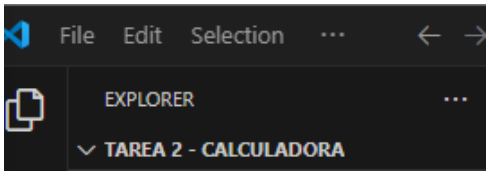
Tarea 2

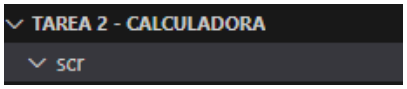
Crear en Visual Studio Code (u otro editor) un proyecto en Java (u otro lenguaje), para simular una calculadora.

1) **El proyecto debe contener una clase principal Calculadora**

Antes de nada, creamos la carpeta de la Tarea 2

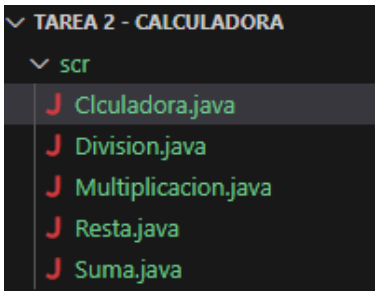


Dentro de esa carpeta, crea otra carpeta llamada scr (en donde pondré mis códigos)

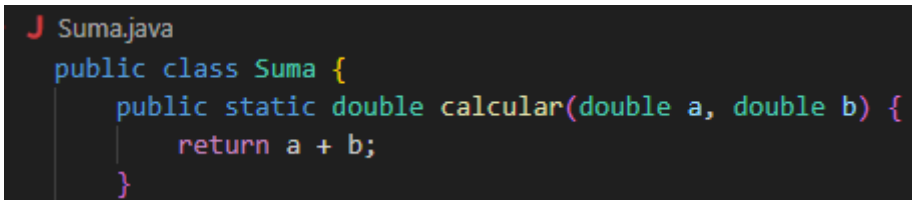


2) **La clase Calculadora debe implementar al menos estas operaciones: Suma, Resta, multiplicación y división.**

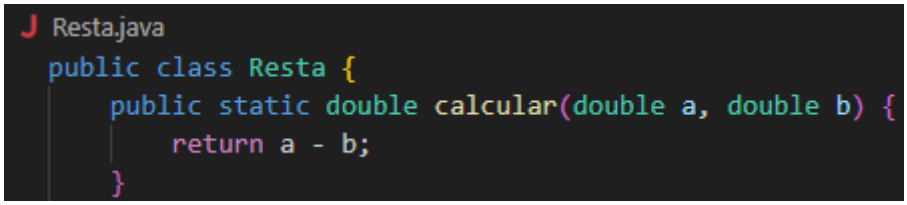
Una vez dentro creamos la calculadora, la suma, la resta la división y la multiplicación.



El código de la Suma:



El código de la Resta:



El código de la Multiplicación:

Multiplicacion.java

```java
public class Multiplicacion {
    public static double calcular(double a, double b) {
        return a * b;
    }
```

El código de la División:

Division.java

```java
public class Division {
    public static double calcular(double a, double b) {
        if (b == 0) {
            System.out.println("Error: División entre 0");
            return 0;
        }
        return a / b;
    }
```

El código de la Calculadora:

Clculadora.java

```java
import java.util.Scanner;

public class Calculadora {
    public static void main(String[] args) {
        Scanner teclado = new Scanner(System.in);

        System.out.print("Ingrese primer número: ");
        double a = teclado.nextDouble();
        System.out.print("Ingrese segundo número: ");
        double b = teclado.nextDouble();

        System.out.println("Suma: " + Suma.calcular(a, b));
        System.out.println("Resta: " + Resta.calcular(a, b));
        System.out.println("Multiplicación: " + Multiplicacion.calcular(a, b));
        System.out.println("División: " + Division.calcular(a, b));
    }
```

3) **Cada operación deberá realizarse en una clase diferente. Corresponderá a una funcionalidad, y por tanto un rama en GIT. Tendrá su propio archivo en el proyecto y en el repositorio.**

Iniciamos el repositorio

```
kamila.gurinagurina@025P113B MINGW64 /e/Despliege Web/Tarea2_Calculadora
$ git init
Initialized empty Git repository in E:/Despliege Web/Tarea2_Calculadora/.git/
```

Enlazamos con repositorio remoto

```
kamila.gurinagurina@025P113B MINGW64 /e/Despliege Web/Tarea2_Calculadora (master
)
$ git remote add origin https://github.com/KamilaGurina/Calculadora.git
```

Se creó el proyecto base de la calculadora en Java con la clase principal Calculadora y las clases de las operaciones básicas (Suma, Resta, Multiplicacion, Division). Se inicializó el repositorio Git local, se hicieron los commits de los archivos y se subió la rama principal (main) al repositorio remoto en GitHub.

```
kamila.gurinagurina@025P113B MINGW64 /e/Despliege Web/Tarea2_Calculadora (master
)
$ git add .

kamila.gurinagurina@025P113B MINGW64 /e/Despliege Web/Tarea2_Calculadora (master
)
$ git commit -m "Tarea 2 subida"
[master (root-commit) 80595a4] Tarea 2 subida
 5 files changed, 41 insertions(+)
 create mode 100644 scr/Clculadora.java
 create mode 100644 scr/Division.java
 create mode 100644 scr/Multiplicacion.java
 create mode 100644 scr/Resta.java
 create mode 100644 scr/Suma.java

kamila.gurinagurina@025P113B MINGW64 /e/Despliege Web/Tarea2_Calculadora (master
)
$ git branch -M main

kamila.gurinagurina@025P113B MINGW64 /e/Despliege Web/Tarea2_Calculadora (main)
$ git push origin main
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 12 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (8/8), 1.03 KiB | 1.03 MiB/s, done.
Total 8 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/KamilaGurina/Calculadora.git
 * [new branch]      main -> main
```

4) **En la máquina local habrá 3 carpetas, que corresponderán a la simulación de 3 Programadores (A, B y C). Reparte el trabajo entre los 3. Demuestra mediante los commits que han trabajado los 3 en el proyecto. Una idea es repartir el trabajo por funcionalidades.**

Crear ramas por funcionalidad

- feature/suma → Programador A
- feature/resta → Programador B
- feature/multdiv → Programador C

```
kamila.gurinagurina@025P113B MINGW64 /e/Despliege Web/Tarea2_Calculadora (main)
$ git checkout -b feature/suma
git checkout main
git checkout -b feature/resta
git checkout main
git checkout -b feature/multdiv
Switched to a new branch 'feature/suma'
Switched to branch 'main'
Switched to a new branch 'feature/resta'
Switched to branch 'main'
Switched to a new branch 'feature/multdiv'
```

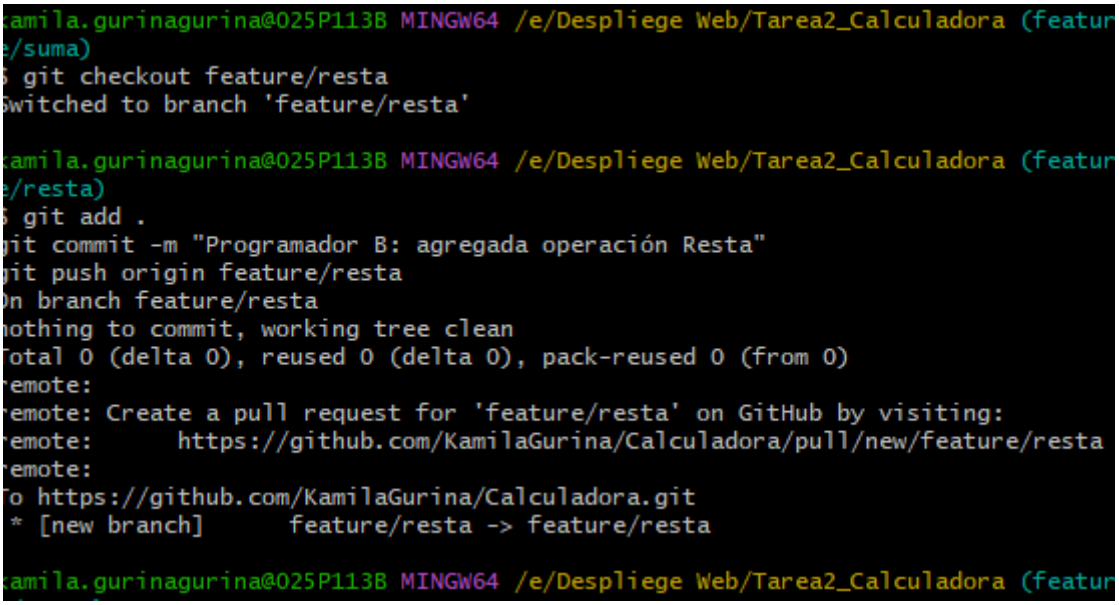
5) **Cada persona diferente deberá modificar su funcionalidad y la clase Calculadora si fuese necesario. Subir sus cambios al mismo repositorio remoto de GitHub**. Al final, deberán fusionar las ramas a la rama principal (master o main), simulando la integración del equipo.

Rama suma → Se creó y subió la rama `feature/suma` para la operación Suma, simulando el trabajo del Programador A, quedando lista para integrarse con la rama principal.

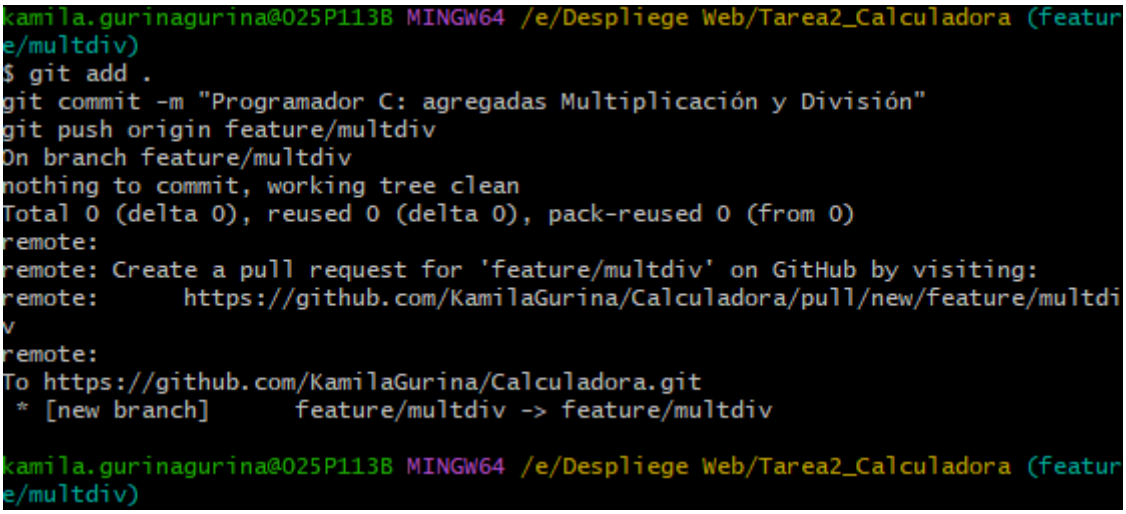
```
kamila.gurinagurina@025P113B MINGW64 /e/Despliege Web/Tarea2_Calculadora (featur
e/suma)
$ git checkout feature/suma
Already on 'feature/suma'
```

```
kamila.gurinagurina@025P113B MINGW64 /e/Despliege Web/Tarea2_Calculadora (featur
e/suma)
$ git add .
git commit -m "Programador A: agregada operación Suma"
git push origin feature/suma
On branch feature/suma
nothing to commit, working tree clean
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'feature/suma' on GitHub by visiting:
remote:      https://github.com/KamilaGurina/Calculadora/pull/new/feature/suma
remote:
To https://github.com/KamilaGurina/Calculadora.git
 * [new branch]      feature/suma -> feature/suma
```

Rama resta



Rama multiplicación y división



6) **Deberá simularse al menos un conflicto, y se demostrará que se ha hecho una resolución a mano del mismo.**

Para generar un conflicto modificamos la línea en el código Suma.java

Suma.java

```java
public class Suma {
    public static double calcular(double a, double b) {
        // Cambio en rama suma
        System.out.println("Cambio rama suma");
        return a + b;
    }
```

Volvemos a Git Bash y commiteamos los cambios

```
kamila.gurinagurina@025P113B MINGW64 /e/Despliege Web/Tarea2_Calculadora (main)
$ git add scr/Suma.java

kamila.gurinagurina@025P113B MINGW64 /e/Despliege Web/Tarea2_Calculadora (main)
$ git add scr/Resta.java

kamila.gurinagurina@025P113B MINGW64 /e/Despliege Web/Tarea2_Calculadora (main)
$ git commit -m "Rama resta: cambio en la misma línea de Suma"
[main 2b8b349] Rama resta: cambio en la misma línea de Suma
 1 file changed, 2 insertions(+)

kamila.gurinagurina@025P113B MINGW64 /e/Despliege Web/Tarea2_Calculadora (main)
$ git push origin feature/resta
Everything up-to-date

kamila.gurinagurina@025P113B MINGW64 /e/Despliege Web/Tarea2_Calculadora (main)
$ git checkout main
Already on 'main'

kamila.gurinagurina@025P113B MINGW64 /e/Despliege Web/Tarea2_Calculadora (main)
$ git merge feature/suma
```

El conflicto se ha generado correctamente

MINGW64:/e/Despliege Web/Tarea2_Calculadora

```
Merge branch 'feature/suma'
# Please enter a commit message to explain why this merge is necessary,
# especially if it merges an updated upstream into a topic branch.
#
# Lines starting with '#' will be ignored, and an empty message aborts
# the commit.
~
```

Una vez resuelto el conflicto los subimos (se me olvidó sacar foto del conflicto a resolver)

```
kamila.gurinagurina@O25P113B MINGW64 /e/Despliege Web/Tarea2_Calculadora (main)
$ git add scr/Suma.java
git commit -m "Conflicto en Suma resuelto"
[main 4a416da] Conflicto en Suma resuelto
 1 file changed, 2 deletions(-)

kamila.gurinagurina@O25P113B MINGW64 /e/Despliege Web/Tarea2_Calculadora (main)
$ git push origin main
Enumerating objects: 19, done.
Counting objects: 100% (18/18), done.
Delta compression using up to 12 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (11/11), 1.04 KiB | 1.04 MiB/s, done.
Total 11 (delta 4), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (4/4), completed with 2 local objects.
To https://github.com/KamilaGurina/Calculadora.git
   80595a4..4a416da  main -> main
```

7) **Deberá simularse un HotFix en la rama master, por ejemplo una comprobación para que no se pueda dividir entre 0.**

Modificamos el método calcular asi:

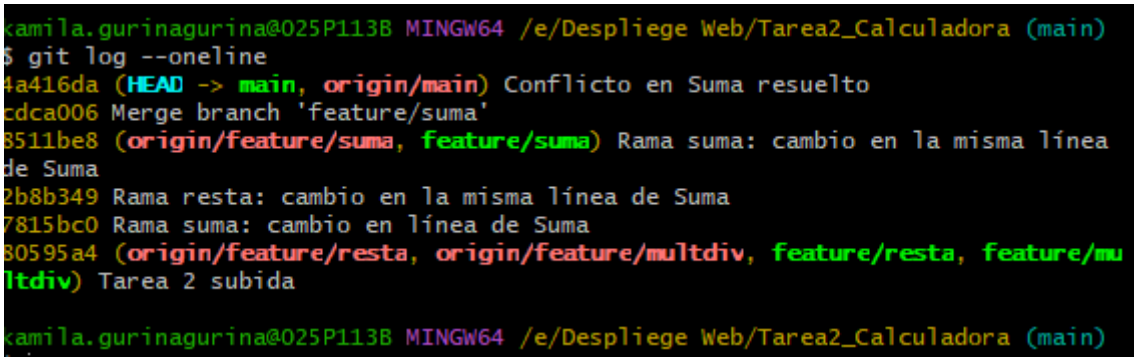
```java
public class Division {
    public static double calcular(double a, double b) {
        if (b == 0) {
            System.out.println("Error: No se puede dividir entre 0");
            return 0; // o cualquier valor que quieras para manejar el error
        }
        return a / b;
    }
```

Y en el GitBash

```
kamila.gurinagurina@O25P113B MINGW64 /e/Despliege Web/Tarea2_Calculadora (main)
$ git add scr/Division.java

kamila.gurinagurina@O25P113B MINGW64 /e/Despliege Web/Tarea2_Calculadora (main)
$ git commit -m "HotFix: evitar división entre 0"
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   scr/Resta.java

no changes added to commit (use "git add" and/or "git commit -a")

kamila.gurinagurina@O25P113B MINGW64 /e/Despliege Web/Tarea2_Calculadora (main)
$ git push origin main
Everything up-to-date
```

8) **El control de commits debe reflejar los cambios realizados.**

Se muestra el historial de commits. Se observa la subida inicial del proyecto, los cambios realizados en cada rama, el merge de la rama suma y la resolución del conflicto manual en `Suma.java`



Conclusión:

Durante el desarrollo, el mayor reto fue comprender cómo generar y resolver conflictos en Git manualmente. También aprendí la importancia de los commits claros y del trabajo en ramas separadas.”