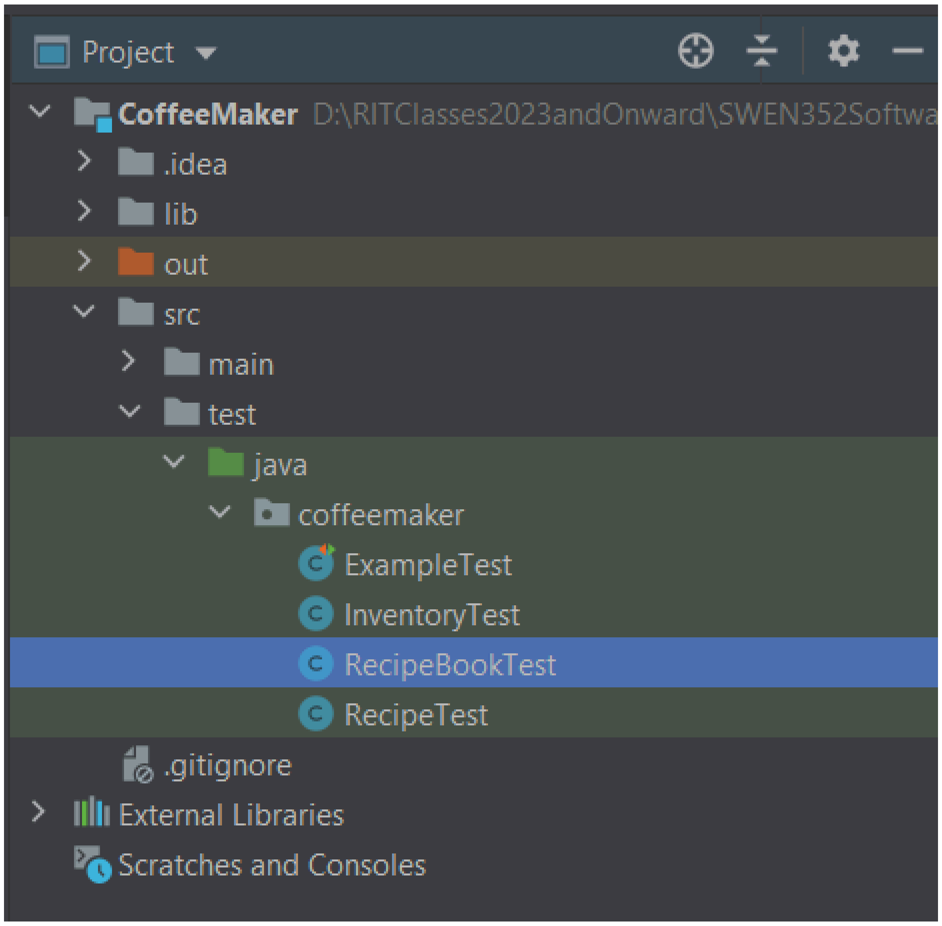
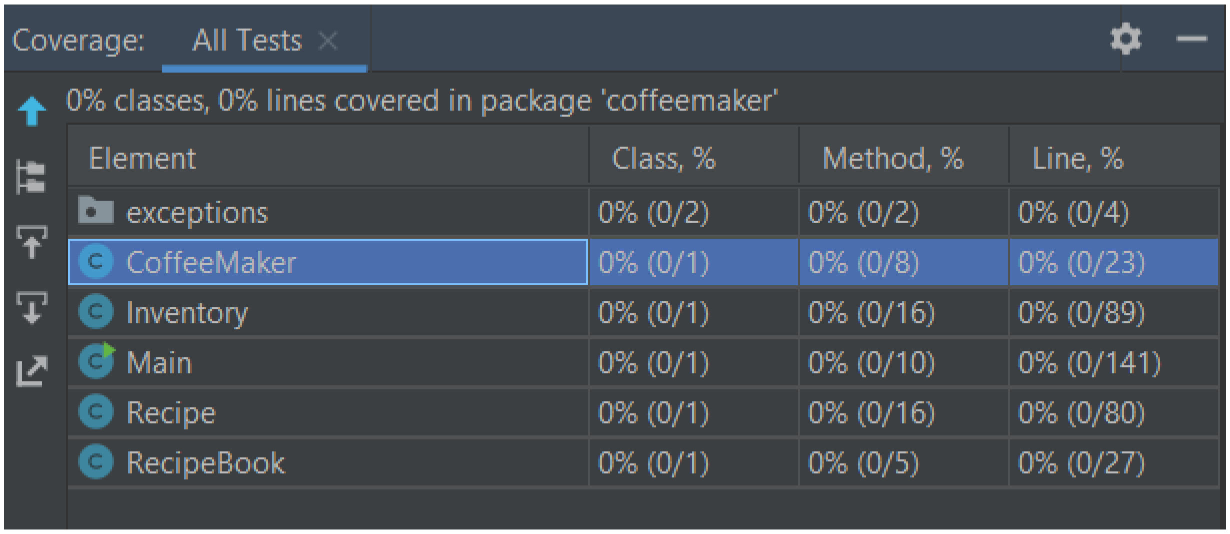
Deliverable 1:



Deliverable 2:



Coverage: All Tests

0% classes, 0% lines covered in package 'coffeemaker'

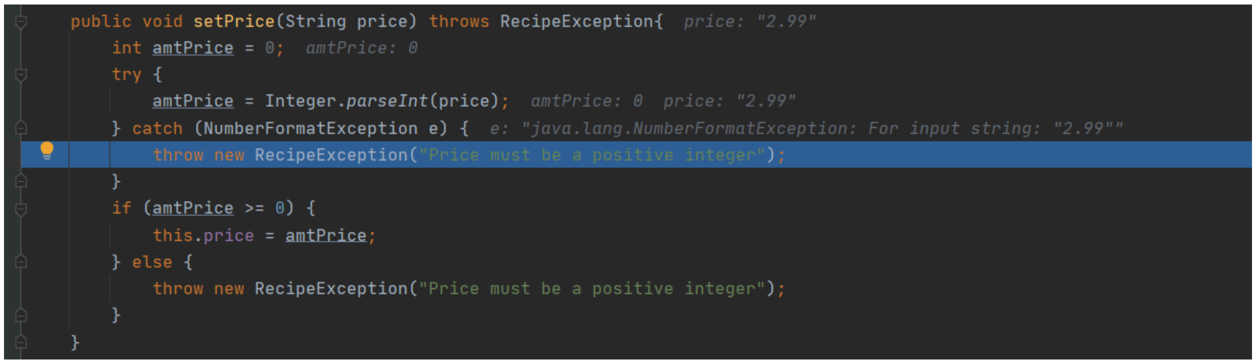
Element	Class, %	Method, %	Line, %
exceptions	0% (0/2)	0% (0/2)	0% (0/4)
CoffeeMaker	0% (0/1)	0% (0/8)	0% (0/23)
Inventory	0% (0/1)	0% (0/16)	0% (0/89)
Main	0% (0/1)	0% (0/10)	0% (0/141)
Recipe	0% (0/1)	0% (0/16)	0% (0/80)
RecipeBook	0% (0/1)	0% (0/5)	0% (0/27)

Deliverable 3:

Defect 1: When adding a recipe and inputting price, using a decimal will give an error and say that it needs a positive integer

Class Defect Is In: Recipe
Function/Method Defect is in: setPrice(String price)

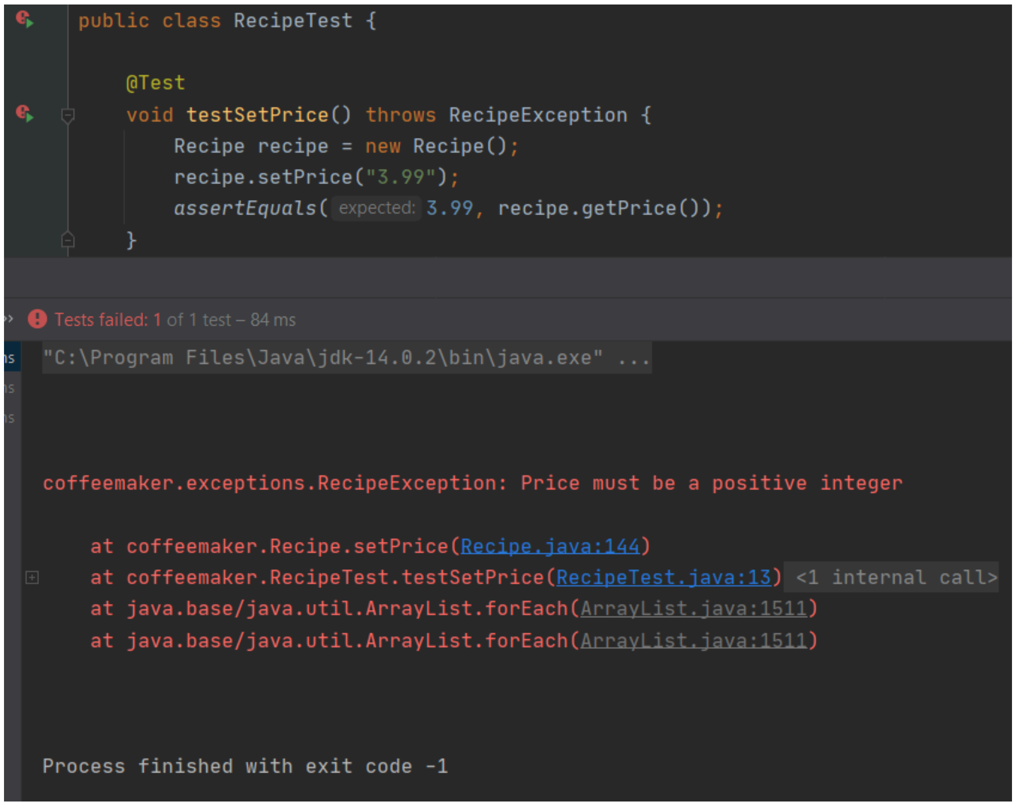
ScreenShot Of Offending Source Code:



```
public void setPrice(String price) throws RecipeException{  price: "2.99"
    int amtPrice = 0;  amtPrice: 0
    try {
        amtPrice = Integer.parseInt(price);  amtPrice: 0  price: "2.99"
    } catch (NumberFormatException e) {  e: "java.lang.NumberFormatException: For input string: "2.99""
        throw new RecipeException("Price must be a positive integer");
    }
    if (amtPrice >= 0) {
        this.price = amtPrice;
    } else {
        throw new RecipeException("Price must be a positive integer");
    }
}
```

Explanation Of Defect:
As seen when running the debugger above, inputting a decimal fails because Integer.parseInt() is being used when we should be using Double.parseDouble(). This defect goes against expected input. Prices in the real world are decimals and not integers. Also, it doesn't prompt the user to enter an integer when entering price.

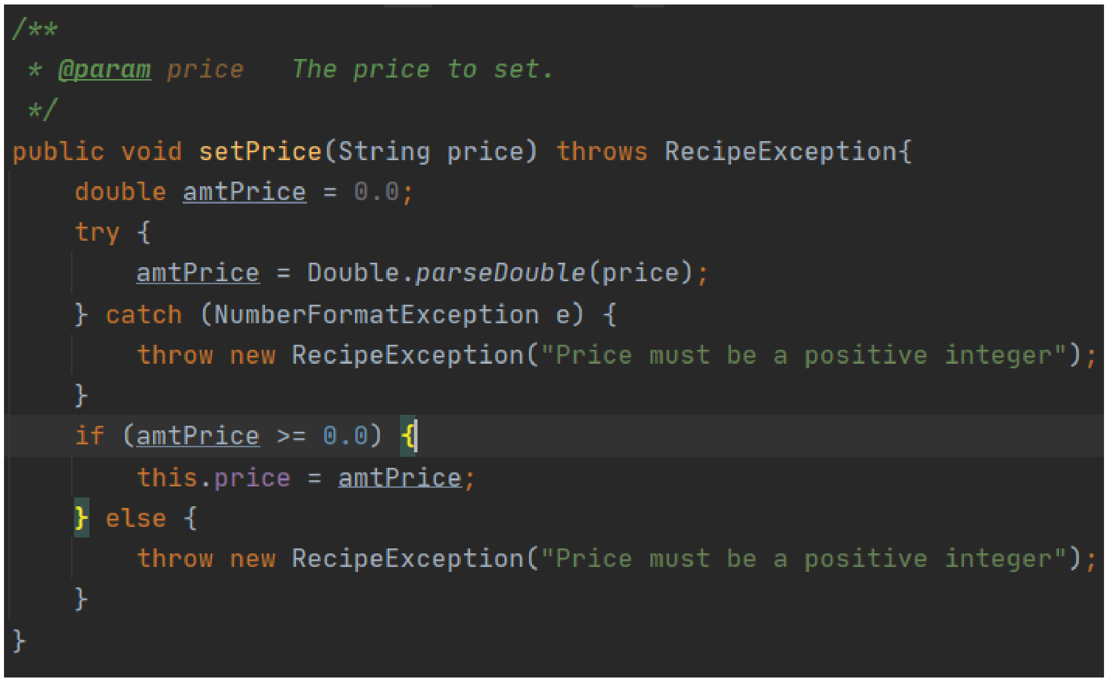
JUnit code catching defect:



```
public class RecipeTest {

    @Test
    void testSetPrice() throws RecipeException {
        Recipe recipe = new Recipe();
        recipe.setPrice("3.99");
        assertEquals( expected: 3.99, recipe.getPrice());
    }
```

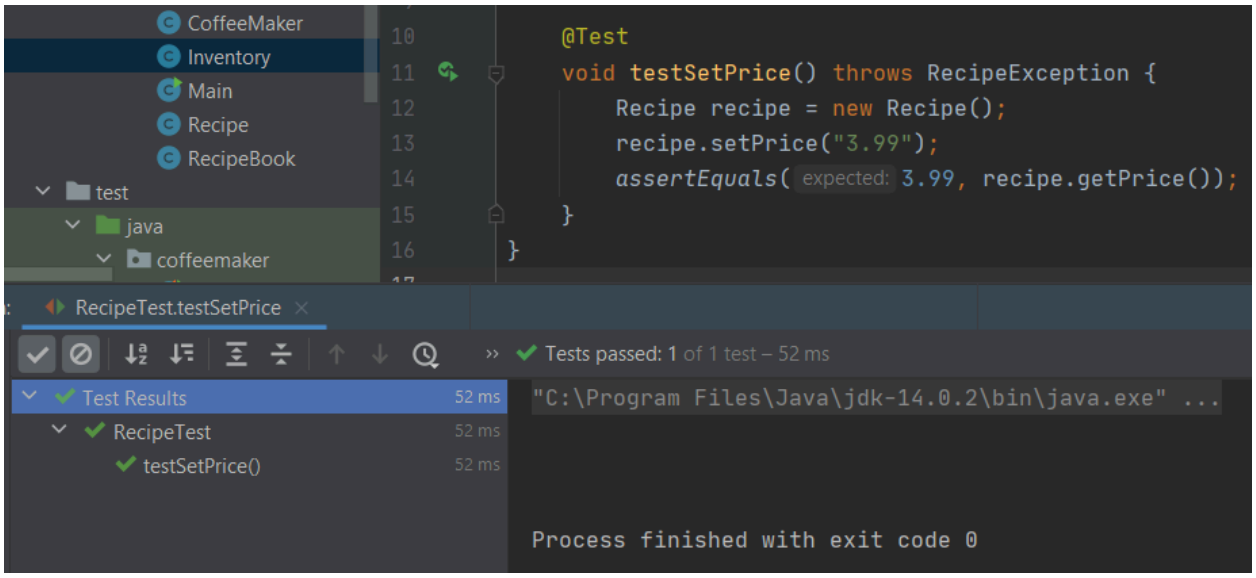
```
Tests failed: 1 of 1 test – 84 ms
"C:\Program Files\Java\jdk-14.0.2\bin\java.exe" ...


coffeemaker.exceptions.RecipeException: Price must be a positive integer

    at coffeemaker.Recipe.setPrice(Recipe.java:144)
    at coffeemaker.RecipeTest.testSetPrice(RecipeTest.java:13) <1 internal call>
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)


Process finished with exit code -1
```

Updated source code with defect fixed:



```
/**
 * @param price   The price to set.
 */
public void setPrice(String price) throws RecipeException{
    double amtPrice = 0.0;
    try {
        amtPrice = Double.parseDouble(price);
    } catch (NumberFormatException e) {
        throw new RecipeException("Price must be a positive integer");
    }
    if (amtPrice >= 0.0) {
        this.price = amtPrice;
    } else {
        throw new RecipeException("Price must be a positive integer");
    }
}
```

Test Method Passing:



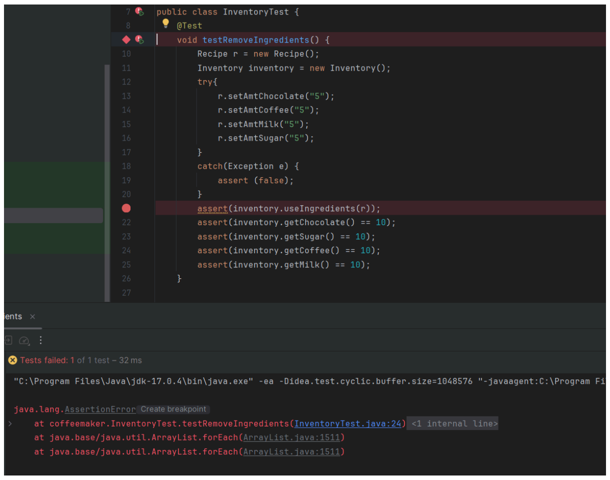
```
    @Test
    void testSetPrice() throws RecipeException {
        Recipe recipe = new Recipe();
        recipe.setPrice("3.99");
        assertEquals( expected: 3.99, recipe.getPrice());
    }
}
```

```
Tests passed: 1 of 1 test – 52 ms
Test Results 52 ms
  RecipeTest 52 ms
    testSetPrice() 52 ms

"C:\Program Files\Java\jdk-14.0.2\bin\java.exe" ...

Process finished with exit code 0
```

Defect 2: Inventory, line 220

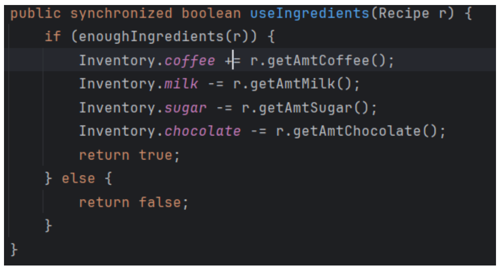
When trying to make a recipe with an inventory, the amount of coffee required is added to the inventory instead of subtracted from. It can easily be fixed just by changing the addition sign to a subtraction sign.

Before:



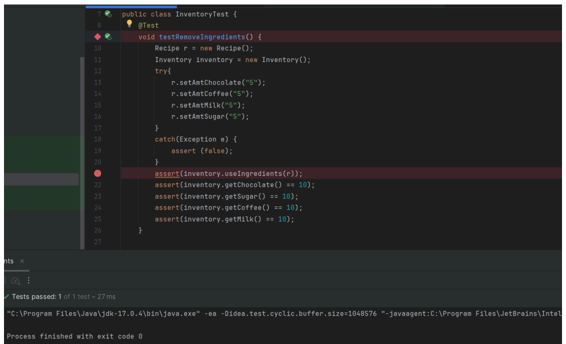
```
public class InventoryTest {
    @Test
    void testRemoveIngredients() {
        Recipe r = new Recipe();
        Inventory inventory = new Inventory();
        try{
            r.setAmtChocolate("5");
            r.setAmtCoffee("5");
            r.setAmtMilk("5");
            r.setAmtSugar("5");
        }
        catch(Exception e) {
            assert (false);
        }
        assert(inventory.useIngredients(r));
        assert(inventory.getChocolate() == 10);
        assert(inventory.getSugar() == 10);
        assert(inventory.getCoffee() == 10);
        assert(inventory.getMilk() == 10);
    }
```

```
Tests failed: 1 of 1 test – 32 ms
"C:\Program Files\Java\jdk-17.0.4\bin\java.exe" -ea -Didea.test.cyclic.buffer.size=1048576 "-javaagent:C:\Program Fi

java.lang.AssertionError Create breakpoint
    at coffeemaker.InventoryTest.testRemoveIngredients(InventoryTest.java:24) <1 internal line>
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)
```



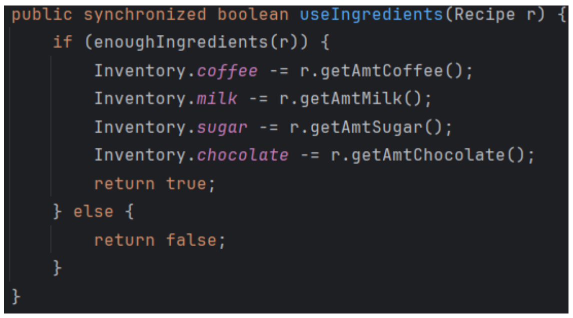
```
public synchronized boolean useIngredients(Recipe r) {
    if (enoughIngredients(r)) {
        Inventory.coffee += r.getAmtCoffee();
        Inventory.milk -= r.getAmtMilk();
        Inventory.sugar -= r.getAmtSugar();
        Inventory.chocolate -= r.getAmtChocolate();
        return true;
    } else {
        return false;
    }
}
```

After:



```
public class InventoryTest {
    @Test
    void testRemoveIngredients() {
        Recipe r = new Recipe();
        Inventory inventory = new Inventory();
        try{
            r.setAmtChocolate("5");
            r.setAmtCoffee("5");
            r.setAmtMilk("5");
            r.setAmtSugar("5");
        }
        catch(Exception e) {
            assert (false);
        }
        assert(inventory.useIngredients(r));
        assert(inventory.getChocolate() == 10);
        assert(inventory.getSugar() == 10);
        assert(inventory.getCoffee() == 10);
        assert(inventory.getMilk() == 10);
    }
```

```
Tests passed: 1 of 1 test – 27 ms
"C:\Program Files\Java\jdk-17.0.4\bin\java.exe" -ea -Didea.test.cyclic.buffer.size=1048576 "-javaagent:C:\Program Files\JetBrains\Intel

Process finished with exit code 0
```



```
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;
    }
}
```

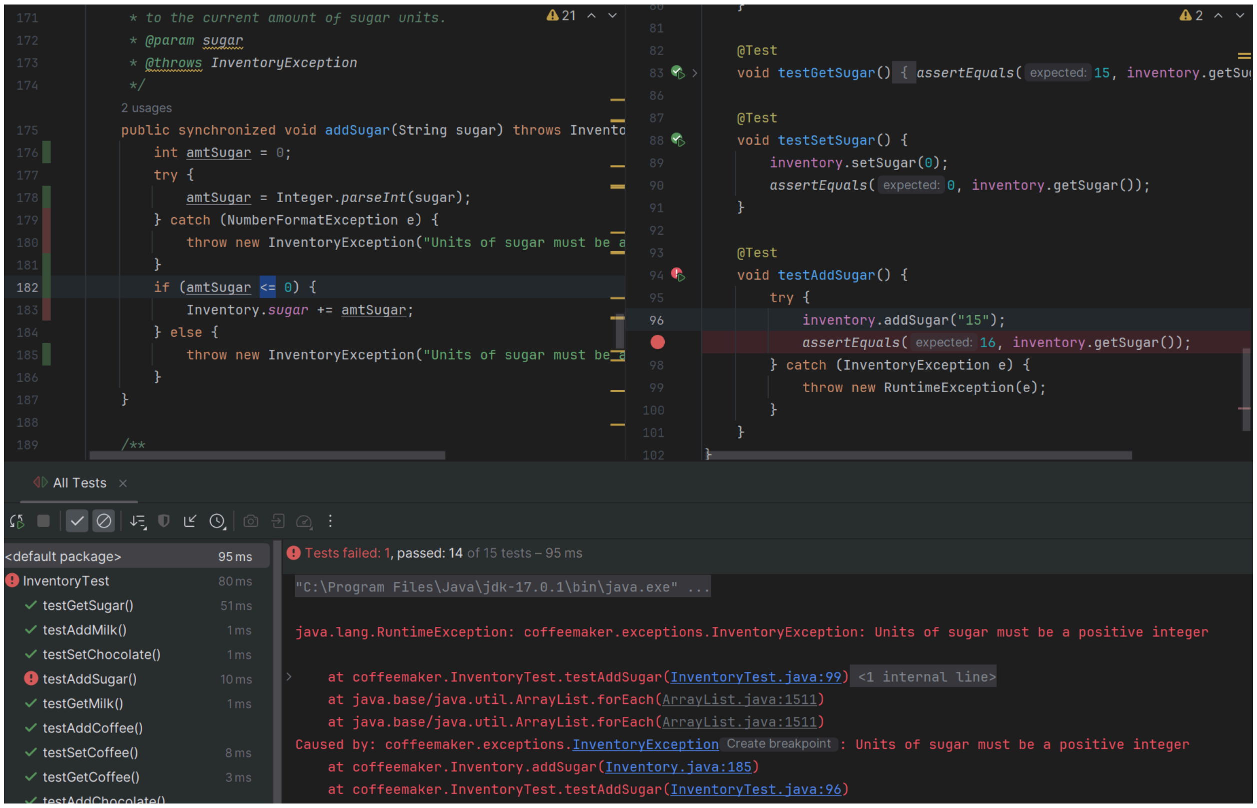
Defect 3: When trying to add to the inventory, the command line interface tells us "Inventory was not added" no matter what integers we provide which is incorrect because checking the inventory afterwards shows that the amount of coffee and milk I've added was added correctly. It's just Sugar and Chocolate that is not being successfully added. This defect will focus on fixing the problem with **adding chocolate** to the inventory.

Class Defect Is In: Inventory
Function/Method Defect is in: setPrice(String price)

Defect 4: When utilizing the addSugar function, the function will always throw an InventoryException because the incorrect inequality is used.

Class Defect Is In: Inventory
Function/Method Defect is in: addSugar(String sugar)

Before:



```
171      * to the current amount of sugar units.
172      * @param sugar
173      * @throws InventoryException
174      */
    2 usages
175     public synchronized void addSugar(String sugar) throws Invento
176         int amtSugar = 0;
177         try {
178             amtSugar = Integer.parseInt(sugar);
179         } catch (NumberFormatException e) {
180             throw new InventoryException("Units of sugar must be a
181         }
182         if (amtSugar <= 0) {
183             Inventory.sugar += amtSugar;
184         } else {
185             throw new InventoryException("Units of sugar must be a
186         }
187     }
188
189     /**
```

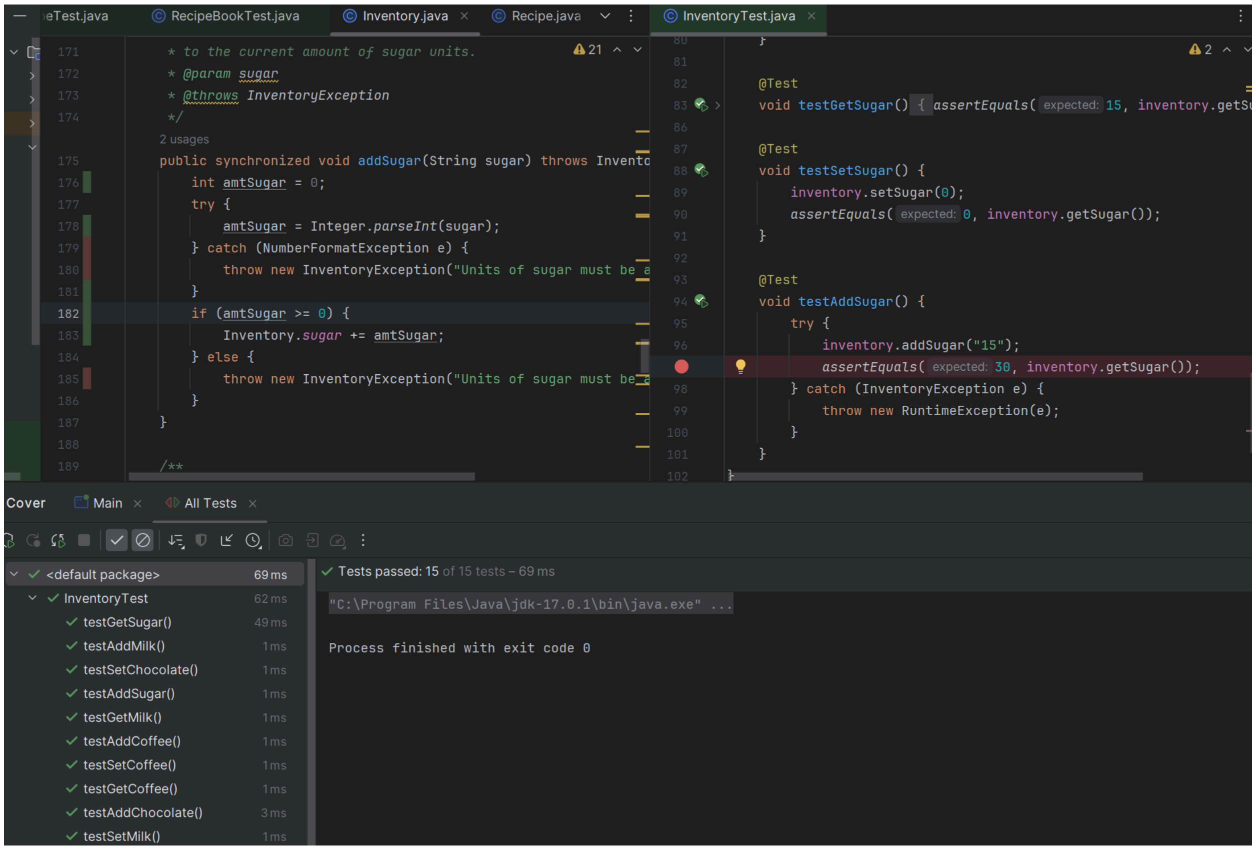
```
81
82     @Test
83     void testGetSugar() { assertEquals( expected: 15, inventory.getSu
86
87     @Test
88     void testSetSugar() {
89         inventory.setSugar(0);
90         assertEquals( expected: 0, inventory.getSugar());
91     }
92
93     @Test
94     void testAddSugar() {
95         try {
96             inventory.addSugar("15");
97             assertEquals( expected: 16, inventory.getSugar());
98         } catch (InventoryException e) {
99             throw new RuntimeException(e);
100        }
101    }
102 }
```

All Tests

```
<default package>        95 ms
InventoryTest            80 ms
  ✓ testGetSugar()       51 ms
  ✓ testAddMilk()         1 ms
  ✓ testSetChocolate()    1 ms
  ! testAddSugar()       10 ms
  ✓ testGetMilk()         1 ms
  ✓ testAddCoffee()
  ✓ testSetCoffee()       8 ms
  ✓ testGetCoffee()       3 ms
  ✓ testAddChocolate()
```

```
Tests failed: 1, passed: 14 of 15 tests – 95 ms

"C:\Program Files\Java\jdk-17.0.1\bin\java.exe" ...

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

    at coffeemaker.InventoryTest.testAddSugar(InventoryTest.java:99) <1 internal line>
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)
Caused by: coffeemaker.exceptions.InventoryException Create breakpoint : Units of sugar must be a positive integer
    at coffeemaker.Inventory.addSugar(Inventory.java:185)
    at coffeemaker.InventoryTest.testAddSugar(InventoryTest.java:96)
```

After:



eTest.java | RecipeBookTest.java | Inventory.java | Recipe.java | InventoryTest.java

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

    /**
```

```java
    }

    @Test
    void testGetSugar() { assertEquals( expected: 15, inventory.getSu

    @Test
    void testSetSugar() {
        inventory.setSugar(0);
        assertEquals( expected: 0, inventory.getSugar());
    }

    @Test
    void testAddSugar() {
        try {
            inventory.addSugar("15");
            assertEquals( expected: 30, inventory.getSugar());
        } catch (InventoryException e) {
            throw new RuntimeException(e);
        }
    }
}
```

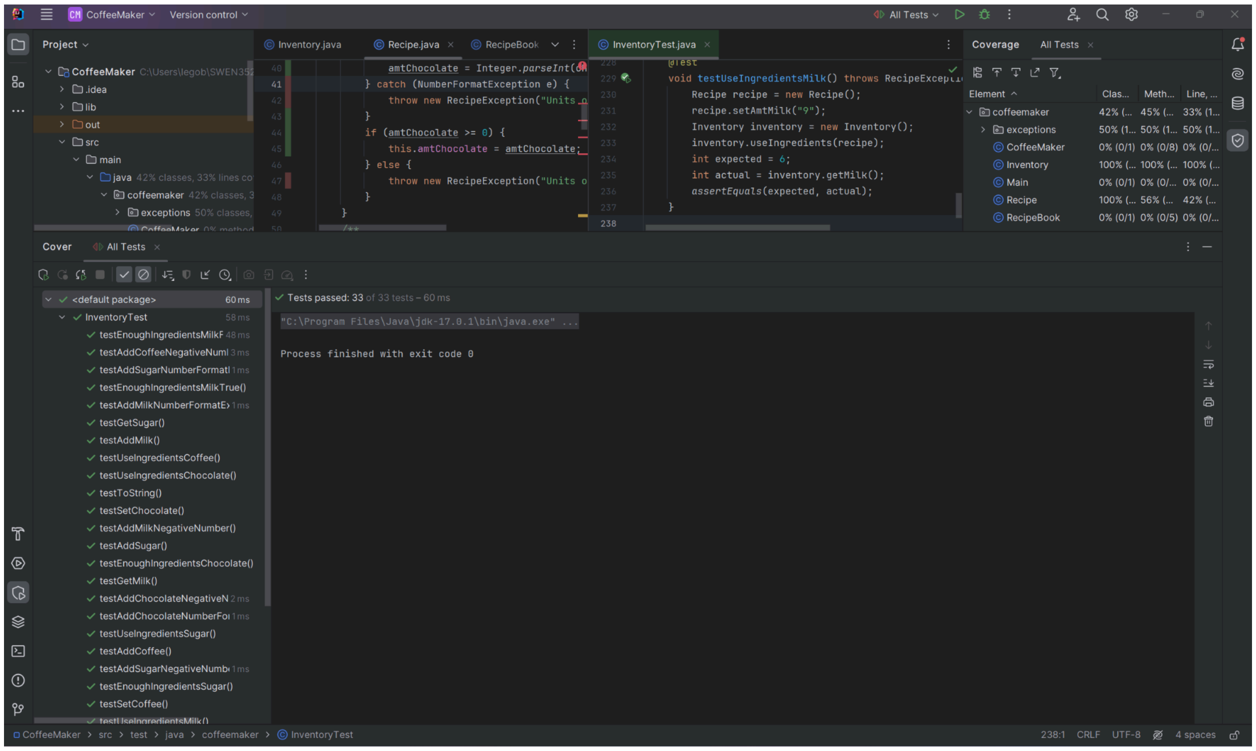
Cover | Main | All Tests

```
<default package>        69 ms
  InventoryTest          62 ms
    testGetSugar()       49 ms
    testAddMilk()         1 ms
    testSetChocolate()    1 ms
    testAddSugar()        1 ms
    testGetMilk()         1 ms
    testAddCoffee()       1 ms
    testSetCoffee()       1 ms
    testGetCoffee()       1 ms
    testAddChocolate()    3 ms
    testSetMilk()         1 ms
```

```
Tests passed: 15 of 15 tests – 69 ms

"C:\Program Files\Java\jdk-17.0.1\bin\java.exe" ...

Process finished with exit code 0
```

For Gabe for deliverable 4: satisfy code coverage for recipe

Coverage for Inventory



```
CoffeeMaker C:\Users\legob\SWEN352
  .idea
  lib
  out
  src
    main
      java 42% classes, 33% lines co
        coffeemaker 42% classes, 3
          exceptions 50% classes,
          CoffeeMaker 0% method
```

```java
                amtChocolate = Integer.parseInt(ch
            } catch (NumberFormatException e) {
                throw new RecipeException("Units o
            }
            if (amtChocolate >= 0) {
                this.amtChocolate = amtChocolate;
            } else {
                throw new RecipeException("Units o
            }
        }
        /**
```

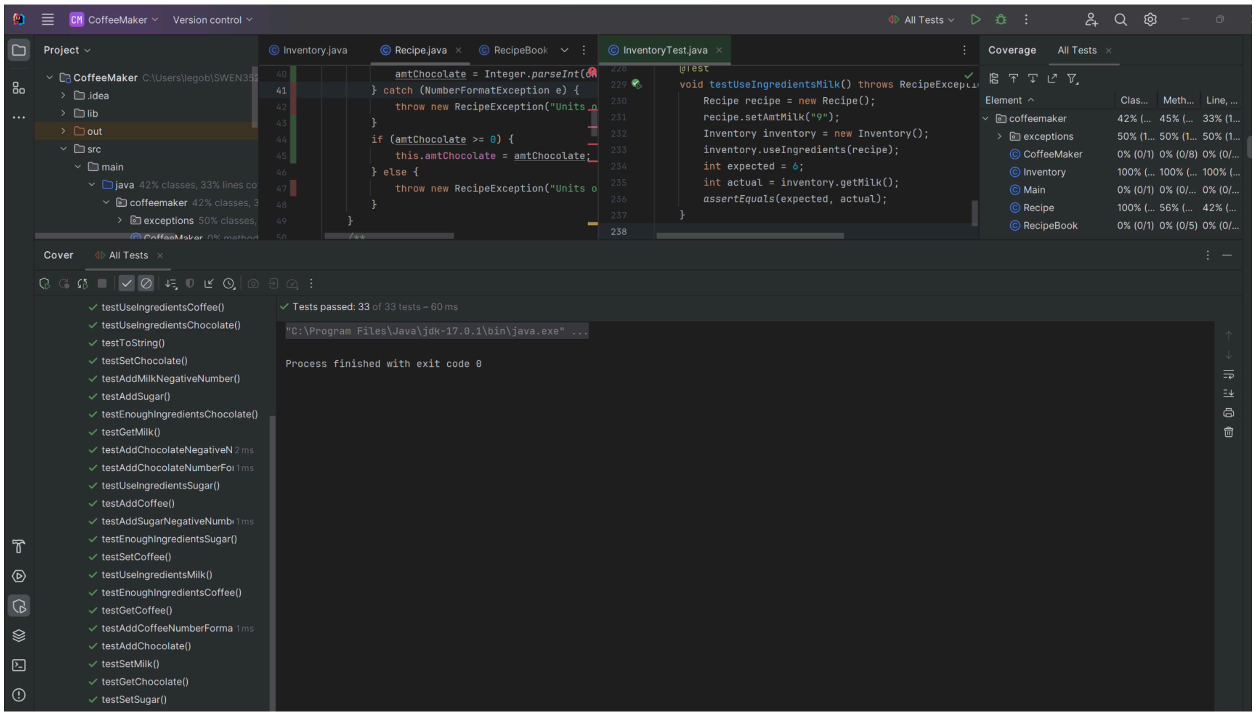
```java
    @Test
    void testUseIngredientsMilk() throws RecipeExceptio
        Recipe recipe = new Recipe();
        recipe.setAmtMilk("9");
        Inventory inventory = new Inventory();
        inventory.useIngredients(recipe);
        int expected = 6;
        int actual = inventory.getMilk();
        assertEquals(expected, actual);
    }
```

Element	Clas...	Meth...	Line, ...
coffeemaker	42% (...	45% (...	33% (1...
exceptions	50% (1...	50% (1...	50% (1...
CoffeeMaker	0% (0/1)	0% (0/8)	0% (0/...
Inventory	100% (...	100% (...	100% (...
Main	0% (0/1)	0% (0/...	0% (0/...
Recipe	100% (...	56% (...	42% (...
RecipeBook	0% (0/1)	0% (0/5)	0% (0/...

```
✓ <default package>                       60 ms
  ✓ InventoryTest                         58 ms
    ✓ testEnoughIngredientsMilkF          48 ms
    ✓ testAddCoffeeNegativeNuml           3 ms
    ✓ testAddSugarNumberFormatl           1 ms
    ✓ testEnoughIngredientsMilkTrue()
    ✓ testAddMilkNumberFormatE>           1 ms
    ✓ testGetSugar()
    ✓ testAddMilk()
    ✓ testUseIngredientsCoffee()
    ✓ testUseIngredientsChocolate()
    ✓ testToString()
    ✓ testSetChocolate()
    ✓ testAddMilkNegativeNumber()
    ✓ testAddSugar()
    ✓ testEnoughIngredientsChocolate()
    ✓ testGetMilk()
    ✓ testAddChocolateNegativeN           2 ms
    ✓ testAddChocolateNumberFoı           1 ms
    ✓ testUseIngredientsSugar()
    ✓ testAddCoffee()
    ✓ testAddSugarNegativeNumb            1 ms
    ✓ testEnoughIngredientsSugar()
    ✓ testSetCoffee()
    ✓ testUseIngredientsMilk()
```

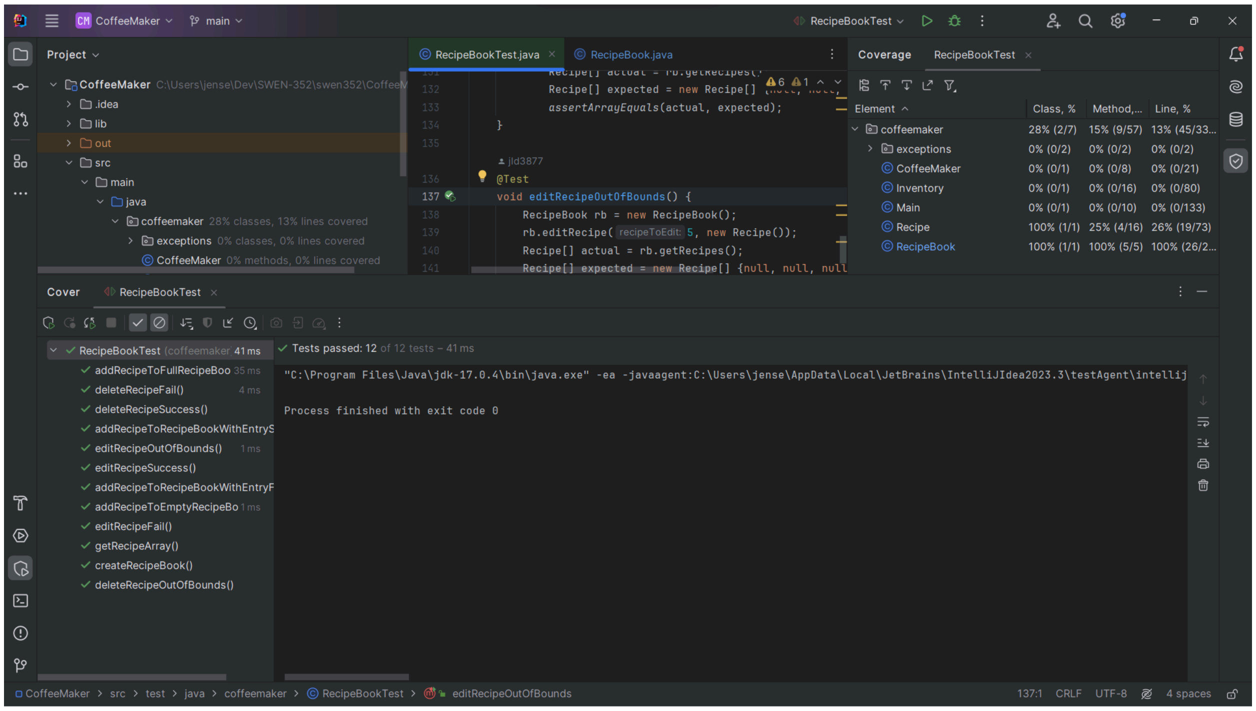
```
✓ Tests passed: 33 of 33 tests – 60 ms

"C:\Program Files\Java\jdk-17.0.1\bin\java.exe" ...

Process finished with exit code 0
```

CoffeeMaker > src > test > java > coffeemaker > InventoryTest   238:1 CRLF UTF-8 4 spaces



```
    ✓ testUseIngredientsCoffee()
    ✓ testUseIngredientsChocolate()
    ✓ testToString()
    ✓ testSetChocolate()
    ✓ testAddMilkNegativeNumber()
    ✓ testAddSugar()
    ✓ testEnoughIngredientsChocolate()
    ✓ testGetMilk()
    ✓ testAddChocolateNegativeN           2 ms
    ✓ testAddChocolateNumberFoı           1 ms
    ✓ testUseIngredientsSugar()
    ✓ testAddCoffee()
    ✓ testAddSugarNegativeNumb            1 ms
    ✓ testEnoughIngredientsSugar()
    ✓ testSetCoffee()
    ✓ testUseIngredientsMilk()
    ✓ testEnoughIngredientsCoffee()
    ✓ testGetCoffee()
    ✓ testAddCoffeeNumberForma            1 ms
    ✓ testAddChocolate()
    ✓ testSetMilk()
    ✓ testGetChocolate()
    ✓ testSetSugar()
```

```
✓ Tests passed: 33 of 33 tests – 60 ms

"C:\Program Files\Java\jdk-17.0.1\bin\java.exe" ...

Process finished with exit code 0
```



CoffeeMaker C:\Users\jense\Dev\SWEN-352\swen352\CoffeeM
- .idea
- lib
- out
- src
  - main
    - java
      - coffeemaker 28% classes, 13% lines covered
        - exceptions 0% classes, 0% lines covered
        - CoffeeMaker 0% methods, 0% lines covered

```java
        Recipe[] expected = new Recipe[]
        assertArrayEquals(actual, expected);
    }

    jld3877
    @Test
    void editRecipeOutOfBounds() {
        RecipeBook rb = new RecipeBook();
        rb.editRecipe( recipeToEdit: 5, new Recipe());
        Recipe[] actual = rb.getRecipes();
        Recipe[] expected = new Recipe[] {null, null, null
```

Element	Class, %	Method,...	Line, %
coffeemaker	28% (2/7)	15% (9/57)	13% (45/33...
exceptions	0% (0/2)	0% (0/2)	0% (0/2)
CoffeeMaker	0% (0/1)	0% (0/8)	0% (0/21)
Inventory	0% (0/1)	0% (0/16)	0% (0/80)
Main	0% (0/1)	0% (0/10)	0% (0/133)
Recipe	100% (1/1)	25% (4/16)	26% (19/73)
RecipeBook	100% (1/1)	100% (5/5)	100% (26/2...

- ✓ RecipeBookTest (coffeemaker 41 ms
  - ✓ addRecipeToFullRecipeBoo 35 ms
  - ✓ deleteRecipeFail() 4 ms
  - ✓ deleteRecipeSuccess()
  - ✓ addRecipeToRecipeBookWithEntryS
  - ✓ editRecipeOutOfBounds() 1 ms
  - ✓ editRecipeSuccess()
  - ✓ addRecipeToRecipeBookWithEntryF
  - ✓ addRecipeToEmptyRecipeBo 1 ms
  - ✓ editRecipeFail()
  - ✓ getRecipeArray()
  - ✓ createRecipeBook()
  - ✓ deleteRecipeOutOfBounds()

✓ Tests passed: 12 of 12 tests – 41 ms

```
"C:\Program Files\Java\jdk-17.0.4\bin\java.exe" -ea -javaagent:C:\Users\jense\AppData\Local\JetBrains\IntelliJIdea2023.3\testAgent\intellij

Process finished with exit code 0
```

CoffeeMaker > src > test > java > coffeemaker > RecipeBookTest > editRecipeOutOfBounds   137:1 CRLF UTF-8 4 spaces