

DIT635

Assignment 2

Date: 25/2-2022

Group 13 :

Marcus Andersson - gusqanuqma@student.gu.se

Carl Andreasson - gusandcafg@student.gu.se

Adam Frison - gusfrisoad@student.gu.se

Felix Mertala - gusmertfe@student.gu.se

Richard Novenius - gusnovri@student.gu.se

1.1

We plan to test edge cases, out of bounds input and invalid inputs, also some expected inputs. This increases the robustness of the system. We don't plan to test scalability, availability nor security since the program is intended to be small, simple and to run locally. The test will be written in java using JUnit 5 as the framework. We plan for a test coverage of 100% (Not possible for now, since the system crashes when trying to add sugar, also when testing hashcode we can't get full coverage despite which values we are testing, the same goes for equals). The test's duration should not exceed 3 seconds. Our pass criteria for the system is at a success rate of 100%. Files not to be tested are: Main and the files in the exceptions folder.

1.2 Tests are in the .Java file which was submitted with this pdf, tests for this problem are under the comment: TESTS FOR PROBLEM 1.2.

1.3 We have not been using any external libraries when testing.

1.4

Detecting faults

- The program crashes when you try to make a coffee bigger than number 4 or smaller or equal 0.
- Delete coffee not working like it should, after deletion it still occupies a recipe space.
- Coffee inventory should decrease when making coffee not increase.
- Cannot add sugar, checks for value less than instead of greater than.

Suggested fixes








- Make it possible to add in a new name when editing a recipe, right now after deleting and editing the empty recipe space I can purchase a coffee without knowing what it is.
- Show the amount of cost for each coffee recipe in the menu when ordering/making.
- Specify amount(dl/ml/cl etc.) to make it more clear for example what 1 unit of milk is.
- Also fix everything we found under **detecting faults**.

2.1

Tests are in the .Java file submitted with this pdf, tests for this problem are under the comment: TESTS FOR PROBLEM 2.1.

2.2

Coverage of tests from both 1.2 and 2.1:

▼ CoffeeMaker/src/main/java		56,5 %	664	512	1 176
▼ edu.ncsu.csc326.coffeemaker		56,2 %	656	512	1 168
> CoffeeMaker.java		91,6 %	76	7	83
> Inventory.java		100,0 %	258	0	258
> Main.java		0,0 %	0	496	496
> Recipe.java		96,0 %	215	9	224
> RecipeBook.java		100,0 %	107	0	107

2.3

Line coverage of nearly 100% was reached by our initial tests, we are not able to reach a test coverage of 100% due to the functions addSugar and Recipes hashCode and equals, otherwise it would be possible and we believe we have reached the highest possible amount of coverage given the domain we are operating in. Which is shown in the picture above for 2.2.