ESPE UNIVERSIDAD DE LAS FUERZAS ARMADAS INNOVACIÓN PARA LA EXCELENCIA	INFORME DE PRÁCTICA DE LABORATORIO

DEPARTAMENTO:		**CARRERA:**	**ITIN**		
ASIGNATURA:	POO	**NIVEL:**	Segundo	**FECHA:**	08/05/2025
DOCENTE:	ING. Jhon Cruz	**PRÁCTICA N°:**		**CALIFICACIÓN:**	

Título o tema de la práctica con una extensión máxima de 20 palabras. Tamaño de letra (TL) 14

Angel Steven Rodriguez Chavez

RESUMEN

Se desarrolló una calculadora en Java aplicando los principios de Programación Orientada a Objetos (POO), incluyendo encapsulamiento, modularidad y validación de entradas. El programa permite realizar operaciones básicas (suma, resta, multiplicación, división) y avanzadas (potencia, raíz cuadrada) mediante un menú interactivo. Se implementaron clases separadas para la lógica de cálculo (Calculadora), validación de datos (Validaciones) y la interfaz de usuario (Menu), demostrando una estructura limpia y mantenible.

Palabras Claves: POO, encapsulamiento, modularidad.

1. **INTRODUCCIÓN:**
   Esta práctica tuvo como objetivo aplicar los conceptos fundamentales de POO en Java, como la encapsulación de atributos, la división de responsabilidades en clases y el manejo de excepciones. La calculadora resultante es un ejemplo claro de cómo diseñar software robusto y escalable.
2. **OBJETIVO(S):**
   2.1 Implementar una calculadora con operaciones básicas y avanzadas.
   2.2 Utilizar clases separadas para lógica, validación y interfaz.
   2.3 Validar entradas numéricas para evitar errores en tiempo de ejecución.
3. **MARCO TEÓRICO:**
   - Encapsulamiento: Los atributos num1 y num2 en Calculadora son privados y se acceden mediante getters/setters.
   - Manejo de excepciones: La clase Validaciones asegura que las entradas sean numéricas.
   - Modularidad: Cada operación está encapsulada en métodos independientes.
4. **DESCRIPCIÓN DEL PROCEDIMIENTO:**
   Materiales y herramientas:
   - NetBeans IDE.
   - JDK 8+.

   Clases implementadas:
   - Main: Punto de entrada. Llama al menú principal.
   - Menu: Gestiona la interfaz de usuario y redirige a las operaciones.
   - Calculadora: Contiene la lógica de las operaciones matemáticas.
   - Validaciones: Valida que las entradas sean números válidos.

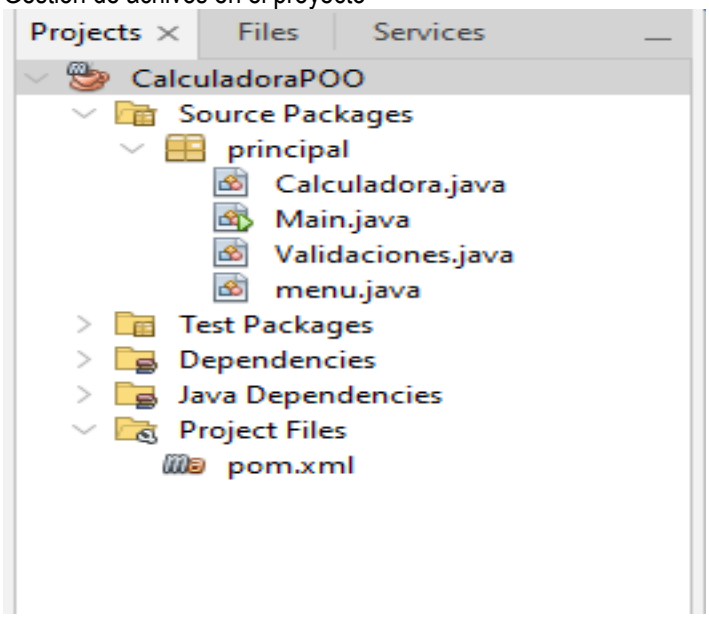
   Flujo del programa:
   - El usuario selecciona una operación en el menú.
   - Se ingresan los números (validados por Validaciones).
   - Calculadora procesa la operación y devuelve el resultado.
5. **ANÁLISIS DE RESULTADOS:**
   Errores manejados:
   - División por cero → Retorna Double.NaN.
   - Raíz de número negativo → Muestra mensaje de error.

- Entradas no numéricas ⟶ Solicita nuevo ingreso.

Resultados exitosos: Las operaciones válidas muestran el resultado con precisión.

6. GRÁFICOS O FOTOGRAFÍAS:

Gestion de achivos en el proyecto



Clase Main:

```java
package principal;

public class Main {

    public static void main(String[] args) {
        menu yeah=new menu();
        yeah.iniciar();
    }
}
```

Clase Calculadora (se encuentra encapsulamiento y la lógica de métodos para usarlos en el proyecto actúa):

```java
package principal;

public class Calculadora {
    Validaciones entrada= new Validaciones();
  private double num1,num2;

    public Calculadora(double num1, double num2) {
        this.num1 = num1;
        this.num2 = num2;
    }

    public double getNum1() {
        return num1;
    }

    public double getNum2() {
        return num2;
    }

    public void setNum1(String imput) {
        this.num1 = entrada.entradaDatos(imput);
    }

    public void setNum2(String imput) {
        this.num2 = entrada.entradaDatos(imput);
    }
     public double suma( double num1, double num2){
    return num1+num2;
    }
     public double resta( double num1, double num2){
    return num1-num2;
    }
     public double multiplicar( double num1, double num2){
    return num1*num2;
    }
    public double dividir( double num1, double num2){
        if(num2== 0){
            System.out.println("error: division para cero. \n digitie de nuevo");
            return Double.NaN;
        }
    return num1/num2;
    }
public double potencia( double num1, double num2){

    return Math.pow(num1, num2);}

       public double raiz( double num1){
           if(num1<0){
               System.out.println("erro: raiz de numero negativo \n digite de nuevo");
               return 0;
           }
    return Math.sqrt(num2);
    }
}
```

Clase de validaciones(sirve para entrada de datos numéricos de tipo Doube):

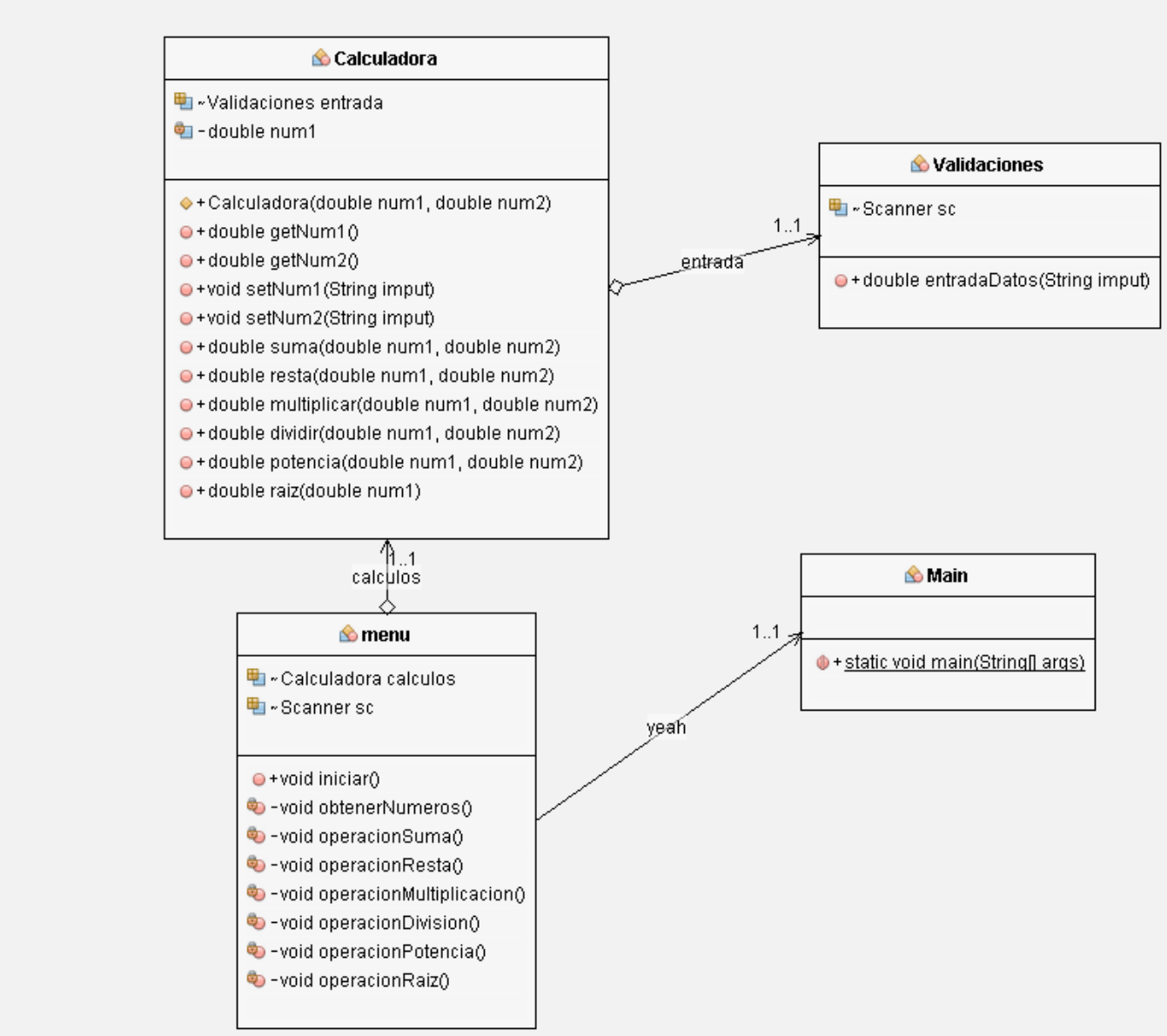
```
1   package principal;
2   import java.util.*;
3   public class Validaciones {
4       Scanner sc=new Scanner(System.in);
5     public double entradaDatos(String imput){
6     while(true){
7     try{
8         return Double.parseDouble(imput);
9     }catch(NumberFormatException e){
10        System.out.println("Entrada valida");
11        imput =sc.nextLine();
12    }
13    }
14    }
15  }
```

Clase Menu (donde generalmente se llamara por medio de objetos en métodos propios de la clase, esto es esencial para dejar el proyecto modulado y listo para mantenimiento si es que se expande el código)

```java
package principal;
import java.util.*;
public class menu {
Calculadora calculos= new Calculadora(0,0);
Scanner sc= new Scanner(System.in);
public void iniciar(){
    char opc;
    System.out.println("==Calculdora==");
do{    System.out.println("1.suma");
System.out.println("2.resta");
System.out.println("3.multiplicacion");
System.out.println("4.division");
System.out.println("5.potencia");
System.out.println("6.raiz");
System.out.println("7.salir");
opc=sc.next().charAt(0);
switch(opc){
    case '1':
        operacionSuma();
        break;
    case '2':
        operacionResta();
        break;
            case '3':
                operacionMultiplicacion();
        break;
    case '4':
        operacionDivision();
        break;
            case '5':
                operacionPotencia();
        break;
    case '6':
        operacionRaiz();
        break;
            case '7':
                System.out.println("BYE BYE");
        break;
            default: System.out.println("Opcion no valida");
}
```

LINK del github: https://github.com/angeldev7/ActividadCalculadoraClase

Diagrama de clases:



7. DISCUSIÓN:

Presentar el análisis de los resultados obtenidos, a través de la comparación entre los conceptos y teorías aprendidas.

8. CONCLUSIONES:

- Se logró una calculadora funcional aplicando POO.
- La separación de responsabilidades mejoró la mantenibilidad.
- Los principios de encapsulamiento y validación aseguraron robustez.

9. BIBLIOGRAFÍA:

- **Deitel, P. y Deitel, H. 2017. Java How to Program. Pearson.**