**LABORATORIO 3 – TDD CREAR UN**

Daniel Ricardo Ruge Gomez

PROYECTO CON MAVEN

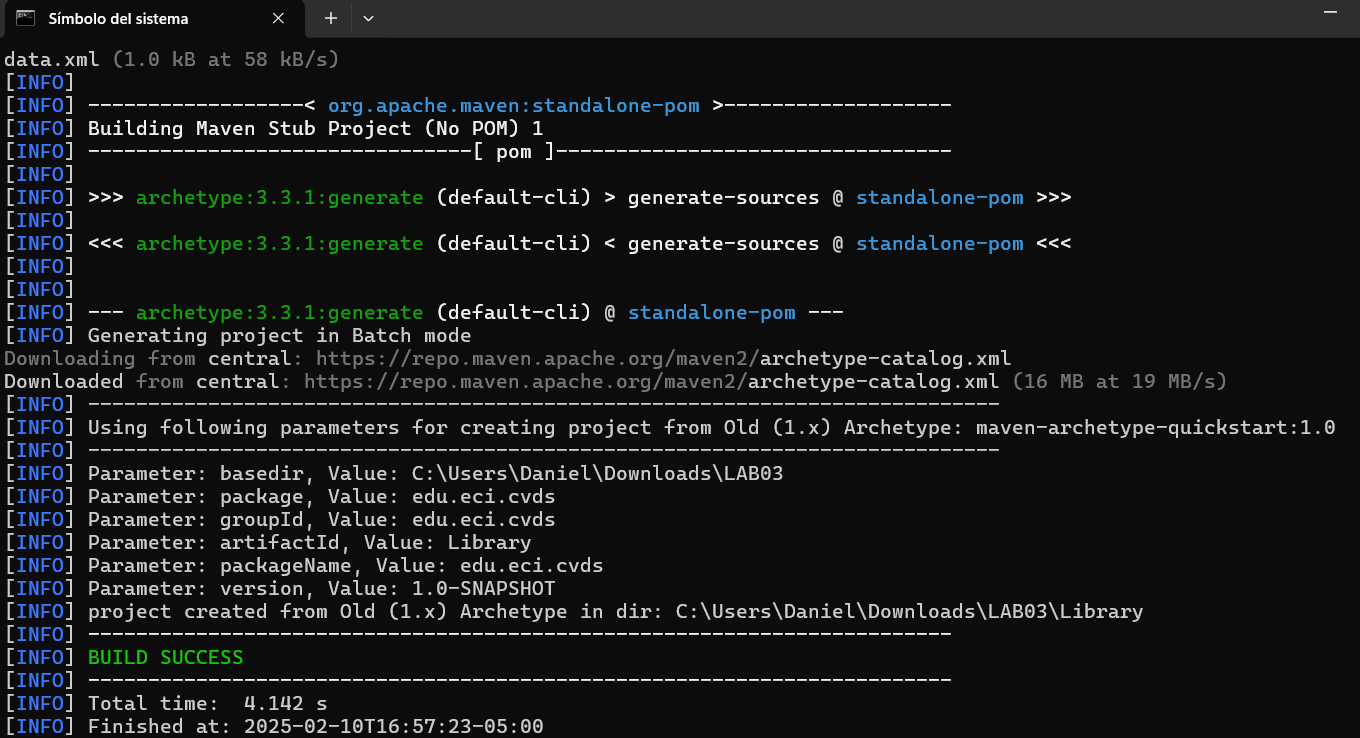
Deben crear un proyecto maven con los siguientes parámetros:

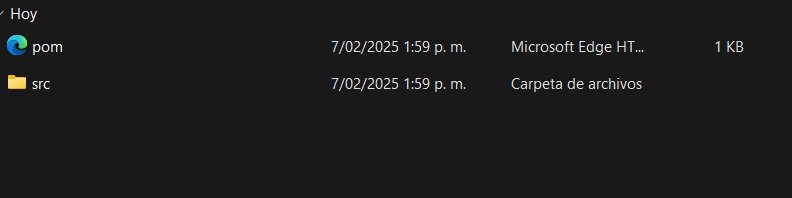
Grupo: edu.eci.cvds

Artefacto: Library

Paquete: edu.eci.cvds.tdd

archetypeArtifactId: maven-archetype-quickstart



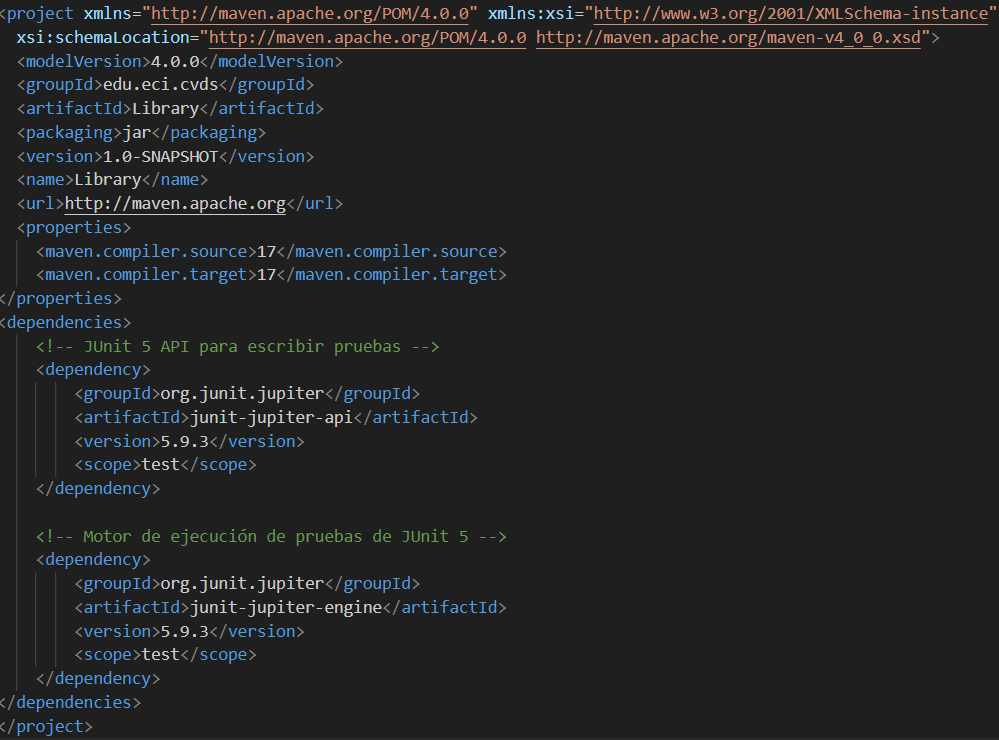


AGREGAR DEPENDENCIA JUNITS

Buscar en maven central la dependencia de JUnit5 en su versión más reciente.

Edite el archivo pom.xml del proyecto para agregar la dependencia.

Verifique que la versión de java sea la 17



AGREGAR ESQUELETO DEL PROYECTO

Cree los siguientes paquetes dentro de edu.eci.cvds.tdd

library

book

loan

user

Estos paquetes también se deben crear en la carpeta de test.

.

│ pom.xml

└───src

├───main

│ └───java

│ └───edu

│ └───eci

│ └───cvds

│ └───tdd

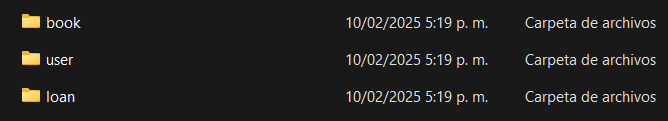
│ └───library

│ └───book

│ └───loan

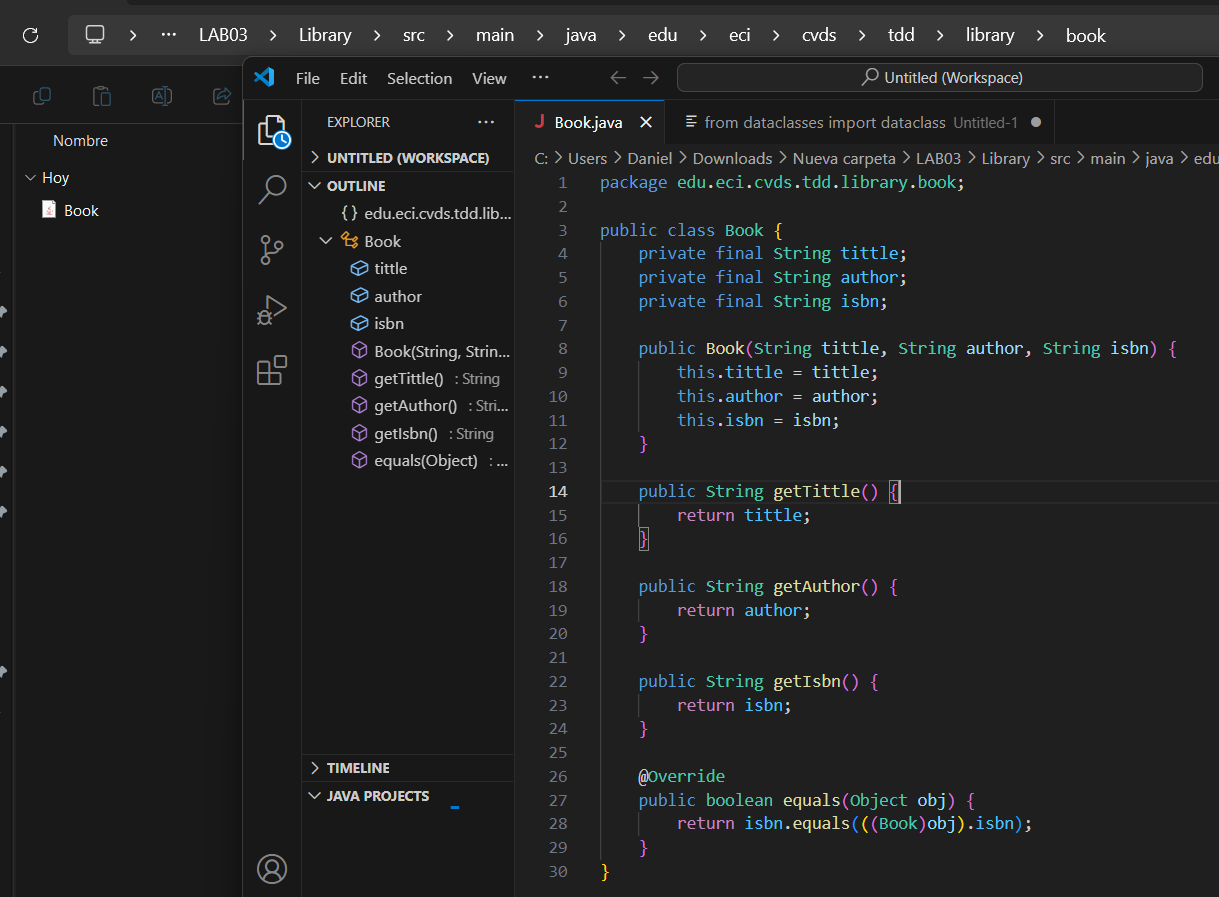
│ └───user

│ App.java

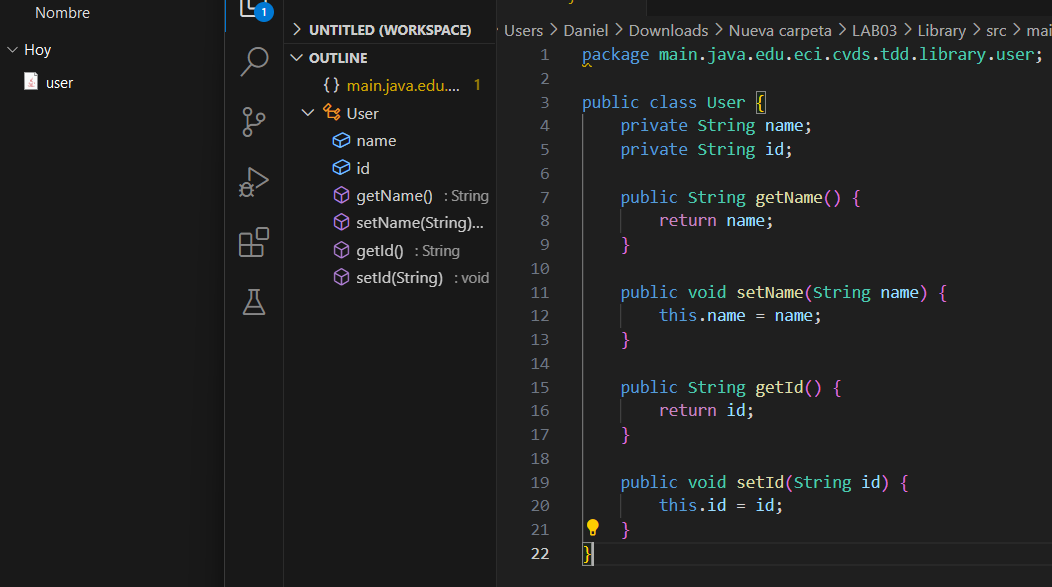


AGREGAR CLASES

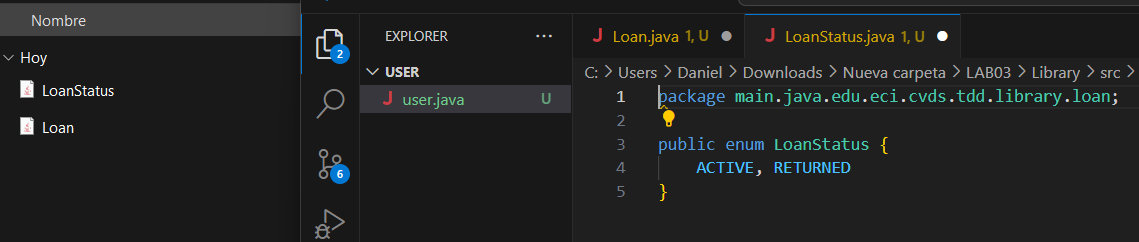
En el paquete edu.eci.cvds.tdd.library.book cree la siguiente clase:



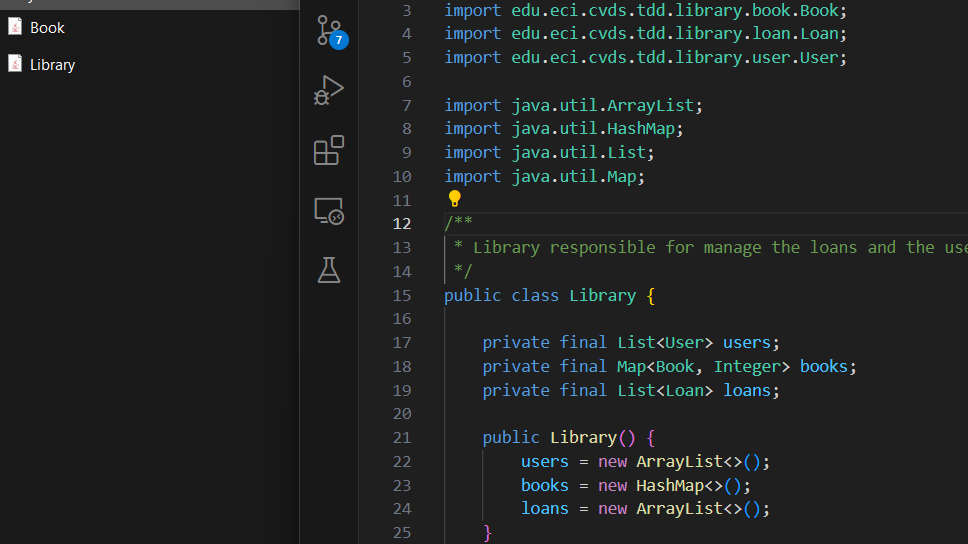
A continuación en el paquete edu.eci.cvds.tdd.library.user cree la siguiente clase:



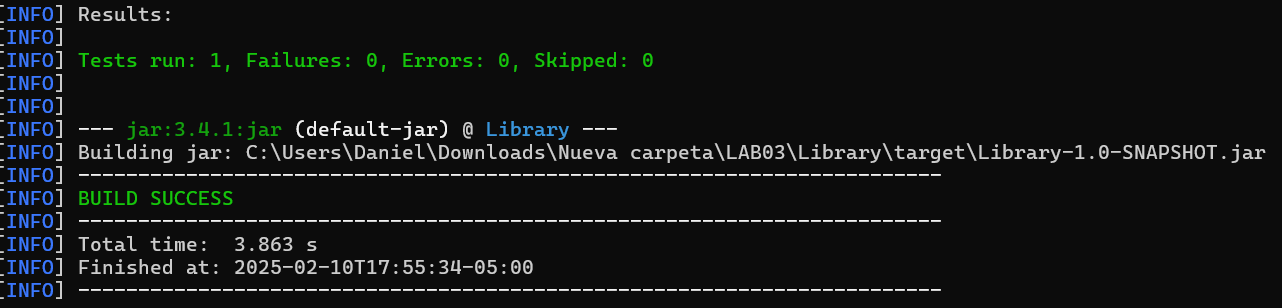
En el paquete edu.eci.cvds.tdd.library.loan se deben crear las clases Loan y el enum LoanStatus:



por último se debe crear la siguiente clase en el paquete edu.eci.cvds.tdd.library



Para validar que la estructura del proyecto está bien se debe compilar usando el comando package.



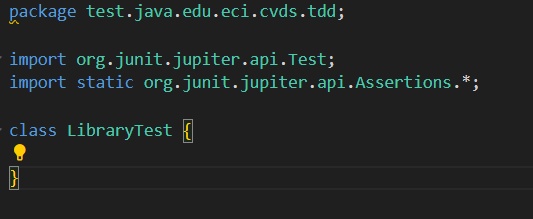
PRUEBAS UNITARIAS Y TDD

CREAR CLASE DE PRUEBA

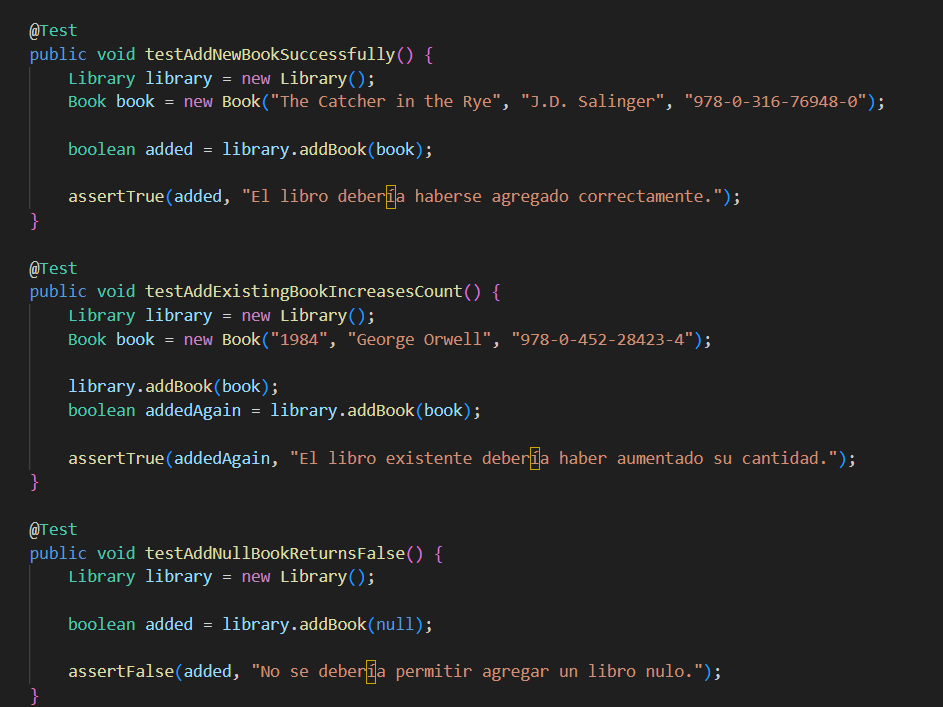
Es necesario crear la clase de prueba para edu.eci.cvds.tdd.Library, la clase debe seguir los estándares de nombres estudiados en clase.

Para pensar en los casos de pruebas lean detenidamente el javadoc de los métodos para reconocer las clases de equivalencia, basados en las clases de equivalencia se debe crear una prueba la cual debe fallar y posteriormente implementar el código necesario para que funcione, este proceso se debe repetir hasta cumplir con la especificación definida en el javadoc.

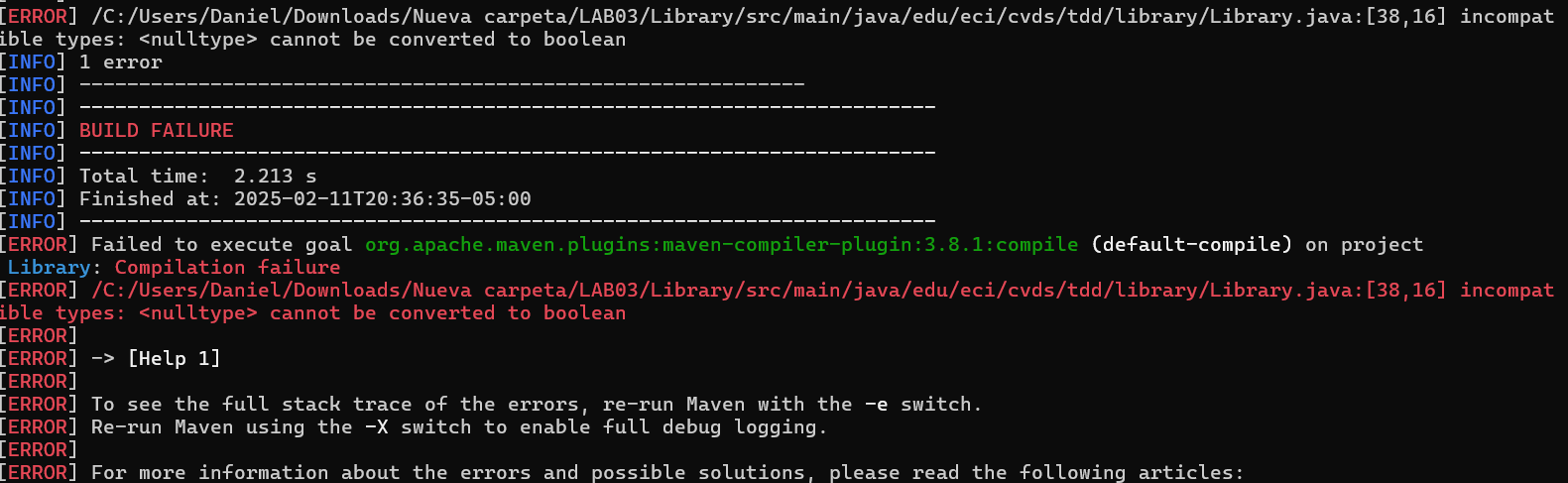
Creamos la clase de prueba:



Siguiendo la documentacion, creamos los test para el metodo addbock:

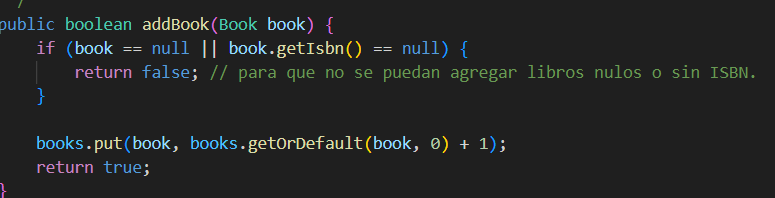


Antes de implementar el metodo:

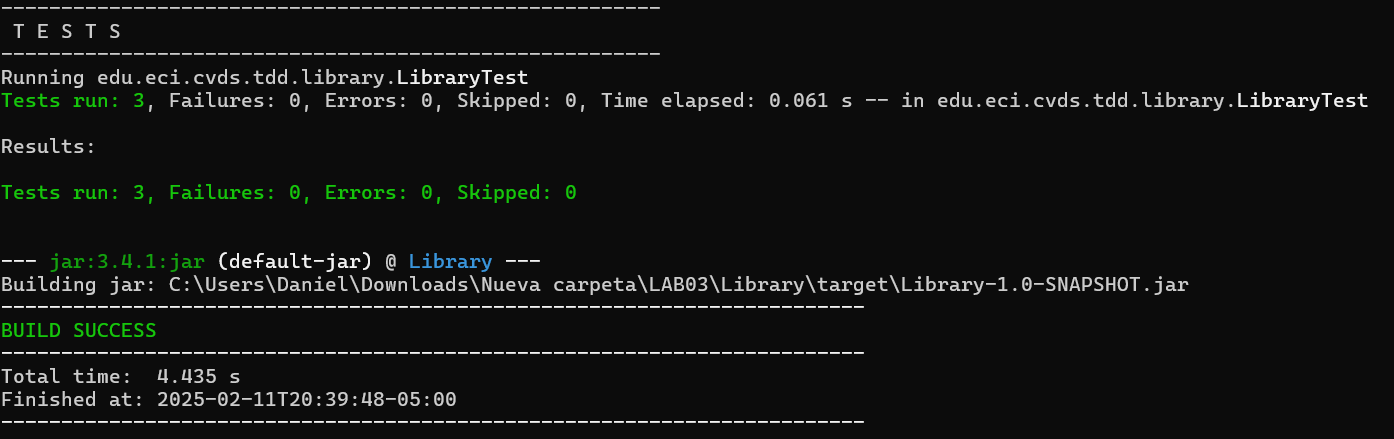


Luego de implementar el metodo:

Implementacion:



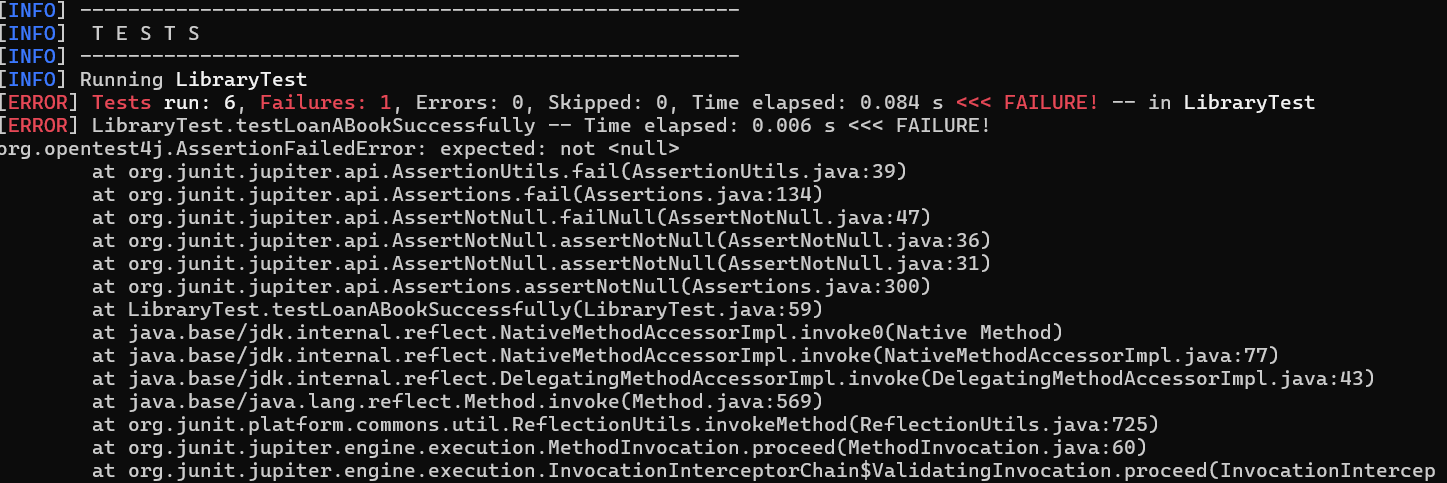
Resultados de las pruebas:



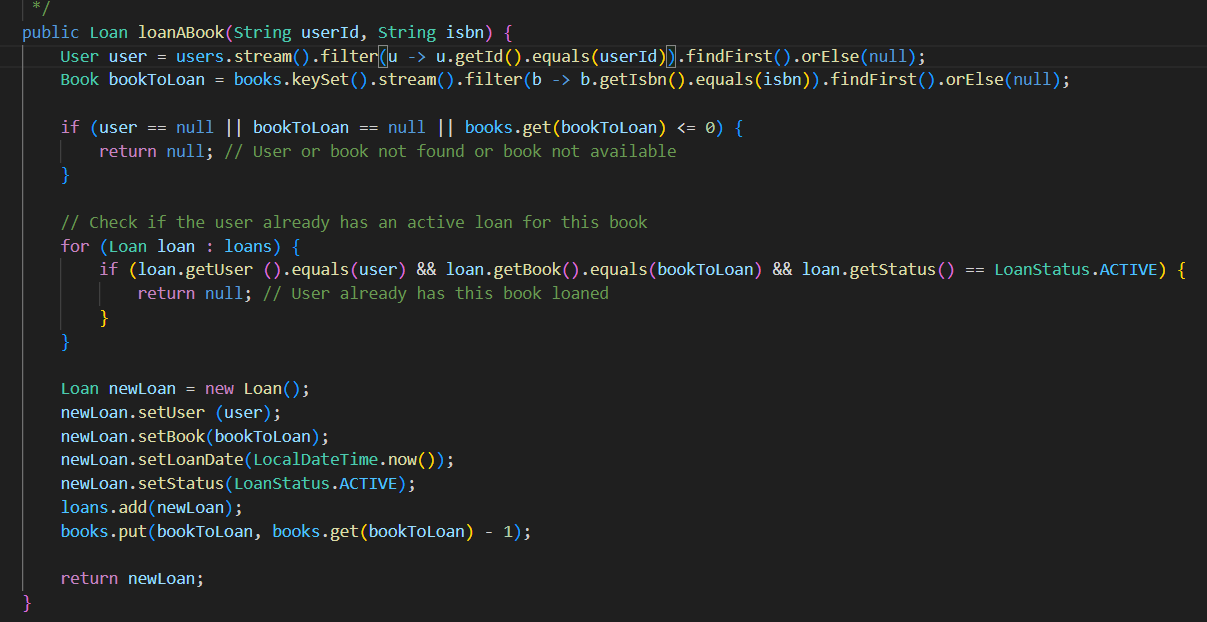
Creamos los test para Loan:



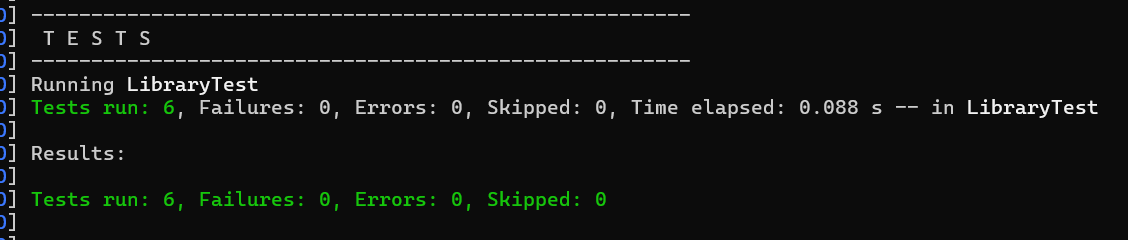
Ante de implementr el metodo:



Implementamos el metodo:



Resultados de las pruebas:



Creamos los test para ReturnLoan

Texto

Descripción generada automáticamente

Antes del método:

Texto

Descripción generada automáticamente

Implementamos método:

Texto

Descripción generada automáticamente

Resultados:

Texto

Descripción generada automáticamente

**COBERTURA**

Ahora al compilar el proyecto en la carpeta target se debe crear una carpeta con el nombre site la cual tiene un index.html, al abrir dicho archivo se debe ver la cobertura total y de cada una de las clases, el objetivo es tener la cobertura superior al 80%.

Tabla

Descripción generada automáticamente con confianza media

Explore los links del reporte en el cual le muestra que partes del código tienen prueba y cuales no.

Pues la mayoría de los que no tienen cobertura son getters y setters por lo tanto en si el código si tiene los test pertinentes