**MARIA CARMEN CORREA HERAS**

**DANIEL DEL RÍO PÉREZ**

**JUAN RAMON VARÓ NÚÑEZ**

**VERÓNICA BONIS MARTÍN**

**UF4 ENTORNOS DE DESARROLLO**

**Actividad 2. Tarea en equipo**

**Control de versiones**

Los integrantes del equipo son:

MARIA CARMEN CORREA HERAS

DANIEL DEL RÍO PÉREZ

JUAN RAMON VARÓ NÚÑEZ

VERÓNICA BONIS MARTÍN

[](https://github.com/carwenblue/EntornosDesarrolloTelefonica)

URL repositorio remoto: <https://github.com/carwenblue/EntornosDesarrolloTelefonica>

A continuación se presentan las tareas a realizar y el método de trabajo elegido:

**Requerimiento 1**

Para resolver el primer requerimiento mantenemos una corta reunión (a través de Discord) en la que definimos la estructura que utilizaremos para implementar el código de la calculadora. Se decide finalmente crear una clase *Main* que nos permitirá probar el funcionamiento de las clases suma, resta, producto y multiplicación mediante un menú ( *switch*) en el que podemos elegir la clase ( Suma “s”, Resta “r”, Producto “p” y Cociente “d”) y dentro de cada clase otro menú (*switch)* que nos permite probar cada método ( 1, 2, 3, 4 ).

Cada integrante instala GIT y crea su usuario en GitHub, compartimos estos usuarios para darnos permiso para trabajar en el repositorio remoto:

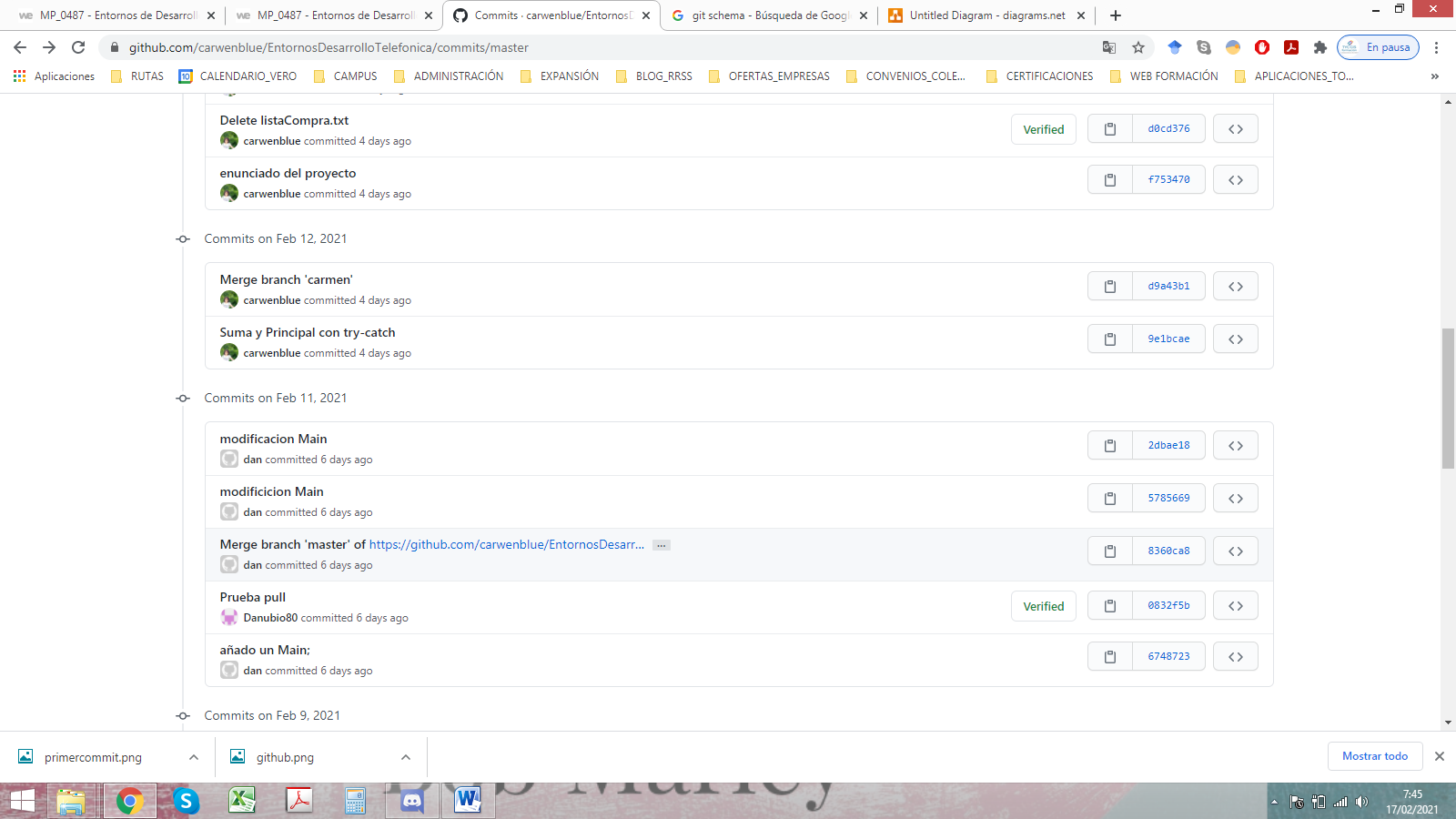
* [María del Carmen Correa Heras](https://github.com/carwenblue/EntornosDesarrolloTelefonica)
* [Juan Ramón Varó](https://github.com/JRVaro)
* [Daniel del Río Pérez](https://github.com/Danubio80)
* [Verónica Bonis Martín](https://github.com/verobm)

Hacemos algunas pruebas con el repositorio remoto y finalmente el flujo de trabajo seguido es el siguiente:

1. Clase principal (Main) y primera clase Producto:



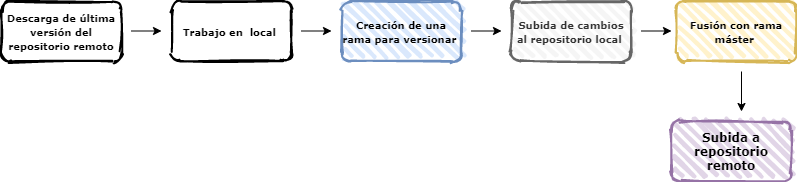
Para empezar a trabajar todos sobre la misma base hacemos el primer commit de la clase principal ( Main) sobre la que debemos trabajar todos los miembros del equipo.



* Primera versión Main y Producto: Daniel del Río Pérez

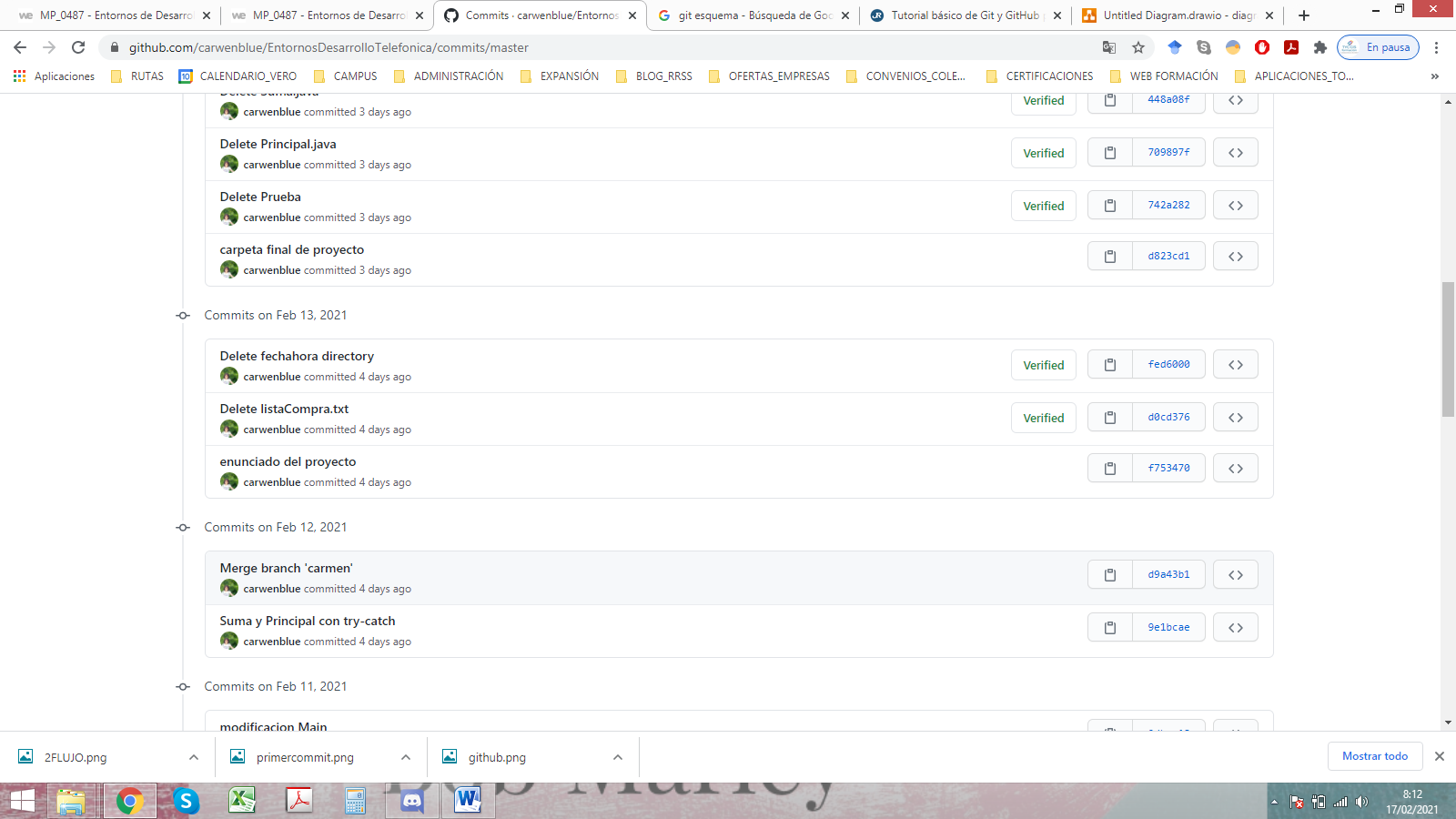
**Requerimiento 2**

Continuamos trabajando sobre la versión ya subida del Main, para ello es necesario otro flujo de trabajo:

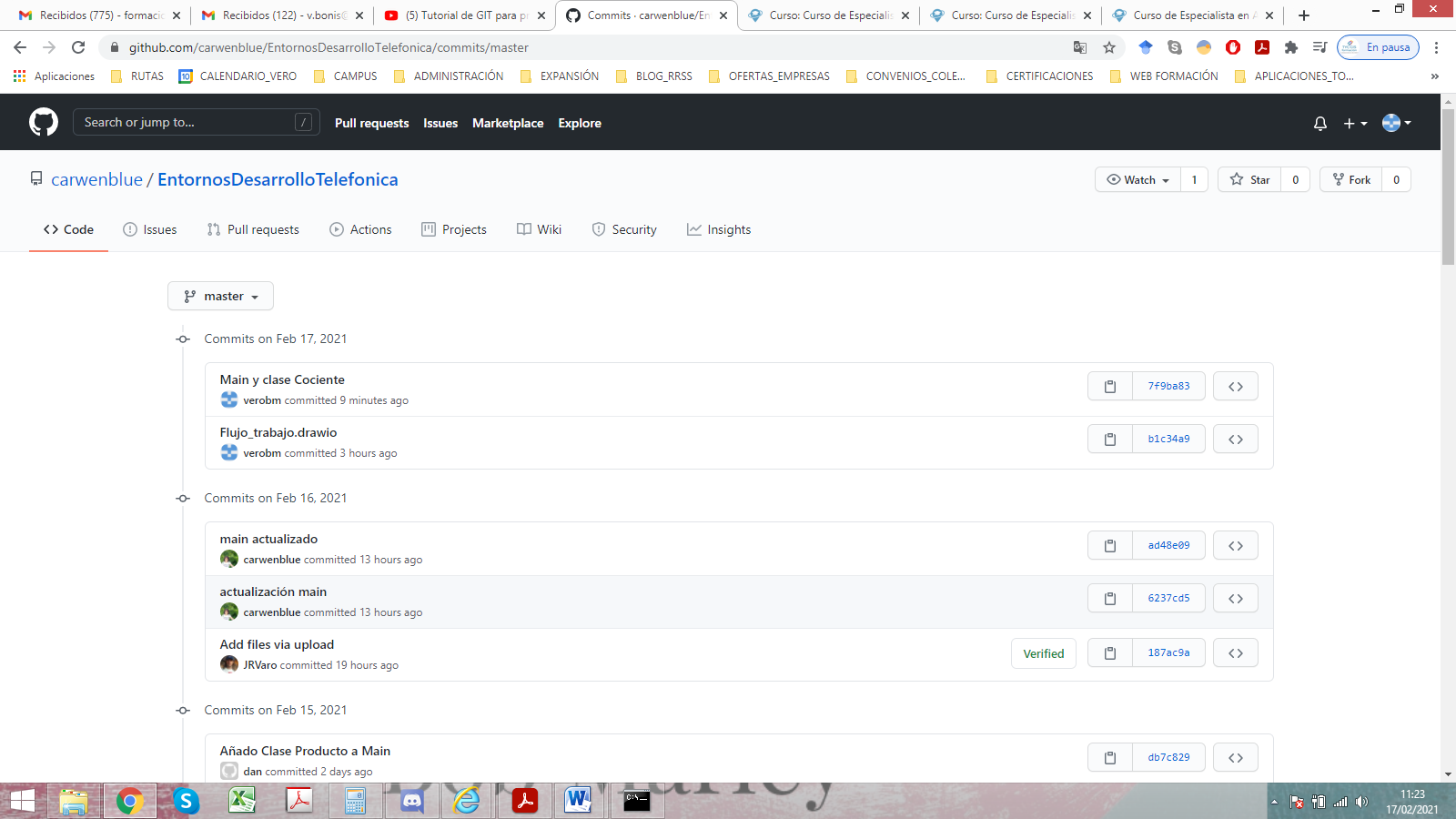


1. *git remote add origin* [*https://github.com/carwenblue/EntornosDesarrolloTelefonica*](https://github.com/carwenblue/EntornosDesarrolloTelefonica)
2. *git pull origin master 🡪 trabajo y cambios en local*
3. *git branch nombredelarama*
4. *git checkout nombredelarama*
5. *git add –A*
6. *git commit –m “Etiqueta del commit”*
7. *git checkout master*
8. *git branch merge nombredelarama*
9. *git push origin master*

En este punto tenemos algún problema de sincronización, ya que necesitamos trabajar todos sobre la clase principal Main, para ello establecemos un orden de trabajo y el siguiente miembro del equipo actualiza la clase principal y sube su clase utilizando para ello su rama y fusionando con la rama máster:



* Segunda versión Main y Suma: María del Carmen Correa Heras



* Tercera versión Main y Cociente: Verónica Bonis Martín
* Cuarta versión Main y Resta : Juan Ramón Varó

**Requerimiento 3**

Para cada método se han realizado algunas modificaciones sobre el código original documentado en el Javadoc:

Suma (indicar cambios)

Resta (indicar cambios)

*Clase Producto*

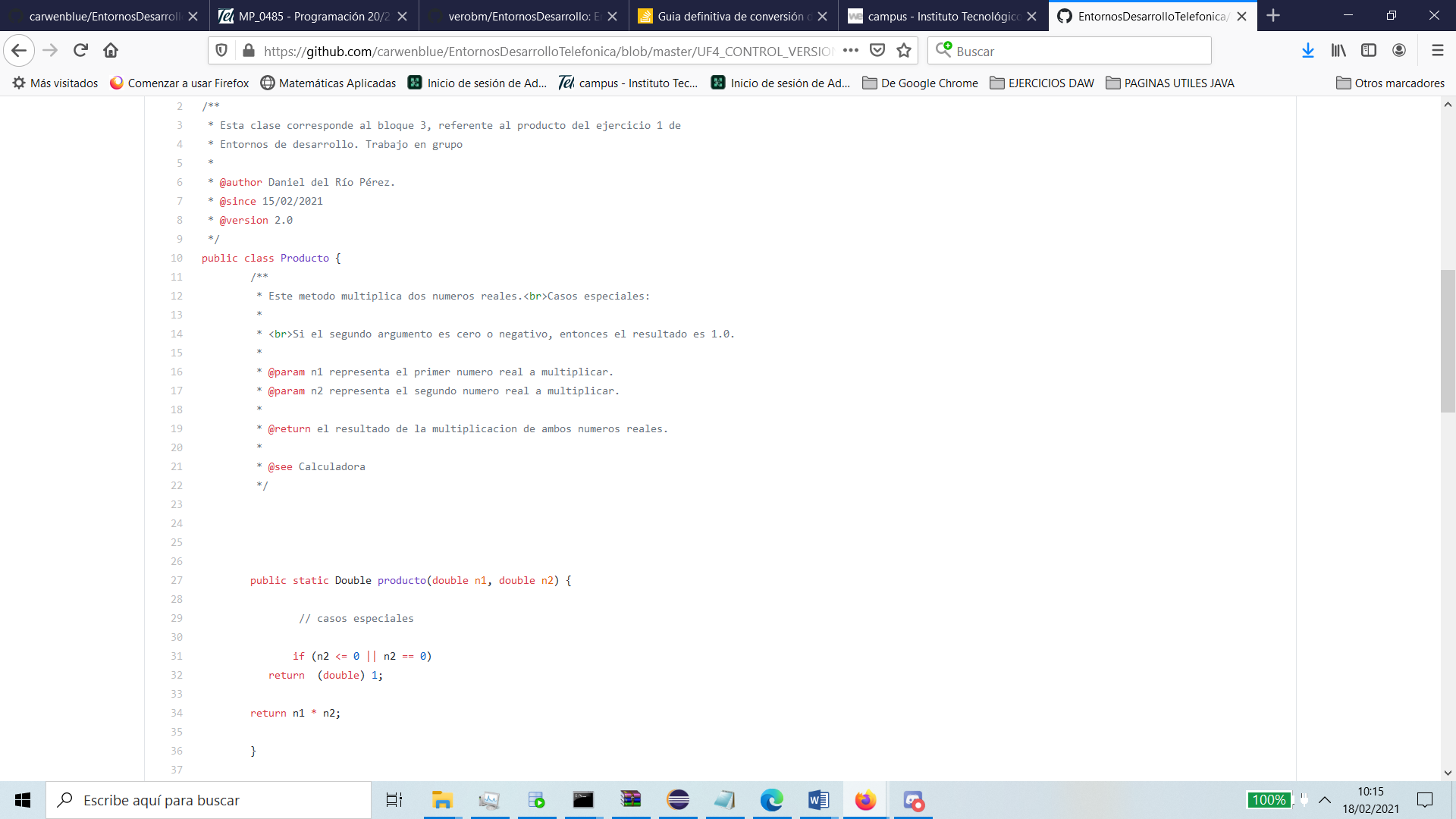
Al evaluar esta clase se ha detectado que la implementacion de todos los metodos que se habian especificado en el **JAVADOC** era excesivo para nuestro nivel,por lo que se ha dejado algunos de los mas usuales.

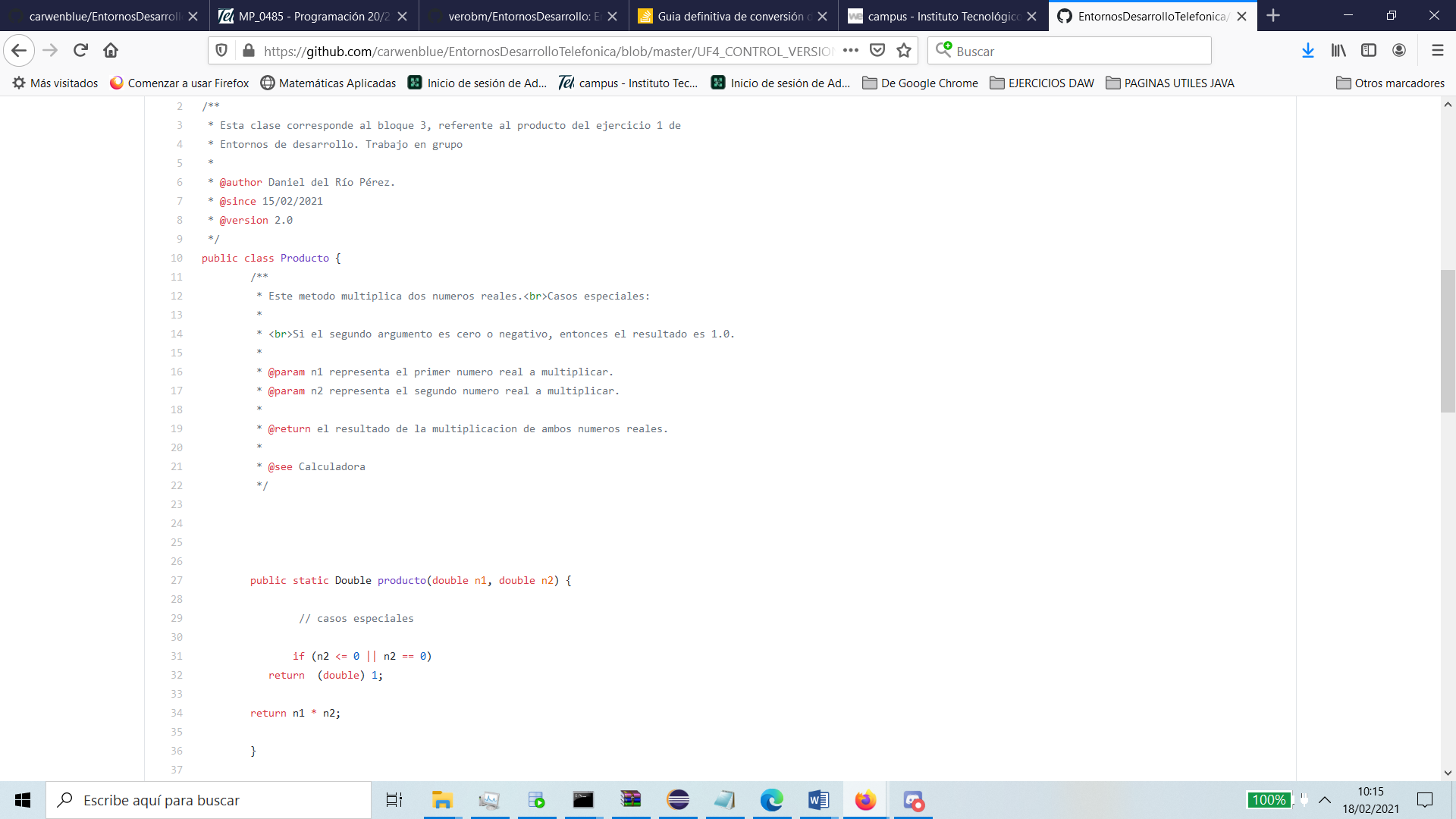
Se observa que cuando los numeros con los que se opera son excesivamente grandes la propia implementacion del algorimo por parte Java, de como resultado *Infinity*.

* Método1 – Producto

Caso especial:

Si el algún argumento es cero o negativo, entonces el resultado es 1.0.

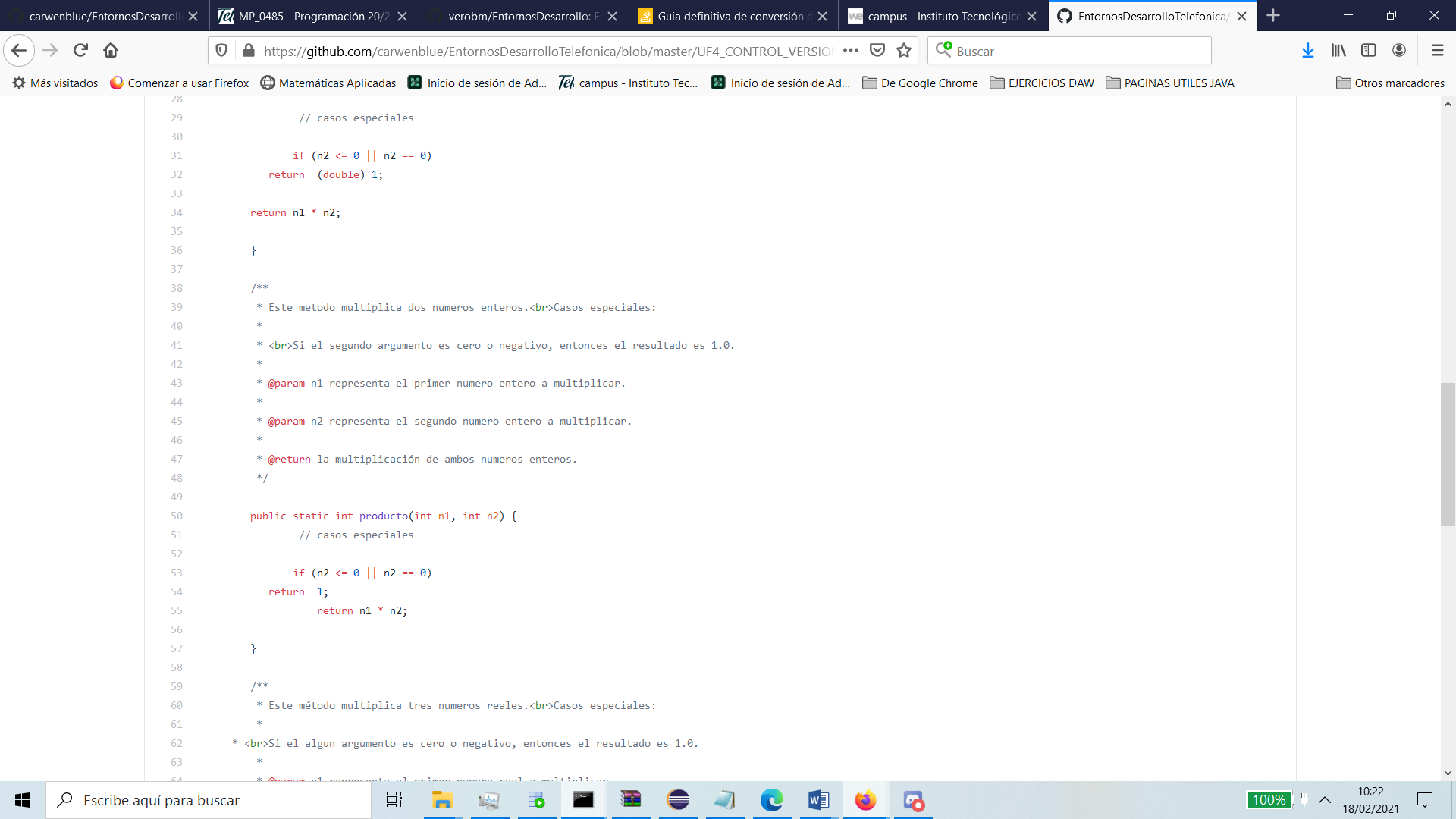




* Método2 – Producto

Caso especial:

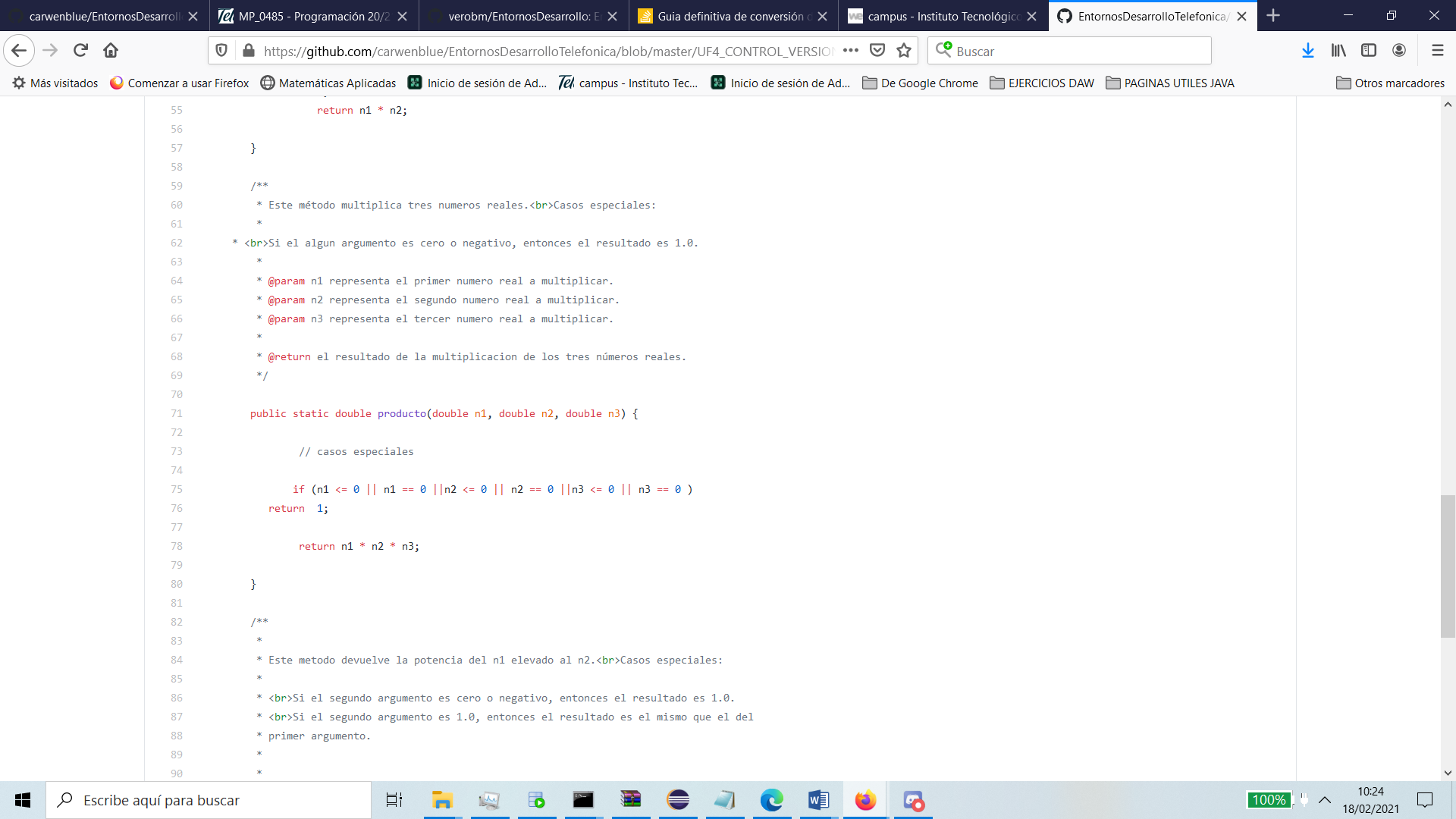
Si el algún argumento es cero o negativo, entonces el resultado es 1.0.



* Método3 – Producto

Caso especial:

Si el algún argumento es cero o negativo, entonces el resultado es 1.0.

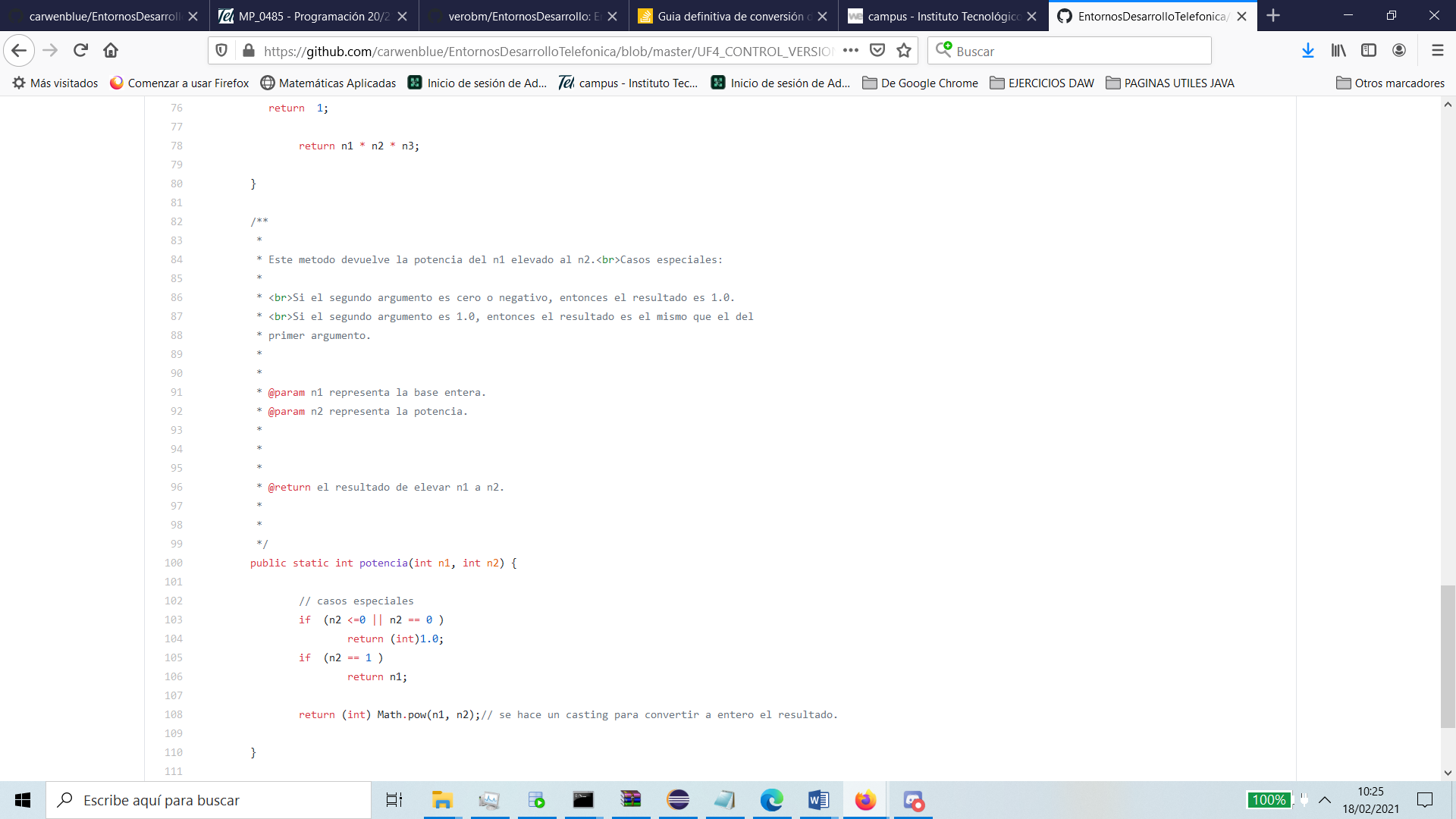


* Método4 – Producto

Caso especial:

Si el algún argumento es cero o negativo, entonces el resultado es 1.0.

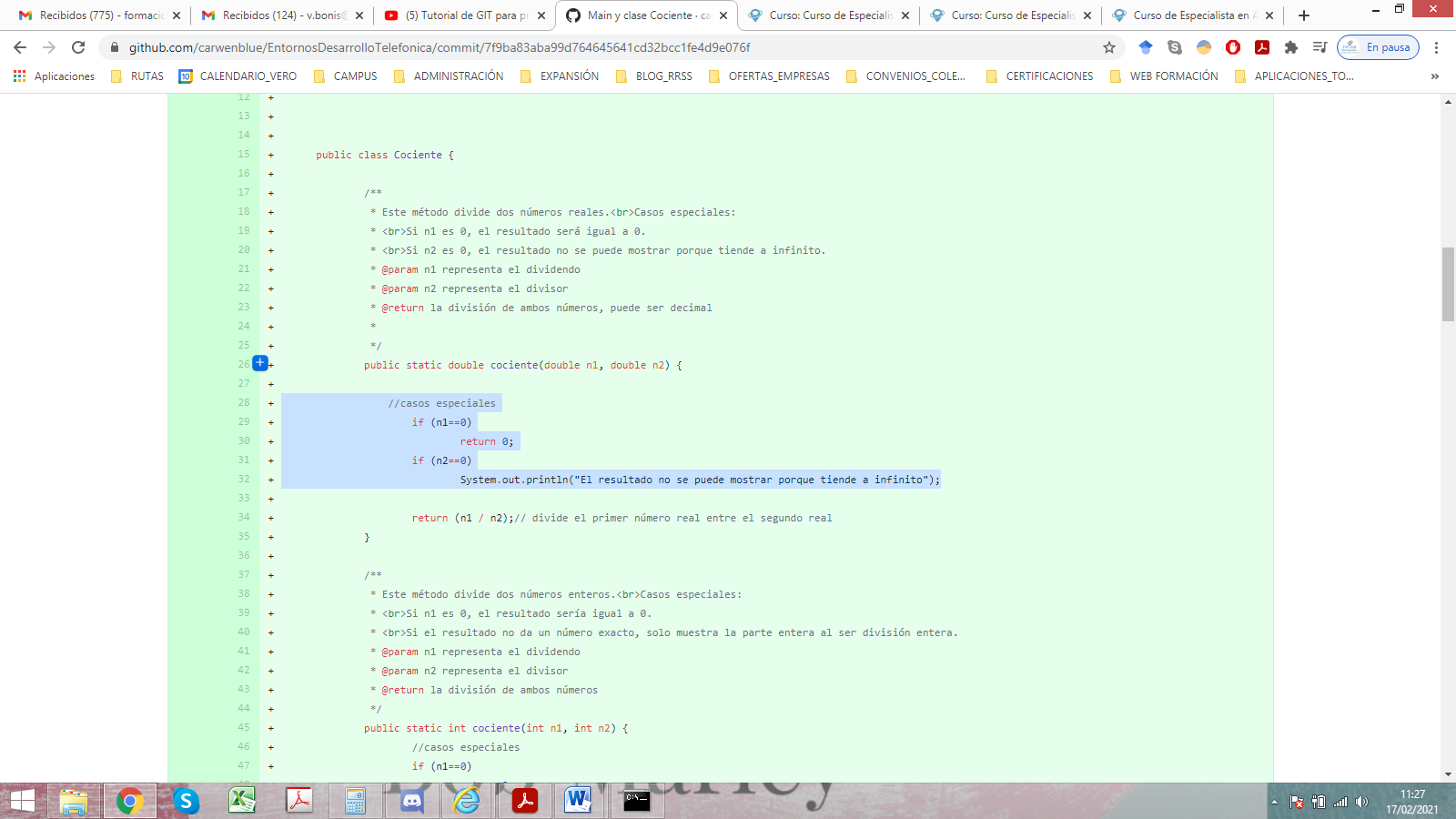
Si el segundo argumento es 1, el resultado será el primer argumento.



*Clase Cociente*

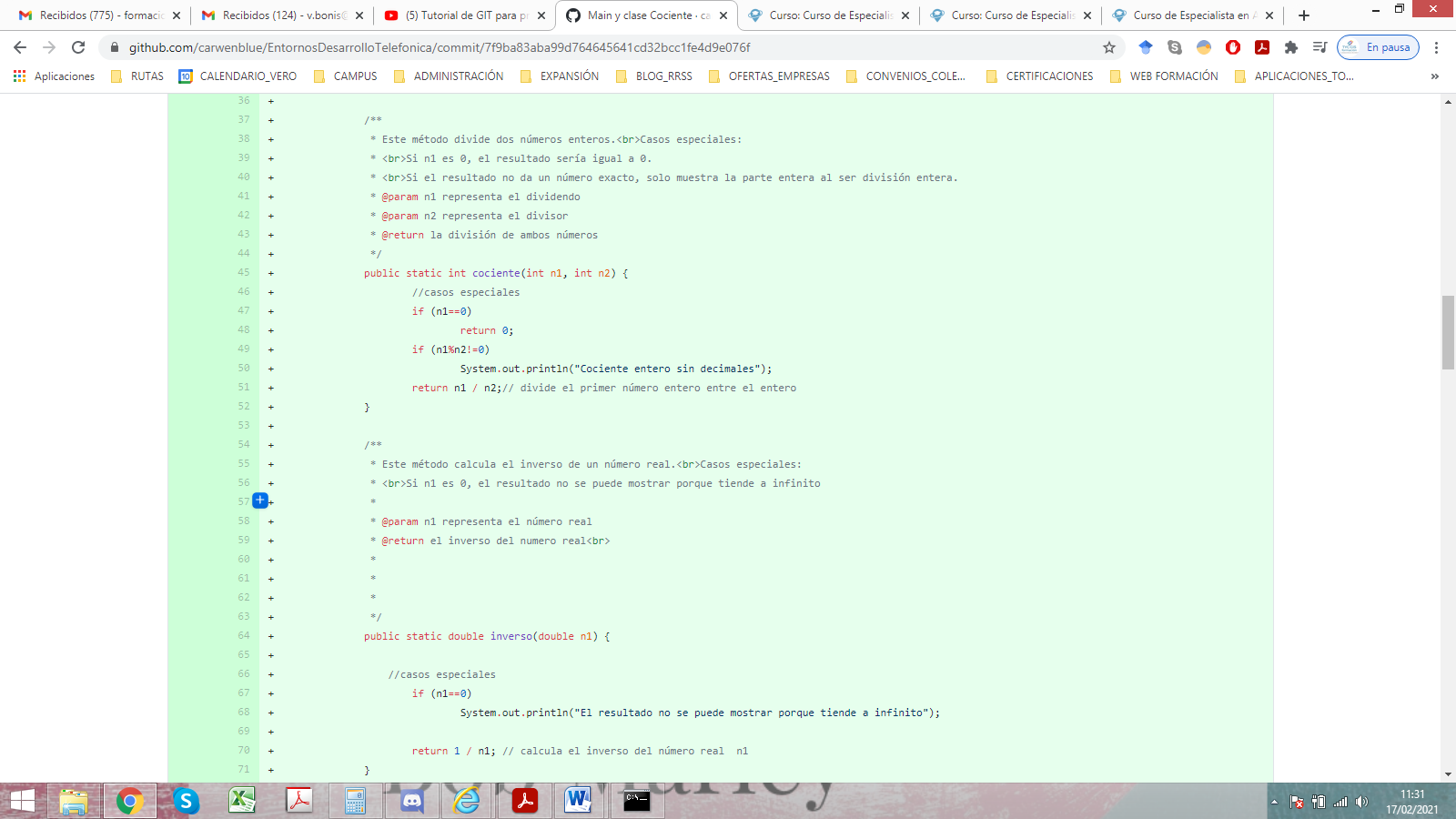
* Método 1 – Cociente

Se implementan dos casos especiales (si n1 es 0 y si n2 es 0) y se elimina el caso en el que cualquiera de los número es NaN



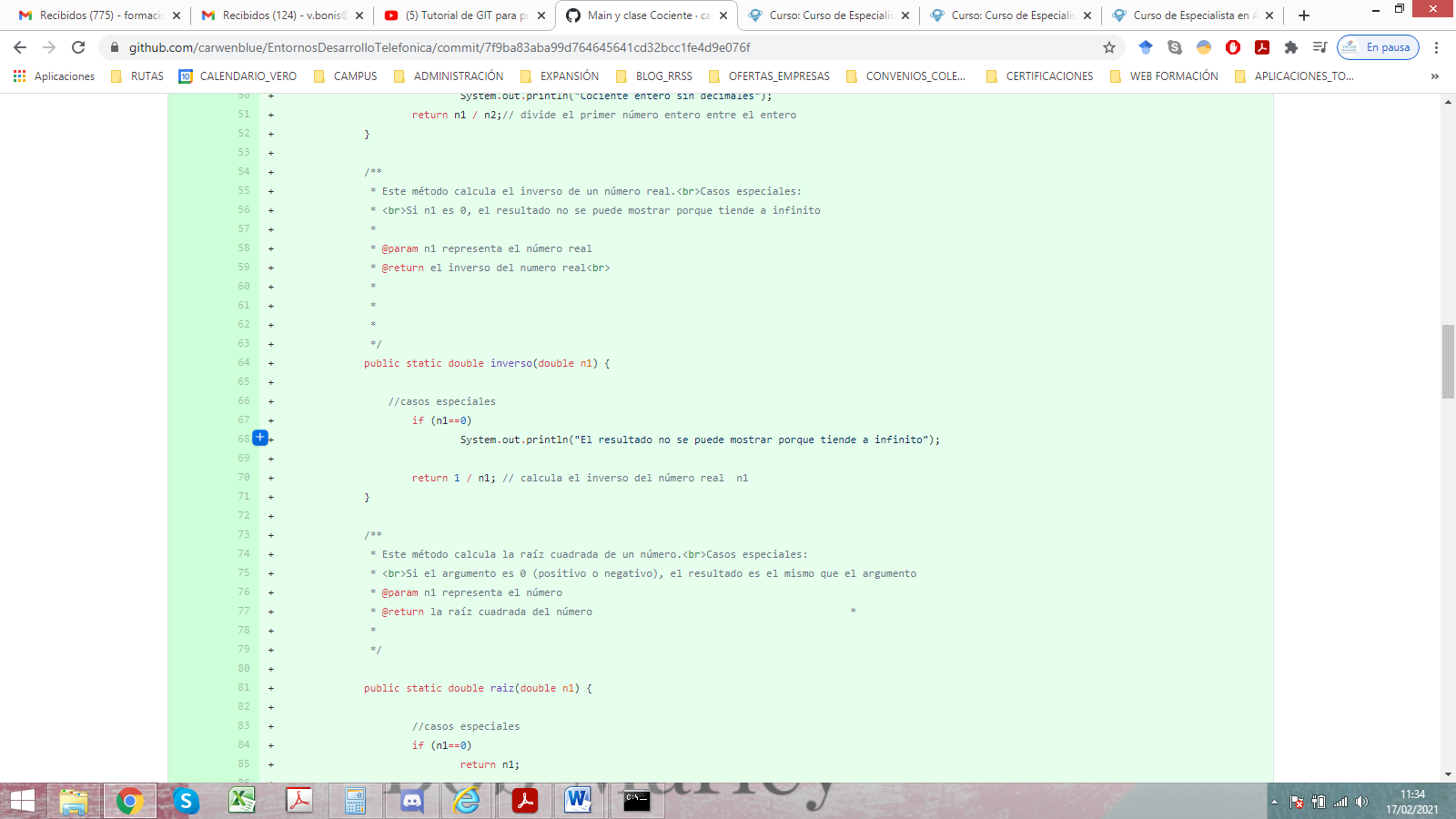
* Método 2 – Cociente

Se implementan dos casos especiales ( si n1 es 0 y si el resultado no da un número exacto) y se elimina el caso en el que cualquiera de los números es NaN



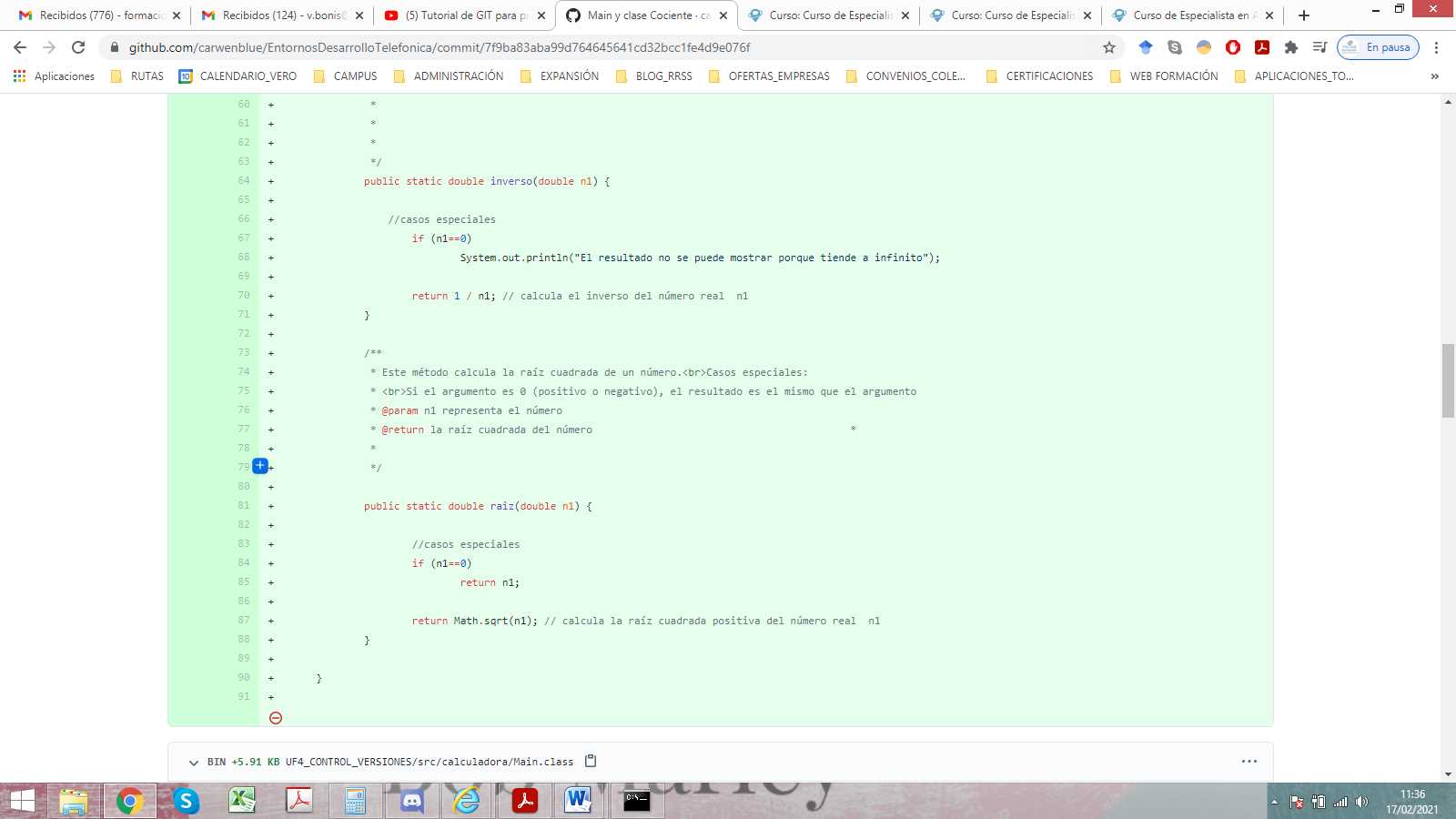
* Método 3 – Inverso

Se implementan un caso especial (si n1 es 0) y se elimina el caso en el que el números es NaN



* Método 4 - Raíz

Se implementan un caso especial (si el argumento es 0) y se elimina el caso en el que el argumento es NaN



Una vez tenemos la versión final ( ) del Main y de cada una de las clases generamos una nueva versión del JavaDoc que también se incluye en el repositorio remoto: