**Entornos Actividad 1**

**Trabajando con GIT**

* **GRUPO 11 :** 
  + Gerard Perujo Buxeda
  + Maria Isabel Martin Simal
  + Noelia Villahermosa Garcia
* **URL REPOSITORIO:**
  + <https://github.com/Gerard-Perujo/Repo1_Actividad1.git>
* El principal problema que hemos tenido ha sido con poder ver los archivos que había cargado el otro compañero en github, al hacer el pull no se actualizaba en eclipse.

Después de estar horas y horas probando y viendo donde teníamos el problema hemos encontrado el problema.

* + **El gitignore**: lo teníamos mal echo y no nos dejaba ver los cambios en eclipse aunque tuviéramos las carpetas de los compañeros, cuando creábamos el repositorio en github creando un readme y un gitignore desde github siempre nos daba problemas.

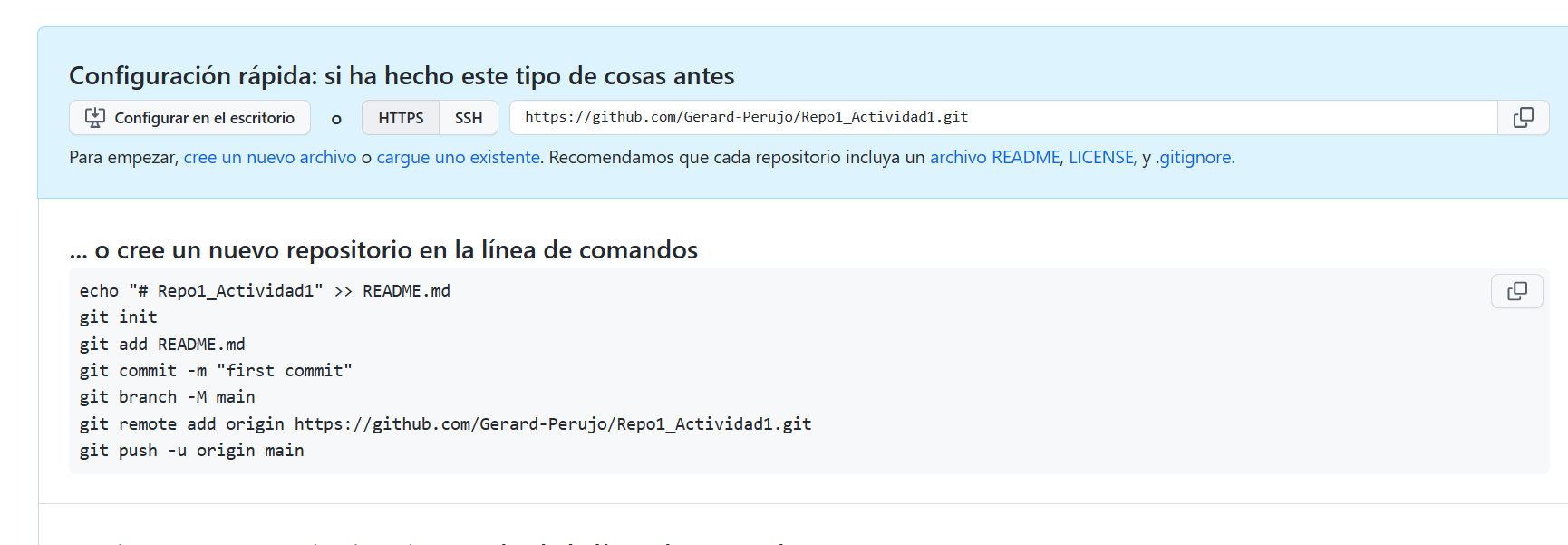
Al final decidimos crear un repositorio vacío solo con el readme y crear el repositorio a través de la línea de comandos.

El archivo gitignore lo hemos creado nosotros manualmente dentro de la carpeta.

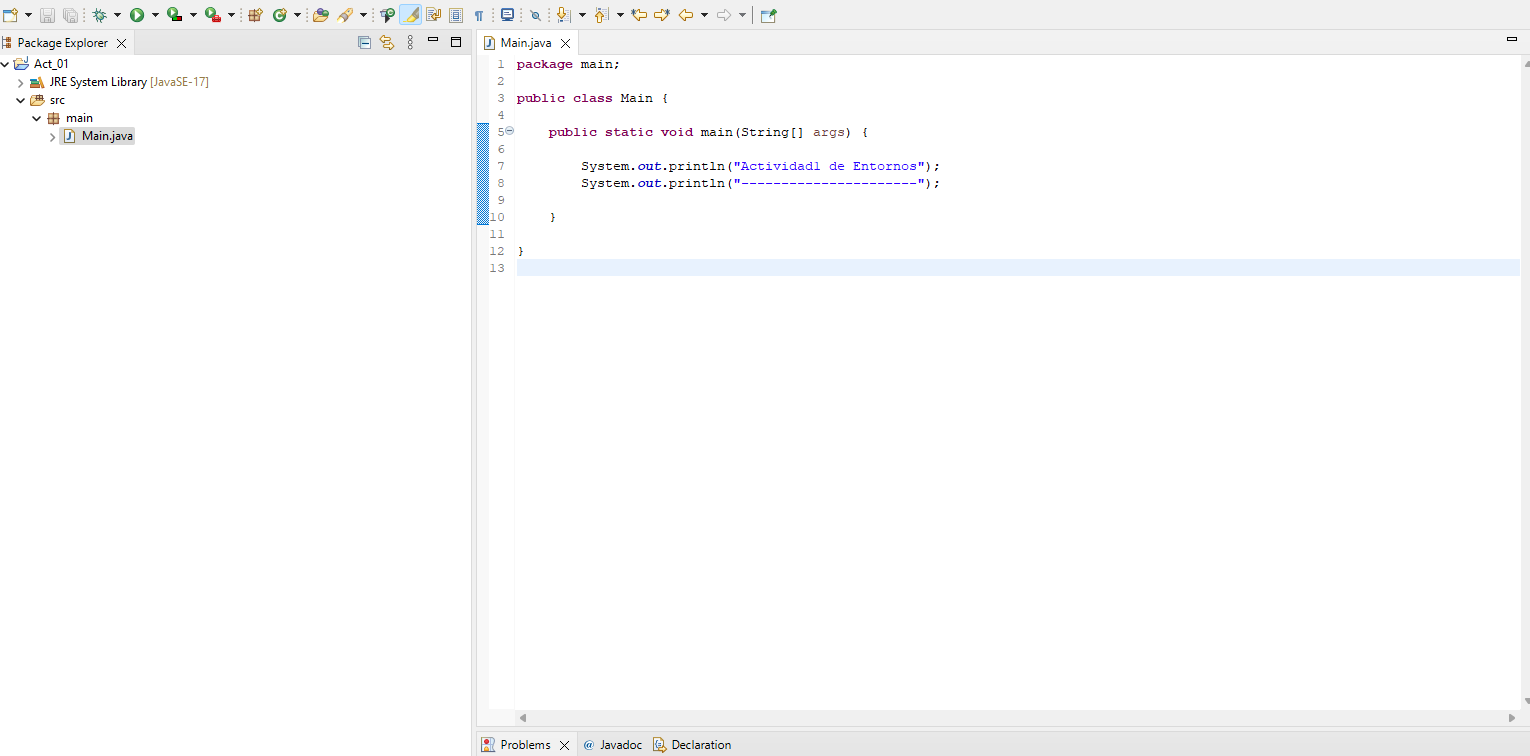
* GERARD:

Requerimiento 1

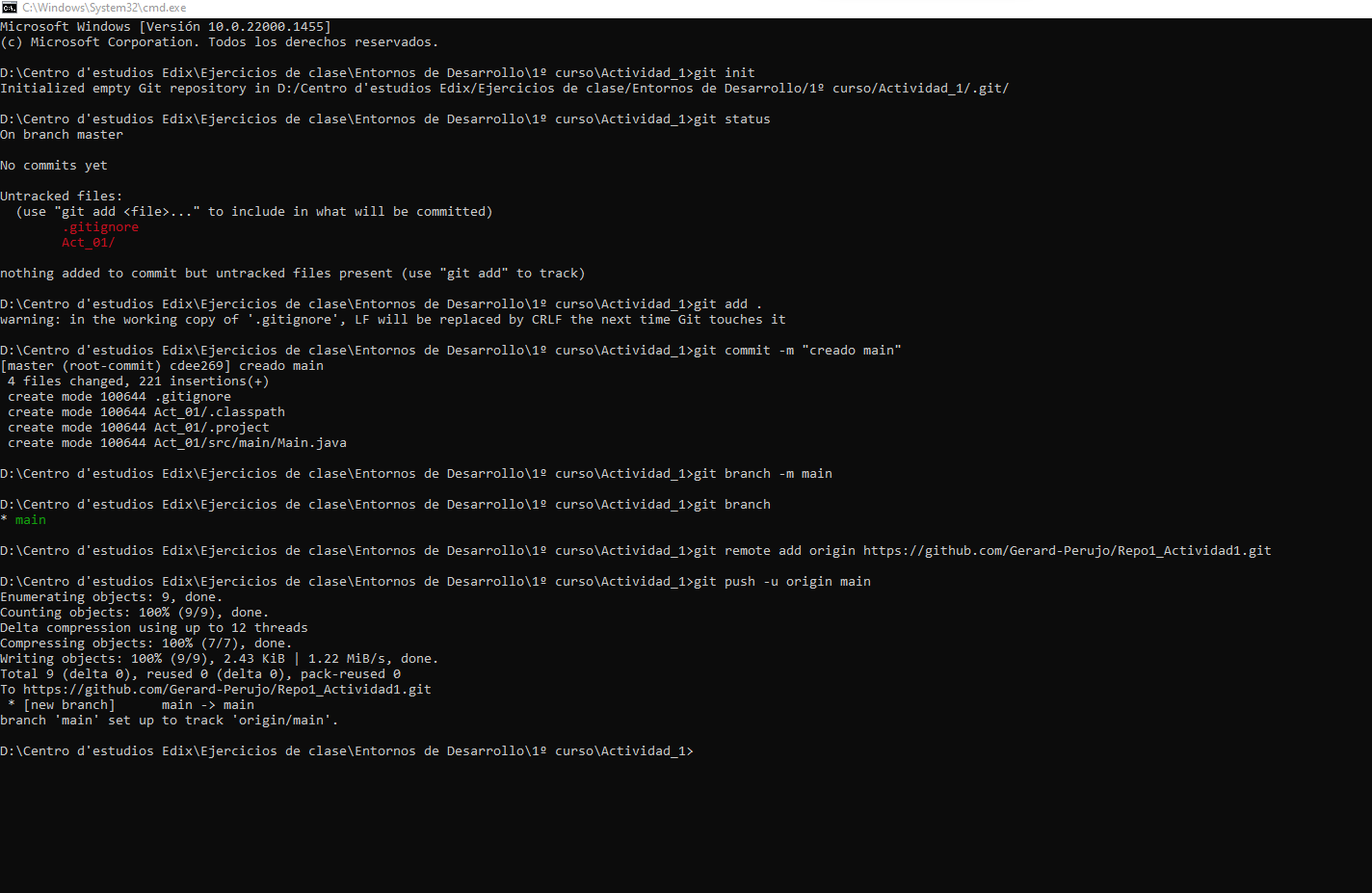
* + Creo el repositorio en mi GitHub



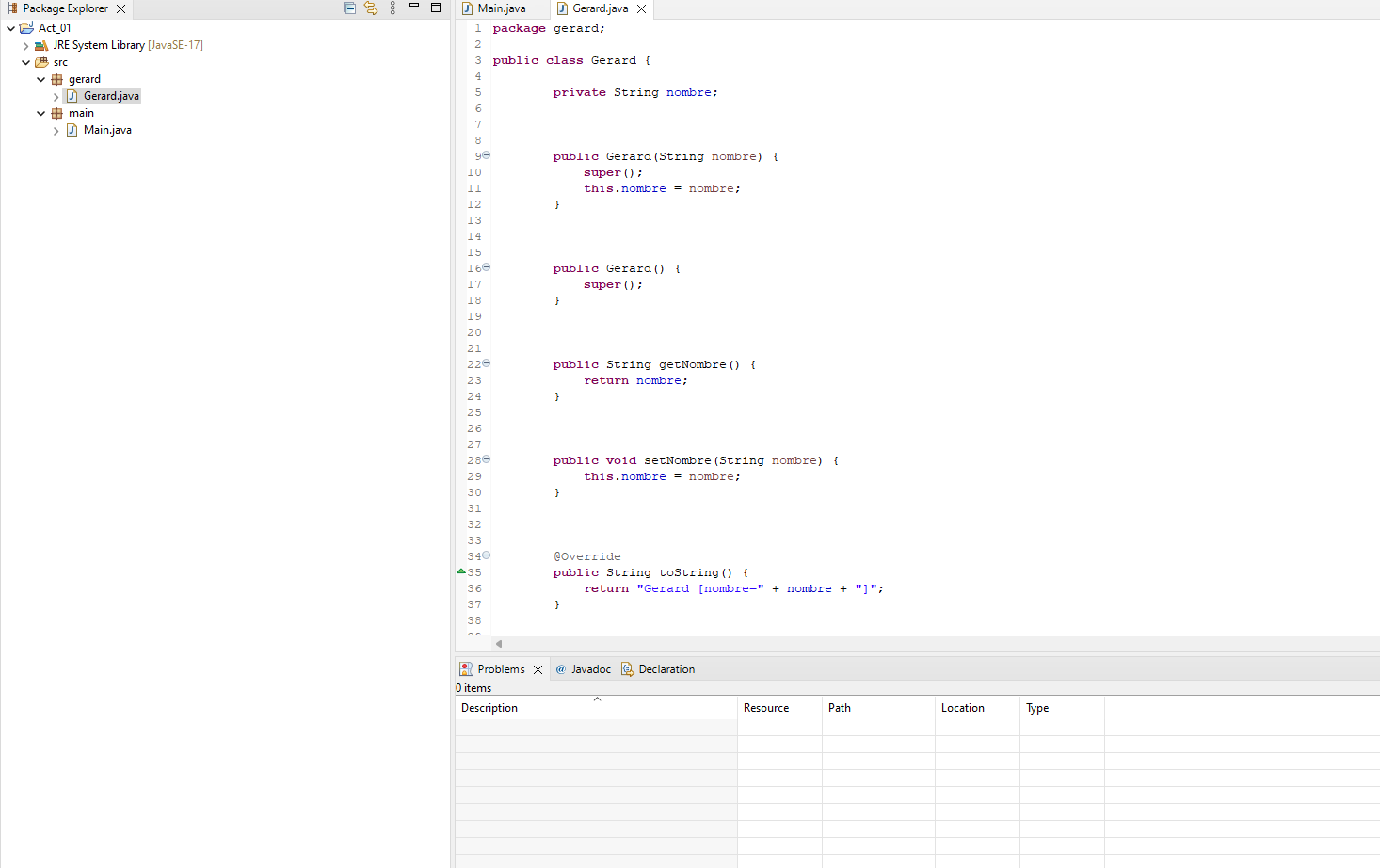
* + Creo proyecto Java en eclipse con la Clase main



* + Creo el repositorio en la línea de comandos y subo la clase main al repositorio estando en la rama main

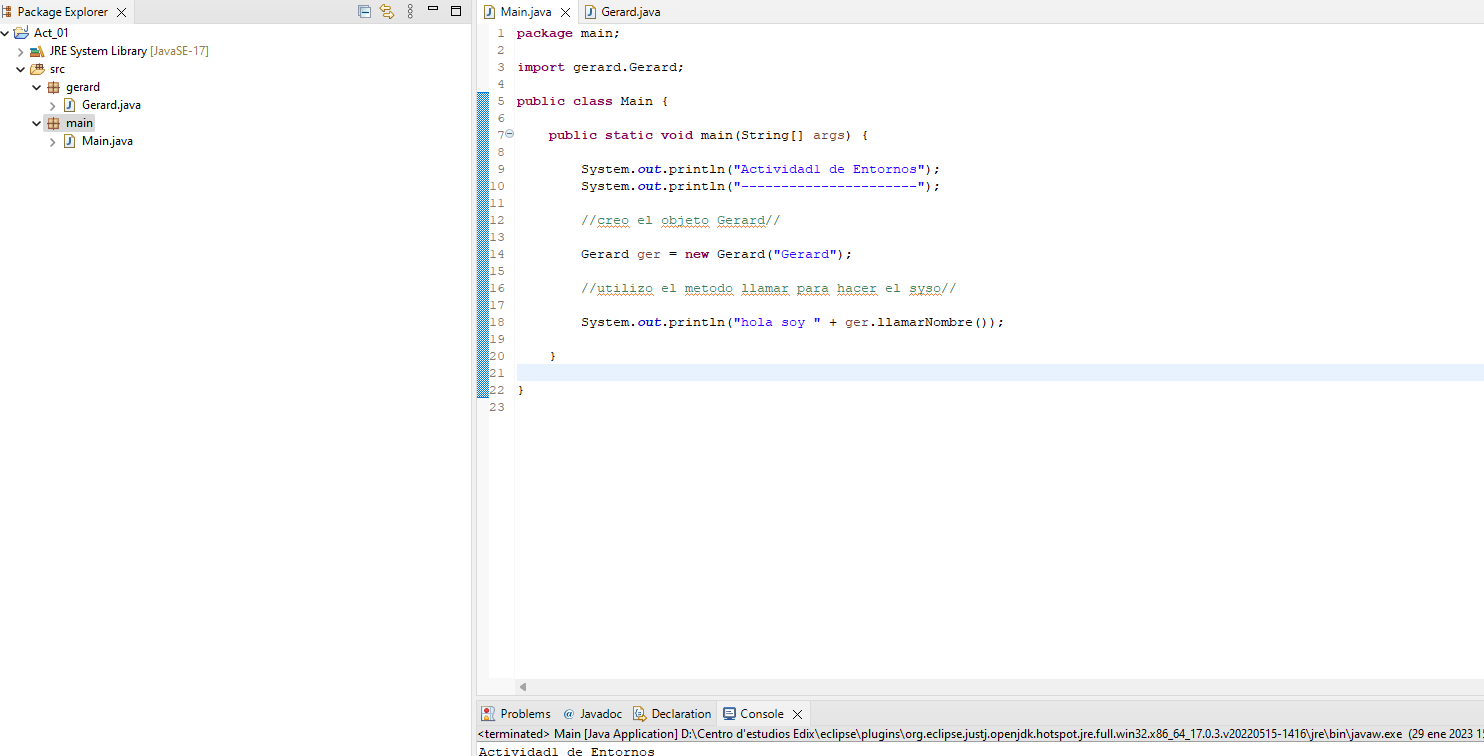


* + Creo la rama develop en git y en eclipse creo un paquete y una clase llamado Gerard

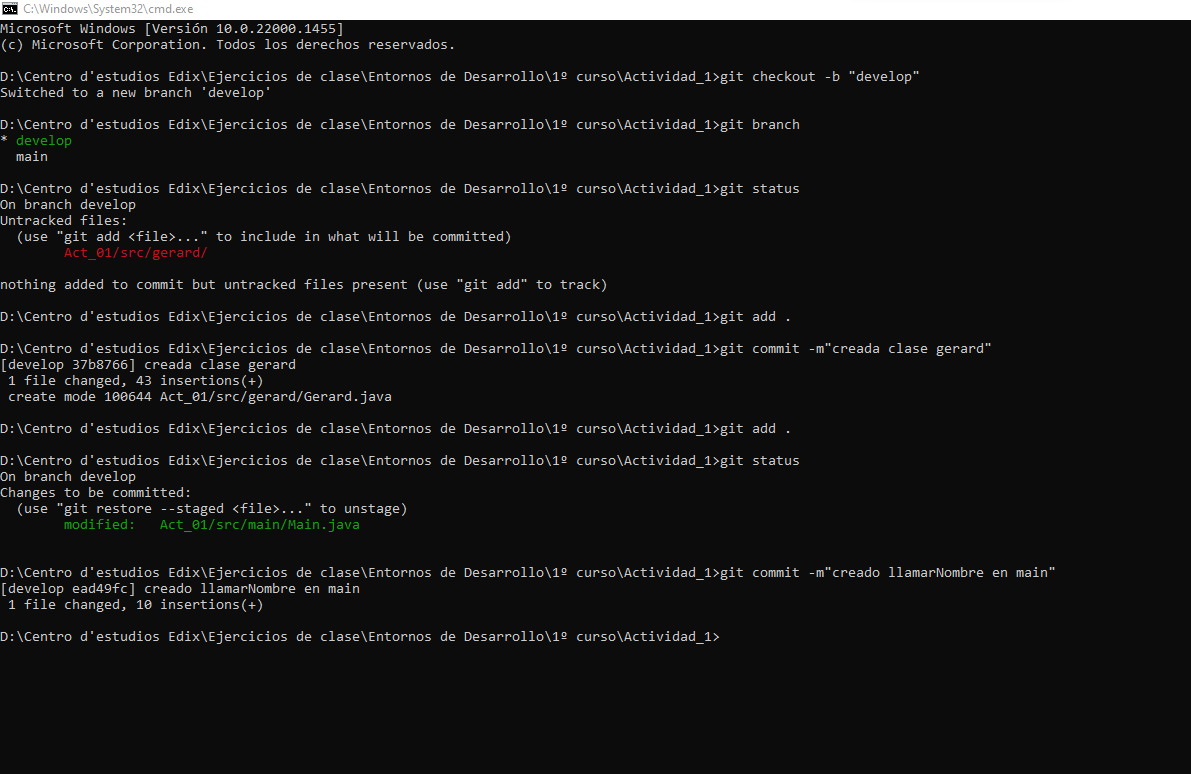


* + Una vez creado el paquete y la clase Gerard con el método llamar nombre

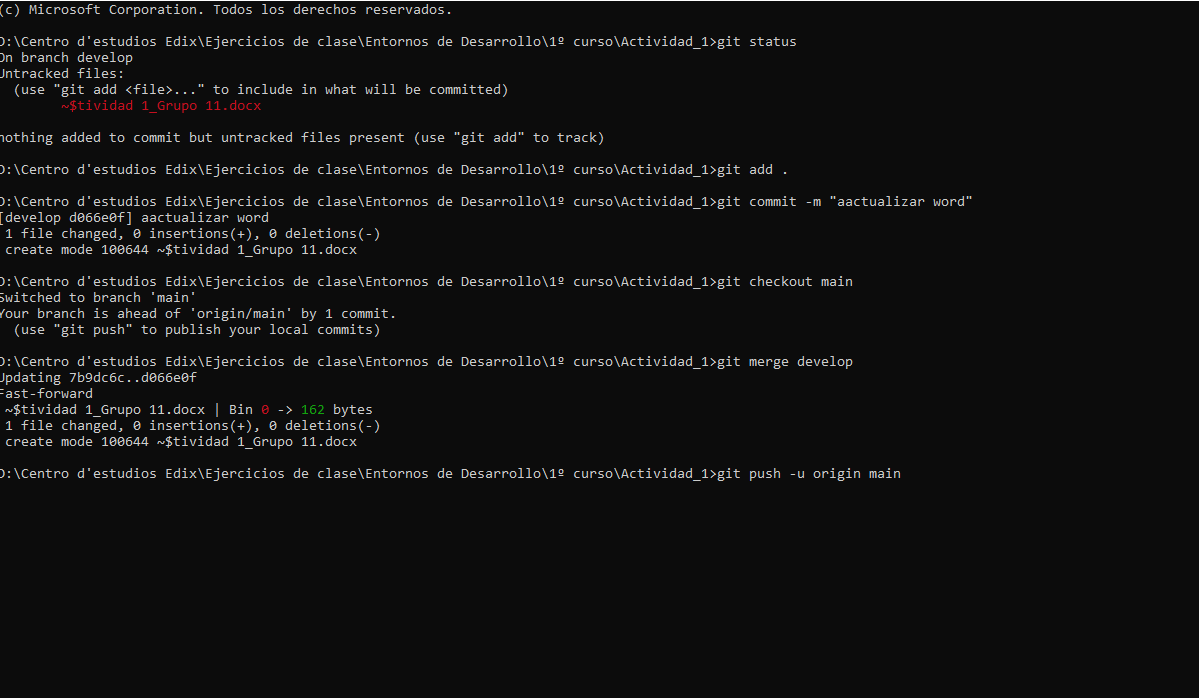
En la clase main creo un objeto ger y utilizo el método llamarNombre(), para hacer el paso 3



* Una vez creado todo tanto la clase Gerard y llamar el método llamarNombre en la clase main, hago un commit en mi rama develop, me cambio a la rama main, para poder hacer el merge con la rama main y la develop y lo subo al repositorio remoto



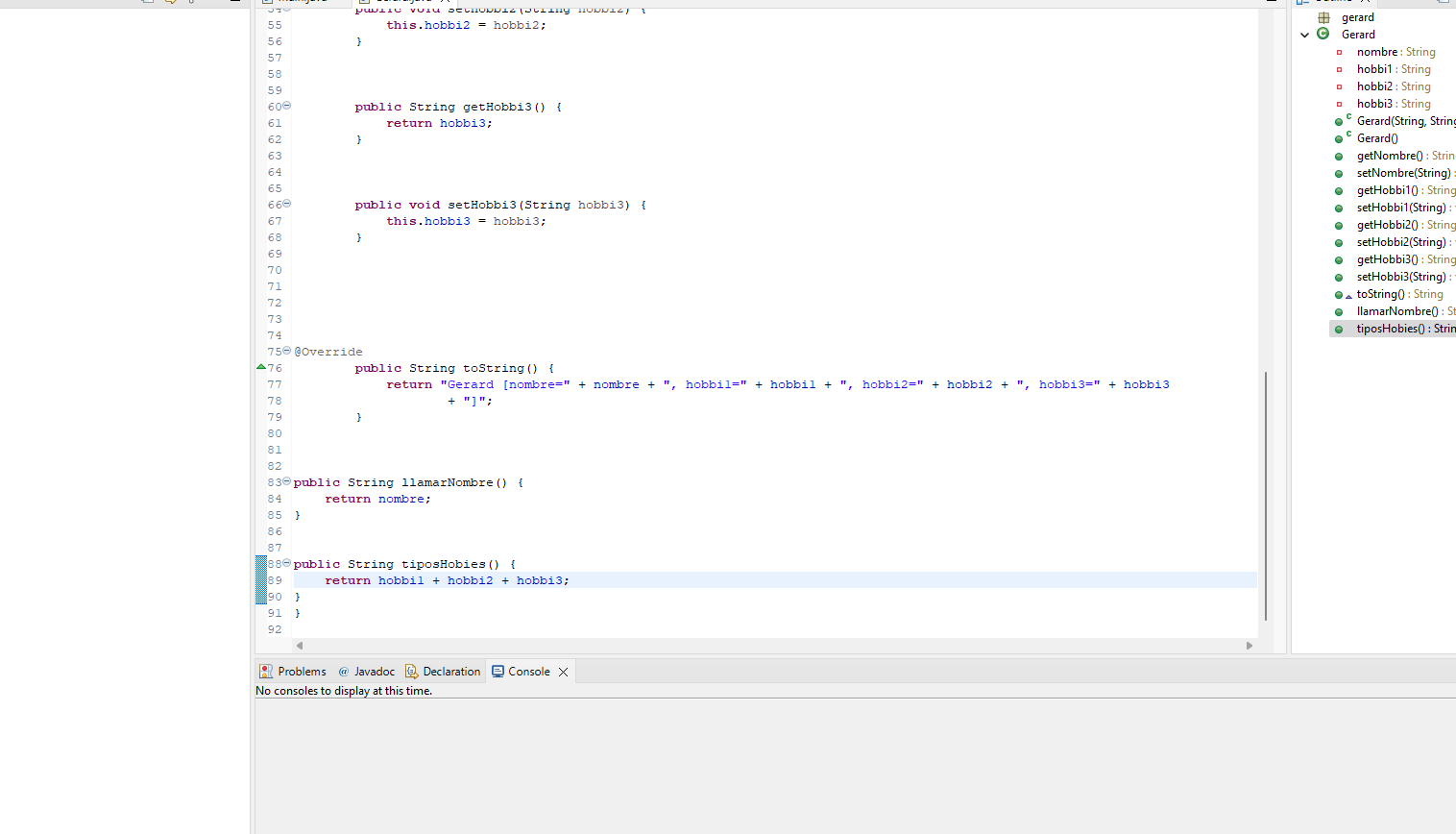
* Una vez que todos los compañeros han hecho push hago un pull para tener los datos de todos.



Requerimiento 2:

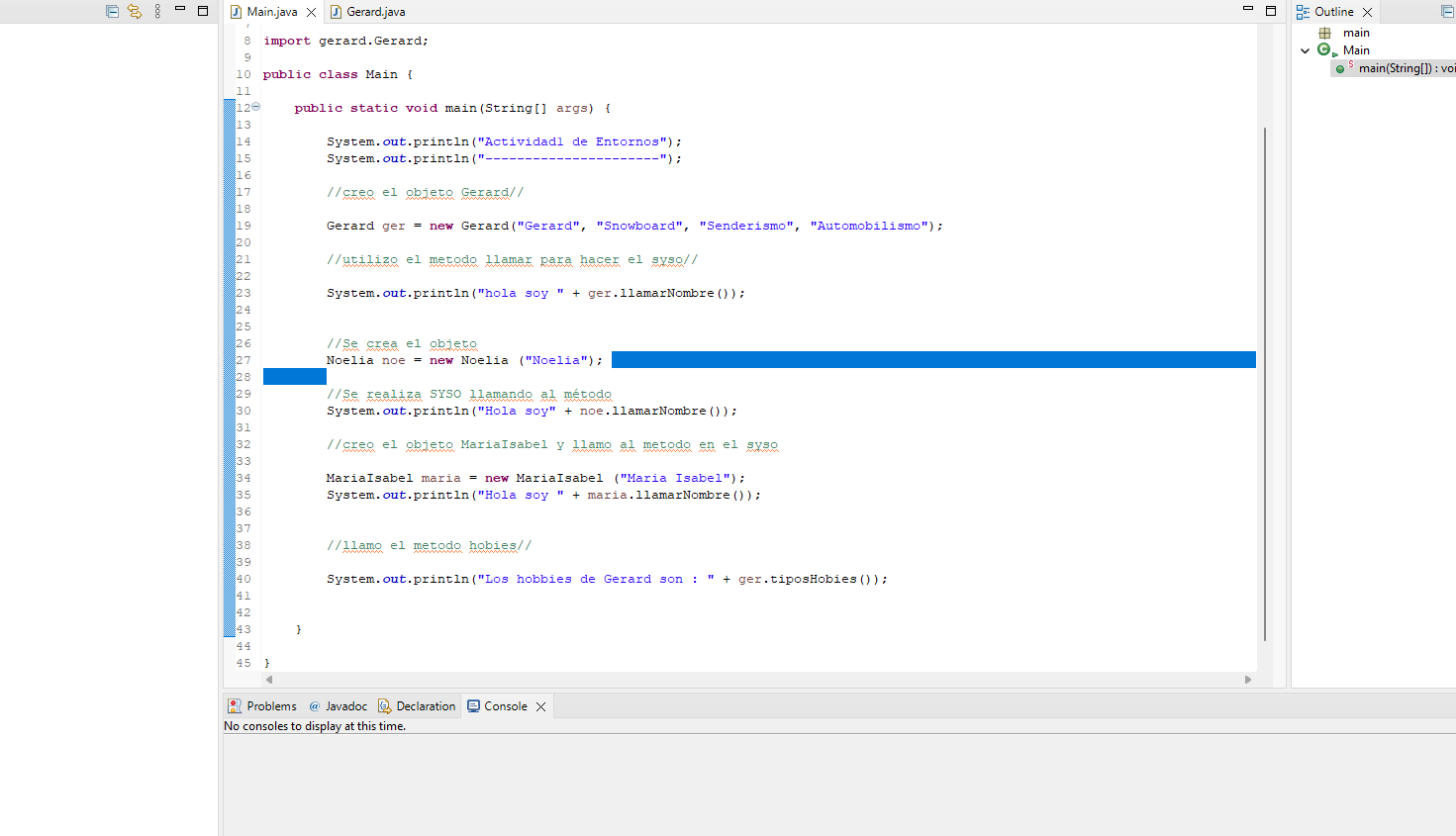
* Dentro de la clase Gerard creo 3 variables de tipo hobie y creo un método

tiposDeHobies() para llamarlo desde el main

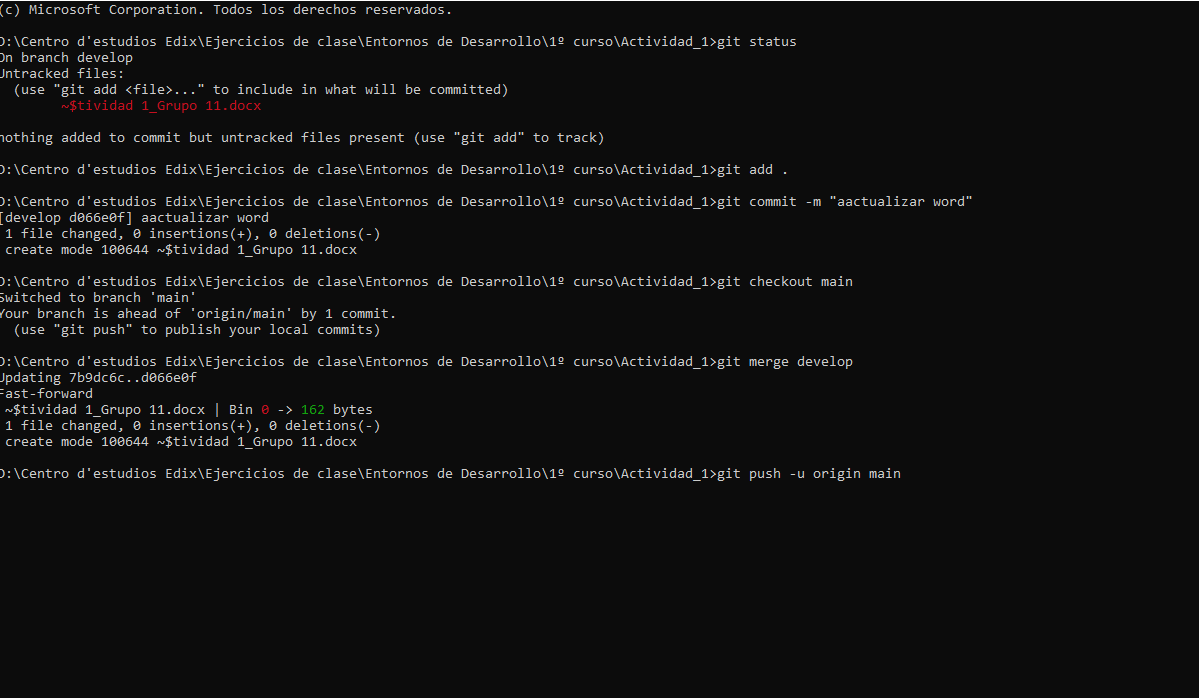


* En el main pongo los hobies dentro del objeto Gerard y hago un syso para imprimir

Los hobbies

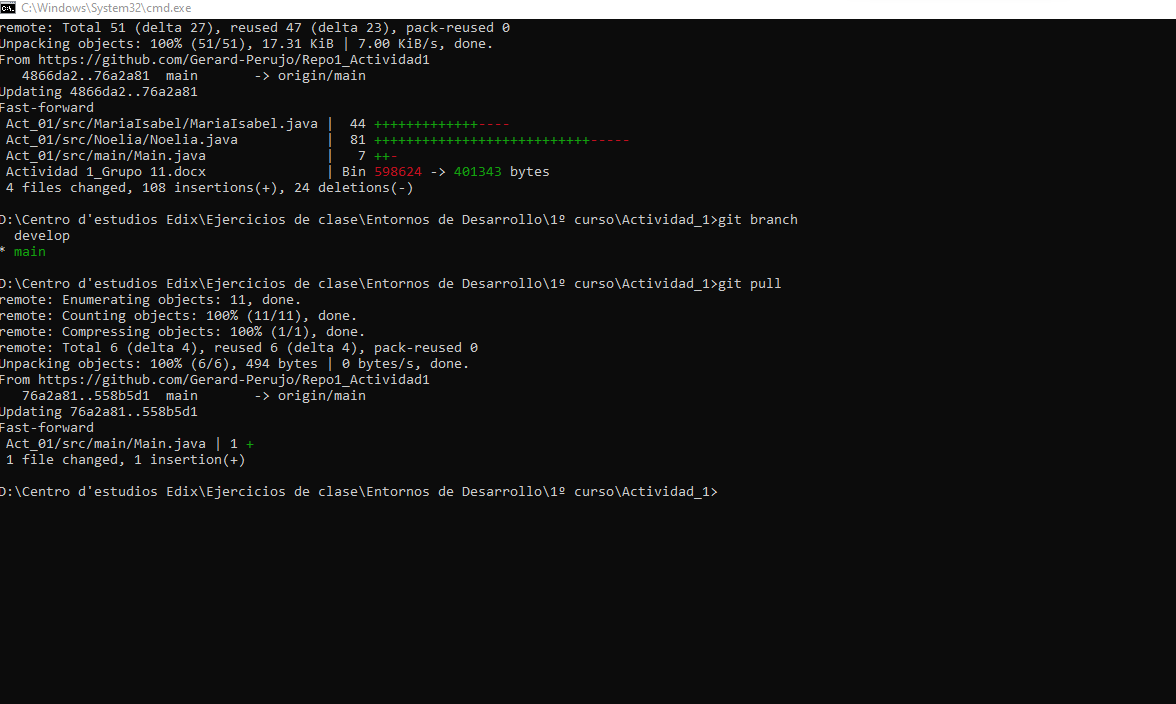


* Una vez terminado hago un push para subirlo al repositorio

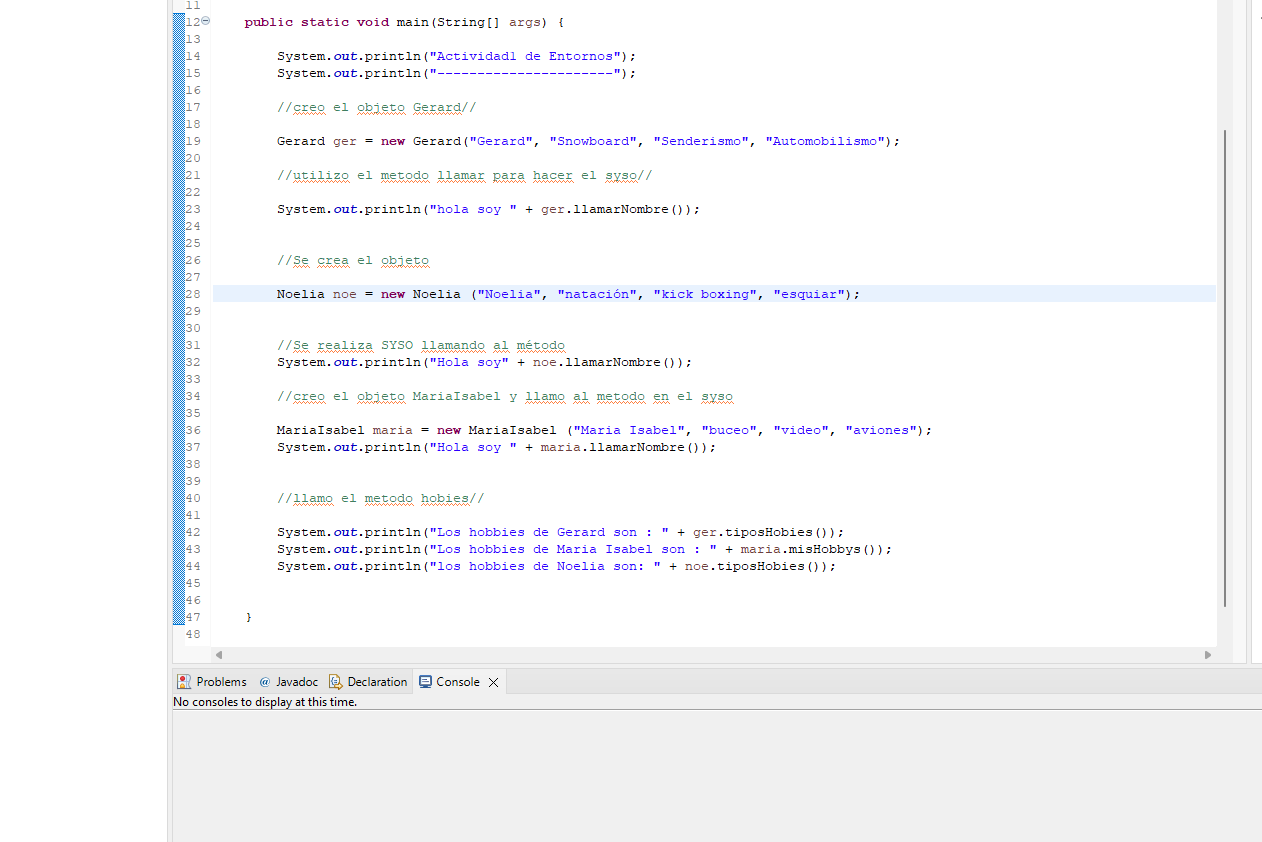


* Una vez todos los compañeros han subido sus datos al repositorio hago un pull

Para descargarme y tener los datos de todos



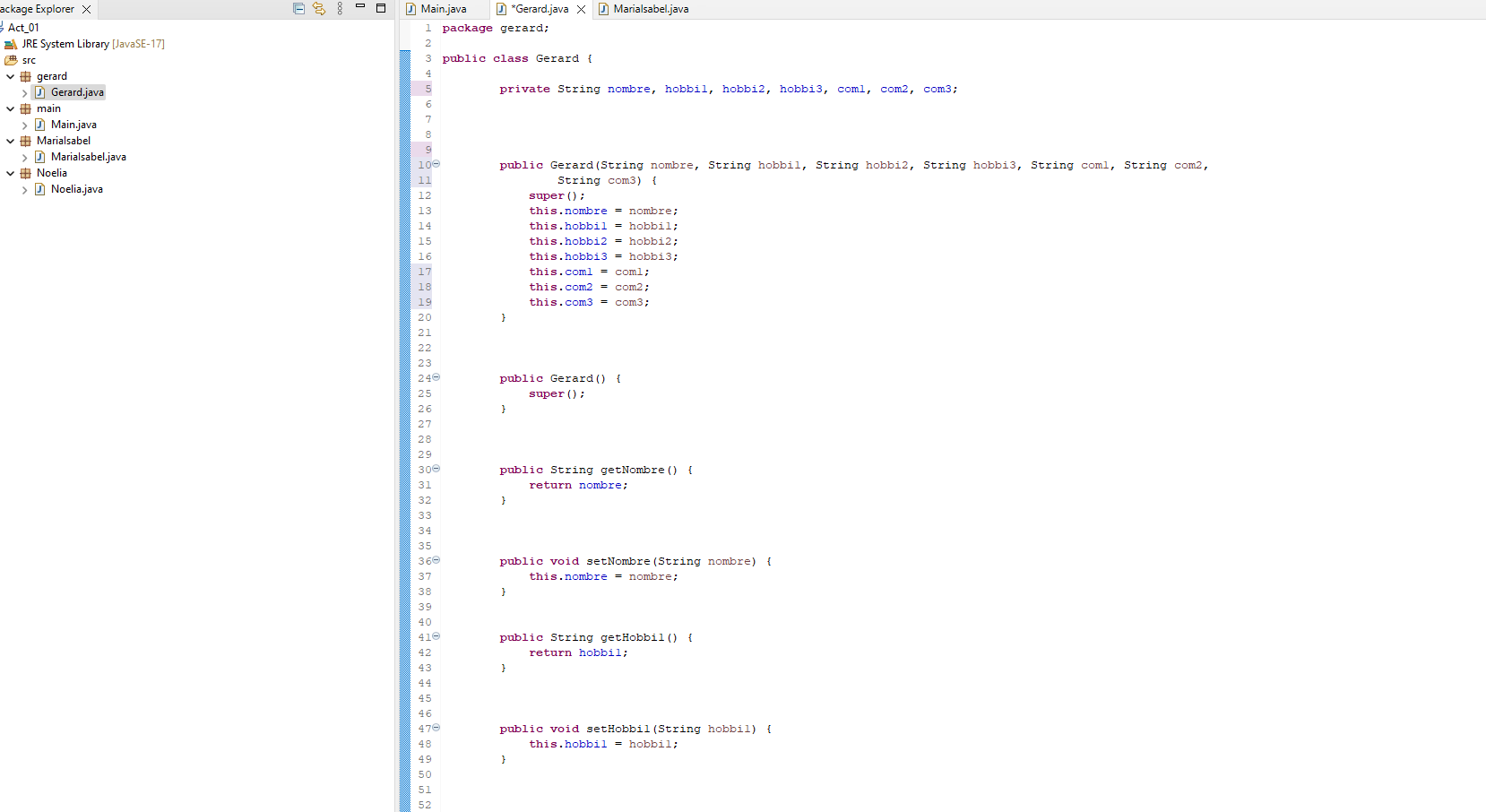
* Aquí se ven los datos de todos los compañeros una vez echo el pull



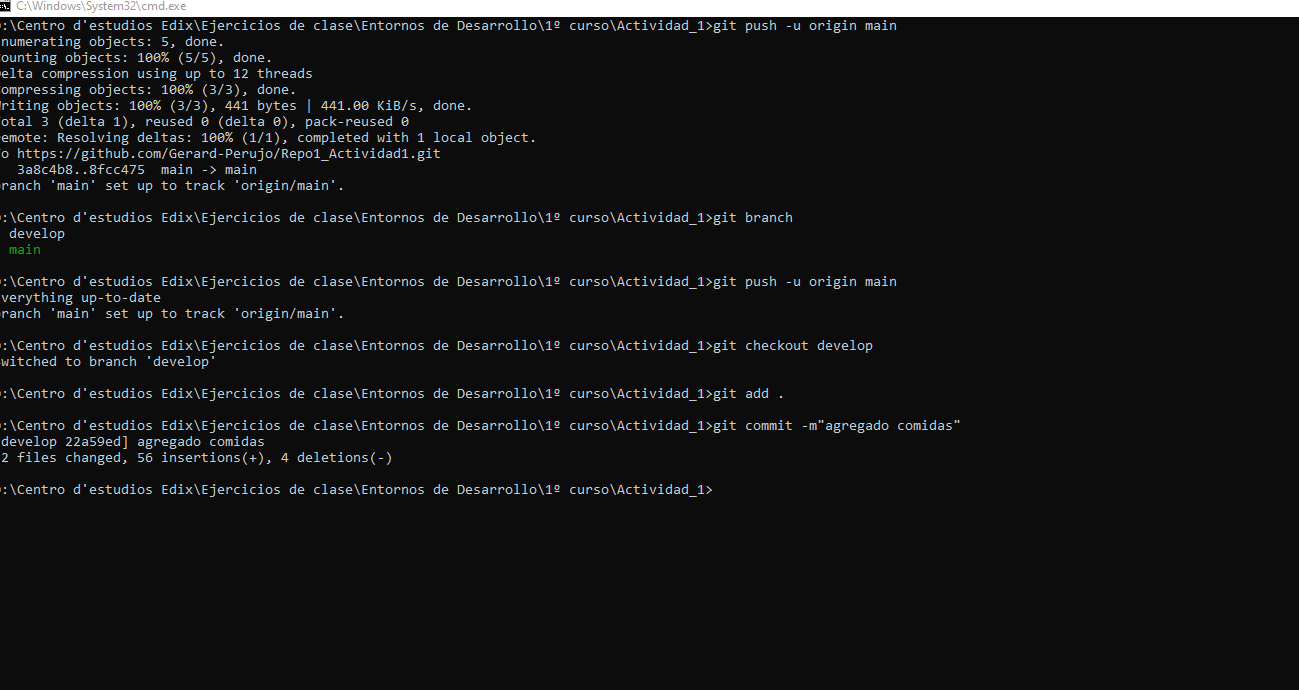
Requerimiento 3:

* Dentro de mi clase Gerard hago cambios y pongo 3 comidas mientras Maribel hace

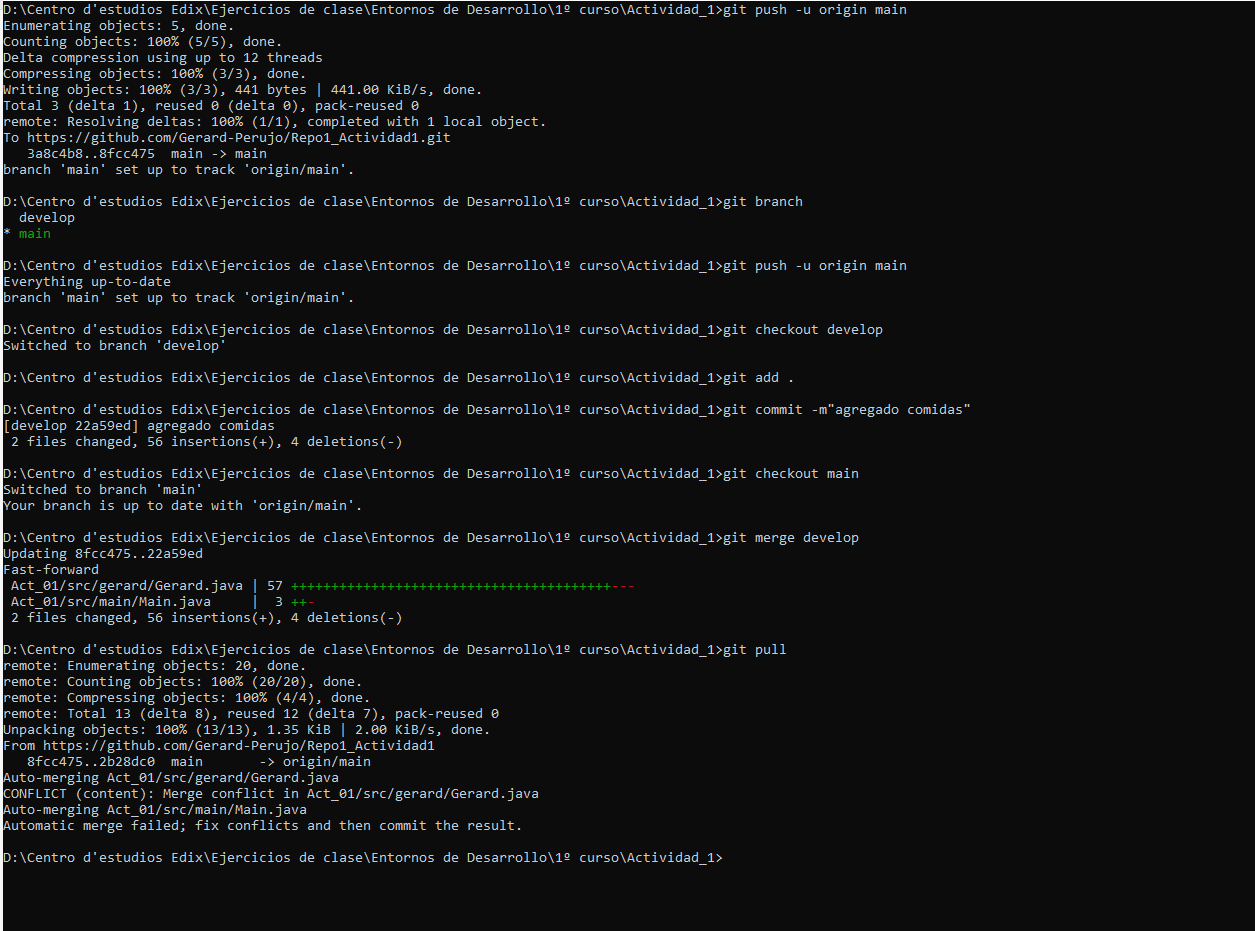
Cambios dentro de mi clase Gerard poniendo otros campos diferentes



* Una vez echos los cambios en mi rama develop hago commit en mi rama



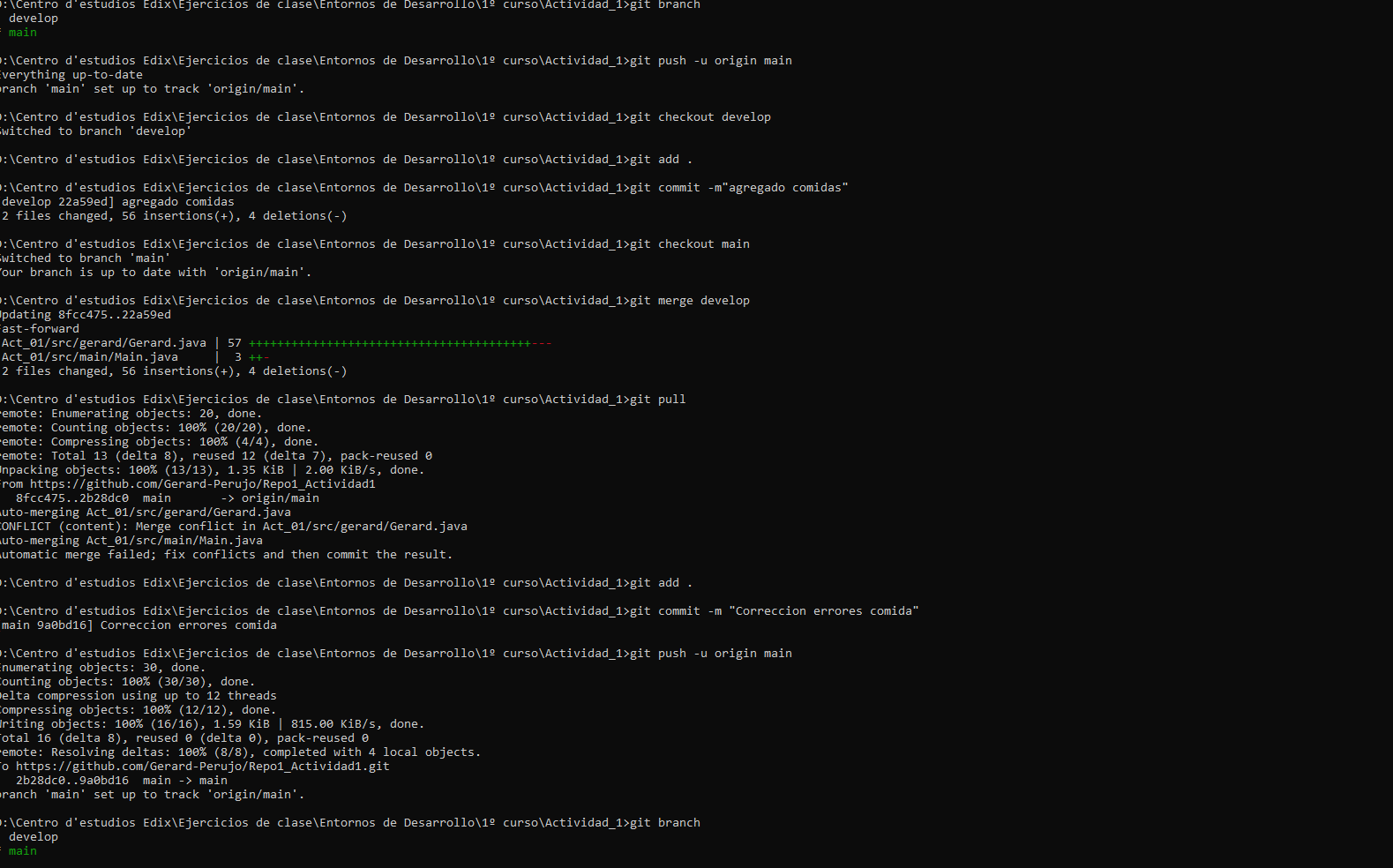
* Una vez Maribel ha terminado los cambios y los sube al repositorio hago un pull desde la rama main para actualizar los cambios y me salen errores



* Voy al eclipse y reparo los errores que me han salido al hacer pull



* Una vez reparado todos los errores vuelvo hacer push y lo envio al repositorio



NOELIA

REQUERIMIENTO 1

1. Clonar el repositorio en la máquina local

Texto

Descripción generada automáticamente

1. Crear la rama DEVELOP.

Texto

Descripción generada automáticamente

1. Texto

   Descripción generada automáticamenteEn la rama DEVELOP, creo un nuevo paquete con mi nombre y dentro de él, creo una clase de Java con mi nombre.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

1. Escribo un método en mi clase que imprima mi nombre en la consola y este método es llamado dentro del método MAIN.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

1. Hago un GIT STATUS para ver que los cambios todavía no están COMMITEADOS.
2. Se realiza un COMMIT de los cambios en la rama DEVELOP.
3. Y vuelvo a realizar un GIT STATUS para ver que se ha versionado el fichero.

Texto

Descripción generada automáticamente

1. Mis compañeros deben hacer PULL en la rama MASTER y asegurarse de que pueden ver los archivos que ha creado el resto de los compañeros en el paso 8.

* Realizo un GIT STATUS para ver si hay cambios pendientes.
* GIT BRANCH para saber en qué rama me encuentro y pasar a la rama MAIN, para pasar a la rama MAIN se hace un CHECKOUT MAIN.
* Hago GIT STATUS para ver los cambios.
* Se realiza un GIT LOG -ALL para saber que el puntero HEAD del COMMIT donde me posiciono se encuentra apuntando a la rama MAIN
* Y por último se realiza GIT PULL para bajar todos los cambios que han realizado mis compañeros

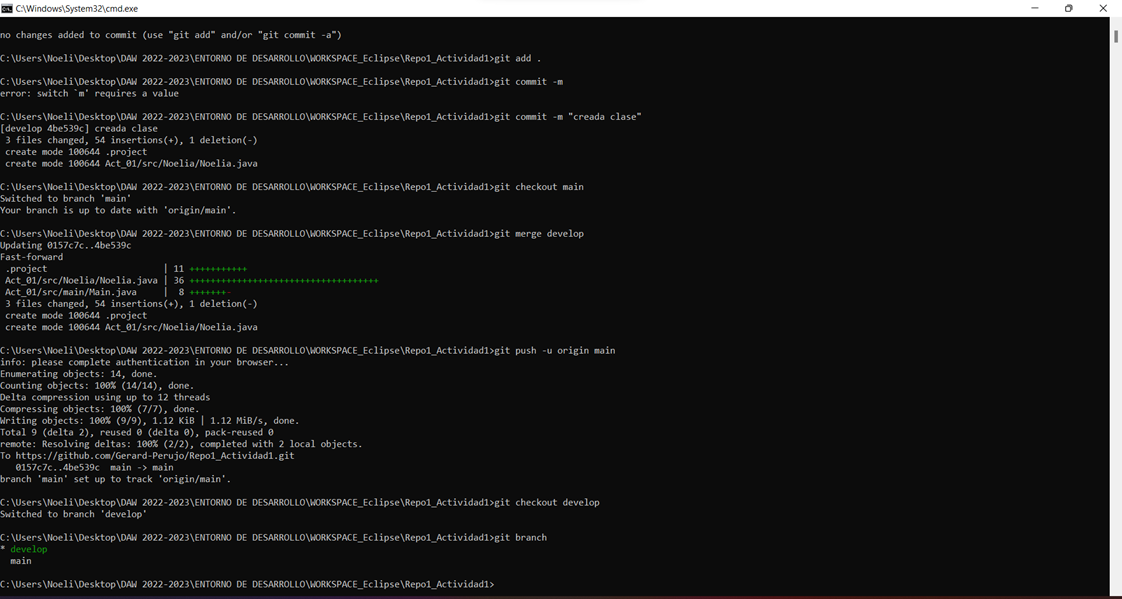
**Texto

Descripción generada automáticamente**

**Texto

Descripción generada automáticamente**

REQUERIMIENTO 2

1. Cada compañero debe pasar los cambios de la rama principal a la su rama DEVELOP para ello debe de hacer un MERGE de la rama principal MASTER a su rama DEVELOP. Ahora en su rama “DEVELOP deberá de poder ver todos los cambios que había en la rama principal ya que ambas ramas apuntan al mismo COMMIT.
2. cambiar su código para que además de imprimir su nombre, ahora también imprima 3 “hobbies” que le gustan hacer.

* Realizo un CHECKOUT DEVELOP para cambiar a mi rama después de crear en la clase Noelia el método hobbies.
* Después se ha hecho un GIT STATUS para ver a posición en la que estaba.
* GIT CHECKOT MAIN para cambiar a la rama MASTER, un GIT ADD para pasar los archivos del WORKSPACE al STAGE.
* GIT COMMIT -m "hobbies" para actualizar los cambios hechos y GIT PUSH para subirlo al repositorio remoto.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

REQUERIMIENTO 3

1. Cada integrante debe pasar los cambios de la rama principal a la su rama “develop” para ello debe de hacer un **merge** de la rama principal “master” a su rama “develop”. Ahora en su rama “develop” deberá de poder ver todos los cambios que había en la rama principal ya que ambas ramas apuntan al mismo commit.
2. 2
3. 3
4. 4

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

RESOLUCION DEL CONFLICTO

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Requerimiento 4:

* **Pros:**  Para trabajar en grupo es ideal porque te facilita muchísimo la labor de compartir archivos con tus compañeros de trabajo, al mismo tiempo también sirve como copia de seguridad si en algún momento pierdes el proyecto en tu ordenador.

También el poder ir para atrás y recuperar una versión antigua en vez de tener que escribirlo tu a mano.

¡¡¡Es gratis!!!

Ayuda mucho poder poner descripciones en los commits para saber a qué hace referencia.

Gracias a la herramienta Gitk te puedes situar.

* **Contras:** hasta que no le cojes el tranquillo es lioso, enrevesado.

Si no hay mucha comunicación entre los compañeros, con los pulls se generan errores.

Hay que hacer muchos pasos para todo, que el programa podría ser más simple.

Es un poco lioso los merge que a veces no sabes en que punto estas.

Como no sabemos hacerlo funcionar bien, necesitamos mucho tiempo en cada paso.

* En definitiva, la experiencia ha sido nefasta, esperemos que al ir trabajando con ella le cojamos el tranquillo y podamos sacar el rendimiento y rapidez a la hora de hacer commits y subir archivos al repositorio remoto.