Parte 1 - Examen Final – Mockito

Suponga que la clase **RecipeBook** aún no está terminada y que tiene la tarea de probar el resto de la aplicación. Deseamos construir pruebas unitarias para un subconjunto de historias de usuarios utilizando una versión simulada de la Clase ***RecipeBook***. Para lo cual ejecutaremos usando una versión donde ***RecipeBook*** es una interfaz abstracta. En este caso, no podemos probar la clase **Recipe**, por lo que no tiene sentido tratar de modificar las pruebas relacionadas con **AddRecipe**, **DeleteRecipe** y **EditRecipe**.

Caso de Uso: Flujo de eventos para el caso de uso de compra de bebidas

Condiciones previas: ninguna

Flujo principal: el usuario seleccionará la bebida que desea comprar. El usuario depositará dinero para pagar la bebida. **[S1] [S2]**

Subflujos:

[S1] El CoffeeMaker verificará si hay suficientes ingredientes en el inventario para hacer la bebida seleccionada. **[E1]**

[S2] El CoffeeMaker se asegurará de que se haya depositado suficiente dinero **[E2]**, se dispensará la bebida y se devolverá cualquier cambio adicional.

Flujos alternativos:

[E1] Si no hay suficiente inventario para hacer la bebida, se mostrará un mensaje, se le devolverá el dinero del usuario y el usuario volverá al menú principal.

[E2] Si el usuario no ingresa suficiente dinero, su dinero será devuelto y el usuario volverá al menú principal.

[E3] Si el usuario selecciona un número que no corresponde a una receta, se le devolverá el dinero del usuario y el usuario volverá al menú principal.

Además, queremos verificar que los métodos ***getAmtChocolate ()***, ***getAmtCoffee ()***, ***getAmtMilk ()*** y ***getPrice ()*** se invoquen una vez para la receta seleccionada.

Descargar el proyecto inicial de GitHub para IDE Netbeans – Maven que contiene dependencias de Mockito y Jacoco, disponible aquí:

https://github.com/jcbergman/mockito_coffeeMaker

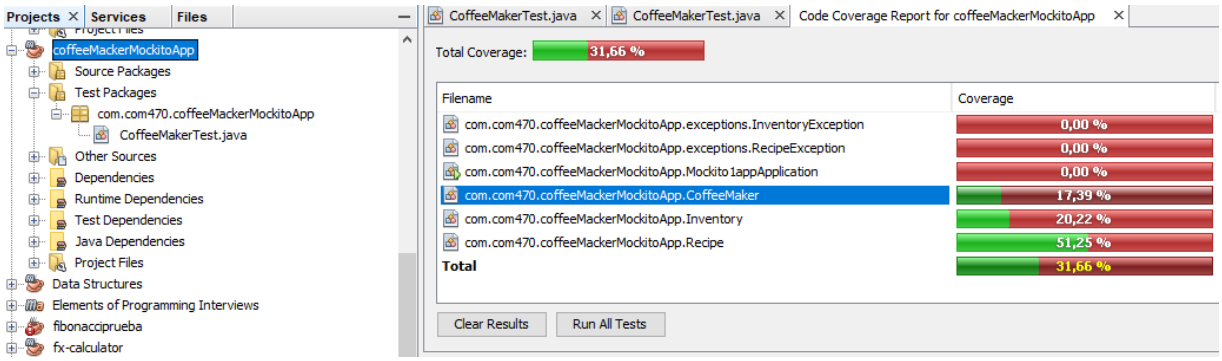
Entregar

Su tarea es utilizar Mockito para crear una versión simulada de ***RecipeBook*** y volver a ejecutar las pruebas unitarias. Queremos encontrar los errores en las clases **CoffeeMaker** e **Inventory** utilizando el ***RecipeBook*** simulado, y verificar que ***RecipeBook*** se esté utilizando de manera adecuada utilizando el soporte de simulación de Mockito.

Para los escenarios de compras y bebidas, esto implicaría verificar que los métodos ***getAmtChocolate (), getAmtCoffee (), getAmtMilk () y getPrice ()*** se invoquen adecuadamente.

Deberá entregar el archivo "CoffeeMakerTest.java" y "Captura de pantalla" a la plataforma ecampus en el tiempo indicado, se calificará en función a la métrica de cobertura alcanzada, y reemplazando el archivo CoffeeMakerTest.java, en el proyecto original (para comprobar la entrega) (solo deberá completar los @Test necesarios y comprobar con la métrica). El proyecto

inicial cuenta con un test el cual ha llegado a una cobertura de 17% , lo que consistirá en su punto de partida.



Ej. de cobertura