Tarea JUnit - Pruebas de Software

Integrantes: Iñaki Oyarzun M. (Enlace a repositorio)
Profesor: Oscar Reyes H.

Diciembre 2022

1 Sobre el documento

El presente entregable tiene por finalidad, recopilar los desarrollos realizados de manera personal para realizar el programa en Java solicitado con las funcionalidades requeridas, además de hacer uso de JUnit para realizar la estructuración de pruebas unitarias y a la vez pruebas de cobertura, las cuales serán detalladas a continuación

2 Desarrollo

2.1 Estructura base

A partir de los siguientes requerimientos para un programa en línea de comandos:

1 Que permita almacenar Libros a partir de una serie de datos indicados

2 Que sea ejecutable en Java

3 Que permita realizar pruebas unitarias y pruebas de cobertura

4 Que sea posible realizar CRUD

De esta forma, se lleva a cabo la estructuración del programa a partir de la creación de dos clases principales:

- **Clase Ubicacion (carpeta moduloReg):** Será la clase encargada de administrar funcionalidades asociadas al manejo de la ubicación del libro, es decir:
  - pisoBiblioteca: El piso donde se encuentra - [Se asume como entero]
  - pasillo: El pasillo donde se encuentra - [Se asume como texto por ejemplo 22-A]
  - estante: El estande en donde se encuentra el libro - [Se asume como texto]
  - posicion: La posicion en donde se encuentra el libro - [Se asume como un array de dos enteros]
- **Clase Libro (carpeta moduloReg):** Será la clase encargada de administrar el manejo del libro tomando en cuenta las otras variables, las cuales son:
  - titulo: Titulo del libro - [Se asume como texto]
  - autores: Autores del libro - [Se asume como texto]
  - fechaEdicion: La fecha en la que se crea el libro - [Se asume como texto]
  - numeroPaginas: La cantidad de paginas que tiene el libro - [Se asume como entero]
  - editorial: La editorial del libro - [Se asume como texto]
  - genero: El genero del libro - [Se asume como texto]
  - isbn: El ISBN del libro - [Se asume como texto]
  - ubicacion: El objeto ubicacion creado con la clase antes señalada
  - estado: El estado en el que se encuentra el libro - [Se asume como texto]

– descripcion: La descripcion del libro - [Se asume como texto]

Es así como utilizando estas dos clases principales, se estructuran en ellas funciones en primer lugar constructoras de la clase, para luego indicar funciones get y set para cada una de las variables señaladas anteriormente de la clase. De forma que en el programa final sean esas las utilizadas durante los manejos CRUD.

2.2 Para el programa

Una vez se tienen las funciones iniciales, se crea el programa que manejará la interacción con el usuario, hallándose todo dentro del código *Main.java*. Allí se encuentran funciones las cuales manejan la interacción del usuario a la hora de crear, actualizar, borrar o ver algún dato de un libro.

En este caso igualmente se maneja un Array para almacenar todos los libros, asumiendo que la instancia del programa alojara los libros en cache durante su ejecución para fines prácticos. La ejecución del programa se puede hallar dentro del repositorio de acuerdo a las instrucciones según la maquina que se ejecute.

3 Pruebas unitarias

Para el desarrollo de las pruebas unitarias, se hace uso en un inicio de VSCode junto con la suite de extensiones de Java para crear las pruebas unitarias en base a JUnit para luego ejecutar las pruebas. Las pruebas a realizar fueron en base a las clases base de Ubicación y Libro, ya que son las que manejan la funcionalidad completa asociada a la creación y edicion del libro según lo indicado por el enunciado de la tarea asignada. Las pruebas son:

- **crearUbicacion:** Se hace una prueba para la creación del objeto Ubicacion dentro de la clase Ubicacion, de manera en que se crea el objeto y luego se realiza un assertTrue comparando todos los componentes de la clase a partir de las funciones get antes señaladas.
- **editarUbicacionInt:** Se hace una prueba para la edición de un campo entero dentro del Objeto de la clase Ubicacion, ya que se considera equivalente el realizar la prueba en el cambio de una variable entera de la clase. De esta forma, se crea el objeto Ubicacion, se le cambia el valor entero del piso y luego se realiza nuevamente un assertTrue para comparar antes y despues si el valor se cambió.
- **editarUbicaionStr:** De igual manera que en la prueba anterior, se ejecuta el mismo procedimiento de edición, pero en este caso para una variable del tipo String dentro de la clase, siendo igualmente equivalente el modificar una de todas las variables string que maneja la clase. Es de esta forma que, se crea un objeto de Ubicacion, para luego cambiar su valor de pasillo utilizando las funciones get y set definidas dentro de la clase y con ello ejecutar un assertTrue para comparar si los valores fueron cambiados correctamente.
- **crearLibro:** Se hace una prueba para la creación de un objeto Libro que esta dentro de la clase Libro. Se comienza creando el Libro, para luego comparar con assertTrue cada una de las entradas dadas al constructor, para corroborar que fueron correctos todos.
- **editarLibroStr:** Al igual que en el caso de editarUbicacionStr, se busca corroborar la modificación de una variable tipo texto dentro de la clase, utilizando una de sus funciones get y set para cambiar el valor del título. Para corroborar con assertTrue.
- **editarLibroInt:** Al igual que en el caso de editarUbicacionInt, se busca corroborar la modificación de una variable tipo entero dentro de la clase, utilizando una de sus funciones get y set para cambiar el valor de paginas.

Con estas pruebas unitarias, se corrobora el correcto funcionamiento para el programa final, donde se demuestra que el uso de las funciones para crear y modificar el libro están correctas (Ver Figura 1). Con ello en una segunda etapa, se comienza a realizar la estructuración del código para el manejo de la interfaz.

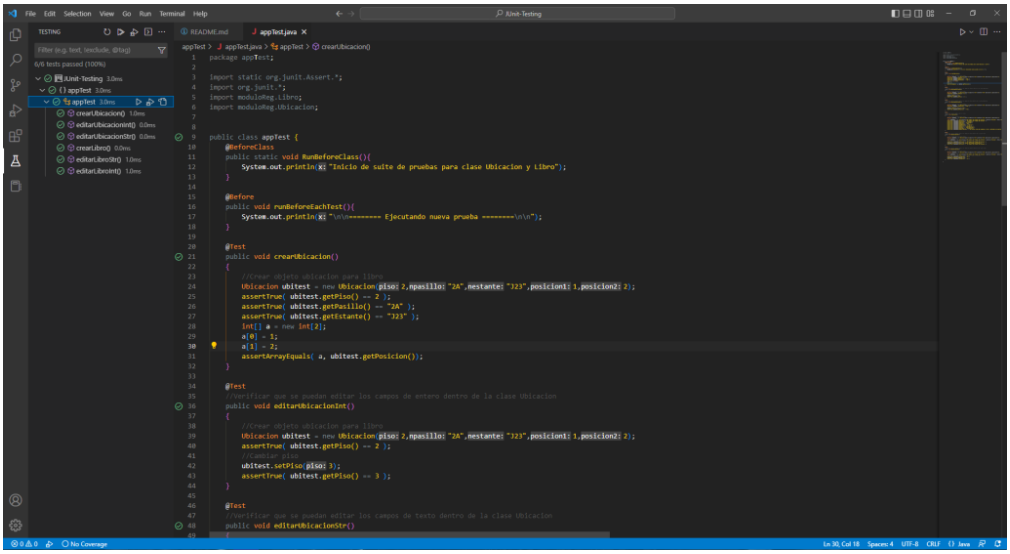

Figure 1: Pruebas escritas dentro de VSCode con la extensión de pruebas en Java, se observa el cumplimiento de las pruebas ejecutadas y parte del código de las pruebas.
Fuente: Elaboración Propia.

4 Interfaz

Se realiza un programa basado en la consola, utilizando Java y la librería Scanner que viene por defecto para el manejo de los input de usuario por la consola para el manejo de la interfaz, teniendo funcionalidades separadas para lo que fueron los manejos solicitados dentro del programa (Ver Figura 2).

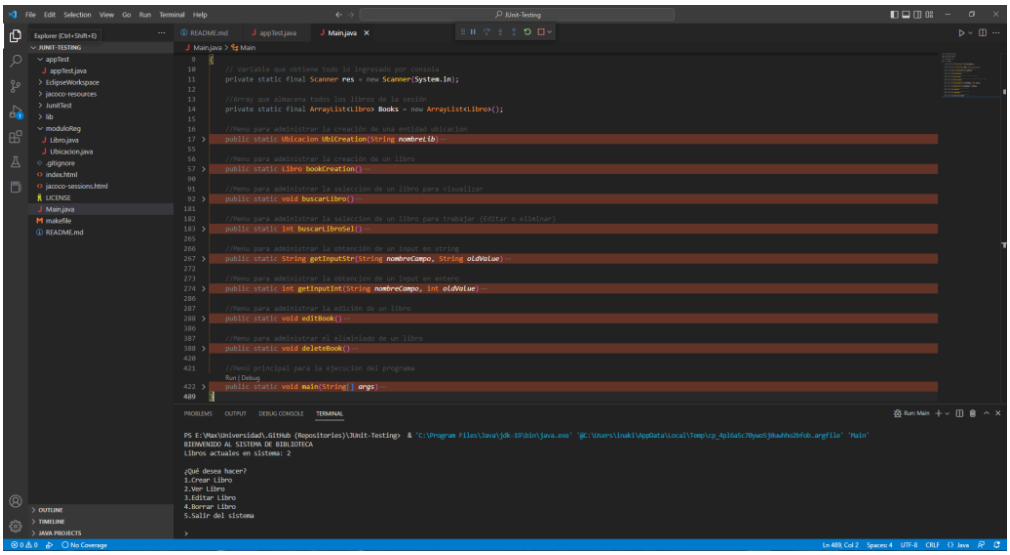

Figure 2: Evidencia del programa principal desarrollado en VSCode, junto a la distribución de los archivos, parte del código y la ejecución mostrando el menú principal en la parte de abajo.
Fuente: Elaboración Propia.

5 Cobertura

Debido a que VSCode no tiene aún una extensión para el manejo de esta funcionalidad es que se hace un traslado de lo realizado a la aplicación de Eclipse IDE que incluye allí una extensión para la ejecución de

pruebas de cobertura, de esta forma una vez montado todo dentro del intérprete, se ejecutan las pruebas de cobertura, obteniendo los siguientes resultados presentados dentro del mismo IDE (Ver Figura 3).

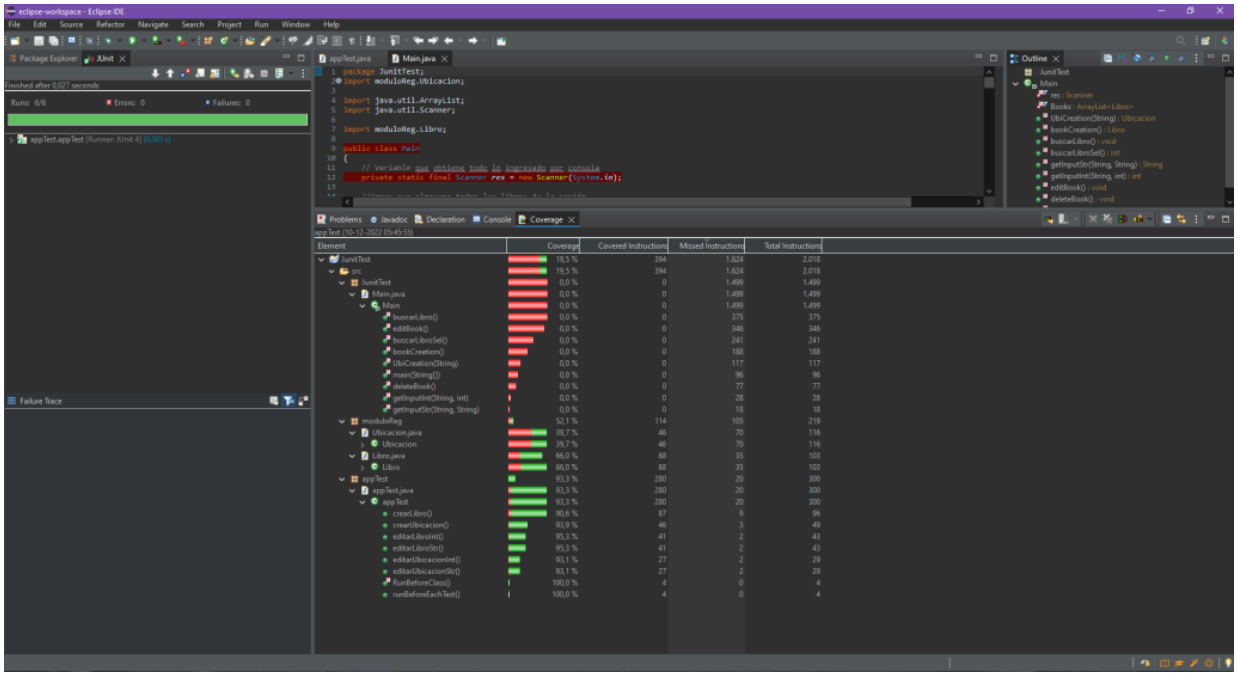

Figure 3: Evidencia de uso de Eclipse y sus extensiones para la comprobación de cobertura al ejecutar las pruebas unitarias, mostrando parte del código y detallando las pruebas con su respectiva cobertura.
Fuente: Elaboración Propia.

Como se observa, la cobertura señala estar muy baja para lo que es el manejo del programa principal, sin embargo, esto se debe a que, como se realizaron las pruebas enfocadas a las clases principales o base, es por ello que estas aparecen en cero, igualmente, no se cubren completamente las clases base creadas debido a que igualmente, solo se hace uso para la edición una de las funciones de edición para string y de entero de cada clase, por lo que, al no utilizar las funciones para cambiar las demás variables dentro de las pruebas, la cobertura se ve afectada. Para obtener un mayor detalle además, se agrega la exportación de esta prueba de cobertura, siendo agregada al repositorio con el nombre de *index.html*.

6 Conclusiones

Como parte del manejo de este stack para el desarrollo de pruebas, se presentaron problemáticas asociadas al aprendizaje e instalación de estos paquetes requeridos, ya que en este caso, es la primera vez que se utilizan, requiriendo tener un extremo cuidado y buscando tutoriales o guías para asegurar un correcto uso e instalación de todo para evitar los problemas a futuro durante el desarrollo.

Sin embargo, como parte de la experiencia desarrollada, se lograron comprender los usos para el desarrollo de pruebas unitarias, principalmente observar lo sencillo que es el uso de JUnit y a la vez lo potente que puede llegar a ser su uso en el desarrollo de aplicaciones, teniendo herramientas como lo son VSCode y Eclipse que facilitan aún más la tarea permitiendo dejar en un click la ejecución de las pruebas o la generación de estos reportes sobre la cobertura.