Introducción a sistemas complejos, JAVA, MVN, GIT

Jose Gutiérrez Marín

January 2021

1 Introduccción

En el taller introducción a sistemas complejos, java, mvn, git, se busca escribir un programa que encuentre algunas medidas estadísticas de un grupo de datos que vienen de un archivo de texto, este programa debe ser escrito en java, además busca afianzar las hábilidades con git y maven.

2 Descripción del diseño

Para solucionar el taller se hizo el uso de 5 clases en java, las cuales fueron creadas para cada funcionalidad del taller.

- App: Es la clase principal en el proyecto, encargada de resolver el problema haciendo uso de las funcionalidades de las otras clases en el proyecto.
- Calculator: Se encarga de realizar los cálculos matemáticos para hallar las medidas estadísticas solicitadas.
- JoseLinkedList: Esta es la clase que se encarga de guardar la información obtenida en nodos del archivo de texto en la estructura de una lista anidada o enlazada.
- Node: Esta clase hace referencia a los nodos que se almacenan en la lista enlazada o anidada, la cual contiene el valor del nodo y el nodo al que apunta en la lista enlazada, en este caso el siguiente.
- Reader: Se encarga de leer el archivo de texto solicitado y devolver la información en una matriz a su solicitante, el cual en este caso es la clase app.

Como se evidencia en la lista, cada clase tiene una función única dentro del diseño propuesto, lo que facilitó al programador elavorar el código fuente de la aplicación, de igual manera, gracias al diseño, la aplicación es intuitiva, pues se puede manejar todo desde la clase App, y así, probar la aplicación en todas sus facetas.

3 Pruebas

Como es de esperarse, la aplicación fue probada en todas sus facetas, para esto se realizaron 5 pruebas las cuales prueban las funcionalidades por cada clase de la aplicación y una que prueba la funcionalidad total de la aplicación, es decir, un caso real donde se tienen que implementar todas las funcionalidades de las clases implementadas en el proyecto, las pruebas realizadas fueron las siguientes:

- shouldReadTheTestFile(): Esta prueba se encarga de probar la funcionalidad de leer archivos de texto a la clase **Reader** por medio de un archivo llamado *testFile.txt*, donde se comprueba que **Reader** no tenga ningún inconveniente leyendo el archivo (Arroje algún tipo de excepción) y por otro lado se comprueba que la matriz que da como respuesta sea la correcta.

- shouldAddNodesInJoseLinkedList(): Encargada de probar que la clase **Reader** agregue nodos correctamente a la lista enlazada.

- shouldReturnCorrectNodesFromJoseLinkedList(): Esta prueba tiene un trabajo similar al de la anterior, pues comprueba que la lista esté agregando nodos y también que los esté almacenando en el orden correcto, es decir, prueba las funcionalidades de **JoseLinkedList** y de **Node**, pues comprueba los valores de los nodos y el orden de ellos, es decir, hacia los nodos a los que se encuentran apuntando.

- JoseLinkedListShouldBeArranged(): Comprueba por medio de los valores de los nodos agregados a una lista anidada que sean coherentes con lo que se está esperando.

- shouldReturnTheCorrectAnswer(): por último esta prueba se encarga de probar la funcionalidad de la aplicación en un caso real, donde tienen que resolver un problema dado.

4 Conclusiones

Después de diseñar, realizar y probar la aplicación es evidente que llevar un orden de acuerdo al diseño y codificación de una aplicación es de vital importancia, así como saber usar las herramientas con las cuales se va a trabajar en el proyecto, además es importante darnos cuenta que, realizar nuestros propios programas para resolver problemas, situaciones que lo requieran es una gran ventaja, pues es automatizar la solución que se requiere y de esta manera estamos ganando tiempo en cuanto a dedicar tiempo para solucionar esos problemas, es decir, estamos facilitándonos la vida por completo, es importante saber manejar las tecnologías que diariamente está evolucionando en nuestro entorno.