

Informe de Laboratorio 02

Tema: GitHub

Nota

Estudiante	Escuela	Asignatura
Hernan Choquehuanca & Eduardo Portugal & Victor Gonzalo & Juan Salas hchoquehuanca@unsa.edu.pe & eporgutalpor@unsa.edu.pe & vmaldonadov@unsa.edu.pe & jsalasag@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Programación Web 2 Semestre: III Código: 1702122

Tarea	Tema	Duración
02	GitHub	2 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2024 - A	Del 28 Abril 2024	Al 07 Mayo 2024

1. Tarea

- Crear su cuenta en github y subir un informe de lo aprendido.
- Deberá enviar capturas de pantalla (jpg, png) de:
 - Su cuenta creada en GitHub.
 - El resultado de ejecutar en línea de comandos: git --version

2. Entregables

- Informe de trabajo
- URL: Repositorio de GitHub
- URL: Video explicativo

3. Equipos, materiales y temas utilizados

- GitHub

4. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- <https://github.com/Eduardo-P/pw2-lab02>

5. URL de video explicativo

- URL del video explicativo en YouTube.
- https://youtu.be/3aKI_PDTDsk

6. Desarrollo del trabajo

6.1. Cuentas en GitHub

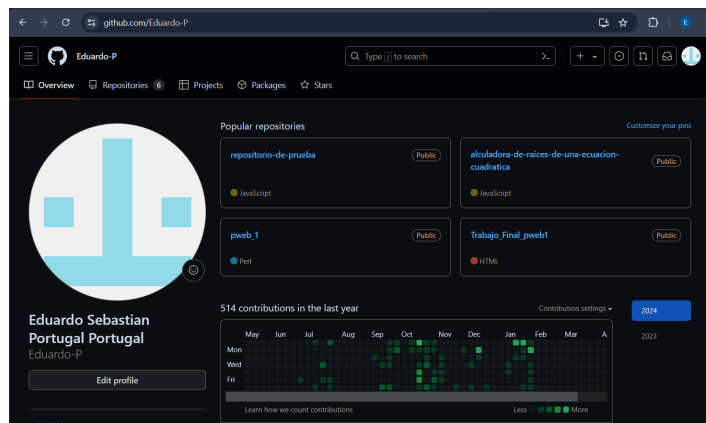


Figura 1: Usuario de Eduardo Portugal

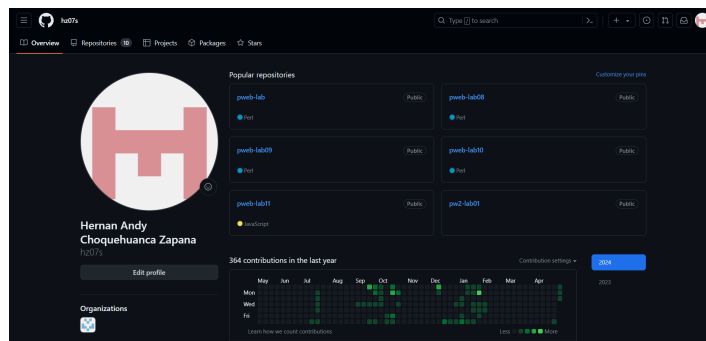


Figura 2: Usuario de Hernan Choquehuanca

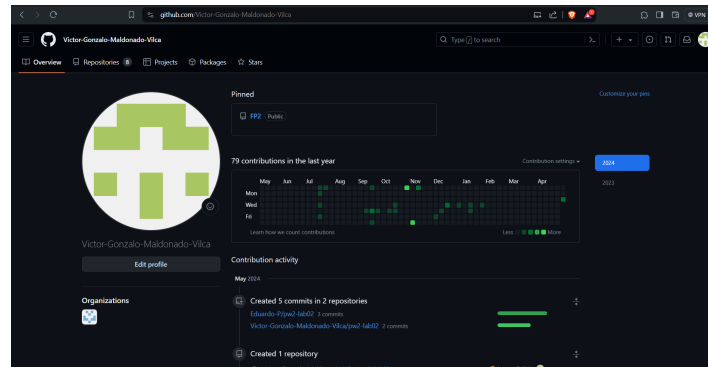


Figura 3: Usuario de Victor Gonzalo

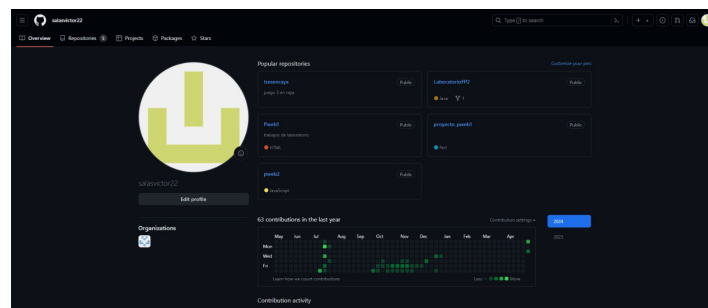


Figura 4: Usuario de Juan Salas



Figura 5: Repositorio creado - pw2-lab02

6.2. git --version

```
PS C:\Users\eduar> git --version
git version 2.43.0.windows.1
PS C:\Users\eduar> |
```

Figura 6: Version de Git

6.3. Creamos un nuevo proyecto en GitHub

6.3.1. Nuevo repositorio en GitHub

- Se creó un nuevo repositorio en GitHub con el nombre de [pw2-lab02](#) al cual se fueron agregados todos los colaboradores del trabajo.



Figura 7: Repositorio creado - pw2-lab02

6.3.2. Crearemos un repositorio local usando git init

```
PS C:\Users\eduar\OneDrive\Documents\Tarea UNSA\Tercer semestre\pweb 2\Laboratorio\pw2-lab02> git init
Initialized empty Git repository in C:/Users/eduar/OneDrive/Documents/Tarea UNSA/Tercer semestre/pweb 2/Laboratorio/pw2-lab02/.git/
```

Figura 8: Comando utilizado

- Este comando se usa para crear un nuevo repositorio Git en una carpeta existente. Al ejecutar este comando, Git establece una estructura de directorios mínima para empezar a llevar el control de versiones.

6.3.3. Crearemos un archivo Readme.md con contenido Markup



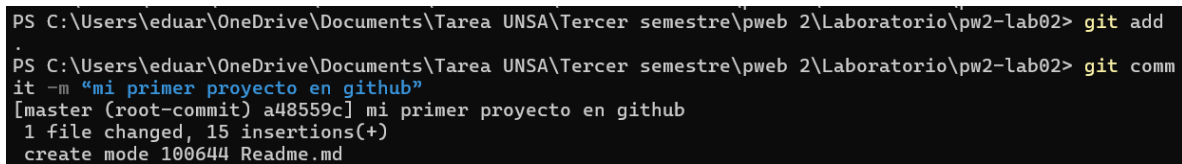
```

1 # Clase Calculator en Java
2
3 Se desea crear una clase Calculator en Java, que tenga las siguientes
operaciones: add, sub, mul, div, mod; estas operaciones recibirán dos
enteros y devolverán un entero.
4 1. Forme grupos de 3 a 5 personas.
5 2. Un integrante del grupo deberá crear el proyecto principal, con el
nombre de su grupo, con la plantilla base:
6
7 ```java
8 class Calculator {
9     int add(int a, int b) { return 0; }
10    int sub(int a, int b) { return 0; }
11    int mul(int a, int b) { return 0; }
12    int div(int a, int b) { return 0; }
13    int mod(int a, int b) { return 0; }
14 }
15
16 3. Comparta el proyecto con sus compañeros de grupo y asigne uno o
dos métodos distintos a cada integrante del grupo.
17 4. Los integrantes del grupo deberán hacer clone, push y pull según
corresponda, de modo que el repositorio contenga la solución final.
18 5. Reportar al profesor que logró culminar la tarea. La tarea debe
ser compartida con el profesor (CarloCorralesD) y entregada usando el
mismo url que se usó para clonar el repositorio.

```

Figura 9: Archivo - Readme.md

6.3.4. Agregaremos este archivo al staging area y hacemos un primer commit



```

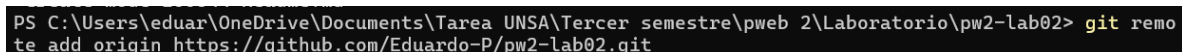
PS C:\Users\eduar\OneDrive\Documents\Tarea UNSA\Tercer semestre\pweb 2\Laboratorio\pw2-lab02> git add .
PS C:\Users\eduar\OneDrive\Documents\Tarea UNSA\Tercer semestre\pweb 2\Laboratorio\pw2-lab02> git comm
it -m "mi primer proyecto en github"
[master (root-commit) a48559c] mi primer proyecto en github
1 file changed, 15 insertions(+)
create mode 100644 Readme.md

```

Figura 10: Comando utilizado

- `git add .` este comando se utiliza para añadir cambios al área de preparación (también llamada "stage" o "index") antes de hacer un commit.
- `git commit` este comando se usa para confirmar (commitear) los cambios añadidos al área de preparación.

6.3.5. Asociamos el repositorio local con el repositorio remoto



```

PS C:\Users\eduar\OneDrive\Documents\Tarea UNSA\Tercer semestre\pweb 2\Laboratorio\pw2-lab02> git remo
te add origin https://github.com/Eduardo-P/pw2-lab02.git

```

Figura 11: Comando utilizado

- `git remote add origin` es el comando que permite establecer una conexión entre un repositorio local y uno remoto, abriendo la puerta a la colaboración, la sincronización, y el respaldo en un entorno de control de versiones.

6.3.6. Actualizamos el repositorio remoto con git push origin master

```
PS C:\Users\eduar\OneDrive\Documents\Tarea UNSA\Tercer semestre\pweb 2\Laboratorio\pw2-lab02> git push origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 720 bytes | 240.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Eduardo-P/pw2-lab02.git
* [new branch]      master -> master
```

Figura 12: Comando utilizado

6.4. Uso de Ramas en GitHub

6.4.1. Crear y trasladarnos a una rama

- `git branch` este comando se utiliza para crear y listar ramas en un repositorio Git.
- `git checkout` este comando tiene dos funciones principales: cambiar de rama y restaurar archivos.

```
PS C:\Users\eduar\OneDrive\Documents\Tarea UNSA\Tercer semestre\pweb 2\Laboratorio\pw2-lab02> git branch div
PS C:\Users\eduar\OneDrive\Documents\Tarea UNSA\Tercer semestre\pweb 2\Laboratorio\pw2-lab02> git branch
div
* master
PS C:\Users\eduar\OneDrive\Documents\Tarea UNSA\Tercer semestre\pweb 2\Laboratorio\pw2-lab02> git checkout div
Switched to branch 'div'
PS C:\Users\eduar\OneDrive\Documents\Tarea UNSA\Tercer semestre\pweb 2\Laboratorio\pw2-lab02> git branch
* div
master
```

Figura 13: Comandos utilizados

- Se creo la rama `div` con `git branch div`, seguidamente se visualizaron las ramas existentes.
- Luego, nos cambiamos a la rama `div` con `git checkout div`

6.4.2. Edicion en la rama `div`

Listing 1: Agregando operación div

```
1 class Calculator {
2     int add(int a, int b){ return 0; }
3     int sub(int a, int b){ return 0; }
4     int mul(int a, int b){ return 0; }
5
6     int div(int a, int b){
7         if (b == 0) {
8             System.out.println("Error: Divisin por cero no est permitida.");
9             return 0;
10        }
11    }
```

6.4.3. Registrar la rama `div` en el repositorio

```
PS C:\Users\eduar\OneDrive\Documents\Tarea UNSA\Tercer semestre\pweb 2\Laboratorio\pw2-lab02> git add .
PS C:\Users\eduar\OneDrive\Documents\Tarea UNSA\Tercer semestre\pweb 2\Laboratorio\pw2-lab02> git commit -m "Agregando operacion de division"
1 file changed, 9 insertions(+), 1 deletion(-)
PS C:\Users\eduar\OneDrive\Documents\Tarea UNSA\Tercer semestre\pweb 2\Laboratorio\pw2-lab02> git push origin div
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 465 bytes | 232.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'div' on GitHub by visiting:
remote:   https://github.com/Eduardo-P/pw2-lab02/pull/new/div
remote:
To https://github.com/Eduardo-P/pw2-lab02.git
 * [new branch]      div -> div
```

Figura 14: Comandos utilizados

- Se registraron los cambios realizados en la rama y estos fueron subidos al repositorio.

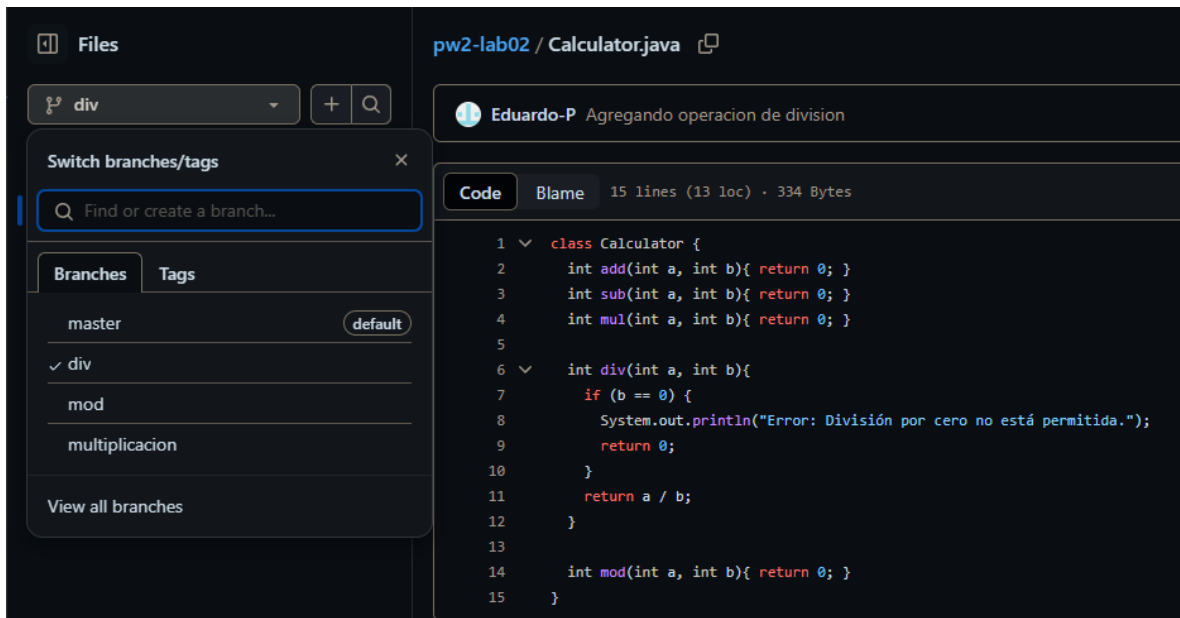


Figura 15: Rama div

- En el repositorio se puede visualizar la nueva rama creada.

6.4.4. Combinando ramas - rama principal **master** con la rama **div**

- **git merge** su propósito principal es integrar cambios de una rama en otra.

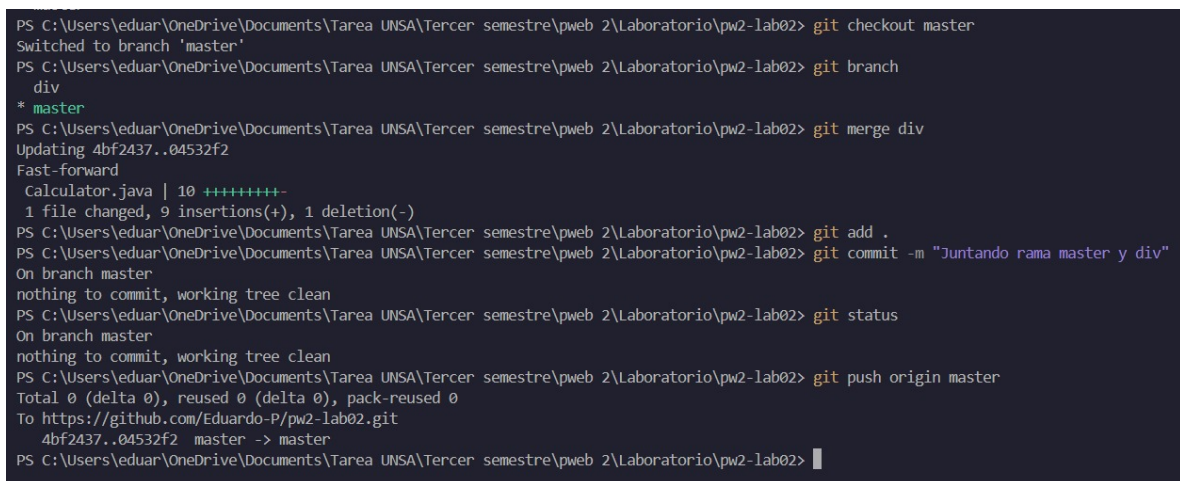
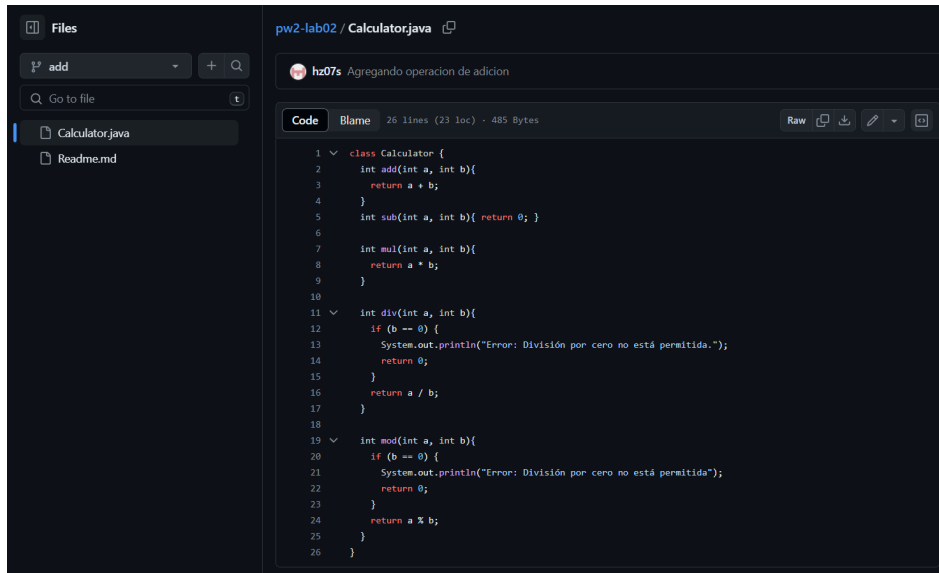


Figura 16: Comandos utilizados

- Primeramente, se cambio a la rama principal **master** para luego utilizar **git merge div** y fusionar los cambios de la rama **div** en **master**.
- Finalmente, lo cambios son agregados al repositorio.

6.5. Todas las Ramas creadas

6.5.1. Rama **add**



Files

add

Go to file

Calculator.java

Readme.md

pw2-lab02 / Calculator.java

hz07s Agregando operacion de adicion

Code Blame 26 lines (23 loc) · 485 Bytes

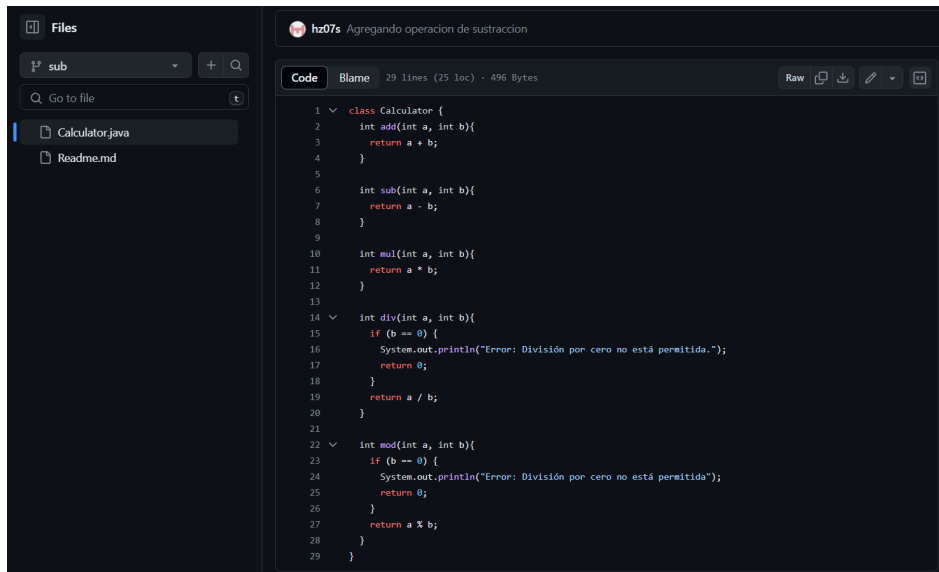
```

1 class Calculator {
2   int add(int a, int b){
3     return a + b;
4   }
5   int sub(int a, int b){ return 0; }
6
7   int mul(int a, int b){
8     return a * b;
9   }
10
11  int div(int a, int b){
12    if (b == 0) {
13      System.out.println("Error: División por cero no está permitida.");
14      return 0;
15    }
16    return a / b;
17  }
18
19  int mod(int a, int b){
20    if (b == 0) {
21      System.out.println("Error: División por cero no está permitida.");
22      return 0;
23    }
24    return a % b;
25  }
26 }

```

Figura 17: Rama add

6.5.2. Rama **sub**



Files

sub

Go to file

Calculator.java

Readme.md

pw2-lab02 / Calculator.java

hz07s Agregando operacion de sustraccion

Code Blame 29 lines (25 loc) · 496 Bytes

```

1 class Calculator {
2   int add(int a, int b){
3     return a + b;
4   }
5
6   int sub(int a, int b){
7     return a - b;
8   }
9
10  int mul(int a, int b){
11    return a * b;
12  }
13
14  int div(int a, int b){
15    if (b == 0) {
16      System.out.println("Error: División por cero no está permitida.");
17      return 0;
18    }
19    return a / b;
20  }
21
22  int mod(int a, int b){
23    if (b == 0) {
24      System.out.println("Error: División por cero no está permitida.");
25      return 0;
26    }
27    return a % b;
28  }
29 }

```

Figura 18: Rama sub

6.5.3. Rama **mul**

Files | pw2-lab02 / Calculator.java | salasvictor22 solucionando error

Switch branches/tags | Find or create a branch...

Branches | Tags

- master (default)
- add
- div
- mod
- ✓ multiplicacion
- sub

View all branches

```

1 class Calculator {
2     int add(int a, int b){ return 0; }
3     int sub(int a, int b){ return 0; }
4
5     int mul(int a, int b){
6         return a * b;
7     }
8
9     int div(int a, int b){
10        if (b == 0) {
11            System.out.println("Error: División por cero no está permitida.");
12            return 0;
13        }
14        return a / b;
15    }
16
17    int mod(int a, int b){ return 0; }
18 }
  
```

Figura 19: Rama **multiplicacion**

6.5.4. Rama **div**

Files | pw2-lab02 / Calculator.java | Eduardo-P Agregando operacion de division

Switch branches/tags | Find or create a branch...

Branches | Tags

- master (default)
- ✓ div
- mod
- multiplicacion

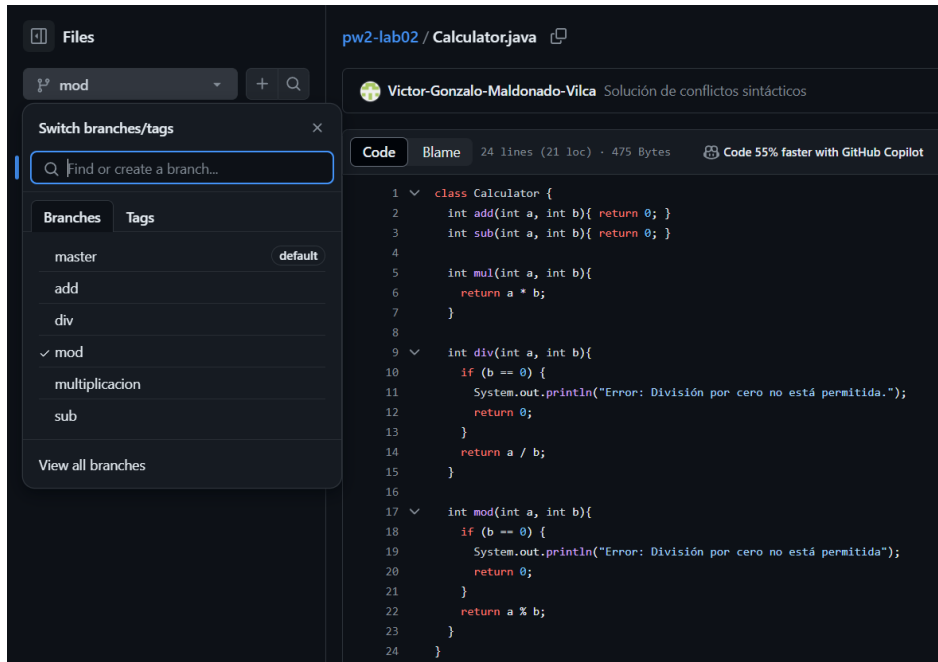
View all branches

```

1 class Calculator {
2     int add(int a, int b){ return 0; }
3     int sub(int a, int b){ return 0; }
4     int mul(int a, int b){ return 0; }
5
6     int div(int a, int b){
7         if (b == 0) {
8             System.out.println("Error: División por cero no está permitida.");
9             return 0;
10        }
11        return a / b;
12    }
13
14    int mod(int a, int b){ return 0; }
15 }
  
```

Figura 20: Rama **div**

6.5.5. Rama **div**



Files

mod

Switch branches/tags

Find or create a branch...

Branches

Tags

master default

add

div

✓ mod

multiplicacion

sub

View all branches

pw2-lab02 / Calculator.java

Victor-Gonzalo-Maldonado-Vilca Solución de conflictos sintácticos

Code Blame 24 lines (21 loc) · 475 Bytes Code 55% faster with GitHub Copilot

```

1  class Calculator {
2      int add(int a, int b){ return 0; }
3      int sub(int a, int b){ return 0; }
4
5      int mul(int a, int b){
6          return a * b;
7      }
8
9      int div(int a, int b){
10         if (b == 0) {
11             System.out.println("Error: División por cero no está permitida.");
12             return 0;
13         }
14         return a / b;
15     }
16
17     int mod(int a, int b){
18         if (b == 0) {
19             System.out.println("Error: División por cero no está permitida.");
20             return 0;
21         }
22         return a % b;
23     }
24 }

```

Figura 21: Rama mod

6.6. Video explicativo

- URL del video explicativo en YouTube.
- https://youtu.be/3aKI_PDTDsk

6.7. Rúbrica para el contenido del Informe y demostración

- El alumno debe marcar o dejar en blanco en celdas de la columna **Checklist** si cumple con el ítem correspondiente.
- Si un alumno supera la fecha de entrega, su calificación será sobre la nota mínima aprobada, siempre y cuando cumpla con todos los ítems.
- El alumno debe autocalificarse en la columna **Estudiante** de acuerdo a la siguiente tabla:

Tabla 1: Niveles de desempeño

	Nivel			
Puntos	Insatisfactorio 25 %	En Proceso 50 %	Satisfactorio 75 %	Sobresaliente 100 %
2.0	0.5	1.0	1.5	2.0
4.0	1.0	2.0	3.0	4.0

Tabla 2: Rúbrica para contenido del Informe y demostración

	Contenido y demostración	Puntos	Checklist	Estudiante	Profesor
1. GitHub	Hay enlace URL activo del directorio para el laboratorio hacia su repositorio GitHub con código fuente terminado y fácil de revisar.	2	X	2	
2. Commits	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4	X	4	
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	X	2	
4. Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2	X	2	
5. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	X	2	
6. Fechas	Las fechas de modificación del código fuente están dentro de los plazos de fecha de entrega establecidos.	2	X	2	
7. Ortografía	El documento no muestra errores ortográficos.	2	X	2	
8. Madurez	El Informe muestra de manera general una evolución de la madurez del código fuente, explicaciones puntuales pero precisas y un acabado impecable. (El profesor puede preguntar para refrendar calificación).	4	X	4	
Total		20		20	

7. Referencias

- <https://git-scm.com/doc>