


	UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA	
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
Aprobación: 2022/03/01	Código: GUIA-PRLE-001	Página: 1

INFORME DE LABORATORIO

(formato estudiante)

INFORMACIÓN BÁSICA					
ASIGNATURA:	PROGRAMACIÓN WEB 2				
TÍTULO DE LA PRÁCTICA:	Git y GitHub				
NÚMERO DE PRÁCTICA:	1	AÑO LECTIVO:	2024	NRO. SEMESTRE:	II
FECHA DE PRESENTACIÓN	21/09/2024	HORA DE PRESENTACIÓN	10/mm/ss		
INTEGRANTE (s) Marron Puma, Maritza Claudia Paredes Saico, Jordán Salas Idme, Nikole Valery Pacheco Esquinarila, Milene				NOTA (0-20)	
DOCENTE(s): Lino José Pinto Oppe					

RESULTADOS Y PRUEBAS
I. EJERCICIOS RESUELTOS: <div style="background-color: #2c3e50; color: white; padding: 10px; margin-top: 10px;"> <pre>PWII-2024B/ ├── src/ │ ├── Calculadora.java │ ├── Metodos/ │ │ ├── Suma.java │ │ ├── Resta.java │ │ ├── Multiplicacion.java │ │ ├── Division.java │ │ └── Modulo.java ├── README.md └── informe.md</pre> </div> <div style="margin-top: 10px;"> Método Suma <div style="background-color: #34495e; color: white; padding: 5px; margin-top: 5px;"> package Metodos; </div> </div>

	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 2</p>

```

@NikoleSalas →/workspaces/PWII-2024B/src/Metodos (Grupo-Miau) $ git add Suma.java
@NikoleSalas →/workspaces/PWII-2024B/src/Metodos (Grupo-Miau) $ git commit -m "Add Suma class and sumar method for adding two numbers"
[Grupo-Miau 6f695e9] Add Suma class and sumar method for adding two numbers
1 file changed, 14 insertions(+)
@NikoleSalas →/workspaces/PWII-2024B/src/Metodos (Grupo-Miau) $ git push
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 2 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 614 bytes | 614.00 KiB/s, done.
Total 5 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/LINOPINTO2023/PWII-2024B
cb24265..6f695e9 Grupo-Miau -> Grupo-Miau
@NikoleSalas →/workspaces/PWII-2024B/src/Metodos (Grupo-Miau) $

```



Metodo Resta

```

package Metodos;

/**
 * Método para restar dos números.
 *
 * @param num1 El primer número.
 * @param num2 El segundo número.
 * @return El resultado de restar ambos números. Devuelve double.
 */
public class Resta {
    public static double restar(double num1, double num2) {
        return num1 - num2;
    }
}

```

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 3</p>

```

• @NikoleSalas →/workspaces/PWII-2024B/src/Metodos (Grupo-Miau) $ git add Resta.java
• @NikoleSalas →/workspaces/PWII-2024B/src/Metodos (Grupo-Miau) $ git commit -m "Add Resta class and restar method for subtracting two numbers"
[Grupo-Miau 84f4ffc] Add Resta class and restar method for subtracting two numbers
1 file changed, 14 insertions(+)
• @NikoleSalas →/workspaces/PWII-2024B/src/Metodos (Grupo-Miau) $ git push
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 2 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 622 bytes | 622.00 KiB/s, done.
Total 5 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/LINOPINTO2023/PWII-2024B
6f695e9..84f4ffc Grupo-Miau -> Grupo-Miau
• @NikoleSalas →/workspaces/PWII-2024B/src/Metodos (Grupo-Miau) $

```

Método Multiplicación

```

package Metodos;

/**
 * Método para multiplicar dos números.
 *
 * @param num1 El primer número.
 * @param num2 El segundo número.
 * @return El resultado de multiplicar ambos números. Devuelve double.
 */
public class Multiplicacion {
    public static double multiplicar(double num1, double num2) {
        return num1 * num2;
    }
}

```

```

• @NikoleSalas →/workspaces/PWII-2024B/src/Metodos (Grupo-Miau) $ git add Multiplicacion.java
• @NikoleSalas →/workspaces/PWII-2024B/src/Metodos (Grupo-Miau) $ git commit -m "Add Multiplicacion class and multiplicar method for multiplying two numbers"
[Grupo-Miau 1827377] Add Multiplicacion class and multiplicar method for multiplying two numbers
1 file changed, 14 insertions(+)
• @NikoleSalas →/workspaces/PWII-2024B/src/Metodos (Grupo-Miau) $ git push
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 2 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 634 bytes | 634.00 KiB/s, done.
Total 5 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/LINOPINTO2023/PWII-2024B
84f4ffc..1827377 Grupo-Miau -> Grupo-Miau
• @NikoleSalas →/workspaces/PWII-2024B/src/Metodos (Grupo-Miau) $

```



Método Division

Metodos > Division.java > ...

```

1 package Metodos;
2
3 /**

```

	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 4</p>

Método Modulo

```

Metodos > Modulo.java > ...
1  package Metodos;
2
3  /**
4   * Método para calcular el módulo de dos números.
5   *
6   * @param num1 El primer número.
7   * @param num2 El segundo número.
8   * @return El resultado del módulo de num1 entre num2. Devuelve double.
9   */
10 public class Modulo {
11     public static double modulo(double num1, double num2) {
12         return num1 % num2;
13     }
14 }

```

II. PRUEBAS

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
Aprobación: 2022/03/01	Código: GUIA-PRLE-001	Página: 5

¿Con que valores comprobaste que tu práctica estuviera correcta? ¿Qué resultado esperabas obtener para cada valor de entrada? ¿Qué valor o comportamiento obtuviste para cada valor de entrada?

```
@NikoleSalas →/workspaces/PWII-2024B (Grupo-Miau) $
s/PWII-2024B_afe316be/bin Calculadora
Calculadora de Operaciones Básicas
=====
Ingrese el primer operando: 5
Ingrese el segundo operando: b
Error: Debes ingresar un número válido.
Ingrese el segundo operando: █
```

```
¿Desea realizar otra operación? (s/n): s
Ingrese el primer operando: 2
Ingrese el segundo operando: 5.2
¿Qué operación desea hacer? (Solo coloque un número)
1: Sumar
2: Restar
3: Multiplicar
4: Dividir
5: Módulo
2
Resultado: -3.2
¿Desea realizar otra operación? (s/n): █
```

```
¿Desea realizar otra operación? (s/n): s
Ingrese el primer operando: 5
Ingrese el segundo operando: 5
¿Qué operación desea hacer? (Solo coloque un número)
1: Sumar
2: Restar
3: Multiplicar
4: Dividir
5: Módulo
3
Resultado: 25
¿Desea realizar otra operación? (s/n): █
```

```
¿Desea realizar otra operación? (s/n): s
Ingrese el primer operando: 5.2
Ingrese el segundo operando: 5
¿Qué operación desea hacer? (Solo coloque un número)
1: Sumar
2: Restar
3: Multiplicar
4: Dividir
5: Módulo
3
Resultado: 26
¿Desea realizar otra operación? (s/n): █
```

```
¿Desea realizar otra operación? (s/n): s
Ingrese el primer operando: 5.2
Ingrese el segundo operando: 5.2
¿Qué operación desea hacer? (Solo coloque un número)
1: Sumar
2: Restar
3: Multiplicar
4: Dividir
5: Módulo
3
Resultado: 27.040000000000003
¿Desea realizar otra operación? (s/n): █
```

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
Aprobación: 2022/03/01	Código: GUIA-PRLE-001	Página: 6

<pre> Calculadora de Operaciones Básicas ===== Ingrese el primer operando: 156 Ingrese el segundo operando: 25 ¿Qué operación desea hacer? (Solo coloque un número) 1: Sumar 2: Restar 3: Multiplicar 4: Dividir 5: Módulo 4 Resultado: 6.24 ¿Desea realizar otra operación? (s/n): █ </pre>	<pre> ¿Desea realizar otra operación? (s/n): s Ingrese el primer operando: 1458 Ingrese el segundo operando: 36 ¿Qué operación desea hacer? (Solo coloque un número) 1: Sumar 2: Restar 3: Multiplicar 4: Dividir 5: Módulo 5 Resultado: 18 ¿Desea realizar otra operación? (s/n): █ </pre>
--	---

<h3>III. CUESTIONARIO</h3>

CONCLUSIONES
<p>Como grupo, seguimos una metodología simple y ordenada para desarrollar el código de la calculadora, asegurándonos de cumplir con los requisitos del problema.</p> <p>La modularización nos ayudó a dividir el código en partes más manejables y fáciles de entender. Además, el uso de Git y GitHub nos permitió trabajar de forma colaborativa, realizar cambios de manera organizada y mantener un control de versiones eficiente. Gracias a esto, pudimos asegurar que nuestro código sea limpio, reutilizable y fácil de modificar en el futuro.</p> <p>Las validaciones y el manejo de errores aseguran que el sistema sea robusto y fácil de usar. Este laboratorio nos dejó una comprensión más profunda sobre cómo organizar proyectos de programación de manera modular y colaborativa, preparándonos mejor para proyectos más complejos en el futuro.</p>

METODOLOGÍA DE TRABAJO
<p><i>Colocar la metodología de trabajo que ha utilizado el estudiante o el grupo para resolver la práctica, es decir el <u>procedimiento/secuencia de pasos en forma general</u>.</i></p> <p>Metodología de Trabajo Utilizada para Resolver la Práctica</p>

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 7</p>

A continuación, se detalla la metodología general que ha seguido nuestro grupo “Miau” para desarrollar y resolver el Laboratorio 01:

1. Análisis del Problema

- **Objetivo:** Queremos implementar una calculadora que realice operaciones matemáticas básicas (suma, resta, multiplicación, división, módulo).
- **Requerimientos Funcionales:** El sistema debería recibir dos operandos e identificar la operación que el usuario deseará realizar.
- **Requerimientos No Funcionales:** Se debía manejar correctamente entradas no válidas y garantizar la continuidad de la ejecución a través de múltiples cálculos y casos.

2. Diseño de la Solución

- **Estructura de la Aplicación:** Se decidió usar una clase Calculadora que encapsularía la funcionalidad principal del programa.
- **Modularización:** Se creó una arquitectura modular en la que cada operación matemática (suma, resta, multiplicación, división y módulo) se delega a clases separadas, a través de métodos estáticos.
- **Control de Flujo:** Se planificó el uso de un ciclo **while** que permitiría al usuario realizar múltiples operaciones hasta que decidiera terminar el programa.

```
boolean continuar = true; // Variable para controlar el bucle

while (continuar) {
    // Lógica de la calculadora
    ...
    // Preguntar si desea realizar otra operación
    System.out.print("¿Desea realizar otra operación? (s/n): ");
    char respuesta = sc.next().charAt(0); // Leer la respuesta del usuario
    continuar = (respuesta == 's' || respuesta == 'S'); // Continuar si la respuesta es 's' o 'S'
}

System.out.println("Gracias por usar nuestra calculadora. ¡Hasta luego!");
```

3. Implementación

- **Entrada de Datos:** Se empleó la clase **Scanner** para manejar la interacción con el usuario. Se implementó estructuras de control para validar los datos ingresados.
 - El programa valida que los números ingresados sean válidos usando el manejo de excepciones (**InputMismatchException**), solicitando nuevamente la entrada cuando se detecta un error.

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 8</p>

```

Scanner sc = new Scanner(System.in);
double num1 = 0, num2 = 0;
boolean inputValido = false;

// Validar el primer operando
while (!inputValido) {
    try {
        System.out.print("Ingrese el primer operando: ");
        num1 = sc.nextDouble();
        inputValido = true; // Entrada válida
    } catch (InputMismatchException e) {
        System.out.println("Error: Debes ingresar un número válido.");
        sc.next(); // Limpiar la entrada incorrecta
    }
}

// Validar el segundo operando
inputValido = false; // Reiniciar validación para el segundo operando
while (!inputValido) {
    try {
        System.out.print("Ingrese el segundo operando: ");
        num2 = sc.nextDouble();
        inputValido = true; // Entrada válida
    } catch (InputMismatchException e) {
        System.out.println("Error: Debes ingresar un número válido.");
        sc.next(); // Limpiar la entrada incorrecta
    }
}

```

- **Menú de Operaciones:** Se creó un menú que ofrece las opciones de operación disponibles (suma, resta, multiplicación, división y módulo). El usuario selecciona la operación introduciendo un número del 1 al 5.

```

System.out.println("¿Qué operación desea hacer? (Solo coloque un número) \n" +
    "1: Sumar\n" +

```


	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 9</p>

```

"2: Restar\n" +
"3: Multiplicar\n" +
"4: Dividir\n" +
"5: Módulo");
int op = sc.nextInt(); // Captura la opción del usuario

```

- **Control de Operaciones:** El código utiliza una estructura switch para seleccionar y ejecutar la operación correspondiente:
 - Cada operación se ejecuta mediante métodos especializados de las clases importadas (Suma, Resta, Multiplicación, etc.).
 - Se implementa manejo de errores específicos, como la verificación de divisiones y módulos por cero, presentando mensajes claros al usuario si se introduce un divisor igual a cero.

```

double resultado = 0;
switch (op) {
case 1:
    resultado = Suma.sumar(num1, num2);
    break;
case 2:
    resultado = Resta.restar(num1, num2);
    break;
case 3:
    resultado = Multiplicacion.multiplicar(num1, num2);
    break;
case 4:
    if (num2 != 0) {
        // resultado = Division.dividir(num1, num2);
    } else {
        System.out.println("Error: No se puede dividir entre cero.");
        continue;
    }
    break;
case 5:
    if (num2 != 0) {
        // resultado = Modulo.modular(num1, num2);
    } else {
        System.out.println("Error: No se puede calcular el módulo con un divisor de cero.");
        continue;
    }
}

```

	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 10</p>

```

    }
    break;
default:
    System.out.println("¡Seleccione un número entre 1 y 5!");
}

```

- **Salida de Resultados:**

- Se agregó una validación que identifica si el resultado es entero o decimal, para presentar el resultado de forma apropiada.

```

boolean esEntero = resultado % 1 == 0; // Verificar si es un entero

if (esEntero) {
    System.out.println("Resultado: " + (int) resultado); // Mostrar como entero
} else {
    System.out.println("Resultado: " + resultado); // Mostrar como decimal
}

```

- **Control de Continuidad:** Al final de cada operación, se pregunta al usuario si desea realizar otra operación, lo que permite mantener un flujo continuo de cálculos hasta que el usuario decida detenerse.

```

System.out.print("¿Desea realizar otra operación? (s/n): ");
char respuesta = sc.next().charAt(0); // Leer la respuesta del usuario
continuar = (respuesta == 's' || respuesta == 'S'); // Continuar si la respuesta es 's' o 'S'

```

4. Pruebas y Validación

	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 11</p>

- **Pruebas de Funcionalidad:** Se realizaron pruebas para garantizar que todas las operaciones (suma, resta, multiplicación, división y módulo) funcionen correctamente.
- **Validación de Entradas:** Se verificó que el sistema maneje correctamente entradas no válidas, como la introducción de caracteres en lugar de números, y que los mensajes de error sean adecuados.
- **Pruebas de Flujo:** Se probó el ciclo de continuidad para asegurar que el programa repita las operaciones correctamente hasta que el usuario desee finalizar.

5. Documentación

- Se añadieron comentarios en el código. Los comentarios ayudan a identificar el flujo del programa y explican el propósito de cada bloque de código, lo que facilita el mantenimiento de proyectos de grandes dimensiones.

REFERENCIAS Y BIBLIOGRAFÍA

1. B. Eckel, *Thinking in Java*, 4th ed. MindView, Inc., 2006.