

|  |
| --- |
| Informe Actividad AE2  Grupo7 |
|  |
| 13 febrero 2022 |

# Indice

|  |
| --- |
| * Componentes del equipo, Grupo 7 (pag 3) * Enunciado de la actividad (pag 4) * Distribución del trabajo en equipo (pag 5) * Pasos realizados por cada desarrollador   + **Pavlo Dudnyk Petrenko (pag 6,7)**   + **David Menéndez Núñez (pag 8,9)**   + Raúl Moreno Rodrigo (pag 10,11) * Dificultades encontradas y resolución de conflictos (pag 12) |
|  |

# Componentes del equipo, Grupo 7

|  |
| --- |
| El equipo de desarrolladores de esta actividad está compuesto por:   * Pavlo Dudnyk Petrenko * David Menéndez Núñez * Raúl Moreno Rodrigo |
|  |
| Enunciado de la actividad  |  | | --- | | AE-2. Control de versiones con GIT y GitHub  **Fecha de entrega** 14 de feb a las 23:59  **Requerimiento 1**  Se pide implementar el código de la actividad 1 mediante Java y Eclipse. Además, hay que tener en cuenta 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. Valoración: 6 puntos sobre 10  **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. Valoración: 3 puntos sobre 10  **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 en esta parte es probable que dé problemas de sincronización, por lo que se espera que los integrantes del equipo hablen y decidan como afrontar el problema.Valoración: 1 puntos sobre 10  Consideraciones  Para toda la actividad se valorará la claridad de código y lo más importante respetar las decisiones documentadas en el Javadoc. En caso de que se decida modificar esa documentación (Ej. se han detectado carencias o errores) se podrá hacer siempre y cuando se justifique.  Para la entrega, es necesario la creación de un pequeño documento formal sobre la actividad (portada, índice, explicación, etc.), indicando los componentes del equipo, la URL del repositorio GITHUB compartido, las decisiones tomadas y la labor de cada integrante del equipo. Para ello cada integrante deberá de exponer los pasos que ha hecho para realizar todos los cambios con su repositorio GIT y el repositorio compartido.  También se valorarán la explicación de los problemas encontrados y su solución. |  Distribución del trabajo  |  | | --- | | Para empezar se acordó repartir el trabajo en base a la actividad 1, por lo que se acuerda que Pablo realice el repositorio común en GitHub, David realice el documento de actividad y Raul implemente en main dos clases en vez de 1 ya que somos 3 en el grupo y hay 4 clases por implementar.  Se mantiene comunicación continua por whatsapp sobre la distribución del trabajo. Se acuerda que la clase main esté en el paquete default y el resto de clases cada una en un paquete, que se importará a la clase main.  En la clase main se crean comentarios para distribuir el paquete en diferentes subclases e intentar evitar el mayor número de conflictos al unir las ramas.  Cada desarrollador del grupo trabajará en su paquete y subirá los cambios, en caso de realizar cambios en el main deberá hacer un push para que todos los desarroladores vean los cambios.  Se acuerda que cada vez que se quiera trabajar en el proyecto, se haga un pull para ver los cambios que han podido realizar los otros desarrolladores en el main.  Cada desarrollador realizará un pequeño documento con los pasos que ha seguido para añadirlo a este documento de Actividad y que será subido también al respositorio. | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | Pasos realizados por cada desarrollador  **Pavlo Dudnyk Petrenko**   1. He creado un repositorio y he invitado a mis compañeros de equipo para que se unan cómo colaboradores:   Captura de pantalla de computadora  Descripción generada automáticamente con confianza media   1. He creado un proyecto local en java, al cual le he copiado las clases Main y Producto de la anterior entrega AE1. 2. He creado el repositorio local de git, seguidamente he hecho un primer commit con la rama máster sin hacer ningún cambio para poder hacer un primer push, para que así exista una rama master cuándo el resto de miembros hagan el primer pull. 3. Ahora he creado una nueva rama llamada producto, sobre la cual he ido haciendo todos los commit:   Interfaz de usuario gráfica, Texto  Descripción generada automáticamente   1. Una vez terminado todo el trabajo, regrese a la rama master para hacer un merge con la rama producto, para seguidamente hacer otro push con todas mis modificaciones. | |  | |  | | Pasos realizados por cada desarrollador  **DAVID MENÉNDEZ NUÑEZ**   1. En base al repositorio que ha creado Pablo, he realizado un clone a una carpeta a mi repositorio local. 2. He añadido un nuevo paquete llamado suma y he añadido la clase Suma.      1. Creada una nueva rama llamada RAMA\_SUMA sobre la que he trabajado.      1. He hecho un commit y luego he realizado un push al repositorio. 2. El siguiente día antes de empezar hice un pull para ver los cambios de los compañeros. 3. He actualizado el main para llamar a los métodos de la clase suma y mostrar en terminal los resultados.      1. He fusionado la RAMA\_SUMA con en master y he realizado un push.      1. He añadido el documento de actividad al repositorio. 2. He comentado el main en la parte de la clase Suma y actualizado el repositorio. | | Pasos realizados por cada desarrollador  **RAÚL MORENO RODRIGO**  Dificultades y resolución de conflictos  Se había creado una carpeta llamada out que se sincronizaba con el respositorio. Como no se quería sincronizar Pablo investigó como hacerlo e introdujo la ruta en su gitignore para evitar que se sincronizase y apareciese en el respositorio de github. | |  | |