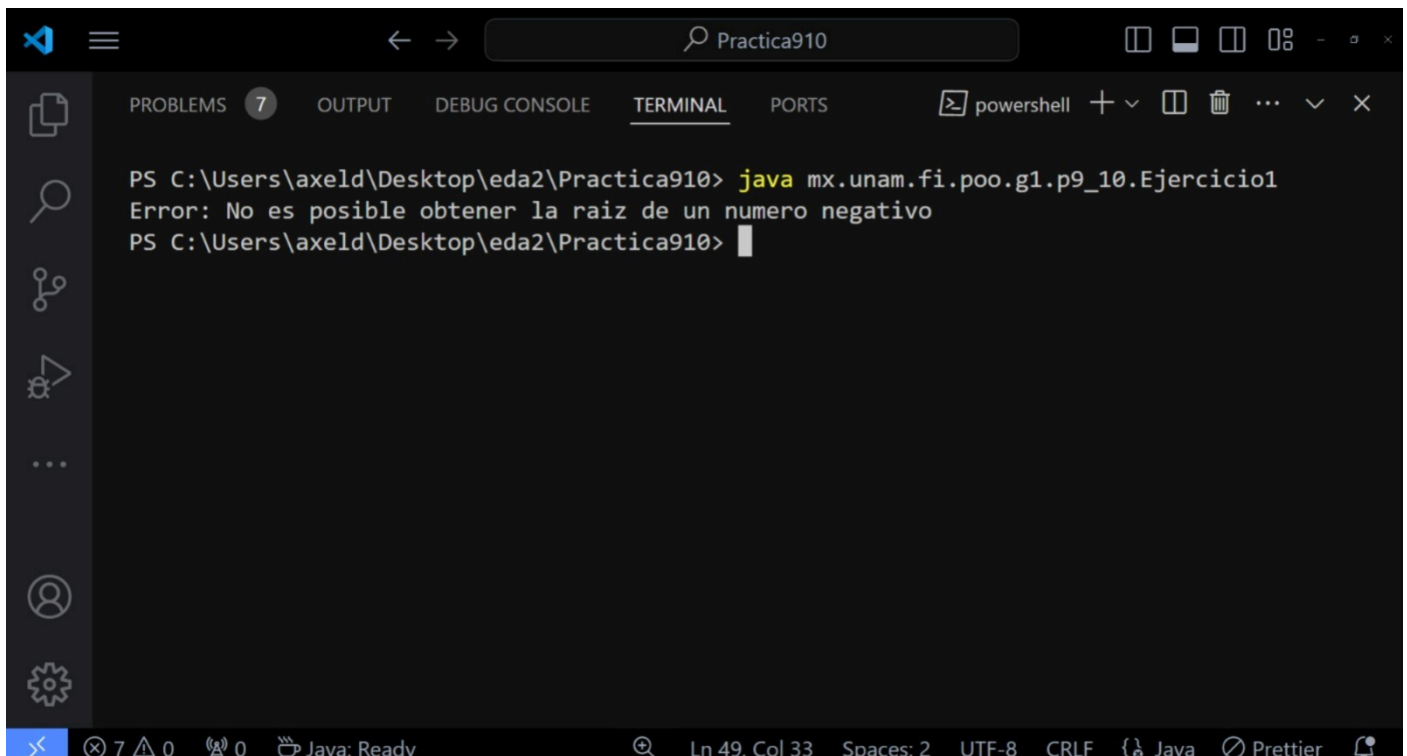
Para el ejercicio de tarea se ingresaron las siguientes cadenas: "Esto ES una PrúebÁ" y "jsjsjsjs", pudiendo distinguir claramente que la segunda no contiene vocales por lo que se espera que se ejecute y muestre en pantalla eso, que la primera contiene vocales y la segunda no.

Resultados:

```

PS C:\Users\axeld\Desktop\eda2\Practica910> java mx.unam.fi.poo.g1.p9_10.Ejercicio0
Error: No es posible dividir entre 0
PS C:\Users\axeld\Desktop\eda2\Practica910>
  
```

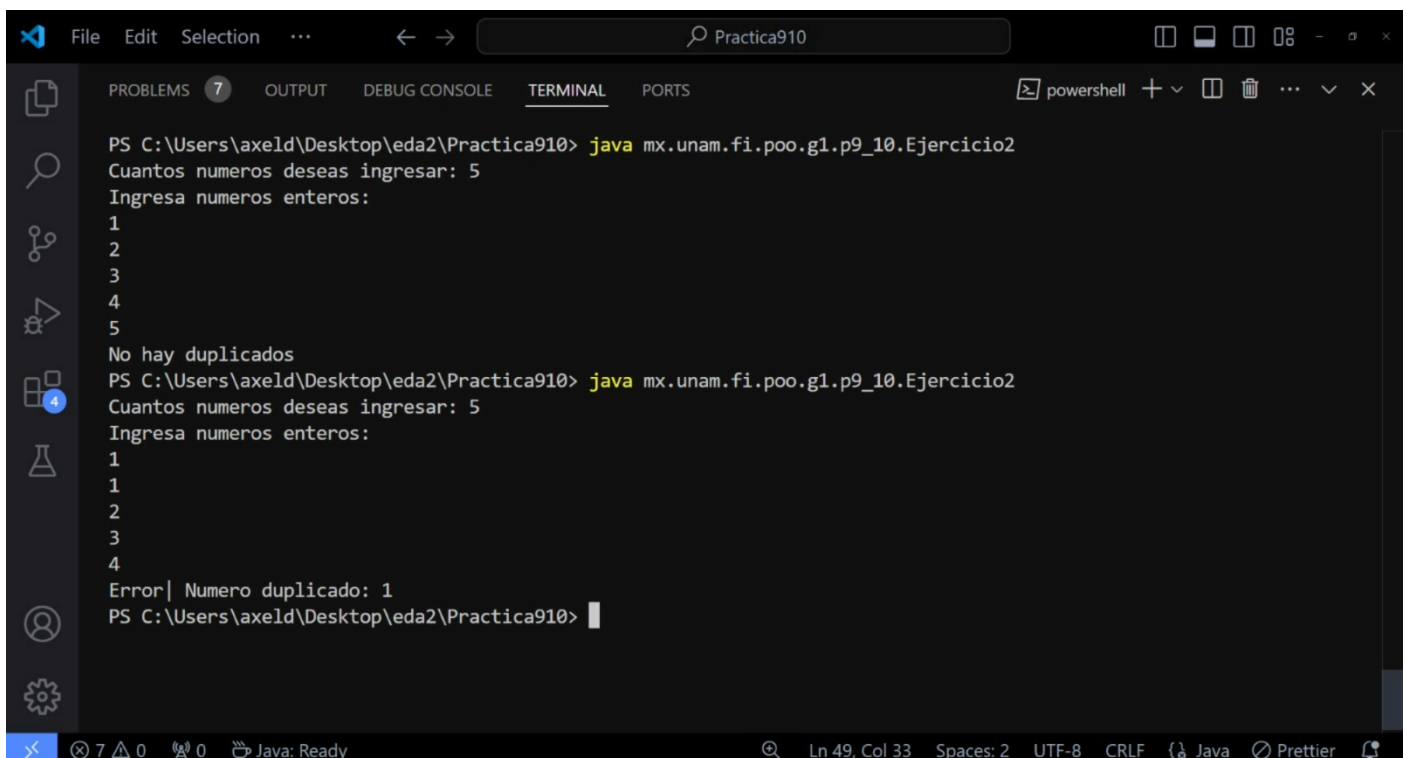
En la imagen anterior se puede observar la ejecución y lo esperado de la ejecución del ejercicio 1.



The screenshot shows a VS Code terminal window with the title bar 'Practica910'. The terminal is in PowerShell mode. The command executed is `java mx.unam.fi.poo.g1.p9_10.Ejercicio1`. The output is an error message: `Error: No es posible obtener la raiz de un numero negativo`. The status bar at the bottom indicates 'Ln 49, Col 33', 'Spaces: 2', 'UTF-8', 'CRLF', and 'Java: Ready'.

```
PS C:\Users\axeld\Desktop\eda2\Practica910> java mx.unam.fi.poo.g1.p9_10.Ejercicio1
Error: No es posible obtener la raiz de un numero negativo
PS C:\Users\axeld\Desktop\eda2\Practica910>
```

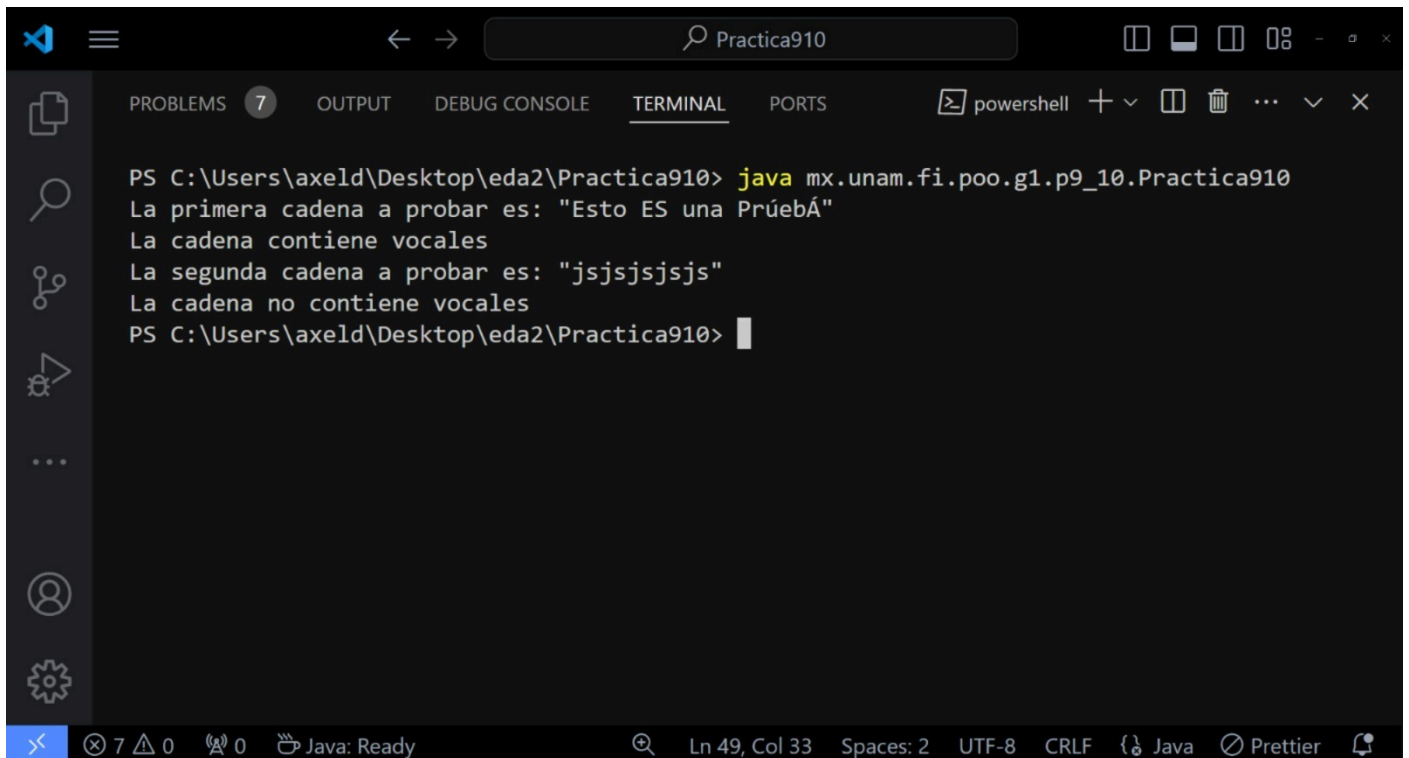
En la imagen anterior se puede observar como fue arrojada la excepción de la raíz cuadrada negativa creada previamente.



The screenshot shows a VS Code terminal window with the title bar 'Practica910'. The terminal is in PowerShell mode. The first command executed is `java mx.unam.fi.poo.g1.p9_10.Ejercicio2`. The output is: `Cuantos numeros deseas ingresar: 5`, `Ingresa numeros enteros:`, followed by the numbers 1, 2, 3, 4, 5, and `No hay duplicados`. The second command executed is `java mx.unam.fi.poo.g1.p9_10.Ejercicio2`. The output is: `Cuantos numeros deseas ingresar: 5`, `Ingresa numeros enteros:`, followed by the numbers 1, 1, 2, 3, 4, and `Error| Numero duplicado: 1`. The status bar at the bottom indicates 'Ln 49, Col 33', 'Spaces: 2', 'UTF-8', 'CRLF', and 'Java: Ready'.

```
PS C:\Users\axeld\Desktop\eda2\Practica910> java mx.unam.fi.poo.g1.p9_10.Ejercicio2
Cuantos numeros deseas ingresar: 5
Ingresa numeros enteros:
1
2
3
4
5
No hay duplicados
PS C:\Users\axeld\Desktop\eda2\Practica910> java mx.unam.fi.poo.g1.p9_10.Ejercicio2
Cuantos numeros deseas ingresar: 5
Ingresa numeros enteros:
1
1
2
3
4
Error| Numero duplicado: 1
PS C:\Users\axeld\Desktop\eda2\Practica910>
```

En la imagen presente se pueden observar las ejecuciones del código, la primera ejecución sin contener números duplicados y la segunda duplicando el primer y segundo elemento.



The screenshot shows a Visual Studio Code interface with a terminal window open. The terminal title is 'Practica910'. The command prompt is 'PS C:\Users\axeld\Desktop\eda2\Practica910>'. The command executed is 'java mx.unam.fi.poo.g1.p9_10.Practica910'. The output shows two test cases: the first string 'Esto ES una Prueba' contains vowels, and the second string 'jsjsjsjsjs' does not. The status bar at the bottom indicates 'Java: Ready' and 'Ln 49, Col 33'.

```
PS C:\Users\axeld\Desktop\eda2\Practica910> java mx.unam.fi.poo.g1.p9_10.Practica910
La primera cadena a probar es: "Esto ES una Prueba"
La cadena contiene vocales
La segunda cadena a probar es: "jsjsjsjsjs"
La cadena no contiene vocales
PS C:\Users\axeld\Desktop\eda2\Practica910>
```

Para el ejercicio de tarea se usaron las cadenas antes mencionadas y como se muestra en pantalla se obtuvo lo esperado.

Conclusiones:

En conclusión, la implementación de los conceptos teóricos relacionados con el manejo de excepciones en Java permitió resolver de manera efectiva los problemas planteados. Como se explicó en la introducción, se desarrollaron diversas excepciones personalizadas para gestionar errores específicos.

El uso de bloques try-catch para capturar y gestionar estos errores garantizó que el programa pudiera continuar su ejecución de manera controlada, sin interrupciones, mientras que los throws facilitaron la propagación de excepciones para que otras partes del programa fueran manejadas de manera adecuada.

La posibilidad de crear excepciones personalizadas refuerza la importancia de este concepto, ya que proporciona un control detallado sobre cómo y cuándo manejar los errores que surgen en un programa. Esto nos proporciona una idea de que el manejo adecuado de excepciones no solo contribuye a la estabilidad del código, sino que también mejora la flexibilidad y el mantenimiento del proyecto.

Referencias:

[1] Oracle, *Java Documentation: Exceptions*. [En línea]. Disponible:

<https://docs.oracle.com/javase/tutorial/essential/exceptions/>. [Accedido: Oct. 20, 2024].

[2] ChatGPT. [En línea]. Disponible: <https://chatgpt.com/>. [Accedido: Oct. 20, 2024]. Pregunta:

“Platícame más acerca de las excepciones y los throws en Java”