TAREA ED04
ANTONIO JIMENEZ SEVILLA

1. Desarrolla el programa que sigue, que se llama en principio, PiedraPapelTijeraBrain (El código está como anexo en esta misma tarea). Intenta entenderlo y escribe el código JavaDoc que explique cada uno de los métodos que hay en el mismo, incrustándolo dentro del código (3 puntos).

Lo primero que hemos realizado es observar el código para realizar la documentación necesaria requerida en la tarea. He analizado los métodos, los bucles, las condiciones etc… para entender como está estructurado el programa. La mayoría de los IDEs utilizan javadoc para generar de forma automática documentación de clases.Veamos en primer lugar qué se debe incluir al documentar una clase:

a) **Nombre de la clase, descripción general, número de versión, nombre de autores**.

b) **Documentación de cada constructor o método** (especialmente los públicos) incluyendo: nombre del constructor o método, tipo de retorno, nombres y tipos de parámetros si los hay, descripción general, descripción de parámetros (si los hay), descripción del valor que devuelve.

Para generar la documentación de un proyecto automáticamente hemos de seguir unas normas a la hora de realizar los comentarios dentro del mismo.

- Los comentarios JavaDoc deben empezar por /** y terminar por */.
- Los comentarios pueden ser a nivel de clase, a nivel de variable y a nivel de método.
- La documentación se genera para métodos public y protected.
- Se puede usar tag para documentar diferentes aspectos determinados del código, como parámetros.

Hemos realizado la siguiente documentación del proyecto **PiedraPapelTijeraBrain.**

Como he indicado en el apartado a) anteriormente he indicado: **Nombre de la clase, descripción general, nombre del autor y versión.**

```java
package piedrapapeltijerabrain;

import java.util.Scanner;

/**
 * PiedraPepelTijeraBrain clase principal
 * Esta clase simula el juego piedra papel tijera
 * @author Antonio Jimenez Sevilla
 * @version 11/03/2021
 *
 */
public class PiedraPapelTijeraBrain {
```

A continuación hemos explicado el método main, y hemos puesto en cada variable y atributo a que corresponden. Las variables de instancia o de clase no se suelen documentar a nivel de javadoc, pero lo he realizado así para entender mejor el programa.

```java
    /**
     * @param args argumentos del programa
     *
     */
    public static void main(String[] args) {
        // TODO code application logic here
        Scanner sc = new Scanner(System.in);//entrada de teclado
        /**
        * numero de chiquipuntos al principio valen 0
        */
        int chiquipuntos = 0;
        /**
        * opcion del jugador
        */
        String opcionJugador = "";
        /**
        *  opción de la maquina
        */
        String opcionBot = "";//
        /**
        * si seconsigue el proposito vale 1
        */
        int exitoEnProposito = 1;//
        /**
        * devuelve el tiempo en que hemos tardado en milisegundos
        */
        long inicio = System.currentTimeMillis();//
```

A continuación hemos explicado el bucle for, y lo que se realiza en cada condición y función.

```
        /**
        * Bucle for para realizar 5 juagadas, se van sumando entre 1 y 5,
        * acumulando el resultado, permitiendo ir aumentado la variable i
        * de 1 en 1,
        * creamos dos variables de tipo int, una para el proposito de la
        * jugada, en la que usamos una funcion( para generar los números
        * aleatoriamente entre 1 y 2, y otra para la opcionbot que nos
        * enseña la jugada en la que también creamos otra función para crear
        * los numeros aleatorios entre 1 y 3
        */
    for (int i = 0; i < 5; i++) {
        //funcion que crea un número aleatorio en 1 y 3
        int proposito = (int) Math.floor(Math.random() * 2 + 1);
        // si equivale a 1
        if (proposito == 1) {
            //se muestra por pantalla la sentencia a realizar
            System.out.println("\n\tIntenta ganar");
        }
        // si equivale a 2
        if (proposito == 2) {
        //se muestra por pantalla la sentencia a realizar
            System.out.println("\n\tIntenta perder");
        }
        // función que crea un número aleatorio en 1 y 3
        int j = (int) Math.floor(Math.random() * 3 + 1);
        //si el numero es 1 la salida por pantalla son tijeras
        if (j == 1) {
            opcionBot = "tijera";
            System.out.println(opcionBot);
            tijera();
        }
        //si el numero es 2 la salida por pantalla es papel
        if (j == 2) {
            opcionBot = "papel";
            System.out.println(opcionBot);
            papel();
        }
        //si el numero es 3 la salida por pantalla es piedra
        if (j == 3) {
            opcionBot = "piedra";
            System.out.println(opcionBot);
            piedra();
        }
```

A continuación explicamos el uso del **do while.** Y el cierre del bucle.

```
        /**
        *Ejecutamos la instrucion, introducir la jugadada
        *
        */
        do {
            System.out.println("Introduce tu jugada");
            opcionJugador = sc.nextLine();
            if (opcionBot.equals(opcionJugador)) {
                System.out.println("No tiene sentido que intentes empatar");
            }
        /**
        *Mientras se cumplan las siguientes condiciones, se sumara o restará
        * 1 a la puntuación
        */
        } while (opcionBot == opcionJugador);
        //
        if (opcionJugador.equals("tijera") && (opcionBot.equals("papel"))) {
            exitoEnProposito = 1;
        }
        if (opcionJugador.equals("papel") && (opcionBot.equals("tijera"))) {
            exitoEnProposito = -1;
        }
        if (opcionJugador.equals("tijera") && (opcionBot.equals("piedra"))) {
            exitoEnProposito = -1;
        }
        if (opcionJugador.equals("piedra") && (opcionBot.equals("tijera"))) {
            exitoEnProposito = 1;
        }
        if (opcionJugador.equals("piedra") && (opcionBot.equals("papel"))) {
            exitoEnProposito = -1;
        }
        if (opcionJugador.equals("papel") && (opcionBot.equals("piedra"))) {
            exitoEnProposito = 1;
        }
        if (proposito == 2) {
            exitoEnProposito *= -1;
        }
        if (exitoEnProposito == 1) {
            chiquipuntos++;

        }

    }//cierre del bucle
```

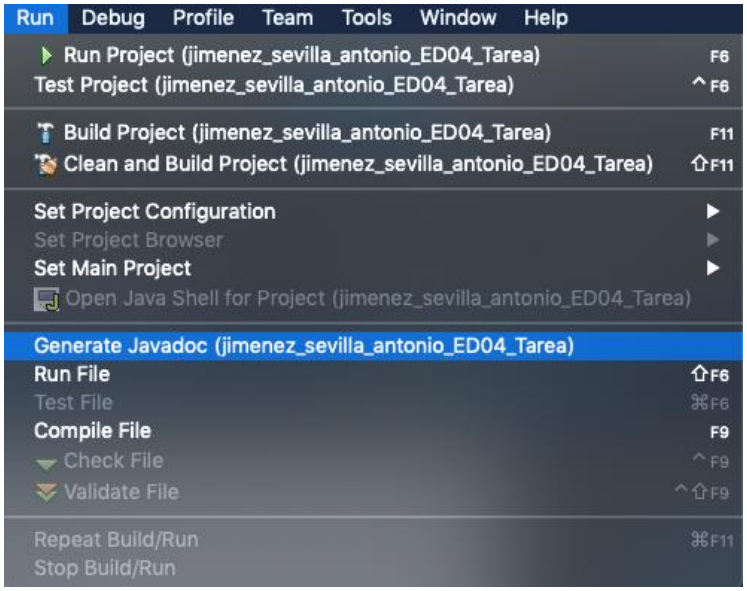
Explicamos las variables la salida por pantalla y el cierre del método.

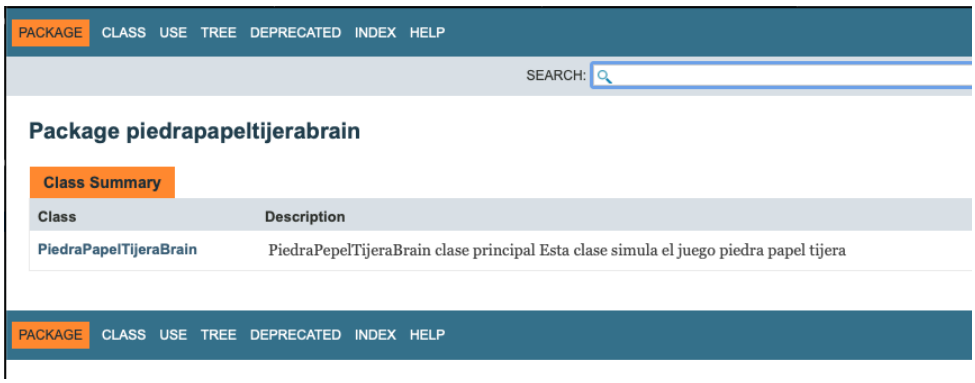
```
        }//cierre del bucle
        /**
        * devuelve el tiempo en que hemos tardado en milisegundos
        */
        long fin = System.currentTimeMillis();
        /**
        * Variable tiempo que calcula el timepo tardado desde el principio
        * hasta el final del programa en segundos
        */
        double tiempo = (double) ((fin - inicio) / 1000);
        //Salida por pantalla del tiempo que hemos tardado en ralizarlo
        System.out.println("Has realizado el ejercicio en " + tiempo + " segundos");
        //numero de fallos se restan
        int nFallos = 5 - chiquipuntos;
        //Salida por pantalla del numero de fallos
        System.out.println("Penalización: " + nFallos + " x 5s = " + nFallos * 5);
        // resta de puntos por fallos
        double tiempoFinal = tiempo + nFallos * 5;
        //salida por pantalla del tiempo final ya calculado en segundos
        System.out.println("Tu tiempo final es de " + tiempoFinal + " segundos");
    }//cierre del método
```

Se documentan los métodos y y el cierre del método y del proyecto.

```
    /**
    * Método que genera la salida por pantalla de tijera
    */
    public static void tijera() {
        System.out.println("'''\n"
                + " _______\n"
                + "---'    ____)____\n"
                + "       ______)\n"
                + "          _____)\n\""
                + "       (____)\n"
                + "---.__(___)\n"
                + "\"'''");
    }//cierre del método
    /**
    * Método que genera la salida por pantalla de papel
    */
    public static void papel() {
        System.out.println("'''\n"
                + " _______\n"
                + "--  ____)____\n "
                + "        ______)\n"
                + "       _______)\n"
                + "      _______)\n"
                + "---.__________)");

    }//cierre del método
    /**
    * Método que genera la salida por pantalla de piedra
    */
    public static void piedra() {
        System.out.println("'''\n"
                + " _______\n"
                + "---'____)\n"
                + "   (_____)\n "
                + "   (_____)\n"
                + "   (____)\n"
                + "---.__(___)\n"
                + "'''");



    }//cierre del método
}//cierre de la clase y proyecto
```

A continuación generaremos el documento JavaDoc





Class PiedraPapelTijeraBrain

java.lang.Object
piedrapapeltijerabrain.PiedraPapelTijeraBrain

```
public class PiedraPapelTijeraBrain
extends java.lang.Object
```

PiedraPepelTijeraBrain clase principal Esta clase simula el juego piedra papel tijera

Constructor Summary

Constructors

Constructor	Description
`PiedraPapelTijeraBrain()`	

Method Summary

All Methods | Static Methods | Concrete Methods

Modifier and Type	Method	Description
static void	`main(java.lang.String[] args)`	
static void	`papel()`	Método que genera la salida por pantalla de papel
static void	`piedra()`	Método que genera la salida por pantalla de piedra
static void	`tijera()`	Método que genera la salida por pantalla de tijera

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Details

PiedraPapelTijeraBrain

```
public PiedraPapelTijeraBrain()
```

Method Details

main

```
public static void main(java.lang.String[] args)
```

Parameters:
args - argumentos del programa

tijera

```
public static void tijera()
```

Método que genera la salida por pantalla de tijera

papel

```
public static void papel()
```

Método que genera la salida por pantalla de papel

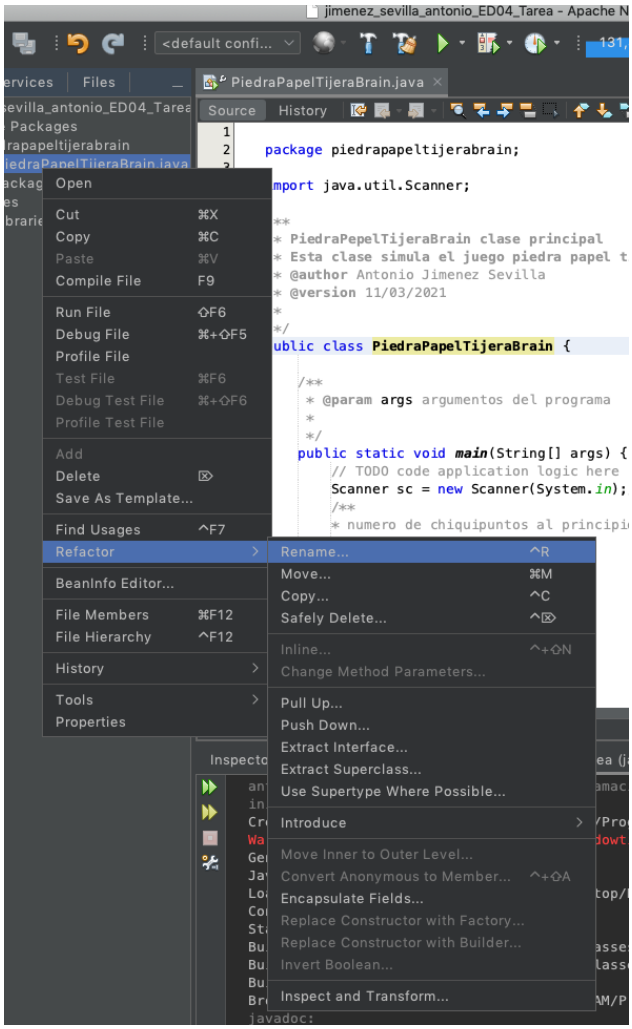
piedra

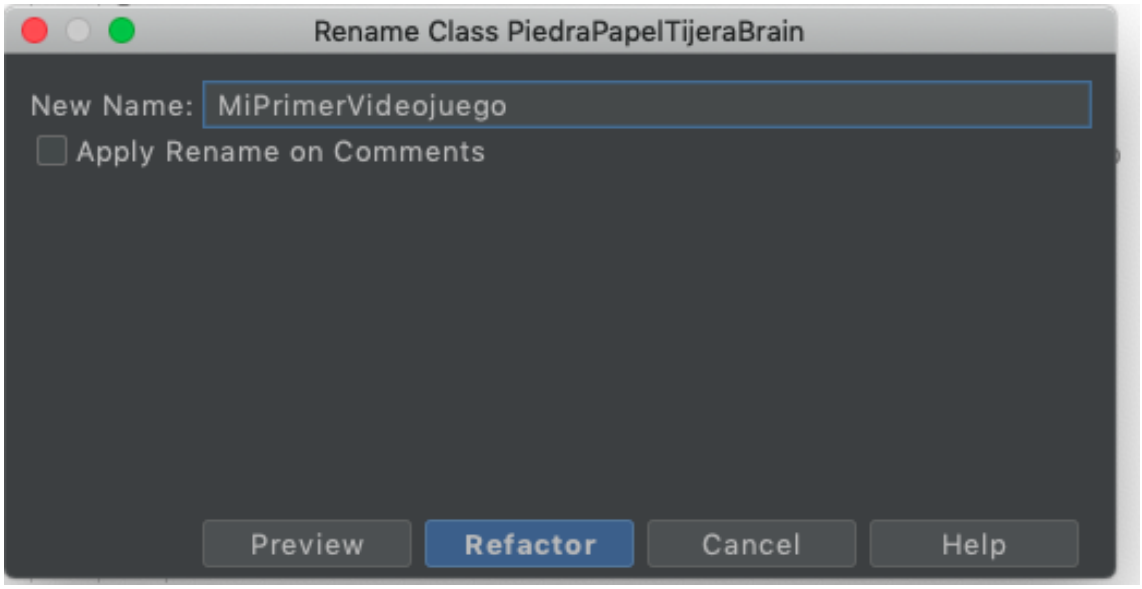
```
public static void piedra()
```

Método que genera la salida por pantalla de piedra

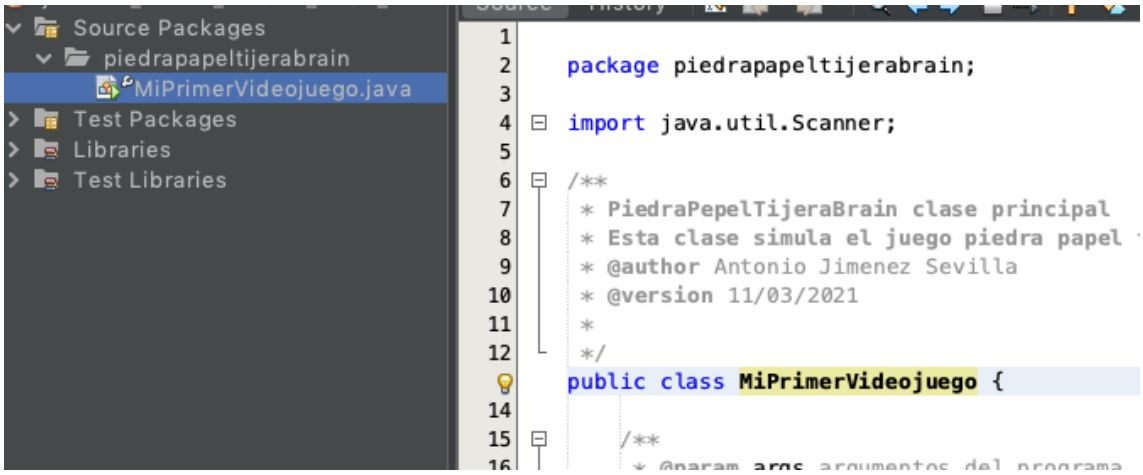
2. Refactoriza el programa para que se llame MiPrimerVideoJuego (3 puntos)

Botón derecho en el nombre del programa, **Refactor** y seleccionamos **rename.**



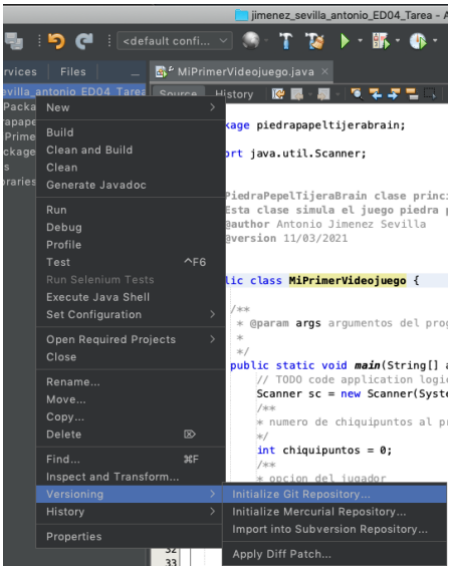


Aquí se puede comprobar el cambio de nombre.

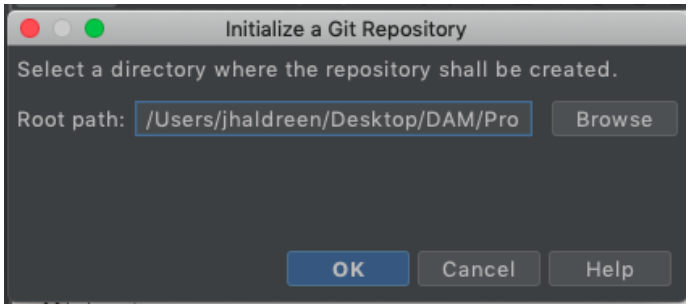


3. Activa el repositorio Git, con Botón Derecho- Versioning – Git Repository- Y después investiga cómo conseguir el siguiente resultado (4 puntos)

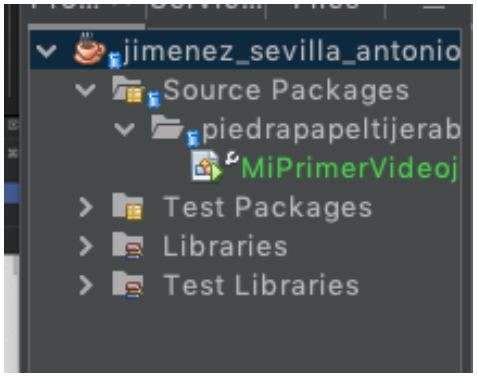
El primer paso para hacer uso del sistema de control de versiones Git en NetBeans es inicializar este ultimo, esto se logra vía el menú Versioning al cual accede presionando el botón derecho del ratón en el proyecto.



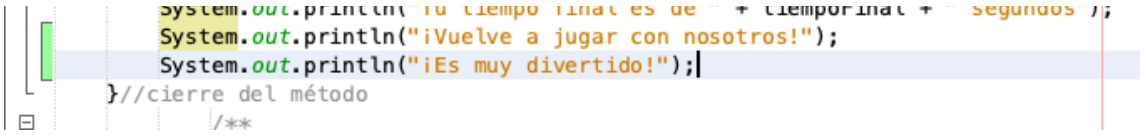
Con esto crearemos un repositorio Git para nuestro proyecto, de ese modo todos los cambios que hagamos podrán guardarse en una forma ordenada y ocupando menos espacio que si hiciéramos una copia del proyecto a cada cambio, esto requiere que indiquemos donde deseamos se cree la carpeta del repositorio, como se ve a continuación.



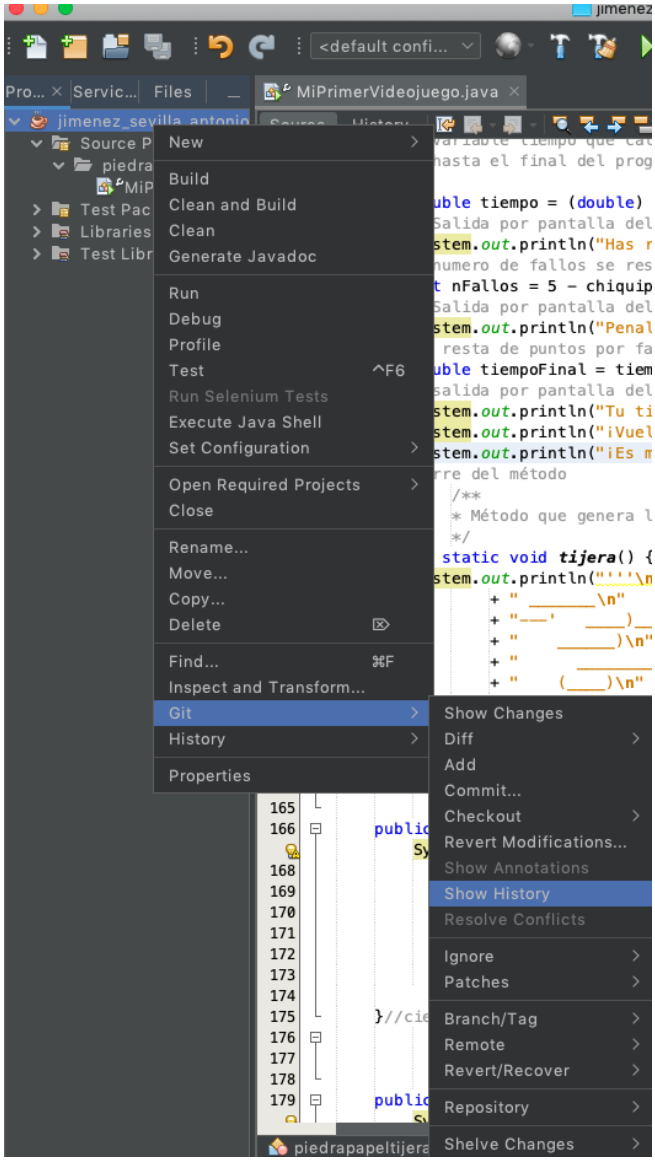
Vemos que ha cambiado de color al activar el GIT. Esta es la forma en que la interfaz de NetBeans le indica que esos archivos se han creado desde la ultima versión guardada.



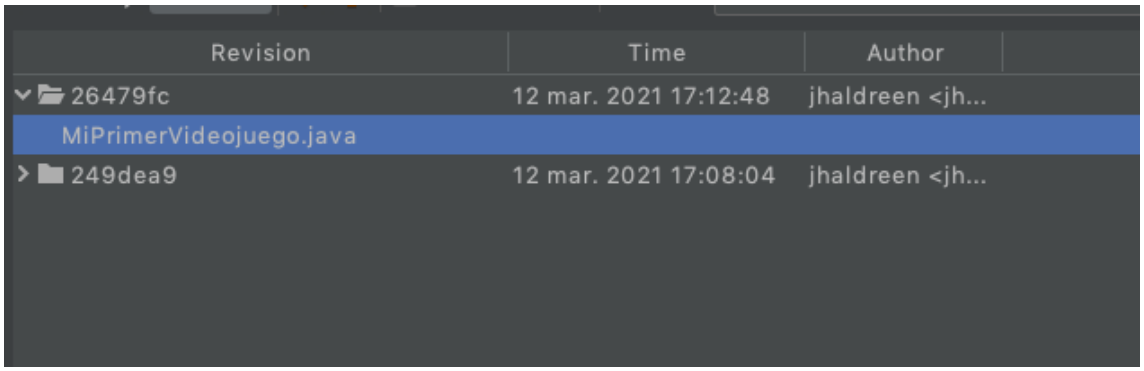
Ahora añadimos la siguiente salida por pantalla, vemos que nos aparece en verde, la zona izquierda.



Ahora con el botón derecho clicamos en el programa, **GIT, show history**



Nos aparece la siguiente ventana seleccionamos MiPrimerPrograma.



Y obtenemos la imagen de antes del cambio y la actual con el cambio realizado. En verde.