### Trabajo de Entornos de Desarrollo

# ACTIVIDAD #2

### Desarrollo de una calculadora en Java en entorno colaborativo.



Presentado por:   
 Nelson Rosales  
 Kyle Lamm

GitHub Respositorio:

<https://github.com/nelson2411/actividad_two_javaDoc_jUnit.git>

[Trabajo de Entornos de Desarrollo 1](#_Toc1138794242)

[ACTIVIDAD #2 1](#_Toc1760457252)

[Desarrollo de una calculadora en Java en entorno colaborativo. 1](#_Toc1597617612)

[Introducción: 2](#_Toc517077865)

[Metodología: 3](#_Toc670295274)

[Enlaces y notas: 4](#_Toc1808338726)

[Problemas y Soluciones: 4](#_Toc1981019943)

[Desglose de Tareas: 4](#_Toc901446009)

[Conclusiones: 5](#_Toc1601034626)

## Introducción:

Dicen que las matemáticas son el lenguaje del universo, pero a veces ese lenguaje parece más un dialecto olvidado de los antiguos dioses del debugging. En este proyecto, nuestro equipo se embarcó en la noble misión de desarrollar una calculadora en Java, con el objetivo de implementar operaciones fundamentales: suma, resta, multiplicación, división, potencia y raíz cuadrada.

Siguiendo las mejores (y a veces desafiantes) prácticas de desarrollo colaborativo, establecimos un repositorio central en GitHub, donde cada miembro del equipo asumió la responsabilidad de una operación específica. Mientras que sumar, restar y multiplicar eran tareas claras, potencia se coló dentro de la clase Multiplicar y raíz cuadrada se escondió en la clase Dividir.

Cada desarrollador trabajó en su entorno preferido (VS Code, Eclipse o IntelliJ) y, una vez completada su tarea, abrió un Pull Request para revisión. Un valiente revisor tenía la responsabilidad de aprobar (o rechazar con diplomacia) los cambios antes de que fueran integrados en la rama principal. Para finalizar, cada integrante actualizó su código con un git pull origin main, asegurando que todos estuviéramos sincronizados y que la calculadora no terminara sumando cuando debía restar.

## Metodología:

* En esta actividad, combinaremos varias herramientas para crear y ejecutar nuestro código.
* En primer lugar, se crea un repositorio remoto en GitHub en el que se puede derivar la base de nuestro trabajo y se establece un lugar de trabajo común.
* A continuación, cada miembro del equipo clonará una copia del repositorio en su sistema local. Llevando a cabo las tareas asignadas a cada uno mediante un **proyecto** en GitHub.
* Por consiguiente, cada miembro creará el código que le haya sido asignado utilizando el IDE de su elección (Intellij, Eclipse, etc.) teniendo en cuenta que debe usar el formato correcto para crear comentarios compatibles con Javadoc. A continuación, cada miembro utilizará Git para enviar su código al repositorio utilizando una rama separada. Esta rama deberá contener un nombre descriptivo.   
  Las ramas se fusionarán y una vez más se extraerán del repositorio a su directorio de trabajo local y, a continuación, se crearán e implementarán las pruebas.
* Las pruebas en esta actividad se llevarán a cabo a través del framework Junit. Después de crear e implementar las pruebas, los miembros del equipo enviarán su trabajo al repositorio para su revisión y el ciclo se repetirá tantas veces como sea necesario hasta que el trabajo esté terminado a satisfacción de cada miembro.

## Enlaces y notas:

Repositotio de GitHub:   
<https://github.com/nelson2411/actividad_two_javaDoc_jUnit>Enlace al proyecto:   
https://github.com/users/nelson2411/projects/1/views/1

Ramas:   
Se ha definido una estructura y estilo de nombramiento de ramas/branch con la nomenclatura:   
***feature/rama***

### Problemas y Soluciones:

* Uno de los problemas que enfentamos fue el de enviar y fusionar correctamente nuestro trabajo en el repositorio desde nuestros espacios de trabajo locales. Esto se resolvió comunicándonos y ayudándonos mutuamente a entender dichas confusiones y dificultades y utilizando adecuadamente los comandos y procesos correctos de Git y GitHub.

### Desglose de Tareas:

1. Kyle Lamm :  
   Creación de la clase Cociente.   
   Aprobar el "Pull Reques” de la clase Suma que fue escrita por Danielrguezm.
2. Nelson:

Crear el repositorio principal de GitHub.

Crear el proyecto para el trabajo en GitHub.  
Crear la clase Producto.   
Crear el Junit test para la clase Producto.

Aprobar diferentes ramas y Pull Requests que no pudieron ser revisados por compañeros que estuvieron ausentes.

## Conclusiones:

El uso de GitHub puede representar un desafío inicial, especialmente al ejecutar los comandos adecuados y seguir los procesos correctos para garantizar una clonación precisa del repositorio y evitar inconsistencias en el código. No obstante, tras familiarizarnos con el flujo de trabajo, la gestión del control de versiones se optimizó significativamente. Git y GitHub demostraron ser herramientas fundamentales para garantizar la integridad del código, facilitando la implementación de medidas de seguridad y minimizando errores en la colaboración.

Asimismo, el uso de JUnit resultó clave en la validación del código, permitiendo la ejecución de pruebas automatizadas para identificar fallos que, de otro modo, podrían haber pasado desapercibidos. Esta herramienta no solo mejoró la detección de errores, sino que también contribuyó a fortalecer la fiabilidad y estabilidad del desarrollo.