//**Deber11**

// **Nombre:** Alex Travez, Mateo Oviedo

// **Fecha de realizacion:**16/12/2024

// **Fecha de entrega:** 17/12/2024

//**Resultados:** El documento es archivo que contiene el proceso para el desarrollo del deber 09, el cual se trataba de una calculadora capaz de realizar 7 operaciones básicas, pero que ha sido tomado para convertirlo a su respectivo lenguaje UML

Paso 0: Comportamiento

Se empleará un sistema para la realización de 7 diferentes tipos de operaciones matemáticas a partir del ingreso de 2 operadores.

Paso 1: Definir Actores

1. Usuario
2. Sistema

Paso 2: Definir los casos de uso, examinando los actores para determinar sus necesidades

Casos de Uso	
Usuario	ingresar dos operandos
Sistema	validar que los operandos no sean mayores a 20,000.
Sistema	realizar operaciones matemáticas
Sistema	realizar manejo de diversas excepciones que pueden ocurrir durante la validación de operandos y las operaciones

Paso 3: para cada caso de uso genere un caso, que este conformado por:

Nombre único
Actores

Condiciones de entrada

Flujo de eventos

Condiciones de salida

Requerimientos especiales

- Caso de Uso: Ingresar dos operandos

Actores: Usuario

Condiciones de Entrada:

El usuario ingresa dos operandos enteros.

Flujo de Eventos:

El usuario introduce dos operandos.

El sistema recibe y almacena los operandos.

Condiciones de Salida: Los operandos son almacenados correctamente en el sistema.

Requerimientos Especiales: Ninguno.

- Caso de Uso: Validar operandos

Actores: Sistema

Condiciones de Entrada: Dos operandos han sido ingresados por el usuario.

Flujo de Eventos:

El sistema verifica que los operandos no sean mayores a 20,000.

Si alguno de los operandos es mayor a 20,000, se lanza una excepción.

Condiciones de Salida:

Los operandos están validados y no exceden el límite permitido (20,000).

Requerimientos Especiales:

El sistema debe manejar excepciones correctamente para valores fuera del rango permitido.

- Caso de Uso: Realizar operaciones matemáticas

Actores: Sistema

Condiciones de Entrada:

Los operandos han sido validados y no son mayores a 20,000.

Flujo de Eventos:

Después de validar los dos operandos, el sistema realiza automáticamente las siguientes operaciones:

Suma

Resta

Multiplicación

División

Módulo

Potenciación

Radicación

Condiciones de Salida:

El sistema muestra los resultados de todas las operaciones al usuario.

Requerimientos Especiales: Ninguno.

- Caso de Uso: Manejo de excepciones

Actores: Sistema

Condiciones de entrada:

Se ha producido un error durante la validación de operandos o la ejecución de una operación.

Flujo de eventos:

El sistema captura la excepción.

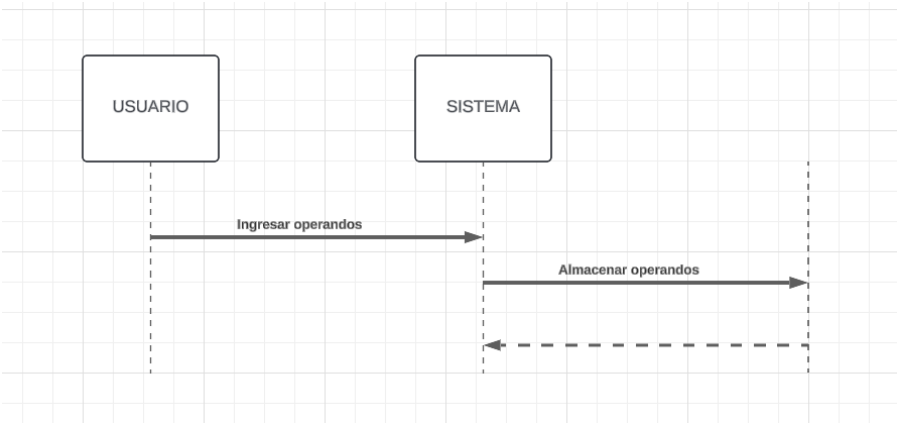
El sistema muestra un mensaje de error al usuario indicando la naturaleza del problema.

Condiciones de salida: ninguna

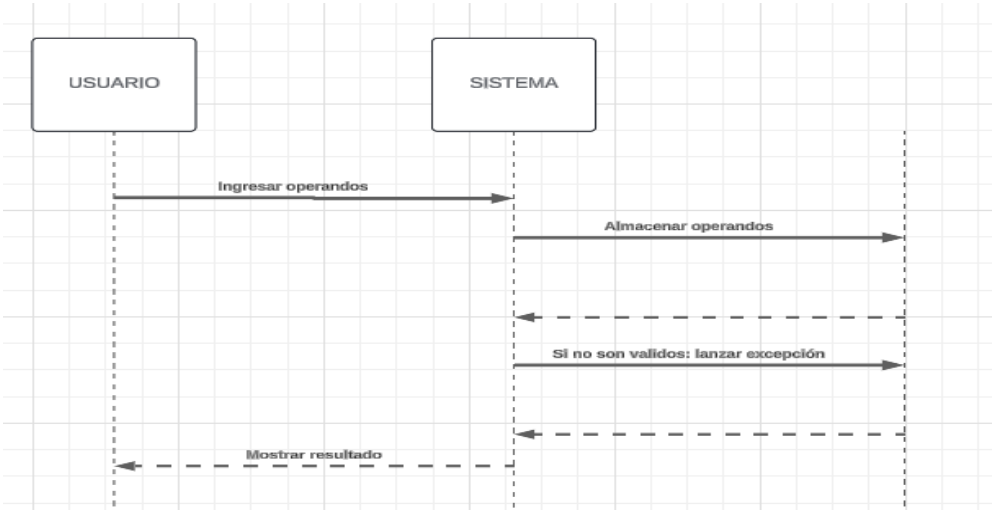
Requerimientos especiales: ninguna

Paso 4: Diagrama de Secuencia por cada caso de uso

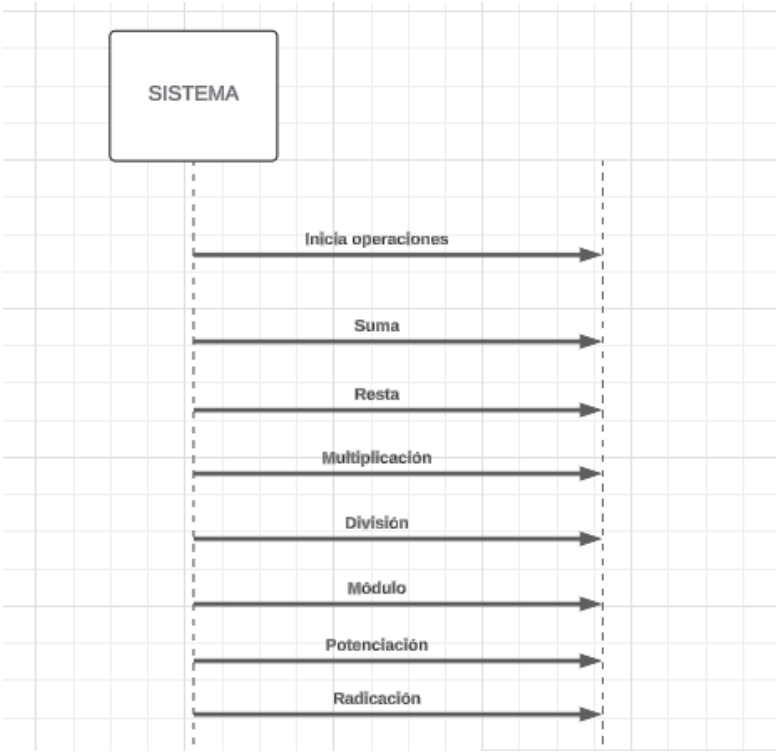
- Caso de Uso: Ingresar dos operandos



- Caso de Uso: Validar operandos



- Caso de Uso: Realizar operaciones matemáticas



- Caso de Uso: Manejo de excepciones

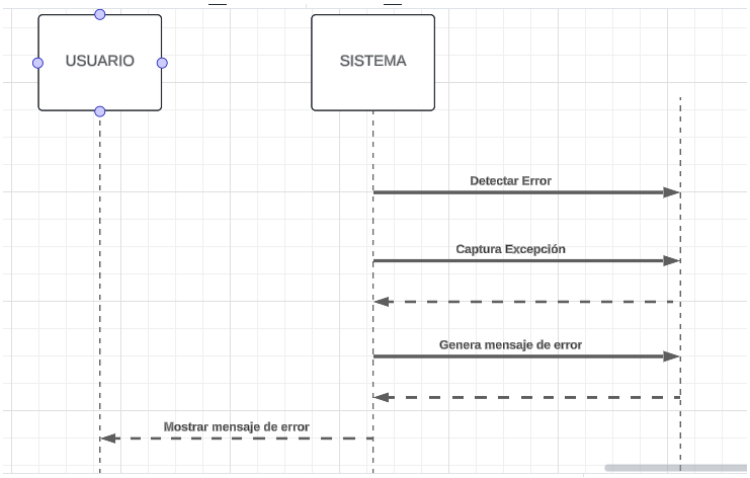
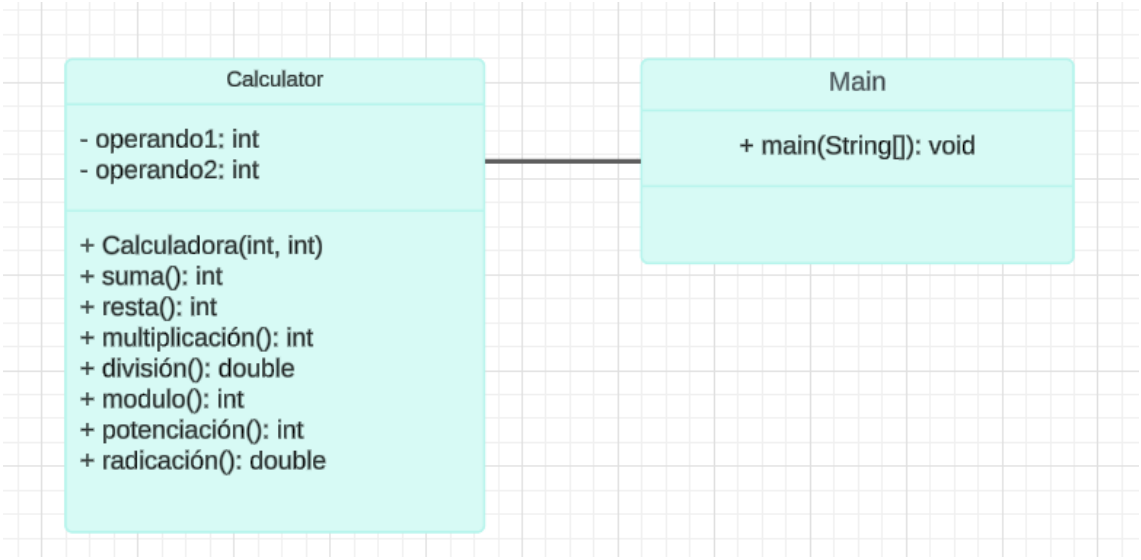


Diagrama de Clases:



//Conclusiones:

- La representación en UML permite estructurar y organizar un proyecto, lo que facilita la comprensión de su funcionamiento y requisitos. En este caso, el diagrama de clases y los casos de uso simplifican la planificación de las operaciones matemáticas y los flujos de validación.
- Incorporar el manejo de excepciones desde las primeras etapas del desarrollo garantiza que el sistema pueda manejar errores de manera eficiente, mejorando la experiencia del usuario y la robustez del programa.
- El desglose de funcionalidades en casos de uso individuales ayuda a crear un sistema modular. Esto no solo mejora la legibilidad del código, sino que también facilita futuras actualizaciones o expansiones, como añadir más operaciones matemáticas.

//Recomendaciones:

- Diseñar pruebas específicas para validar cada caso de uso (por ejemplo, límites de operandos y manejo de excepciones) antes de la implementación final y ejecución del código.
- En lugar de verificar manualmente si los operandos superan el límite de 20,000, se recomienda implementar una función genérica que centralice esta validación. Esto facilita la reutilización del código y reduce el margen de error al asegurar consistencia en todas las operaciones.
- Aunque los casos de uso están bien definidos, se recomienda incluir descripciones más específicas sobre los posibles errores y excepciones. Por ejemplo, detallar qué mensajes de error se mostrarán al usuario en cada escenario que permitirá que los desarrolladores comprendan mejor las interacciones del sistema.

//Bibliografía:

[1] Aula Virtual - EPN: Análisis, Diseño y Lenguaje UML. (s/f). Edu.ec. Recuperado el 16 de diciembre de 2024, de https://aulasvirtuales.epn.edu.ec/course/view.php?id=66230&section=11

[2] Introducción a UML - Diagramas y Modelado. (s/f). Visual Paradigm. Recuperado el 16 de diciembre de 2024, de

https://www.visual-paradigm.com/es/guide/uml-unified-modeling-language/