Desafío Sesión 2
Mini calculadora de enteros

Reto 1

- **¿Qué solución presenta el código?**
    - Una calculadora simple donde se ingresan dos números y pueden realizarse operaciones aritméticas de suma, resta, multiplicación y división. Luego de realizada la operación se muestra en pantalla el resultado.

- **¿Cómo comprobaron que no había errores de sintaxis?**
    - Además de evaluar estáticamente el código, se procedió a limpiar la solución y compilarla. Al compilar, lo realizo sin errores y luego se procedió a ejecutarla, acción que también se realizó sin problemas. De dicha manera se evidenció que no existen problemas de sintaxis en el programa.

- **¿Cómo comprobaron que no había errores semánticos?**
    - También se evaluó estáticamente el código y se procedió a realizar pruebas básicas del funcionamiento de una calculadora para evidenciar que el resultado obtenido por las operaciones es el correcto. Como errores de semántica se encontró que los saltos de línea son incorrectos. Hay que incluir validaciones defensivas y preventivas.

Reto 2

- **¿Qué hace la prueba unitaria?**
    - Evalúa la funcionalidad de una instrucción atómica, es decir de una función o un método. Para que las pruebas unitarias sean efectivas, la codificación debe de seguir estándares y buenas prácticas.

- **¿Cuántas pruebas realizaron para demostrar que funciona bien?**
    - Se debería de realizar pruebas por método realizable con el fin de tener mejor cobertura. Si el código se hubiese separado en método por operación aritmética más método por funcionalidad auxiliar como el menú y la toma de datos, deberían de haberse realzado al menos 6 pruebas unitarias de happy path. Para las pruebas unitarias de las operaciones aritméticas se debería de evaluar que dado dos números enteros se realice con éxito la operación esperada.

Reto 3

- **¿Para qué sirven las excepciones? ¿Cuándo se ejecuta finally?**
  - Las excepciones indican que ha ocurrido algún error en la ejecución, es una manera de controlar y manejar errores, así como evitar que el programa termine abruptamente. Finally es una instrucción o bloque de código que siempre se ejecutará al final del programa sin importar que hayan existido errores.

- **¿Qué estrategias defensivas usaron y por qué?**
  - Se agregaron estructuras selectivas para agregar validaciones en el ingreso de valores para mitigar/evitar errores por datos incorrectos.

- **¿Cómo convertirían el código en clases para poderlo reutilizar?**
  - Separaría primero en funciones y métodos los cuales serian incluidos en una clase. Los separaría en métodos operativos que ayudarán a la funcionalidad lógica del programa y en métodos auxiliares que apoyarán el manejo de valores entre clases. Aplicaría las características del paradigma POO.

Reto 4

- **¿Qué es un caso de pruebas?**
  - Un caso de prueba es una guía para el usuario que realice las pruebas para poder contar con entradas definidas y salidas esperadas para el proceso a validar y verificar. Pueden existir casos de prueba técnicos y casos de prueba de negocio.

- **¿Lo usuarios validaron o verificaron? ¿Por qué?**
  - Los usuarios finales validan las funcionalidades con base a casos de uso de negocio, ya que ellos estan orientados a ver si el sistema desarrollado cumple con lo solicitado y soluciona el problema que fue planteado.

- **¿Qué hubiera pasado si la validación no acepta el sistema? Dé sus recomendaciones**
  - Si la validación realizada por los usuarios los lleva a no aceptar el sistema, se debe recibir la retroalimentación para tener una mejor visión de lo que se necesitaba y asi poder trazar un plan para realizar las correcciones necesarias y así poder solucionar el problema y cumplir con lo esperado.