Problem 2

Allen Bates, Hao Chen, Marcos Moraes, Krishna Sindhuja

1. Generate invalid, valid-but-not-useful, useful, equivalent, and non-equivalent mutants for methods of your choice in the CoffeeMaker code (before you apply any of your fixes). You do not have to use the same methods for all mutant categories. Apply at least one mutation operator from each of the three categories in Figure 16.2 in the textbook. (20 Points)

Invalid (Program shouldn't even execute)

In CoffeeMaker.java,

```
public static void mainMenu() {
    System.out.println("1. Add a recipe");
    System.out.println("2. Delete a recipe");
    System.out.println("3. Edit a recipe");
    System.out.println("4. Add inventory");
    System.out.println("5. Check inventory");
    System.out.println("6. Make coffee");
    System.out.println("0. Exit\n");
```

can be changed to

```
public static void Menu() {
    System.out.println("1. Add a recipe");
    System.out.println("2. Delete a recipe");
    System.out.println("3. Edit a recipe");
    System.out.println("4. Add inventory");
    System.out.println("5. Check inventory");
    System.out.println("6. Make coffee");
    System.out.println("0. Exit\n");
```

The program never continues

Valid but not useful (Syntactically correct)

In Inventory.java,

```
if (amtSugar <= 0) {
                    Inventory.sugar += amtSugar;
                }
```

Can be changed to

```
if (amtSugar >= 0) {
                    Inventory.sugar += amtSugar;
                }
```

The mutation will not effect the execution of the program

addSugarTest() and addSugarErrorTest() in InventoryTest.Java figures this mutant out.

Useful (the mutant should correct the program)

In Recipebook.java

```
public synchronized String deleteRecipe(int recipeToDelete) {
                if (recipeArray[recipeToDelete] != null) {
                        String recipeName = recipeArray[recipeToDelete].getName();
                        recipeArray[recipeToDelete] = new Recipe();
                        return recipeName;
                }
```

Can be changed to

```
public synchronized String deleteRecipe(int recipeToDelete) {
                if (recipeArray[recipeToDelete] != null) {
                        String recipeName = recipeArray[recipeToDelete].getName();
                        recipeArray[recipeToDelete] = null;
                        return recipeName;
                }
```

The program used to not delete the entry with the delete function, with this change the program executed correctly deleting the entry called this function on.

In recipeBookTest.java, the testDeleteRecipeNull() checks for that issue and reports it.

Equivalent (mutant does not change the execution of the program)

In inventory.java

```
public synchronized boolean useIngredients(Recipe r) {
        if (enoughIngredients(r)) {
                Inventory.coffee += r.getAmtCoffee();
```

Can be changed to

```
public synchronized boolean useIngredients(Recipe r) {
        if (enoughIngredients(r)) {
                Inventory.coffee -= r.getAmtCoffee();
```

That's an important change but it does not change the functionality.

In InventoryTest.java, useIngredientsTest() handles this mutant.

Non-Equivalent (mutant does change the execution of the program)

In main.java

```
if(Inventory.coffee < r.getAmtCoffee()) {
   isEnough = false;
}
```

can be changed to:

```
if(Inventory.coffee < r.getAmtMilk()) {
   isEnough = false;
}
```

The whole execution changes

2. Non-Equivalent

Class: Inventory

Change the code from:

```
if(Inventory.coffee < r.getAmtCoffee()) {
        isEnough = false;
 }
```

To:

```
if(Inventory.coffee < r.getAmtMilk()) {
        isEnough = false;
 }
```

In RecipeBook:

If you change the setName() to setPrice():

```
public synchronized String editRecipe(int recipeToEdit, Recipe newRecipe) {
                if (recipeArray[recipeToEdit] != null) {
                        String recipeName = recipeArray[recipeToEdit].getName();
                        newRecipe.setName("");
                        recipeArray[recipeToEdit] = newRecipe;
                        return recipeName;
                } else {
```

```
                return null;
            }
        }
```

to:

```
public synchronized String editRecipe(int recipeToEdit, Recipe newRecipe) {
            if (recipeArray[recipeToEdit] != null) {
                String recipeName = recipeArray[recipeToEdit].getName();
                newRecipe.setPrice("");
                recipeArray[recipeToEdit] = newRecipe;
                return recipeName;
            } else {
                return null;
            }
        }
```

3.

The mutation for Inventory can be checked at the enoughIngredientsTest().

The mutation for RecipeBook can be checked at the methods: testEditRecipeName() and testEditRecipeNull().