**Requerimiento 1**

Se pide implementar el código de la actividad 1 mediante java y eclipse. Además, hay que tener en cuenta que hay que cada integrante del equipo tendrá que trabajar con GIT y todo el equipo tendrá un repositorio GIT común (Github).

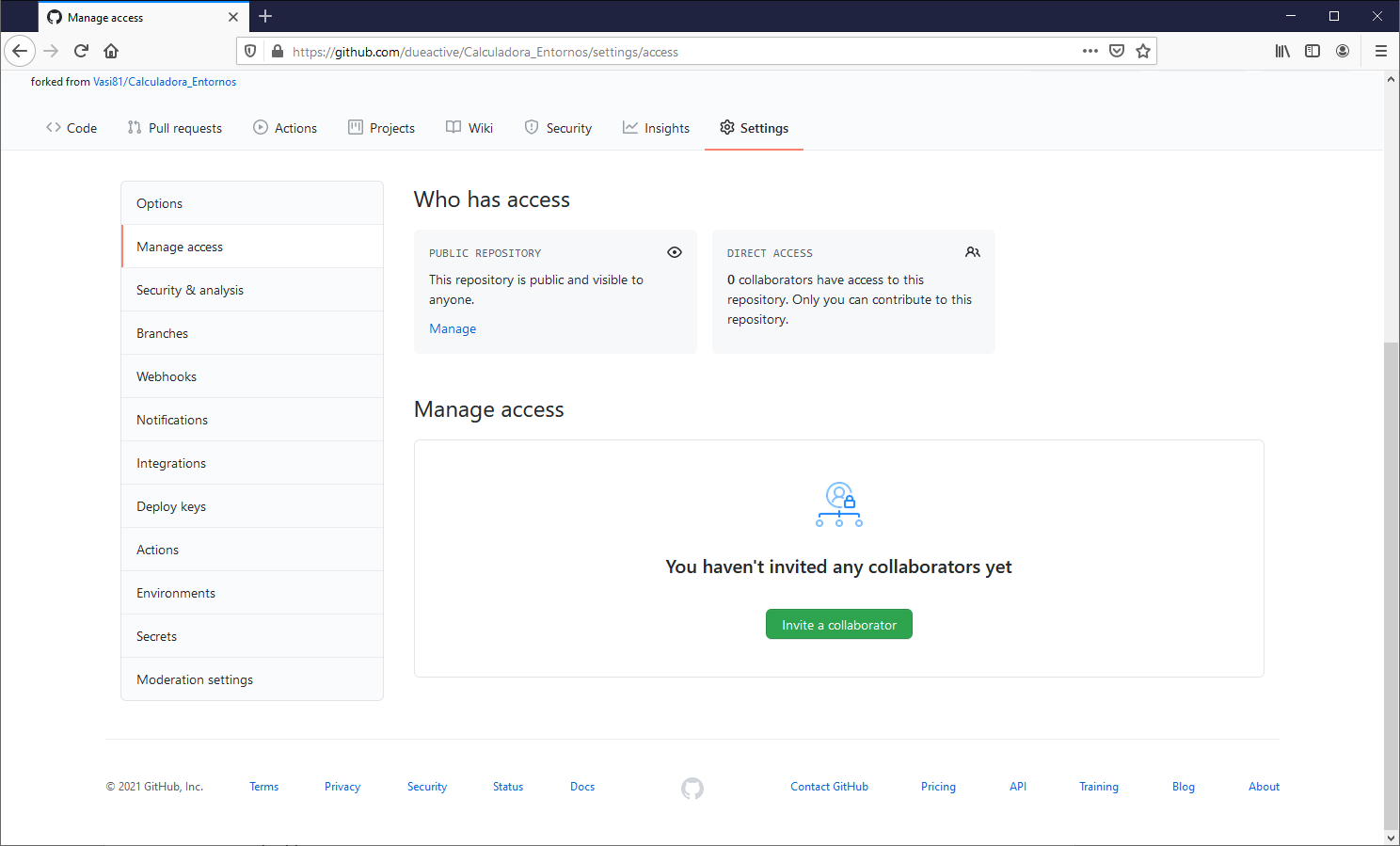
Cada integrante del equipo tendrá la labor de implementar una clase completa haciendo los commits necesarios a su repositorio local y sincronizarse con el resto del equipo a través del repositorio remoto.

\*\*Antes de nada, unificamos el nombre del paquete para nuestras clases.

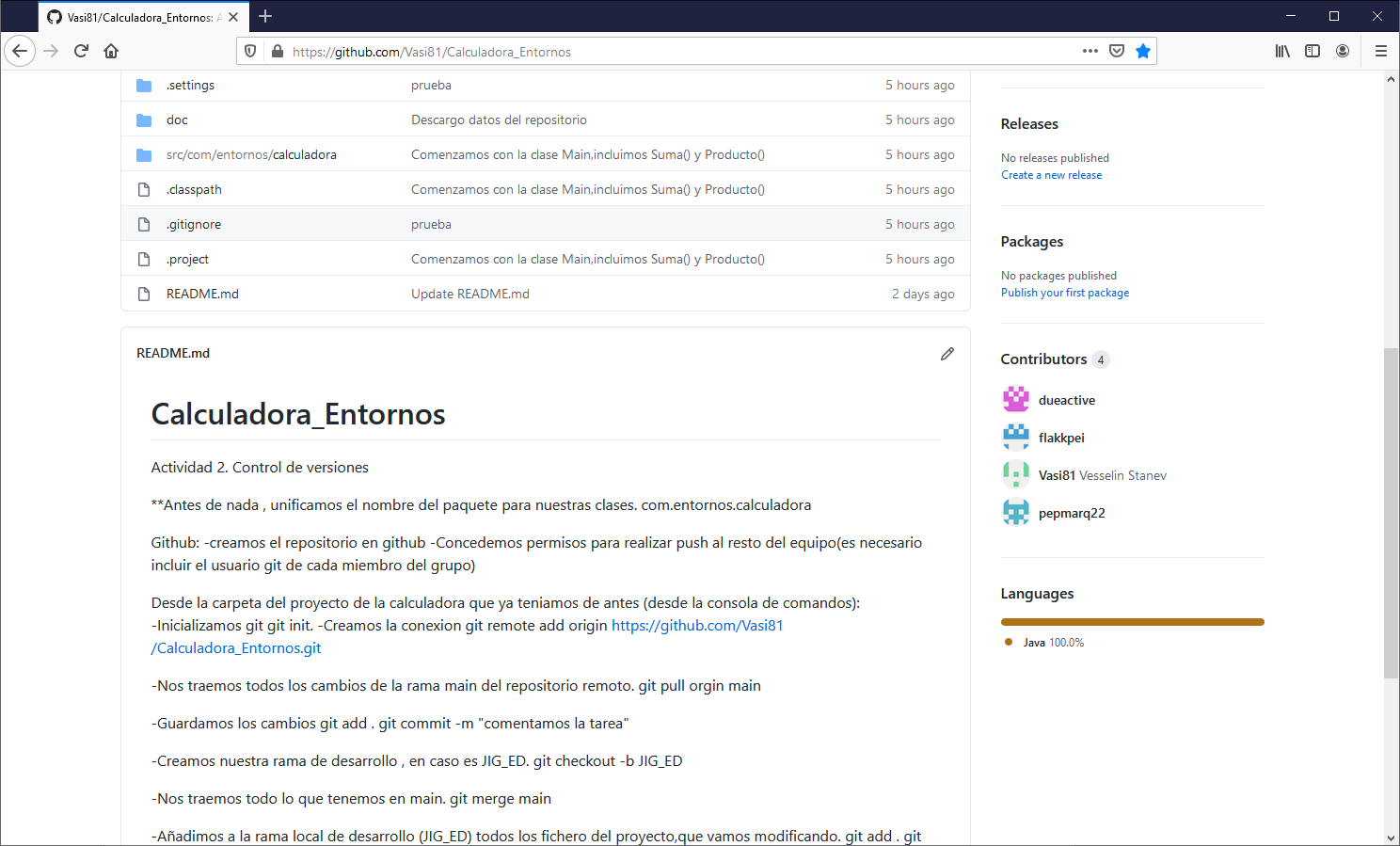
**com.entornos.calculadora**

-creamos el repositorio en Github (<https://github.com/Vasi81/Calculadora_Entornos.git>).

-Asignamos los integrantes al proyecto para que tengan permisos sobre ese repositorio, de lo contrario no podrán realizar acciones como PUSH.



Integrantes:



\*\*Importante, para evitar futuros problemas con el fichero .gitignore (del repositorio remoto ) lo definimos de la siguiente manera:

##############################

## Eclipse

##############################

.settings/

bin/

tmp/

.metadata

.classpath

.project

De esta manera solo subiremos al repositorio los ficheros .java y la carpeta doc , cuando generemos la nueva documentación en JavaDoc y no tendremos problemas con el resto de las configuraciones del proyecto de cada miembro del equipo.

Una vez que tenemos el repositorio remoto, comenzamos a nivel local cada integrante del equipo.

Desde la carpeta que contiene el proyecto de la Calculadora de cada miembro, desde la consola de comando escribimos:

**git init** (Inicializa el proyecto con git).

Como en el repositorio remoto tenemos la rama main (master), creamos nuestra rama main en nuestro repositorio local :

🡪 git checkout -b main (Como rama master).

🡪git checkout -b Nombre Rama para desarrollo (Como rama de desarrollo, en esta rama realizaré todos los cambios en el desarrollo).

A continuación, verificamos las ramas activas que tenemos, **git branch**.

Texto

Descripción generada automáticamente

Después podemos conectarnos al repositorio remoto que usaremos los integrantes del equipo para ir subiendo los cambios (pull) o descargando los cambios realizados por otrp compañero (push) , Creamos la conexión:

**git remote add origin** [**https://github.com/Vasi81/Calculadora\_Entornos.git**](https://github.com/Vasi81/Calculadora_Entornos.git)

**git checkout -b JIG\_ED** (creamos una rama para desarrollo).

**git pull orgin main** : Descargamos el repositorio remoto en la rama de desarrollo para verificar si tenemos algún cambio por descargar.

Es posible que nos encontremos con este problema:

**fatal: refusing to merge unrelated histories**

**Se produce cuando se fusionan dos proyectos no relacionados (es decir, proyectos que no conocen la existencia del otro y que tienen historiales de commit que no coinciden).**

Solución:

git pull origin main --allow-unrelated-histories

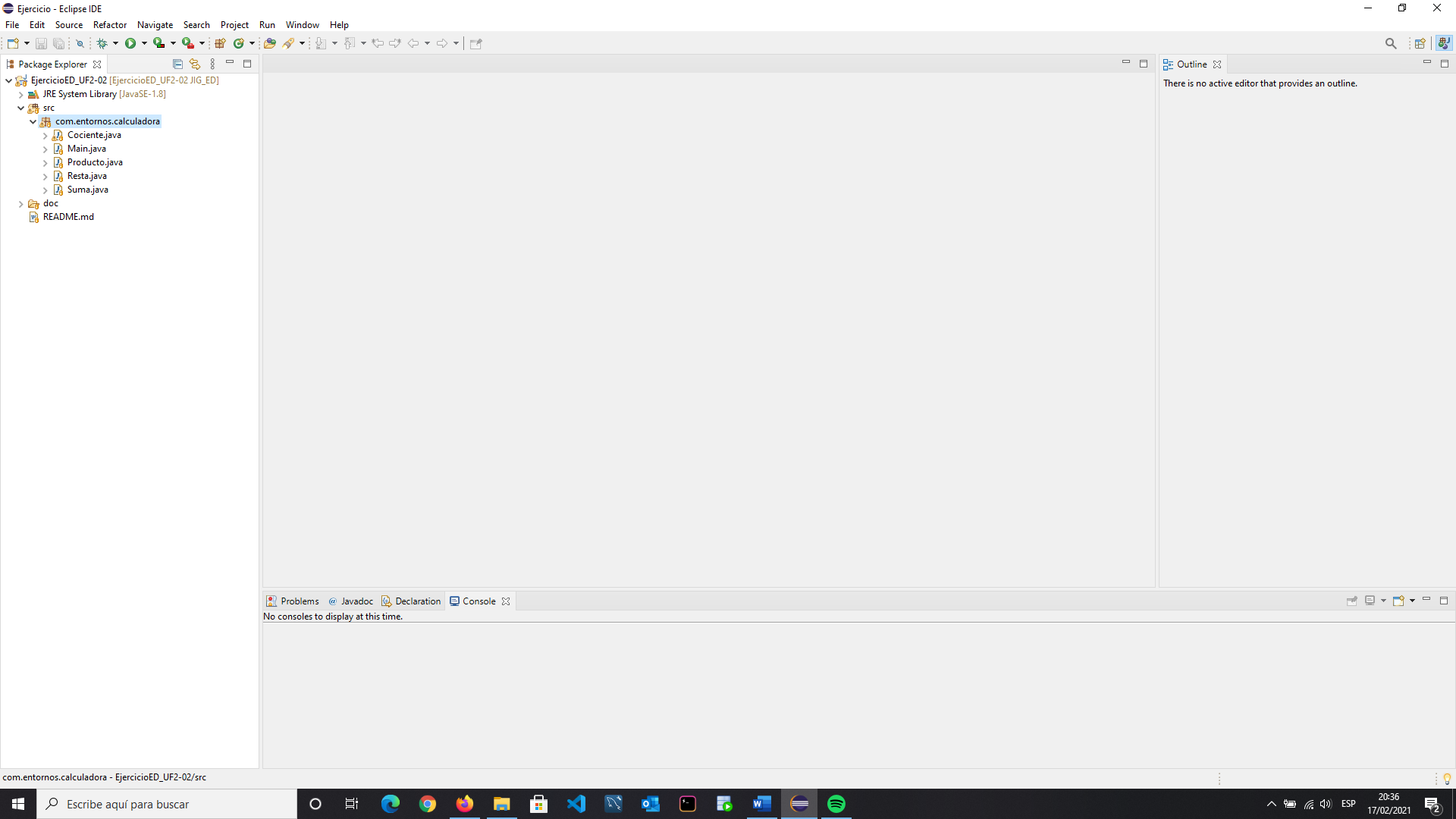
Realizará el Auto-merging y tendremos que editar el fichero afectado y ver con que cambios nos queremos quedar. “<<<<<<< HEAD”.

A continuación, escribo el primer commit para comenzar con la implementación del código.

**git add .**

**git commit -m “incluimos los cambios que se vayan aplicando”.**

Desde Eclipse comenzamos con los cambios o la implementación del código y si nos fijamos en Eclipse nos indicará la rama en la que estamos trabajando.



**Requerimiento 2**

Se pide que cada integrante trabaje con una rama propia, pero cada vez que haga una subida al repositorio compartido fusione su rama con la rama máster para poder ir viendo los cambios del resto del equipo.

Comenzamos con los cambios y según los realizamos los podemos ir incluyendo en git o podemos aglutinar varios cambios en un solo commit.

**git add .** (Para que aplique a todos los ficheros afectados o podemos indicar el fichero que queremos incluir).

**git commit -m “Comentario sobre el cambio”**

Esto lo tendríamos que realizar por cada cambio o conjunto de cambios que vayamos realizando.

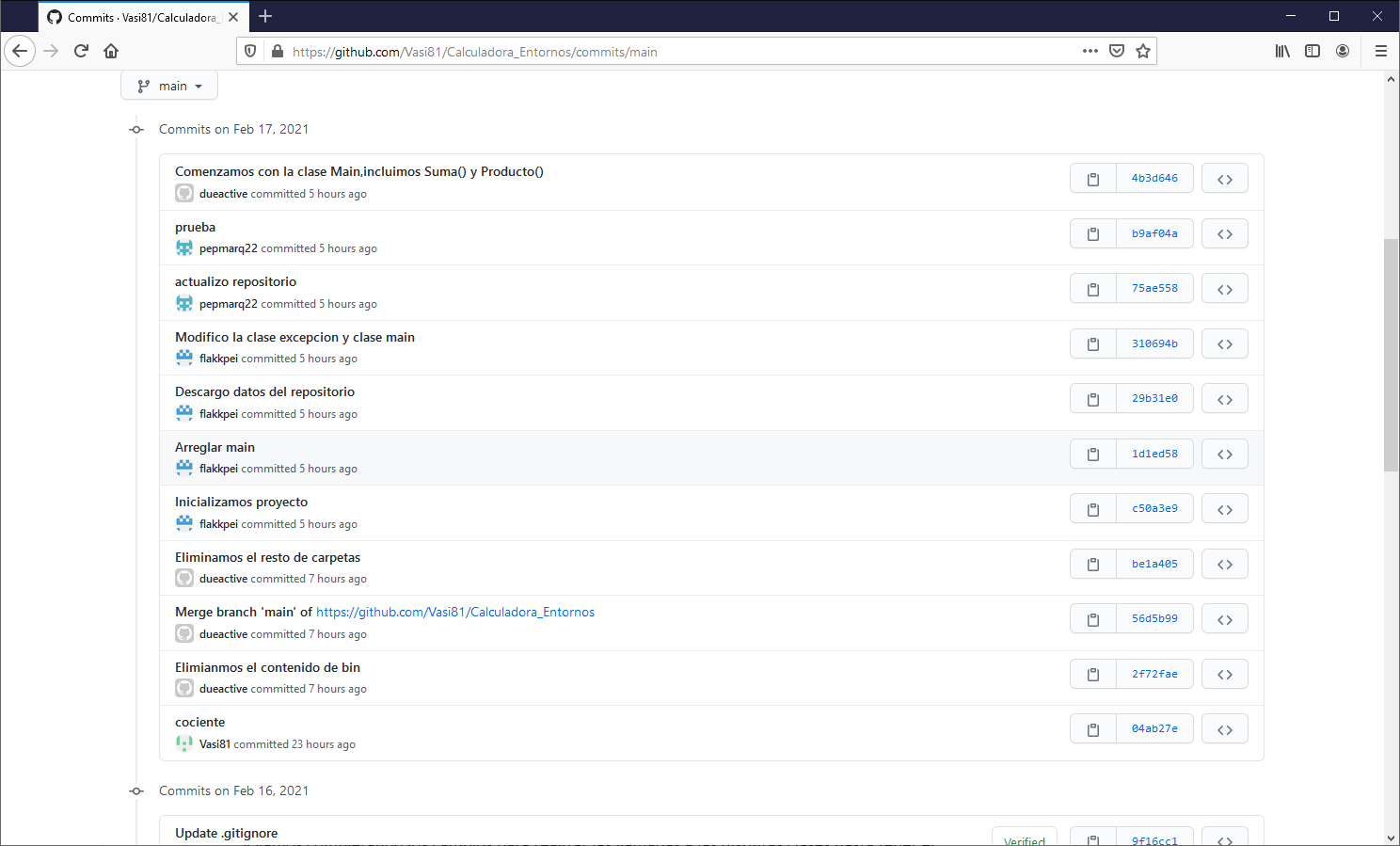
**Como en la actividad anterior no quedamos satisfechos por como realizamos la documentación de JavaDoc , hemos repasado todo lo referente a la documentación con JavaDoc, por lo tanto esto también se refleja en los commits.**

Una vez tengamos todos los cambios con sus respectivos commits , nos cambiamos de rama , a la rama main de nuestro repositorio local.

Desde esta rama realizamos git merge Rama de desarrollo, para fusionar lo que tenemos en la rama de desarrollo con la rama main.

Una vez que ambas ramas tienen el mismo contenido, podemos subir dichos cambios al repositorio remoto.

**git push origin main (Subimos los cambios al repositorio remoto).**



Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Gitk

**Requerimiento 3**

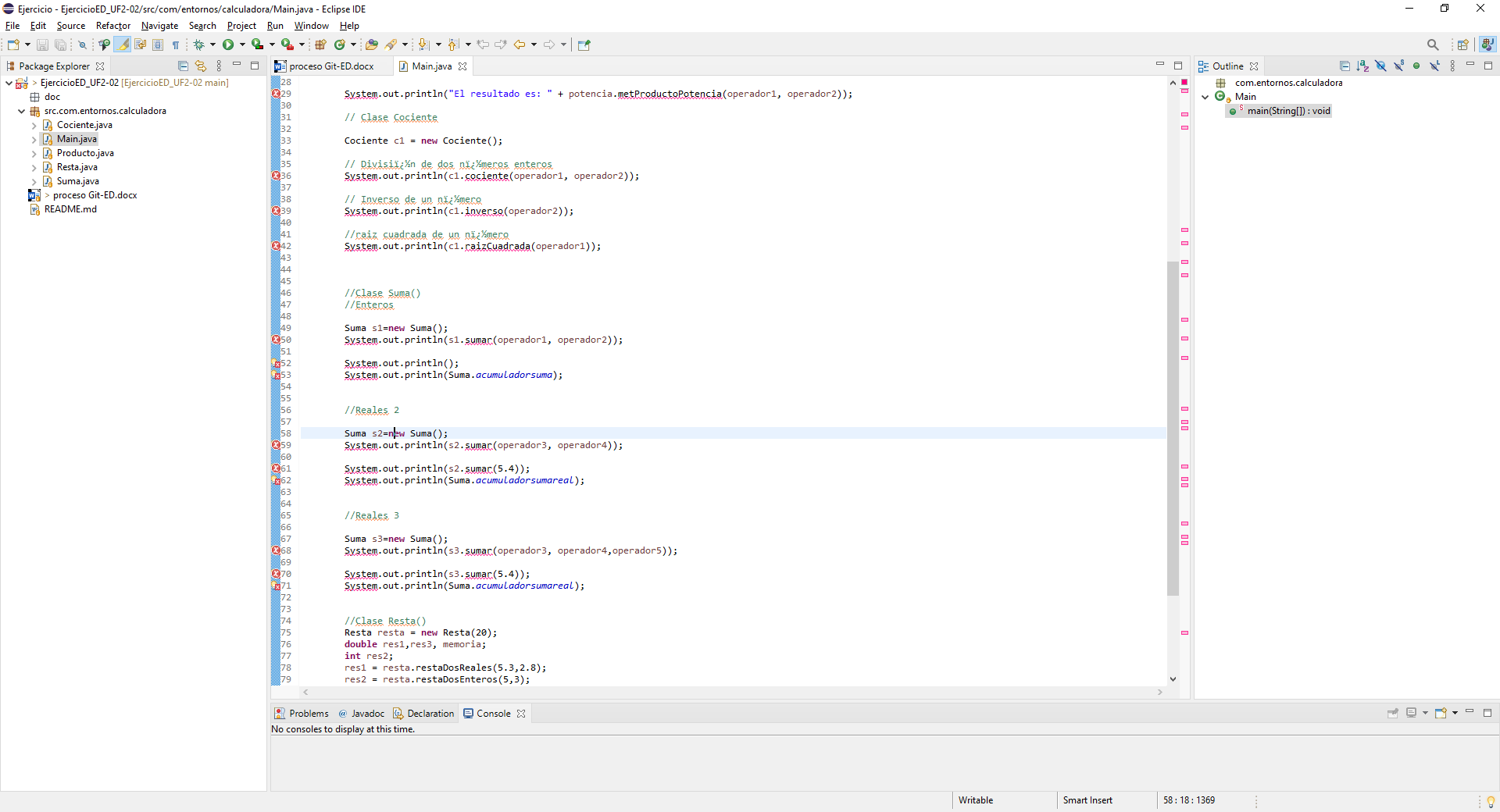
Se pide una clase main que permita probar las clases de la calculadora. Para ello cada integrante tendrá que modificar el código que llama a los métodos de su clase en la clase main, es decir, todos los integrantes del equipo tendrán que trabajar sobre la misma clase main. Con que cada integrante pruebe uno de los métodos que ha realizado, es más que suficiente.

Nótese, que esta parte es probable que, de problemas de sincronización, por lo que se espera que los integrantes del equipo hablen y decidan como afrontar el problema.

Para este caso nos organizamos para subir una primera versión de la clase main y a continuación, cada uno de los miembros, de manera ordenada, nos descargamos dicha clase e iremos completando los cambios para realizar las llamadas a las distintas clases hasta tener el main con todas las clases implementadas.

Git checkout JIG\_ED

Modifico la clase Main , para incluir la llamada a la clase , en este caso la clase Suma.:



Git add .

Git commit -m “Incluimos el uso de la clase Suma()”

Git checkout main

Git merge JIG\_ED

Git push origin main

Una vez finalizada esta parte el resto de los compañeros descargaremos dicha versión para tener el mismo main.

**git pull origin main** (nos indicará que ficheros sufren cambios).

Texto

Descripción generada automáticamente

Con la documentación operamos de la misma forma, una vez tengamos todas las clases en el repositorio podemos generar la documentación de todas las clases.

Una vez que tengamos la documentación (doc) subimos los cambios:

🡪git add .

🡪git commit -m “Generamos la documentación JavaDoc”.

🡪git checkout main

🡪git merge JIG\_ED

🡪git push origin main.

Pasos realizados por los integrantes del equipo:

Vesselin Stanev:

1. Una vez asignados los permisos al repositorio he creado mi repositorio local (git init) y la conexión con el repositorio remoto (git remote add <https://github.com/Vasi81/Calculadora_Entornos>)
2. He realizado una clonación del repositorio remoto con el comando git clone <https://github.com/Vasi81/Calculadora_Entornos>)
3. He creado una rama de desarrollo en mi repositorio local (git checkout -b vbs\_ed) sobre la que empiezo a trabajar para implementar la clase Cociente.
4. Después de realizar la implementación, he añadido los cambios al stage (git add . ) y he realizado los commits oportunos (git commit -m”……”)
5. Una vez todo comiteado y listo traslado el puntero HEAD a mi rama principal: git checkout main.
6. Una vez posicionado en la rama main, hago una fusión con mi rama de desarrollo: git merge vbs\_ed
7. Después de hacer la fusión subo los cambios al repositorio remoto: git push origin main
8. En los posteriores cambios y modificaciones de la clase Main, he seguido prácticamente los mismos pasos, pero antes de empezar a trabajar en mi rama, he descargado la última versión del repositorio remoto con el comando git pull