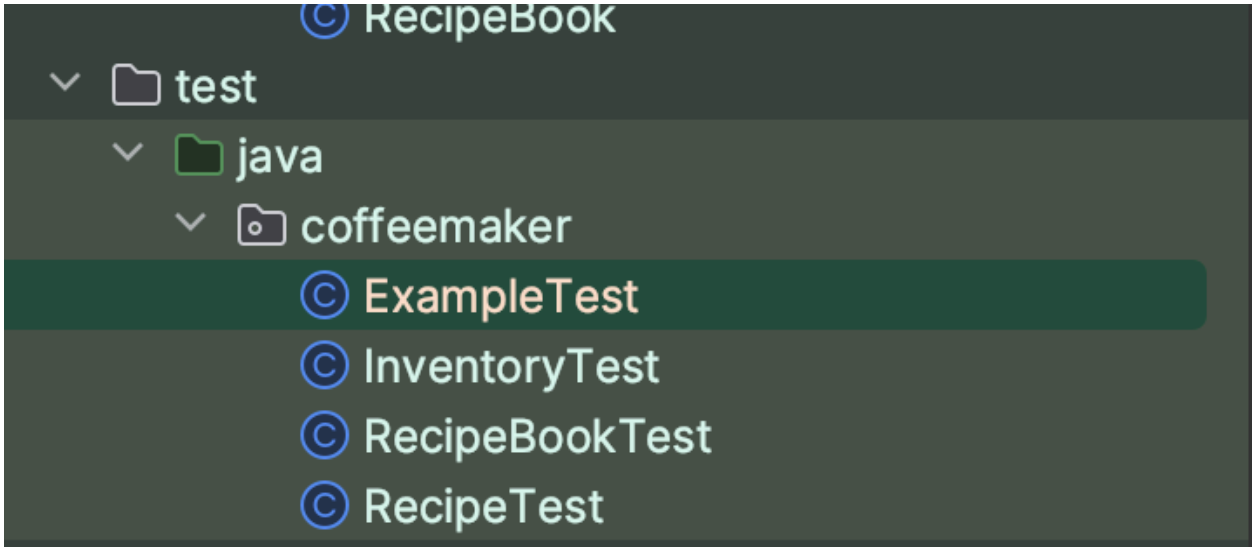
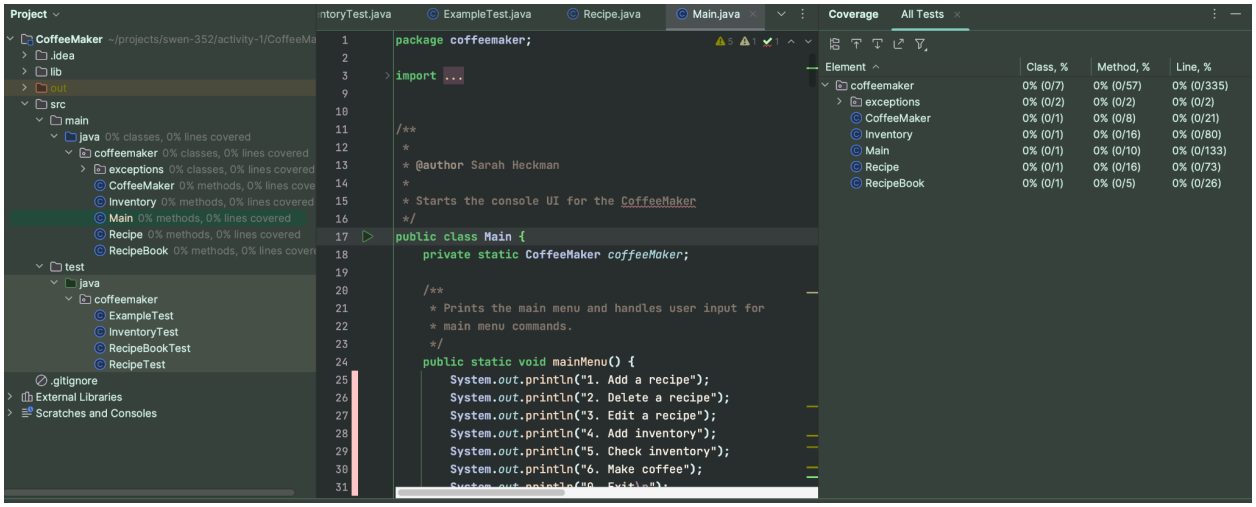
STEP 1



STEP 2



STEP 3

Defect 1

Class Name: RecipeBook
Method name: editRecipe

Original Source code with defect:

```java
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;
    }
}
```

1–2 sentences explaining why it *is* a defect, and how it can be fixed:

newRecipe.setName does not do anything in the original code. This is a problem as it makes it so that you are just replacing the recipe with a new one, not editing all the old data. We can fix it by changing setName() to the old name so that all the attributes are changed but the recipe name will not.

JUnit code that catches defect:

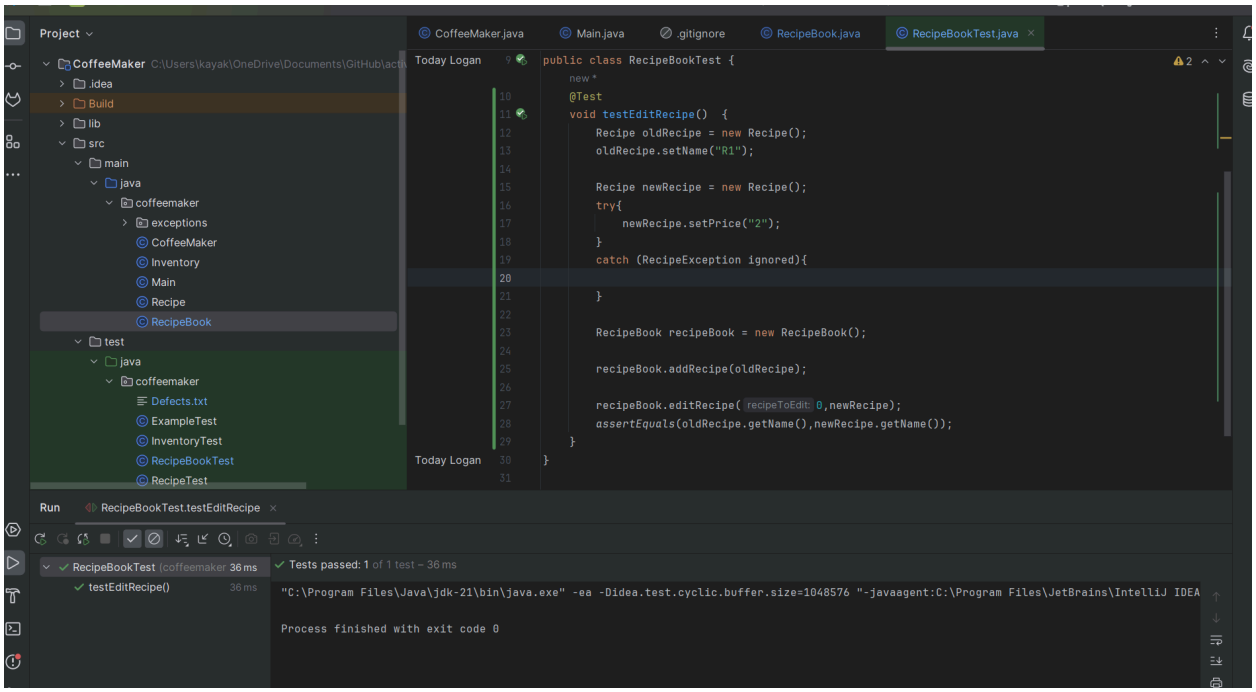
```java
public class RecipeBookTest {
    @Test
    void testEditRecipe()  {
        Recipe oldRecipe = new Recipe();
        oldRecipe.setName("R1");

        Recipe newRecipe = new Recipe();
        try{
            newRecipe.setPrice("2");
        }
        catch (RecipeException ignored){

        }

        RecipeBook recipeBook = new RecipeBook();

        recipeBook.addRecipe(oldRecipe);

        recipeBook.editRecipe( recipeToEdit: 0,newRecipe);
        assertEquals(oldRecipe.getName(),newRecipe.getName());
    }
}
```

```
Tests failed: 1 of 1 test – 49 ms
"C:\Program Files\Java\jdk-21\bin\java.exe" -ea -Didea.test.cyclic.buffer.size=1048576 "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA

org.opentest4j.AssertionFailedError:
Expected :R1
Actual   :
<Click to see difference>
```

Fixed code:

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

JUnit passing:



Defect 2

Class Name: Inventory
Method name: addSugar
Original Source code with defect:

```java
public synchronized void addSugar(String sugar) throws InventoryException {
    int amtSugar = 0;
    try {
        amtSugar = Integer.parseInt(sugar);
    } catch (NumberFormatException e) {
        throw new InventoryException("Units of sugar must be a positive integer");
    }
    if (amtSugar <= 0) {
        Inventory.sugar += amtSugar;
    } else {
        throw new InventoryException("Units of sugar must be a positive integer");
    }
}
```

1–2 sentences explaining why it *is* a defect, and how it can be fixed: The sugar amount when adding sugar is only incrementing the sugar amount whenever the amount added is less than 0, so any positive integer causes the function to throw an error. To fix this, the function should use ">=" rather than "<=".

JUnit code that catches defect:

```java
@Test
public void addSugarTestHappy() {
    int initialSugar = inventory.getSugar();

    try {
        inventory.addSugar(AMOUNT);
    } catch (InventoryException ie) {
        Assertions.fail("Error should not have been thrown");
    }

    int expectedSugar = inventory.getSugar();

    Assertions.assertNotEquals(initialSugar, expectedSugar);
    Assertions.assertEquals(expectedSugar, actual: initialSugar * 2);
}
```

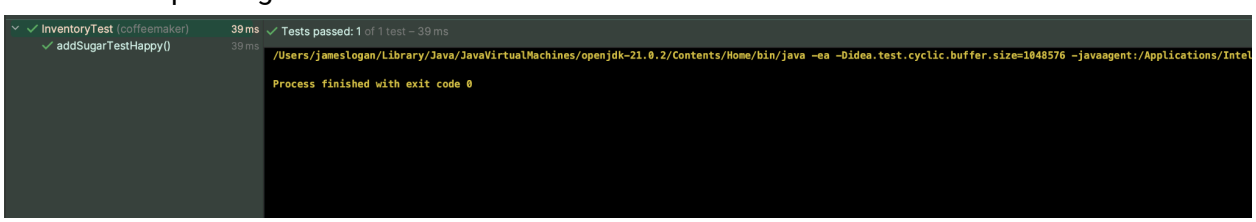
JUnit failing:

```
InventoryTest (coffeemaker)   36 ms   Tests failed: 1 of 1 test – 36 ms
  addSugarTestHappy()         36 ms

coffeemaker.exceptions.InventoryException: Units of sugar must be a positive integer

    at coffeemaker.Inventory.addSugar(Inventory.java:185)
    at coffeemaker.InventoryTest.setup(InventoryTest.java:22) <1 internal line>
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1596)
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1596)
```

Fixed code:

```java
/**
 * Add the number of sugar units in the inventory
 * to the current amount of sugar units.
 * @param sugar
 * @throws InventoryException
 */
public synchronized void addSugar(String sugar) throws InventoryException {
    int amtSugar = 0;
    try {
        amtSugar = Integer.parseInt(sugar);
    } catch (NumberFormatException e) {
        throw new InventoryException("Units of sugar must be a positive integer");
    }
    if (amtSugar >= 0) {
        Inventory.sugar += amtSugar;
    } else {
        throw new InventoryException("Units of sugar must be a positive integer");
    }
}
```

JUnit passing:



Defect 3

Class Name: Inventory
Method name: useIngredients
Original Source code with defect:

```java
/**
 * Removes the ingredients used to make the specified
 * recipe.  Assumes that the user has checked that there
 * are enough ingredients to make
 * @param r
 */
public synchronized boolean useIngredients(Recipe r) {
    if (enoughIngredients(r)) {
        Inventory.coffee += r.getAmtCoffee();
        Inventory.milk -= r.getAmtMilk();
        Inventory.sugar -= r.getAmtSugar();
        Inventory.chocolate -= r.getAmtChocolate();
        return true;
    } else {
        return false;
    }
}

/**
```

<u>1–2 sentences explaining why it *is* a defect, and how it can be fixed:</u>

The coffee amount is being increased instead of removed. It can be fixed by changing the "+=" operation to a "-=" operation.

JUnit code that catches defect:

```java
@Test
public void useIngredientsTestEnough() {
    int initialCoffee = inventory.getCoffee();
    int initialChocolate = inventory.getChocolate();
    int initialMilk = inventory.getMilk();
    int initialSugar = inventory.getSugar();

    boolean result = inventory.useIngredients(recipe);

    int expectedCoffee = inventory.getCoffee();
    int expectedCohocolate = inventory.getChocolate();
    int expectedMilk = inventory.getMilk();
    int expectedSugar = inventory.getSugar();

    Assertions.assertTrue(result);
    Assertions.assertTrue( condition: expectedCoffee < initialCoffee);
    Assertions.assertTrue( condition: expectedCohocolate < initialChocolate);
    Assertions.assertTrue( condition: expectedSugar < initialSugar);
    Assertions.assertTrue( condition: expectedMilk < initialMilk);
}
}
```

JUnit failing:

```
InventoryTest (coffeemaker)                35 ms   Tests failed: 1 of 1 test – 35 ms
    useIngredientsTestEnough()             35 ms
/Users/jameslogan/Library/Java/JavaVirtualMachines/openjdk-21.0.2/Contents/Home/bin/java -ea -Didea.test.cyclic.buffer.size=1048576 -javaagent:/Applications/Intel

org.opentest4j.AssertionFailedError:
Expected :true
Actual   :false
<Click to see difference>

<6 internal lines>
    at coffeemaker.InventoryTest.useIngredientsTestEnough(InventoryTest.java:57) <1 internal line>
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1596)
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1596)
```

Fixed code:

```java
/**
 * Removes the ingredients used to make the specified
 * recipe.  Assumes that the user has checked that there
 * are enough ingredients to make
 * @param r
 */
public synchronized boolean useIngredients(Recipe r) {
    if (enoughIngredients(r)) {
        Inventory.coffee -= r.getAmtCoffee();
        Inventory.milk -= r.getAmtMilk();
        Inventory.sugar -= r.getAmtSugar();
        Inventory.chocolate -= r.getAmtChocolate();
        return true;
    } else {
        return false;
    }
}
```

JUnit passing:

```
InventoryTest (coffeemaker)        34 ms   Tests passed: 1 of 1 test – 34 ms
  useIngredientsTestEnough()       34 ms
/Users/jameslogan/Library/Java/JavaVirtualMachines/openjdk-21.0.2/Contents/Home/bin/java -ea -Didea.test.cyclic.buffer.size=1048576 -javaagent:/Applications/Intel

Process finished with exit code 0
```

Defect 4

Class Name: Recipe.java
Method name: setName
Original Source code with defect:

```java
/**
* @param name   The name to set.
*/
6 usages  James Logan
public void setName(String name) {
    if(name != null) {
        this.name = name;
    }
}
```

1–2 sentences explaining why it *is* a defect, and how it can be fixed:

Recipes that have the same names but with extra spaces should be treated as the same name. You should not be able to add two recipes with the same name but extra spaces to the recipe book.

JUnit code that catches defect:

```java
James Logan *
public class RecipeTest {
    new *
    @Test
    void testAddRecipeWithSameName() {
        CoffeeMaker coffeeMaker = new CoffeeMaker();
        String name1 = "AriGregLogan";
        Recipe recipe1 = new Recipe();
        recipe1.setName(name1);
        String name2 = "AriGregLogan  ";
        Recipe recipe2 = new Recipe();
        recipe2.setName(name2);
        coffeeMaker.addRecipe(recipe1);

        assertEquals(recipe1, recipe2);
        assertFalse(coffeeMaker.addRecipe(recipe2));
    }
```

JUnit failing:

```
Run: Main × All Tests ×
Tests failed: 1, passed: 4 of 5 tests – 23 ms

<default package>                23 ms
  RecipeTest                     19 ms
  ExampleTest                     4 ms
    testAdd()                     3 ms
    testSetException()            1 ms
    testIsEmpty()

java.lang.AssertionError: expected:<AriGregLogan> but was:<AriGregLogan  >
Expected :AriGregLogan
Actual   :AriGregLogan
<Click to see difference>

<1 internal line>
  at org.junit.Assert.failNotEquals(Assert.java:835) <2 internal lines>
  at coffeemaker.RecipeTest.testAddRecipeWithSameName(RecipeTest.java:19) <1 internal line>
  at java.base/java.util.ArrayList.forEach(ArrayList.java:1596)
  at java.base/java.util.ArrayList.forEach(ArrayList.java:1596)
```

Fixed code:

```
/**
* @param name   The name to set.
*/
6 usages   James Logan *
public void setName(String name) {
    if(name != null) {
        this.name = name.strip();
    }
}
/**
```

JUnit passing:

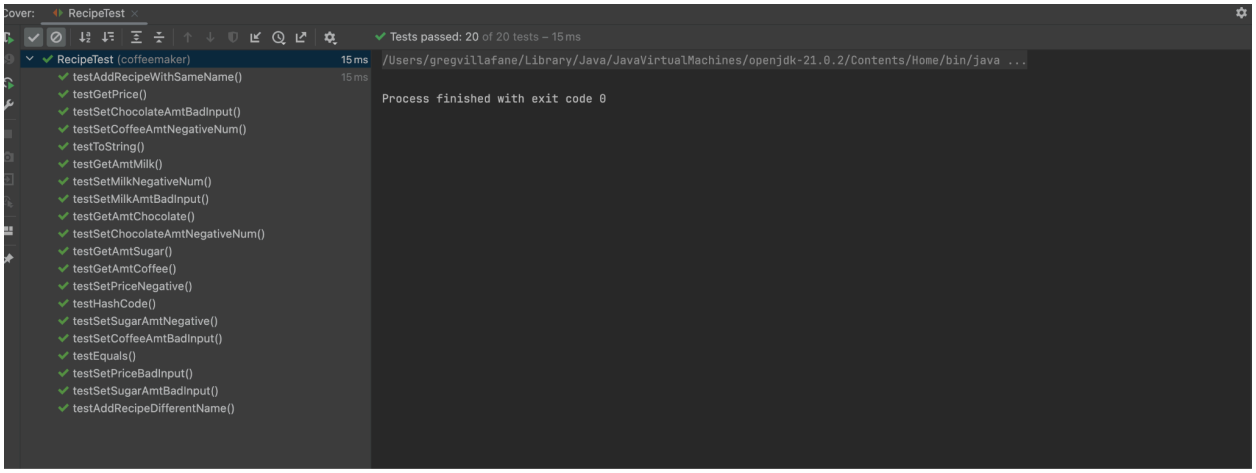
```
Tests passed: 5 of 5 tests – 14 ms

<default package>                14 ms
  RecipeTest                     11 ms
  ExampleTest                     3 ms
    testAdd()                     2 ms
    testSetException()            1 ms
    testIsEmpty()

/Users/gregvillafane/Library/Java/JavaVirtualMachines/openjdk-21.0.2/Contents/Home/bin/java ...

Process finished with exit code 0
```

STEP 4

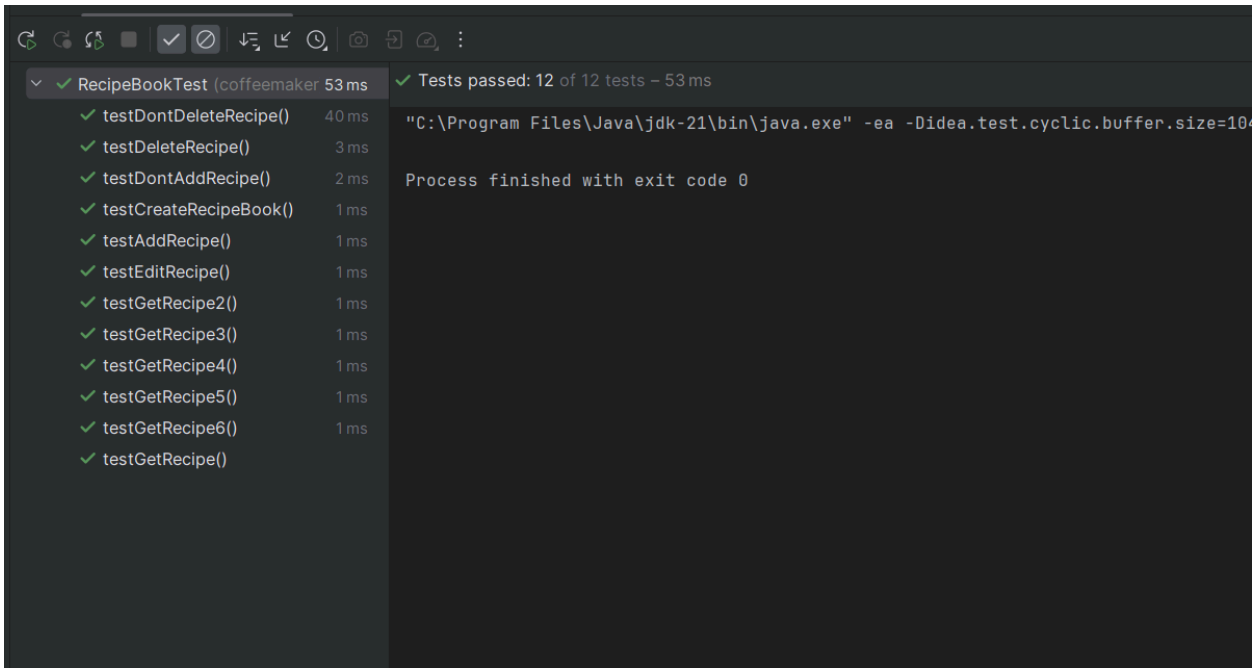
CoffeeMaker	100% (1/1)	25% (2/8)	19% (4/21)
Inventory	100% (1/1)	100% (16/16)	100% (80/80)
Main	0% (0/1)	0% (0/10)	0% (0/133)
Recipe	100% (1/1)	100% (16/16)	100% (73/73)
RecipeBook	100% (1/1)	100% (5/5)	100% (26/26)

STEP 5

Recipe Tests:



RecipeBook Tests:



Inventory Tests:

InventoryTest (coffeemaker)	55 ms
useIngredientsTestEnough()	22 ms
addCoffeeTestInvalidNumber()	2 ms
addSugarTestInvalidNumber()	2 ms
setChocolateTest()	1 ms
addMilkTestInvalidNumber()	2 ms
addMilkTestHappy()	2 ms
getCoffeeTest()	1 ms
addChocolateTestHappy()	3 ms
testToStringHappy()	3 ms
addCoffeeTestNegativeNumber()	1 ms
addCoffeeTestHappy()	2 ms
setCoffeeTest()	1 ms
getSugarTest()	1 ms
getChocolateTest()	1 ms
setMilkTest()	1 ms
getMilkTest()	1 ms
addSugarTestNegativeNumber()	1 ms
addChocolateTestNegativeNumber	1 ms
setSugarTest()	1 ms
addMilkTestNegativeNumber()	2 ms
addChocolateTestInvalidNumber()	1 ms
useIngredientsTestNotEnough()	1 ms
addSugarTestHappy()	2 ms

Tests passed: 23 of 23 tests – 55 ms

```
/Users/jameslogan/Library/Java/Jav

Process finished with exit code 0
```