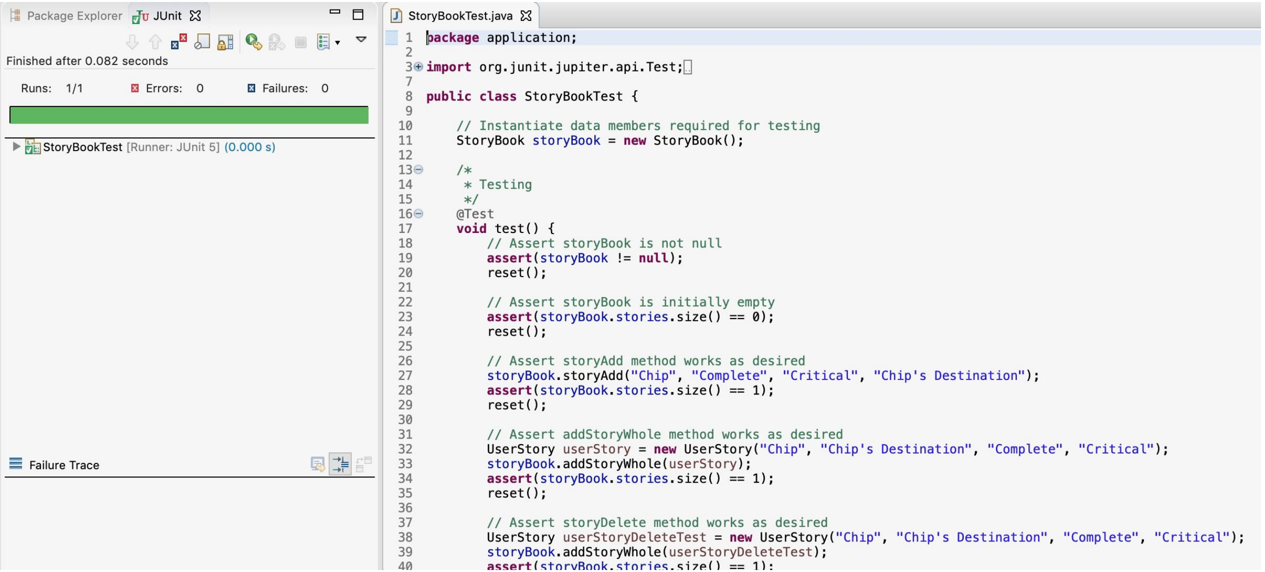
Acceptance Tests

Preconditions	Steps	Postconditions	Results
- 3 stories are created	- Stories are initialized with random inputs (header, content, priority, client) - Each story holds unique ID - Client(s) update	- All stories are present on Server/other clients	Works as expected
- Stories Modified/Deleted	- Stories modified by different client then who created story - Each story stays robust, only changes when one client updates at a time - Deleted stories are updated across Clients/Server	- All stories appropriately reflected on Server/other clients	Works as expected
- Burndown Chart initialized	- Stories are initialized randomly - Server updates and holds all stories	- All stories are appropriately reflected on the BurnDown chart	Works as expected

Integration Tests (Unit Testing)

Screenshot of Results/Code



```
Finished after 0.082 seconds
Runs: 1/1    Errors: 0    Failures: 0
StoryBookTest [Runner: JUnit 5] (0.000 s)
Failure Trace
```

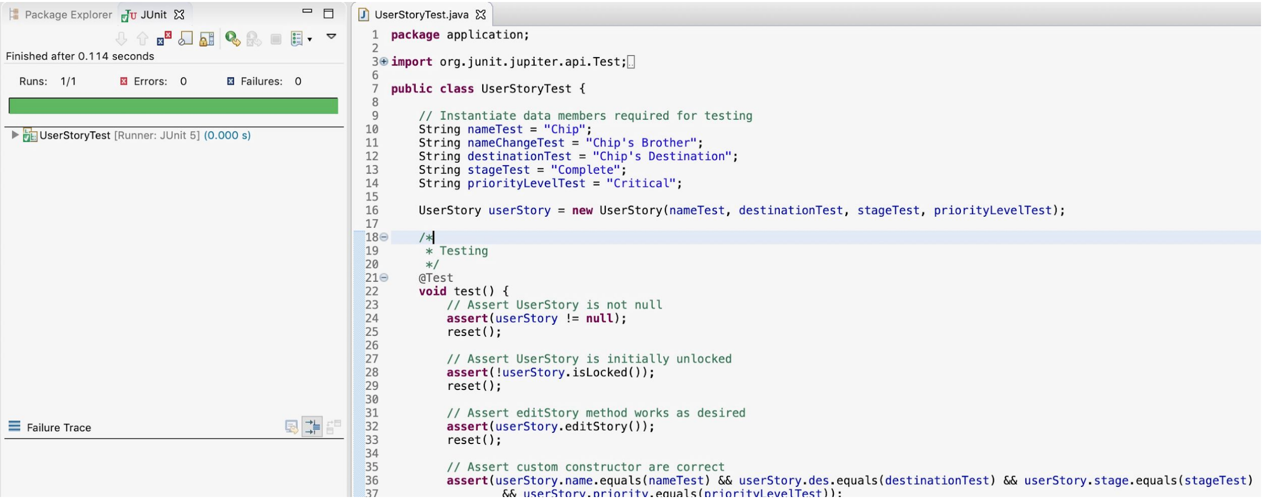
```java
package application;

import org.junit.jupiter.api.Test;

public class StoryBookTest {

    // Instantiate data members required for testing
    StoryBook storyBook = new StoryBook();

    /*
     * Testing
     */
    @Test
    void test() {
        // Assert storyBook is not null
        assert(storyBook != null);
        reset();

        // Assert storyBook is initially empty
        assert(storyBook.stories.size() == 0);
        reset();

        // Assert storyAdd method works as desired
        storyBook.storyAdd("Chip", "Complete", "Critical", "Chip's Destination");
        assert(storyBook.stories.size() == 1);
        reset();

        // Assert addStoryWhole method works as desired
        UserStory userStory = new UserStory("Chip", "Chip's Destination", "Complete", "Critical");
        storyBook.addStoryWhole(userStory);
        assert(storyBook.stories.size() == 1);
        reset();

        // Assert storyDelete method works as desired
        UserStory userStoryDeleteTest = new UserStory("Chip", "Chip's Destination", "Complete", "Critical");
        storyBook.addStoryWhole(userStoryDeleteTest);
        assert(storyBook.stories.size() == 1);
```

Test # - Justification

1 - Test StoryBook and functionality

Integration Tests (Unit Testing)

Screenshot of Results/Code



```
Finished after 0.114 seconds
Runs: 1/1    Errors: 0    Failures: 0
UserStoryTest [Runner: JUnit 5] (0.000 s)
Failure Trace
```

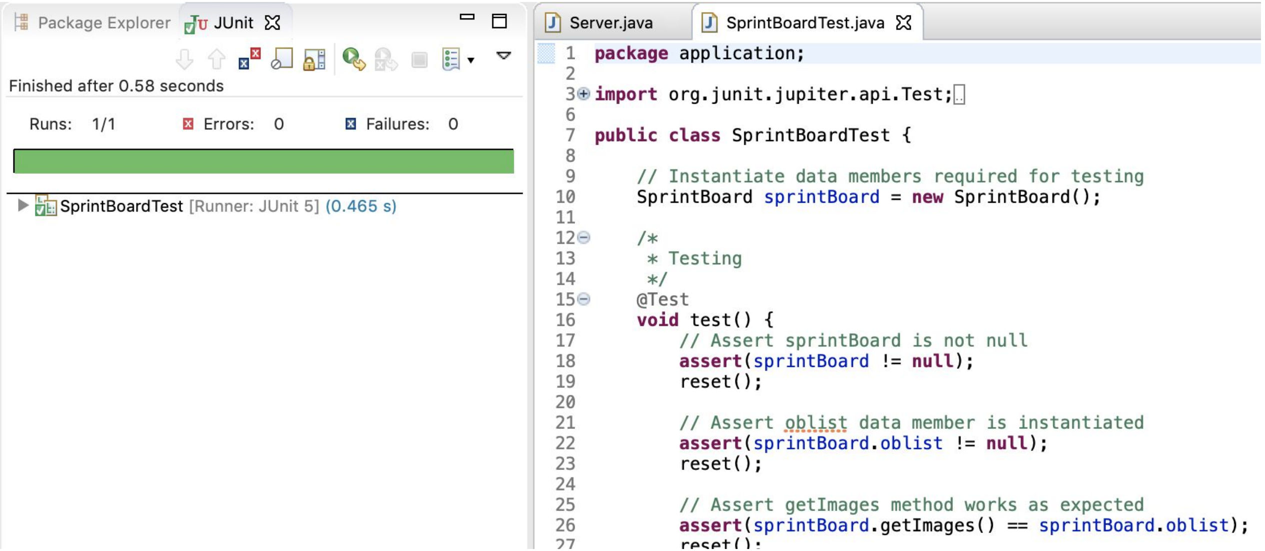
```java
package application;

import org.junit.jupiter.api.Test;

public class UserStoryTest {

    // Instantiate data members required for testing
    String nameTest = "Chip";
    String nameChangeTest = "Chip's Brother";
    String destinationTest = "Chip's Destination";
    String stageTest = "Complete";
    String priorityLevelTest = "Critical";

    UserStory userStory = new UserStory(nameTest, destinationTest, stageTest, priorityLevelTest);

    /*
     * Testing
     */
    @Test
    void test() {
        // Assert UserStory is not null
        assert(userStory != null);
        reset();

        // Assert UserStory is initially unlocked
        assert(!userStory.isLocked());
        reset();

        // Assert editStory method works as desired
        assert(userStory.editStory());
        reset();

        // Assert custom constructor are correct
        assert(userStory.name.equals(nameTest) && userStory.des.equals(destinationTest) && userStory.stage.equals(stageTest)
                && userStory.priority.equals(priorityLevelTest));
```

Test # - Justification

2 - Test the functionality of the lock and function of UserStory

Integration Tests (Unit Testing)

Screenshot of Results/Code



```
Finished after 0.58 seconds

Runs:   1/1        Errors:   0        Failures:   0

SprintBoardTest [Runner: JUnit 5] (0.465 s)
```

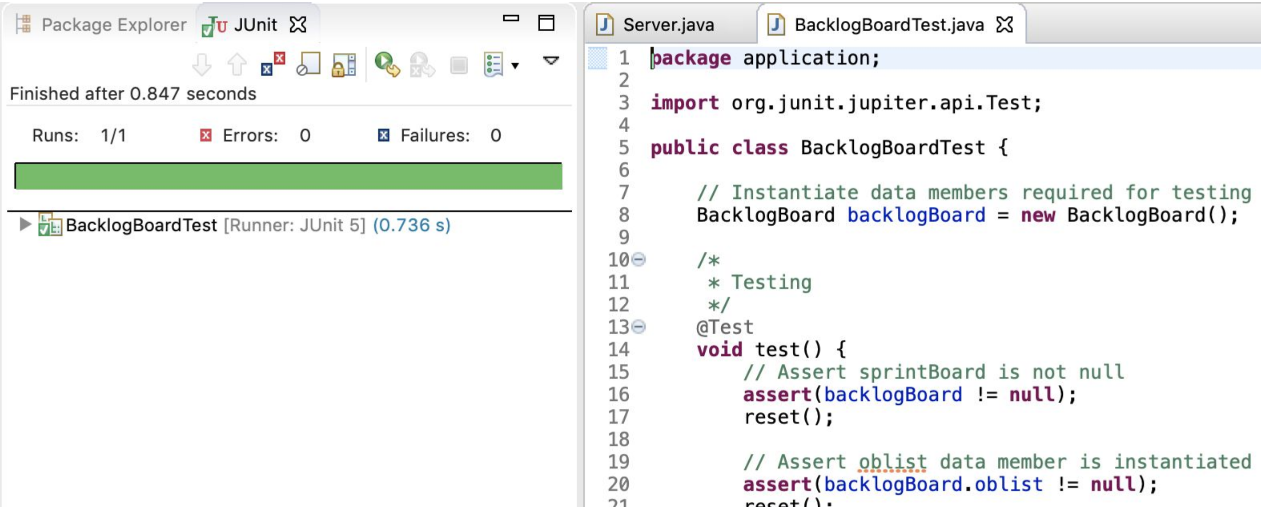
```java
package application;

import org.junit.jupiter.api.Test;

public class SprintBoardTest {

    // Instantiate data members required for testing
    SprintBoard sprintBoard = new SprintBoard();

    /*
     * Testing
     */
    @Test
    void test() {
        // Assert sprintBoard is not null
        assert(sprintBoard != null);
        reset();

        // Assert oblist data member is instantiated
        assert(sprintBoard.oblist != null);
        reset();

        // Assert getImages method works as expected
        assert(sprintBoard.getImages() == sprintBoard.oblist);
        reset();
```

Test # - Justification

3 - Test SprintBoard and functionality

Integration Tests (Unit Testing)

Screenshot of Results/Code



```
Finished after 0.847 seconds

Runs:   1/1        Errors:   0        Failures:   0

▶ BacklogBoardTest [Runner: JUnit 5] (0.736 s)
```

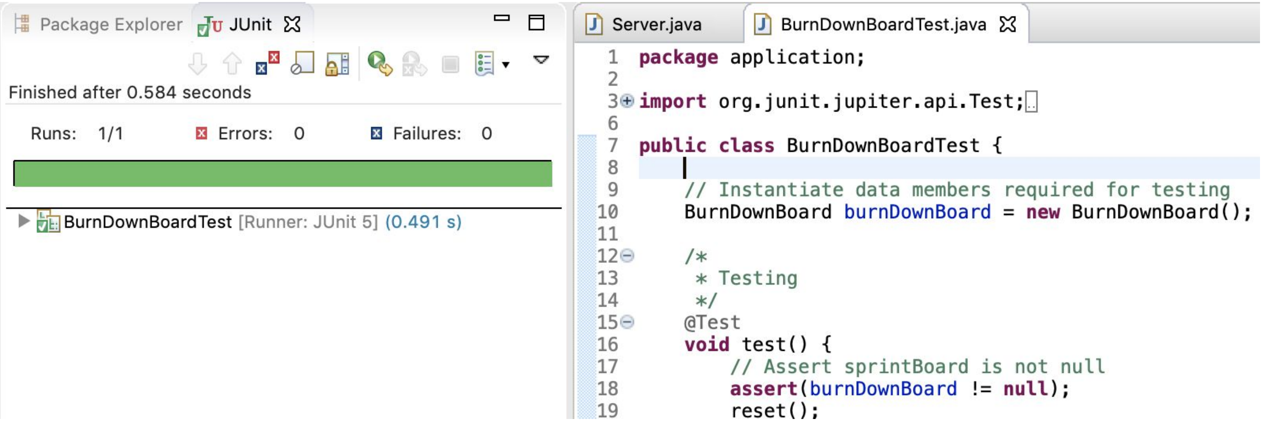
```java
package application;

import org.junit.jupiter.api.Test;

public class BacklogBoardTest {

    // Instantiate data members required for testing
    BacklogBoard backlogBoard = new BacklogBoard();

    /*
     * Testing
     */
    @Test
    void test() {
        // Assert sprintBoard is not null
        assert(backlogBoard != null);
        reset();

        // Assert oblist data member is instantiated
        assert(backlogBoard.oblist != null);
        reset();
```

Test # - Justification

4 - Test BacklogBoard and functionality

Integration Tests (Unit Testing)

Screenshot of Results/Code



```
Finished after 0.584 seconds

Runs:   1/1        Errors:   0        Failures:   0

BurnDownBoardTest [Runner: JUnit 5] (0.491 s)
```

```java
package application;

import org.junit.jupiter.api.Test;

public class BurnDownBoardTest {
    
    // Instantiate data members required for testing
    BurnDownBoard burnDownBoard = new BurnDownBoard();

    /*
     * Testing
     */
    @Test
    void test() {
        // Assert sprintBoard is not null
        assert(burnDownBoard != null);
        reset();
```

Test # - Justification

5 - Test BurnDownBoard and functionality