Brady Maes
CS449

Sprint 0 Documentation

1. Key Decisions of the SOS Project

Object-oriented programming language	Java
GUI library (strongly encouraged)	JavaFX (Subject to change if I decide it does not meet my needs)
IDE (Integrated Development Environment)	IntelliJ IDEA Community Edition
xUnit framework (e.g., JUnit for Java)	JUnit version 5 (I also might use Mockito for testing small game functionalities). In the case of JUnit 5 with IntelliJ Community Edition, I will be using Maven.
Programming style guide (must read it carefully)	Google Java Style Guide (I am a big fan of Camel and Pascal casing as well as limiting characters on each line). ● Also worth noting that IntelliJ, like many IDEs, has the option to be configured to a specific style guide. This can be done for my project with the Google Java Format Plugin.
Project hosting site	GitHub (github.com)
Other decisions if applicable	I make some other decisions found in my **brainstorming** section below.

Brainstorming:

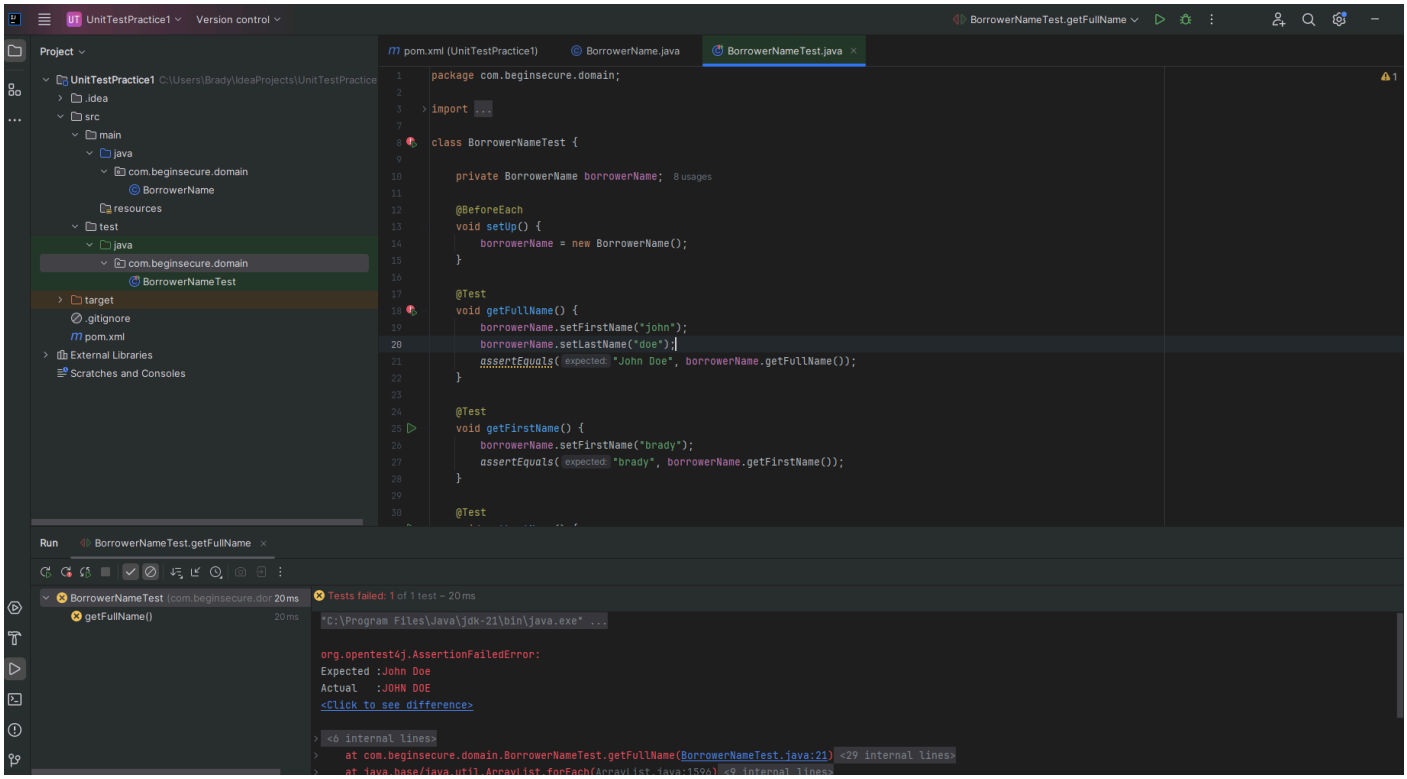
- Additional login feature that allows users to create an account. If this account is an admin account, debug features will be present. If the user has an account, their records will be stored (Google Drive or something for simplicity). If the user selects the guest account option, then nothing is stored long-term. Maybe have leaderboards?
- Use Java if can figure it out
- Try to use an OOP approach that reuses functionality and utilizes classes
- Bot to play against?
- Custom board size?- Might conflict with Bot's search algorithm (complexity skyrockets with board size)

2. Unit testing (4 points

The tutorial that I followed can be found here:
https://youtu.be/jBeu8BDvM48?si=l1dqLOHjNldS1qBC

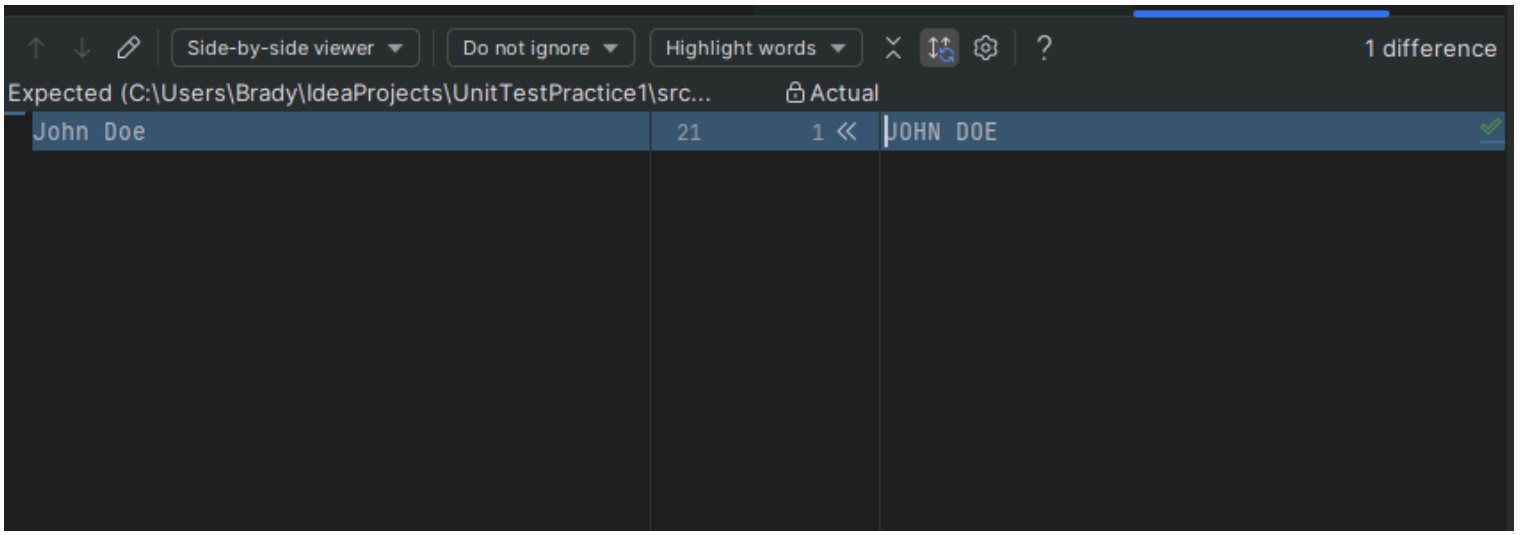
This example highlighs several methods (getters and setters) being tested in an example class.

We manaully run a test on our getFullName() method, but we intentionally make a mistake to show the unit testing:



Test One Fail

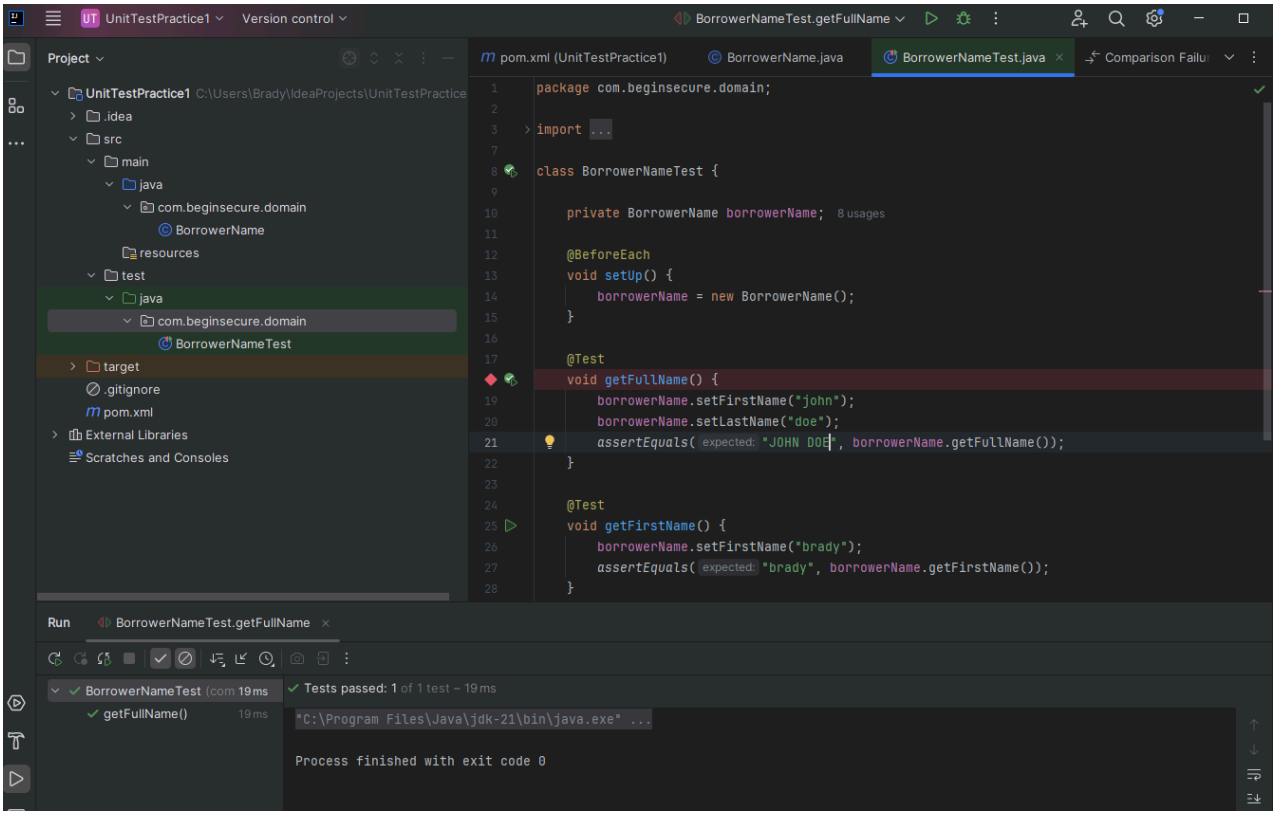
As you can see, our test failed. We can see that it expected "John Doe" but received "JOHN DOE." We can also view this in our side by side viewer window, which can be useful in more complex problems:



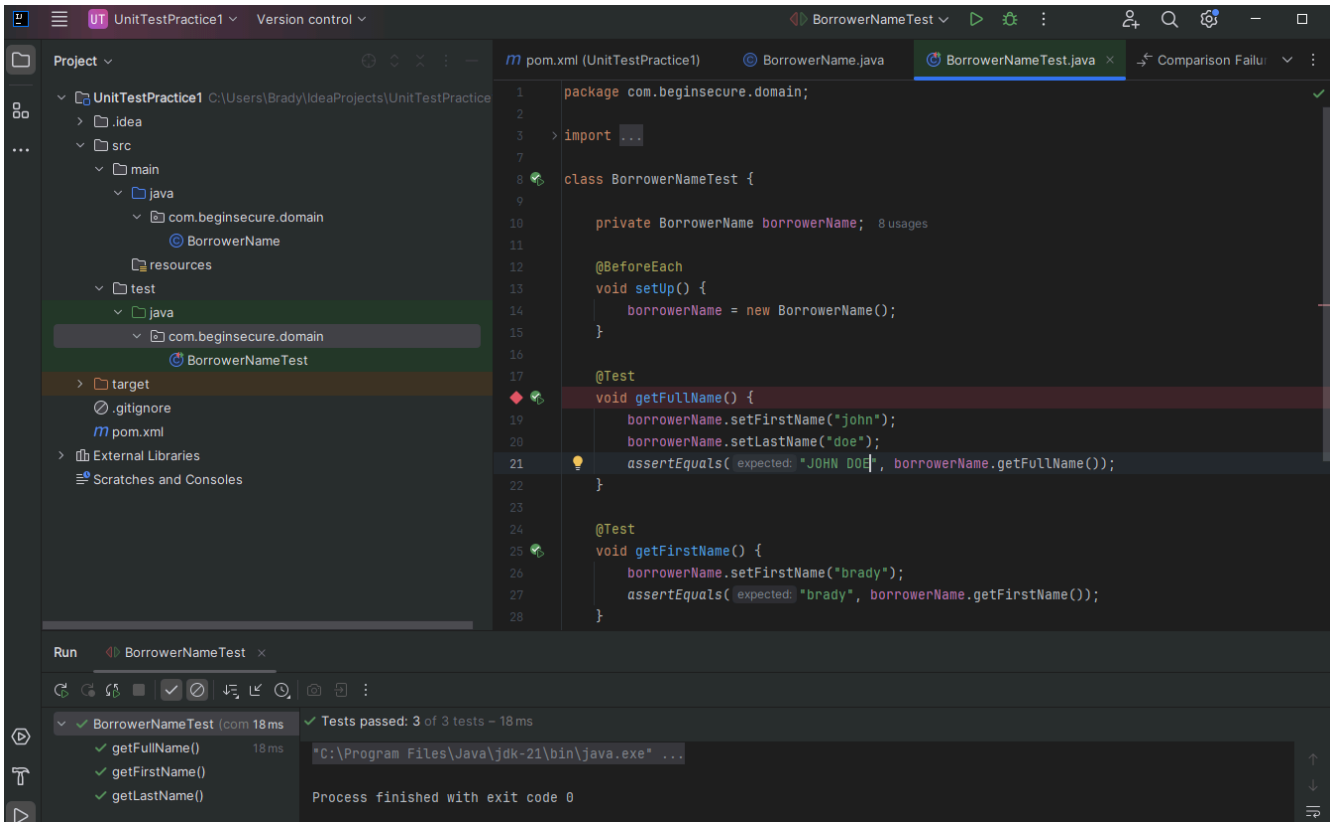
Side By Side View (Failure)

After looking at this failure, it is clear that our line
:return firstName.toUpperCase() + " " + lastName.toUpperCase();

is causing the error because it is setting the entirity to both strings to uppercase. We can either break these into substrings to fix their capitilization or use some functionality from StringUtils. For simplicity mistake, I just change the expected output to expect fully capitilized words, and the test passes:



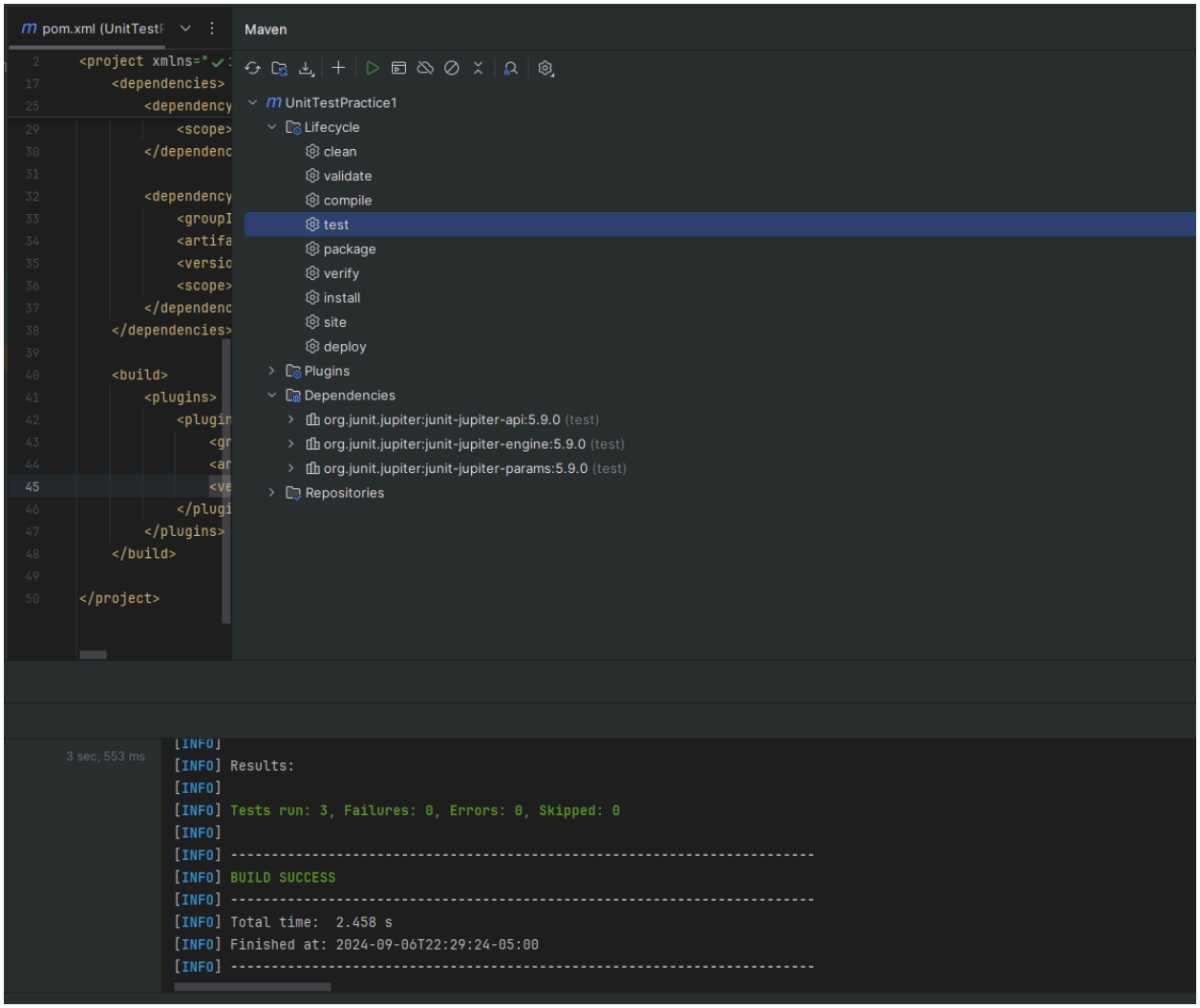
For my second unit test, I test all of my classes methods at the same time (and they pass):



Second Test, All Pass

Note on the bottom left side that the checkmarks next to the methods indicate which ones were tested and their status.

Finally, I also use Maven itself to run a test of my code:



Maven Test Pass

I tried to have my code comply with Google's Java Style Guide. However, my code lacks documentation as their is essentially no complexity to it. As I write more complex and personalized code for my project, I will make sure to include Javadoc comments.

Unit Testing Source Code:

pom.xml (UnitTestPractice1) | BorrowerName.java | BorrowerNameTest.java

```java
package com.beginsecure.domain;

public class BorrowerName {  2 usages

    private String firstName;  3 usages

    private String lastName;  3 usages

    public BorrowerName() {  1 usage
    }

    public String getFullName() {  1 usage
        return firstName.toUpperCase() + " " + lastName.toUpperCase();
    }

    public String getFirstName() {  1 usage
        return firstName;
    }

    public void setFirstName(String firstName) {  2 usages
        this.firstName = firstName;
    }

    public String getLastName() {  1 usage
        return lastName;
    }

    public void setLastName(String lastName) {  2 usages
        this.lastName = lastName;
    }
}
```

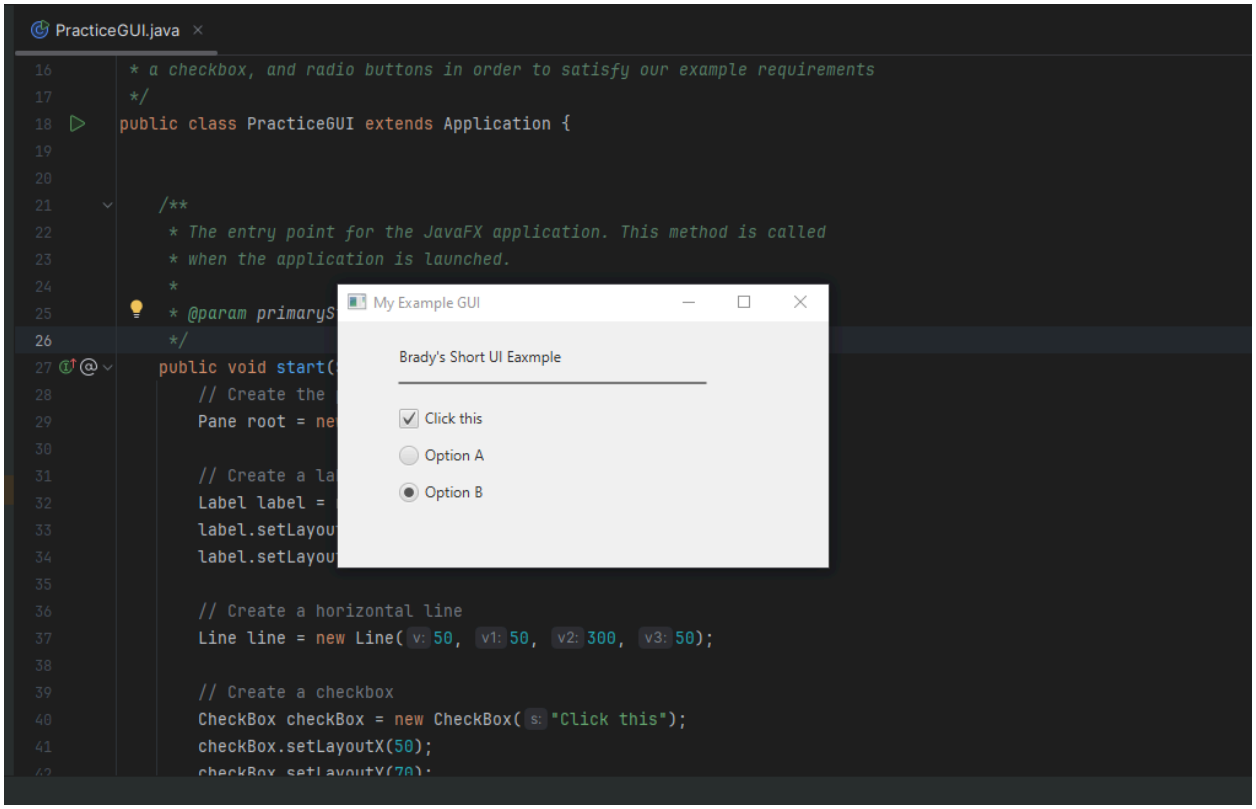
```java
package com.beginsecure.domain;

import ...

class BorrowerNameTest {

    private BorrowerName borrowerName;  8 usages

    @BeforeEach
    void setUp() {
        borrowerName = new BorrowerName();
    }

    @Test
    void getFullName() {
        borrowerName.setFirstName("john");
        borrowerName.setLastName("doe");
        assertEquals( expected: "JOHN DOE", borrowerName.getFullName());
    }

    @Test
    void getFirstName() {
        borrowerName.setFirstName("brady");
        assertEquals( expected: "brady", borrowerName.getFirstName());
    }

    @Test
    void getLastName() {
        borrowerName.setLastName("maes");
        assertEquals( expected: "maes", borrowerName.getLastName());
    }
}
```

POM xml for Unit Testing:

```xml
    <dependencies>
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-api</artifactId>
            <version>5.9.0</version>
            <scope>test</scope>
        </dependency>

        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-engine</artifactId>
            <version>5.9.0</version>
            <scope>test</scope>
        </dependency>

        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-params</artifactId>
            <version>5.9.0</version>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-surefire-plugin</artifactId>
                <version>3.0.0-M7</version>
            </plugin>
        </plugins>
    </build>

</project>
```

3. GUI programming (4 points)

I made a small JavaFX program that includes text, lines, a check box, and radio buttons. This is what it looks like when running:



GUI Running With Buttons Pressed

GUI Source Code:

PracticeGUI.java

```java
package com.beginsecure.maventest.guiexample1;

import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.CheckBox;
import javafx.scene.control.Label;
import javafx.scene.control.RadioButton;
import javafx.scene.control.ToggleGroup;
import javafx.scene.layout.Pane;
import javafx.scene.shape.Line;
import javafx.stage.Stage;


/**
 * JavaFX application that displays a simple GUI with text, a line,
 * a checkbox, and radio buttons in order to satisfy our example requirements
 */
public class PracticeGUI extends Application {


    /**
     * The entry point for the JavaFX application. This method is called
     * when the application is launched.
     *
     * @param primaryStage the primary stage for this application
     */
    public void start(Stage primaryStage) {
        // Create the pane (root layout)
        Pane root = new Pane();

        // Create a label (text)
        Label label = new Label( s: "Brady's Short UI Eaxmple");
        label.setLayoutX(50);
        label.setLayoutY(20);

        // Create a horizontal line
        Line line = new Line( v: 50, v1: 50, v2: 300, v3: 50);

        // Create a checkbox
        CheckBox checkBox = new CheckBox( s: "Click this");
        checkBox.setLayoutX(50);
        checkBox.setLayoutY(70);

        // Create radio buttons with a ToggleGroup
        ToggleGroup group = new ToggleGroup();

        RadioButton radioButton1 = new RadioButton( s: "Option A");
        radioButton1.setToggleGroup(group);
        radioButton1.setLayoutX(50);
        radioButton1.setLayoutY(100);

        RadioButton radioButton2 = new RadioButton( s: "Option B");
        radioButton2.setToggleGroup(group);
        radioButton2.setLayoutX(50);
        radioButton2.setLayoutY(130);

        // Add components to the root layout
        root.getChildren().addAll(label, line, checkBox, radioButton1, radioButton2);

        // Create the scene with the root layout and set dimensions
        Scene scene = new Scene(root, v: 400, v1: 200);

        // Set the stage title and scene, then show it
        primaryStage.setTitle("My Example GUI");
        primaryStage.setScene(scene);
        primaryStage.show();
    }


    /**
     * The main method, which launches the JavaFX App
     *
     * @param args the command-line arguments
     */
    public static void main(String[] args) {
        launch(args);
    }
}
```