Programming Assignment: Adding Mocks and Stubs

✓ Passed · 10/10 points

Deadline The assignment was due on Apr 30, 11:59 PM PKT
You can still pass this assignment before the course ends.

Instructions **My submissions** **Discussions**

Task

Suppose that the RecipeBook class is not yet finished and you are tasked with testing the rest of the application. We wish to construct unit tests for a *subset* of the user stories using a stubbed version of the RecipeBook. During the grading phase, we will run a version of the Coffee Maker that leaves the RecipeBook as an abstract interface. In this instance, we can't test the Recipe class, so there is no point in trying to modify tests relating to AddRecipe, DeleteRecipe, and EditRecipe.

A good place to focus is on the PurchaseBeverage tests:

UC7: Flow of Events for the Purchase Beverage Use Case

7.1 Preconditions: None

7.2 Main Flow: The user will select the beverage they wish to purchase. The user will deposit money to pay for the beverage. [S1][S2]

7.3 Subflows:

[S1] The CoffeeMaker will check if there are enough ingredients in the inventory to make the selected drink. [E1]

[S2] The CoffeeMaker will make sure enough money was deposited [E2], the beverage will be dispensed, and any extra change will be returned.

7.4 Alternative Flows:

[E1] If there is not enough inventory to make the beverage, a message will be displayed, the user's money will be returned, and the user will be returned to the main menu.

[E2] If the user does not enter enough money, their money will be returned, and the user will be returned to the main menu.

[E3] If the user selects a number that does not correspond to a recipe, the user's money will be returned, and the user will be returned to the main menu.

In addition, we want to verify that the getAmtChocolate(), getAmtCoffee(), getAmtMilk(), and getPrice() methods are being called once for the selected recipe and zero times for the other recipes.

To get started, we have an initial gradle Eclipse project containing Mockito dependencies available here:

CoffeeMaker_Mock_Assign
TGZ File

CoffeeMaker_Mock_Assign
ZIP File

The instructions for importing the project are the same as the previous assignment with JaCoCo, so if you need to refresh your memory, look there for instructions.

Deliverable

Your task is to use Mockito to create a mocked/stubbed version of RecipeBook and re-run the unit tests that are not focused on RecipeBook, such as PurchaseBeverage. We want to find the bugs in the CoffeeMaker and Inventory classes using the stubbed RecipeBook, and to check that RecipeBook is being used appropriately using the mocking support of Mockito.

For the purchaseBeverage scenarios, this would involve checking that the getAmtChocolate(), getAmtCoffee(), getAmtMilk(), and getPrice() methods are being called appropriately for the scenario. We will add some mutant versions of the CoffeeMaker that will check this aspect of the system.

How will this be graded?

In order to measure the adequacy of your tests, we will run your tests against an incomplete implementation of the CoffeeMaker in which RecipeBook does not have any functionality. Thus, a mocked/stubbed version of this class is necessary for tests to run correctly.

You will get 50% credit for submitting tests which accurately report a working correct implementation. You will achieve incremental additional credit for each mutation in the CoffeeMaker and Inventory class that are caught by the mocked/stubbed tests.

How to submit

When you're ready to submit, you can upload files for each part of the assignment on the "My submissions" tab.