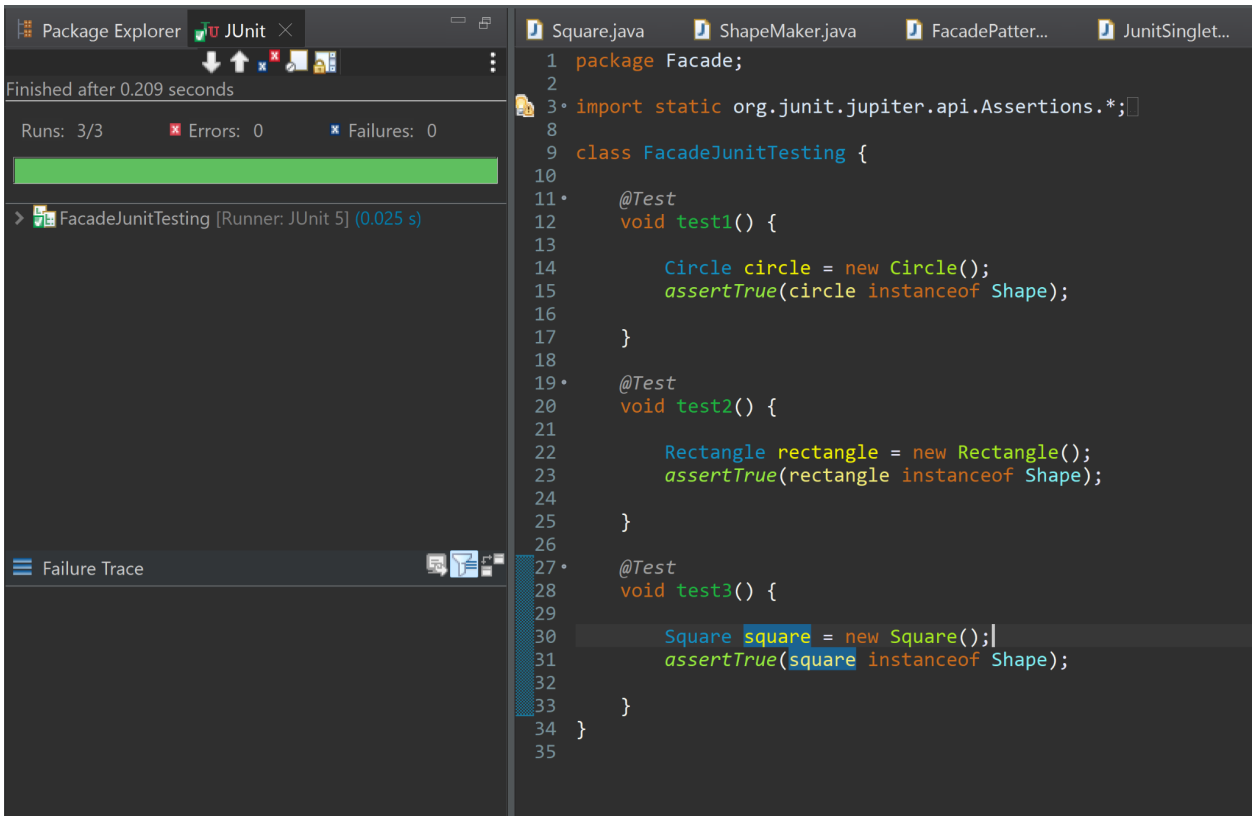
Name: Shannon Noah
Student Number: 300163898

Screen Capture of Junit Tests

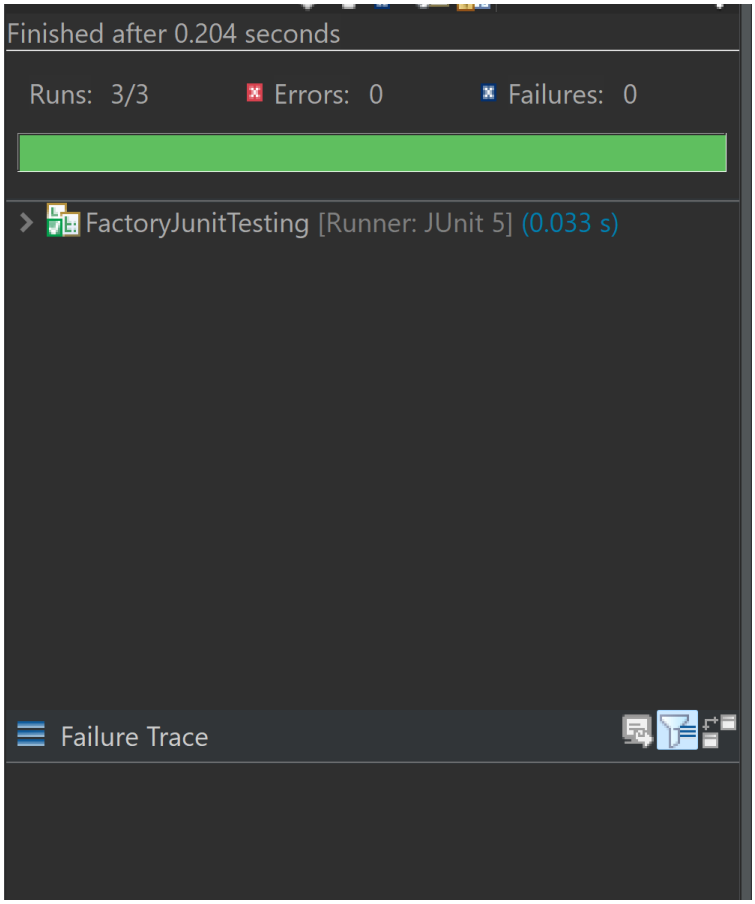
Part 1

Facade



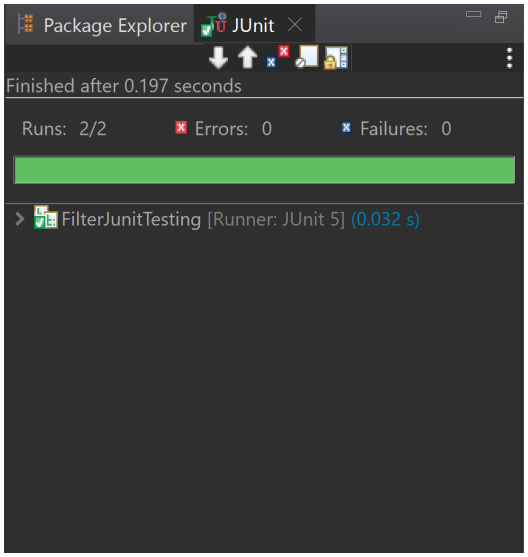
Factory

```
class FactoryJunitTesting {

    @Test
    void test1() {

        ShapeFactory shapeFactory = new ShapeFactory();

        Shape shape = shapeFactory.getShape("CIRCLE");

        assertTrue(shape instanceof Circle);
    }

    @Test
    void test2() {

        ShapeFactory shapeFactory = new ShapeFactory();

        Shape shape = shapeFactory.getShape("RECTANGLE");

        assertTrue(shape instanceof Rectangle);
    }


    @Test
    void test3() {

        ShapeFactory shapeFactory = new ShapeFactory();

        Shape shape = shapeFactory.getShape("SQUARE");

        assertTrue(shape instanceof Square);
    }
```

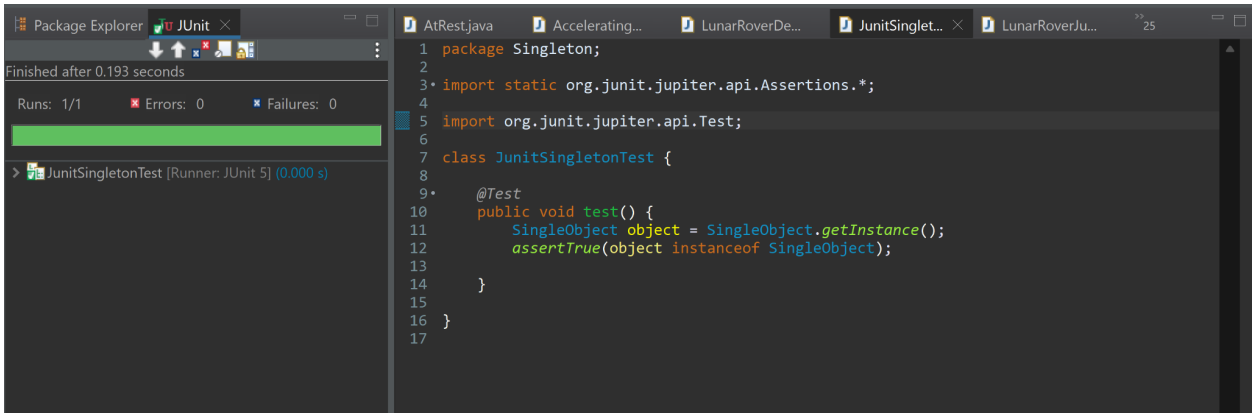

Finished after 0.204 seconds
Runs: 3/3
Errors: 0
Failures: 0
FactoryJunitTesting [Runner: JUnit 5] (0.033 s)
Failure Trace

Filter

```java
    @Test
    void test1() {


        List<Person> persons = new ArrayList<Person>();
        persons.add(new Person("Jane", "Female", "No"));
        persons.add(new Person("Bob", "Male", "No"));


        Criteria female = new CriteriaFemale();

        List<Person> actual = female.meetCriteria(persons);
        List<Person> expected = new ArrayList<Person>();

        expected.add(new Person("Jane", "Female", "No"));

        assertEquals(expected.get(0).getName(), actual.get(0).getName());

    }

    @Test
    void test2() {

        List<Person> persons = new ArrayList<Person>();
        persons.add(new Person("Jane", "Female", "No"));
        persons.add(new Person("Bob", "Male", "No"));

        Criteria male = new CriteriaMale();

        List<Person> actual = male.meetCriteria(persons);
        List<Person> expected = new ArrayList<Person>();

        expected.add(new Person("Bob", "Male", "No"));

        assertEquals(expected.get(0).getName(), actual.get(0).getName());
```



Singleton



Part 2

```java
package Part2;

import static org.junit.jupiter.api.Assertions.*;

class LunarRoverJunitTesting {


    // When the left pedal was pressed once it accelerates the buggy forward.
    @Test
    void test1() {
        LunarRoverContext lunarRover = new LunarRoverContext();
        lunarRover.l1();
        String output = lunarRover.getState().toString();
        assertEquals(output, "Accelerating Forwards State");
    }

    //If accelerating forward and you press right pedal twice it deaccelerates.
    @Test
    void test2() {
        LunarRoverContext lunarRover = new LunarRoverContext();
        lunarRover.l1();
        lunarRover.r2();
        String output = lunarRover.getState().toString();
        assertEquals(output, "Deaccelerating State");
    }
```

```
29
30      //To achieve constant forward speed press the right pedal for more than 5 seconds.
31      @Test
32      void test3() {
33          LunarRoverContext lunarRover = new LunarRoverContext();
34          lunarRover.l1();
35          lunarRover.rm5();
36          String output = lunarRover.getState().toString();
37          assertEquals(output, "Constant Forward Speed State");
38      }
39
40      //If the buggy is at rest and the left pedal is pressed for more than
41      //accelerate backwards.
42      @Test
43      void test4() {
44          LunarRoverContext lunarRover = new LunarRoverContext();
45          lunarRover.lm5();
46          String output = lunarRover.getState().toString();
47          assertEquals(output, "Accelerating Backwards State");
48      }
49
```