Logotipo, Icono

Descripción generada automáticamente

**Yara Velasco Vázquez**

**CONTROL DE VERSIONES CON GIT**

# **ÍNDICE**

**Introducción 2**

**[1ª Parte: Código sin comentarios 6](#_1ª_Parte:_Código)**

**2ª Parte: Documentación con Javadoc 9**

**3ª Parte: Pruebas con JUnit 19**

## **Introducción**

Logotipo, nombre de la empresa

Descripción generada automáticamentePara realizar los pasos de este trabajo, utilizaremos una cuenta disponible de **Github** para situar nuestro proyecto y el programa **Git Bash** que se encontrará ya instalado, el cual nos permitirá descargar el programa del repositorio e ir actualizándolo con comentarios en modo local para subirlo todo de manera conjunta al repositorio remoto.

Contamos con el ejercicio descargado sobre el que querremos trabajar.

Texto

Descripción generada automáticamente con confianza media

Para realizar las 3 fases de este trabajo, decidí crear 3 carpetas con la carpeta del **Ejercicio\_13** metida dentro de cada una de ellas, esto me servirá para probar la funcionalidad de cada una con **Git Bash** e ir modificando los ficheros de manera local para posteriormente, subirlos al repositorio remoto e ir observando los cambios.

Una vez hecho esto, inicio sesión con mi cuenta de **Github** para crear un repositorio desde la opción **New.**

Interfaz de usuario gráfica

Descripción generada automáticamente

Creo un repositorio que llamaré **TR13\_Velasco\_Vazquez\_Yara.**

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Aquí nos mostrará una ventana que nos dirá si queremos crear un nuevo fichero o subir un fichero existente. También nos comenta sobre las opciones de crear un repositorio a partir de comandos, hacer **push** a un repositorio existente o bien importar código desde otro repositorio. Para este trabajo, escojo la opción de subir un fichero existente: **“Uploading an existing file”.**

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Arrastro los ficheros que quiero subir.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Podemos observar que las carpetas se encuentran subidas al repositorio.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Ahora crearé una carpeta llamada **“prueba”** para descargar lo que tiene el repositorio e ir realizando los cambios.

Imagen que contiene Texto

Descripción generada automáticamente

Iniciamos **Git Bash** y nos situamos en la carpeta con el comando **“cd”** y la arrastramos para meter su ruta.

Texto

Descripción generada automáticamente

Ahora copiamos el enlace del repositorio, escribimos **“git clone”** (comando que nos permite descargar contenidos) y pegamos el enlace.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Texto, Carta

Descripción generada automáticamente

Abrimos la carpeta y vemos que se ha descargado lo que se encontraba en el repositorio.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

## **1ª Parte: Código sin Documentar**

Para esta parte, solamente visualizaremos el código del ejercicio sin comentarios.

Para ello, abriremos la carpeta y entraremos en el directorio **“Código sin Documentar”** y accedemos a abrir los ficheros con **Eclipse.**

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Contamos con la clase **Program** para comprobar el funcionamiento de la clase principal y pedirá un valor al usuario.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

La clase **Sucesiones** que se basará en mostrar la sucesión de **Fibonacci** dado una posición que introduzca el usuario. Esta sucesión comienza con los números 0 y 1 y a partir de estos, cada elemento es el resultado de la suma de los dos anteriores.

Texto

Descripción generada automáticamente

La clase **Teclado** necesaria para la lectura de datos.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Texto

Descripción generada automáticamente

El resultado del funcionamiento del programa.



## **2ª Parte: Documentación con Javadoc**

Para esta parte, nos posicionaremos con **Git Bash** en la ruta de los códigos del directorio **“Código documentado”** con el comando **“cd”.**

Captura de pantalla de un celular

Descripción generada automáticamente

Una buena práctica para trabajar con **Git** es tener la rama principal **(main)** con una versión estable de nuestro proyecto, y tener otra rama **“Develop”** en la que poder trabajar.

Con **“git branch -a”** observamos todas las ramas que hay disponibles y en cuál de ellas nos encontramos, como no se ha creado ninguna rama solo nos encontraremos en el **“main”** que es la rama principal. Ahora, creamos una rama llamada **“Develop”** escribiendo **“git branch Develop”.**

Texto

Descripción generada automáticamente

Nos colocamos en esa rama con **“git checkout Develop”.**

Texto

Descripción generada automáticamente

Podemos crear otra rama dentro de **“Develop”** para poder tener varias versiones del código a la vez, en mi caso crearé una nueva rama llamada **“Documentacion”** y me sitúo en ella.

Texto

Descripción generada automáticamente**Javadoc** nos permite detectar errores y añadir nuevas funcionalidades por lo que trabajaremos colocando los comentarios y etiquetas correspondientes en los códigos para su documentación.

Tabla

Descripción generada automáticamente

Abriremos los códigos de la carpeta **“Código Documentado”** con **Eclipse.**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

El resultado con los *tags* y comentarios con **Javadoc** sería el siguiente para la clase **Sucesiones.**

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Para la clase **Program.**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Y para la clase **Teclado.**

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ahora añadiremos al repositorio local los cambios realizados sobre los códigos con las etiquetas y comentarios de **Javadoc** con el comando **“git add”** y añadimos un comentario con **“git commit -m”** sobre lo que se hizo.

Hacemos esto con los 3 ficheros.

Texto, Carta

Descripción generada automáticamente

Texto, Carta

Descripción generada automáticamente

Texto, Carta

Descripción generada automáticamente

Ahora empezaremos a subir ramas hasta llegar a la principal donde subiremos las modificaciones desde el local al remoto, para ello, primero nos cambiamos a la rama **“Develop”** que se encuentra antes de la rama **“Documentacion”.**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Subimos los cambios de la rama de abajo que es **“Documentacion”** a **“Develop”** con el comando **“git merge Documentacion”.**

Texto, Carta

Descripción generada automáticamente

Hacemos el mismo proceso para la rama **“main”;** nos cambiamos de rama y subimos los cambios que contiene **“Develop”**.

Texto, Carta

Descripción generada automáticamente

Finalmente, escribimos **“git push”** para subir los cambios al repositorio remoto.

Texto, Carta

Descripción generada automáticamente

Miramos en nuestro repositorio remoto y vemos que se acaba de actualizar el contenido con comentarios incluidos.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Para finalizar esta parte de **Javadoc,** podemos generar la documentación con **“Generate Javadoc”** desde **Project** en el menú. También crearemos la carpeta destino donde se meterá este contenido, por ejemplo, en el escritorio.

Imagen que contiene Texto

Descripción generada automáticamente

Escogemos las clases a documentar del proyecto e indicaremos con la opción **Browse** la carpeta destino donde se almacenará el código HTML y le damos a **Next.**

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Escribimos el título del documento HTML que se va a generar y finalizaríamos este proceso con **Finish.**

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Ya tendríamos la documentación generada de los códigos del ejercicio, podemos subir la carpeta creada al repositorio con **“Add file”** y **“Upload files”.**

Interfaz de usuario gráfica, Texto, Chat o mensaje de texto

Descripción generada automáticamente

Para aceptar los cambios le damos a **“Commit changes”.**



Vemos que se ha actualizado.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

## **3ª Parte: Pruebas con JUnit**

**JUnit** es una herramienta utilizada para realizar pruebas unitarias en Java, la usaremos para realizar pruebas con el código del programa y así evaluar si el funcionamiento de cada uno de los métodos de las clases se comporta como se espera.

Nos vamos a la carpeta **“Código con Pruebas”** escribiendo **“cd”** y su ruta. Estando en la rama **“Develop”** de **Git Bash,** creamos una nueva rama para realizar las pruebas que la llamaremos **“Pruebas”** y nos situamos en ella.

Imagen que contiene Texto

Descripción generada automáticamente

En **Eclipse** abriremos el proyecto de la carpeta **“Código con Pruebas”.** Nos vamos al menú eny seleccionamos **File>Import>General>Projects from Folder or Archive.**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Aquí buscamos el directorio que contiene el ejercicio sobre el que queremos realizar pruebas y le damos a **Finish.**

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Seleccionamos la clase **Sucesiones** del paquete; **clic derecho>New>JUnit Test Case.**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Cambiamos a la opción **“New Junit 4 test”.**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Aquí seleccionaremos los métodos que queremos probar, en este caso solo tiene uno que es el de **Fibonacci.**

Interfaz de usuario gráfica

Descripción generada automáticamente

Se crea una clase de prueba que se muestra así por defecto.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Con el código siguiente, cambiamos el método **void** a un **string** y añadimos una variable para realizar la clase de prueba.

Texto

Descripción generada automáticamente

Modificamos también el programa.

Texto

Descripción generada automáticamente

Añadimos el código a la clase de prueba para que compruebe el programa.

Texto

Descripción generada automáticamente

Este sería el resultado que nos indica que la prueba ha sido exitosa.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Añadimos cada archivo modificado y escribimos un comentario en relación a los hechos.

Texto, Carta

Descripción generada automáticamente

Texto, Carta

Descripción generada automáticamente

Captura de pantalla de un celular

Descripción generada automáticamente

Nos cambiamos a la rama **“Develop”** y subimos el contenido.

Texto, Carta

Descripción generada automáticamente

Borramos las ramas que creamos con **“git branch -d”.**

Texto

Descripción generada automáticamente

Nos cambiamos a la rama principal y subimos el contenido de **“Develop”** y lo borramos.

Texto, Carta

Descripción generada automáticamente

Texto

Descripción generada automáticamente con confianza media

Y finalizaríamos este proceso con un **push** para subir los cambios a remoto.

Texto, Carta

Descripción generada automáticamente

Así quedaría el repositorio con los cambios realizados. También se podrían añadir para complementar el trabajo este propio documento y una presentación.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente