Task 1:

It is in SVN (google code of Vending Machine project)

Task 2

For this task, complete the acceptance testing table (located in the reqs directory) for the acceptance tests necessary to complete the User Stories.

Acceptance Tests

Test ID	Description	Expected Results	Actual Results
addRecipe1	Precondition: Run CoffeeMaker Enter: Menu option 1, "Add a recipe " Name: Coffee Price: 50 Coffee: 3 Milk: 1 Sugar: 1 Chocolate: 0 Return to main menu.	Coffee successfully added.	Coffee successfully added.
deleteRecipe1	Precondition: addRecipe1 has run successfully Enter: Menu option 2, "Delete a recipe " Select: Coffee Return to main menu.	Successfully deleted	Successfully deleted
editRecipe1	Precondition: addRecipe1 has run successfully Enter: Menu option 3, "Edit a recipe " Select: Coffee Price: 50 Coffee: 3 Milk: 1 Sugar: 1 Chocolate: 0 Return to main menu.	Coffee successfully added.	Coffee successfully added.

addInventory 1	Precondition: not enough ingredients in the Inventory Enter: Menu option 4, “Add inventory" Coffee: 10 Milk: 10 Sugar: 10 Chocolate: 10 Return to main menu.	Add Inventory successful	Coffee: 25 Milk: 25 Sugar: 25 Chocolate: 25
checkInventory	Precondition: before making beverages make sure amount of ingredients present Enter: Menu option 5, “Check inventory" Return to main menu.	Checked the inventory amount	Coffee: 25 Milk: 25 Sugar: 25 Chocolate: 25
purchaseBeverage 1 (makeCoffee)	Preconditions: checkInventory, addRecipe1 should run successfully : checkInventory: 5 Add Recipe : Coffee; Please enter the recipe price: $100 Coffee: 2 Milk: 1 Sugar: 1 Chocolate: 0 Enter: Menu option 6, "Make coffee " Select: select the number of the recipe to purchase: menu option 4 enter the amount you wish to pay: 300 Your change is: 200 Return to main menu.	CanMakeCoffee should be true	Your change is: 200

checkOptions0	Precondition: None	Please press the number that corresponds to what you would like the coffee maker to do.	Please press the number that corresponds to what you would like the coffee maker to do

Task 3

Once you have coded the acceptance tests from Task 2 and ensure that the pass within JUnit, run the EclEmma code coverage tool and generate an HTML test report (as you did in Task 1) Task3.html that should be included (along with the dependent files) in the reqs folder within your CoffeeMaker project.

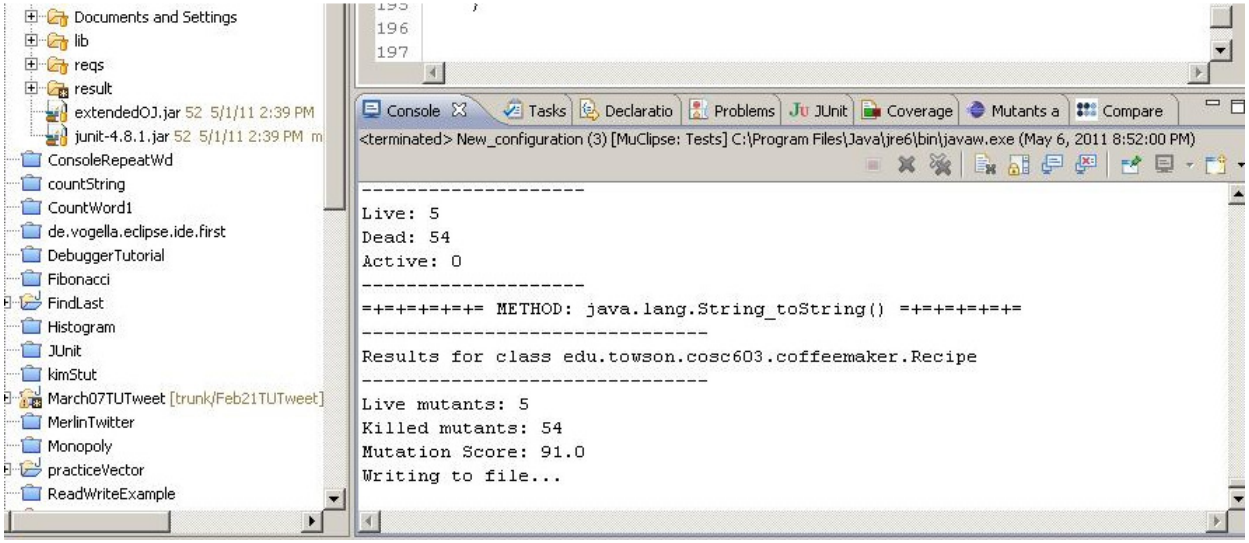
It is in SVN (Google code).

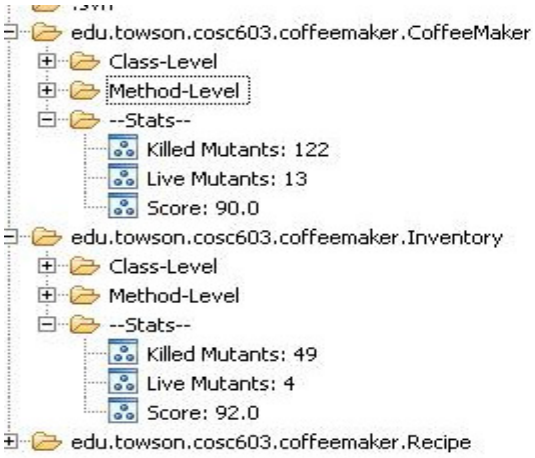
Task 4:

Once you are familiar with how to use MuClipse, use its mutation testing results to kill as many mutants as possible and increase your Mutation Score. I would, ideally, like to see a resulting Mutation Score of 90. Once you think you are finished with this step, capture a screen show displaying your mutation score

and include it in your Project Report.

For this task, briefly describe how mutation testing aided in developing your unit test cases.





Mutation testing was really a great testing tool which helped in finding out all the exceptional test cases which I did not implement during acceptance testing or unit testing. The mutants , those were alive after mutation test run, were not very easy to remove all the time. Sometimes it was not difficult to remove, but it needs through understanding of the code.

Task 6 – Summing it All Up. Upon completion, each student is to submit a short report that includes

your write-ups from the previous tasks (clearly labeled) as well as:

- **A description (2-3 paragraphs) of what you learned from this project**

  In this project I have learned about coverage tool for unit testing and how to apply coverage criteria for tesing by installing eclipse plugin EclEmma. I have also learned how to determine if I have written sufficient Junit test methods or not to cover the code 100%. After runnig coverage as Juint test in Eclipse, the details percentage coverage for each class and each method can be seen in the console. This was very helpful to determine extra test methods for coverage.

I have also learned about Mutation testing. Muclipse is also a plugin, but installation of Muclipse was not easy. It did not work properly in first trial. It took some time to install properly. But the testing tool is very helpful to determine exceptional test methods.

- **A description (2-3 paragraphs) of what you liked and didn't like about EclEmma and MuClipse's**

  support for unit testing.

EclEmma is a good tool for coverage, but I got some warning about coverage tool is not installed after I used it for many times. So I had to uninstall and install it two to three times. Other than that, It is a very good tool and easy to use.

Muclipse is also a very good tool. But its setup was little bit error prone. Otherwise, using Muclipse is very easy.