Sebastian Hernandez 95764679

En el siguiente TP,se desarrollo una aplicacion en en el cual por medio de la consola se podran generar una serie de tareas segun el usuario requiera se podran visualizar las diferentes opciones a selecionar las siguientes opciones:

1. Registro de recursos académicos
2. Clasificación de recursos
3. Evaluación de relevancia
4. Generación de informes
5. Filtrado avanzado de recursos

y luego se podra ver por pantalla los resultados o descargar un archivo segun se requiera , se implementaron los pilares e la programaicon orientada objetos tales como:

El poliformismo: se realizan diferentes implementacion en diferentes clases utilizando el mismo metodo heredado, asi mismo tambien por medio de la implementacion de las interfaces.

herencia: existe la clase RecursoAcademico, de la cual la mayoria de las clases heredan sus atributos y metodos.

Emcapsulamiento: a lo largo del proyecto se pueden ver que se usa este principio en diferentes formas privado para metodos y variables de algunas clases, publico para los metodos de las interfaces y abstracto para la clase abstracta de RecursoAcademico.

Abstraccion: se abstrae los metodos que se necesitan sin enfocarnos de su implementacion , solo en el resultado.

Asi mismo se agrega la libreria lombok (no requerido), pero la utilice para dar obtener un codigo mas limpio y menos repetitivo en las clases como lo son getters, setter , toString, entre otros.

trate de modularizar el codigo lo mas posible, para que cada clase tuviera una tarea especifica, y asi cumpliendo con algunos de los criterios de los principios SOLID, particularmente el single responsability.

Se trabajo con inyecciones de dependencias, asi se le quita responsablidad a la clase principal, en este caso se agregaron las constuctor las depedencias requeridas para el funcionamiento aqui se cumplieron otros princios SOLID como: Dependency Inversion Principle en la cual se dependen de de abstracciones (interfaces o clases abstractas).

En el uso de las interfaces tambien se cumplio con el principio de Interface Segregation Principle en cual las interfaces solo tienen los metodos necesarios para el funcionamiento de la misma, sin la necesidad de implementar un metodo que no se utilice.