EBU5304 – Software Engineering

TDD

- Test Driven Development in Agile Process
- Using JUnit

Test Driven Development in Agile process

TDD cycle

- TDD: write tests prior to write the production code
- TDD is a simple, short-cycled mechanism
  - Write a specification, in code and in the form of a unit test. The test verifies a functional unit of your code.
  - Demonstrate test failure.
  - Write code to meet the specification.
  - Demonstrate test success.
  - Refactor the code, to ensure that the system still has an optimally clean code base.
- Run all tests against the entire system at all time.

JUnit

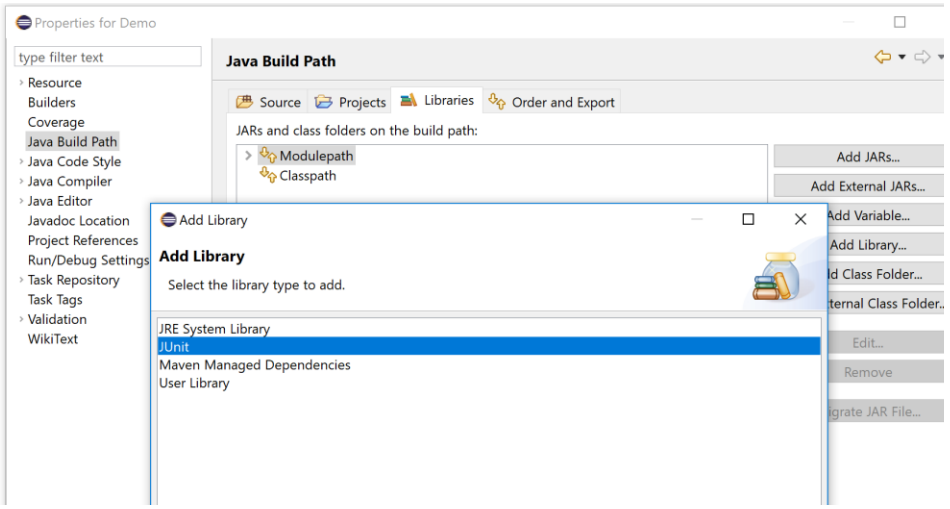
- Junit is a simple unit-testing framework for supporting TDD
- Download JUnit from https://junit.org/
- Current version: JUnit 5
- Comes with major IDE, e.g. Eclipse

Tutorial

- Write a StudentTest class: test code
- Write a Student class: product code
- Using Eclipse with JUnit 5

Eclipse set up

1. Create a new Java project
2. Right click the project then choose > Properties > Java Build Path > Libraries
3. Click the button "Add Library" then select "JUnit"



JARs and class folders on the build path:

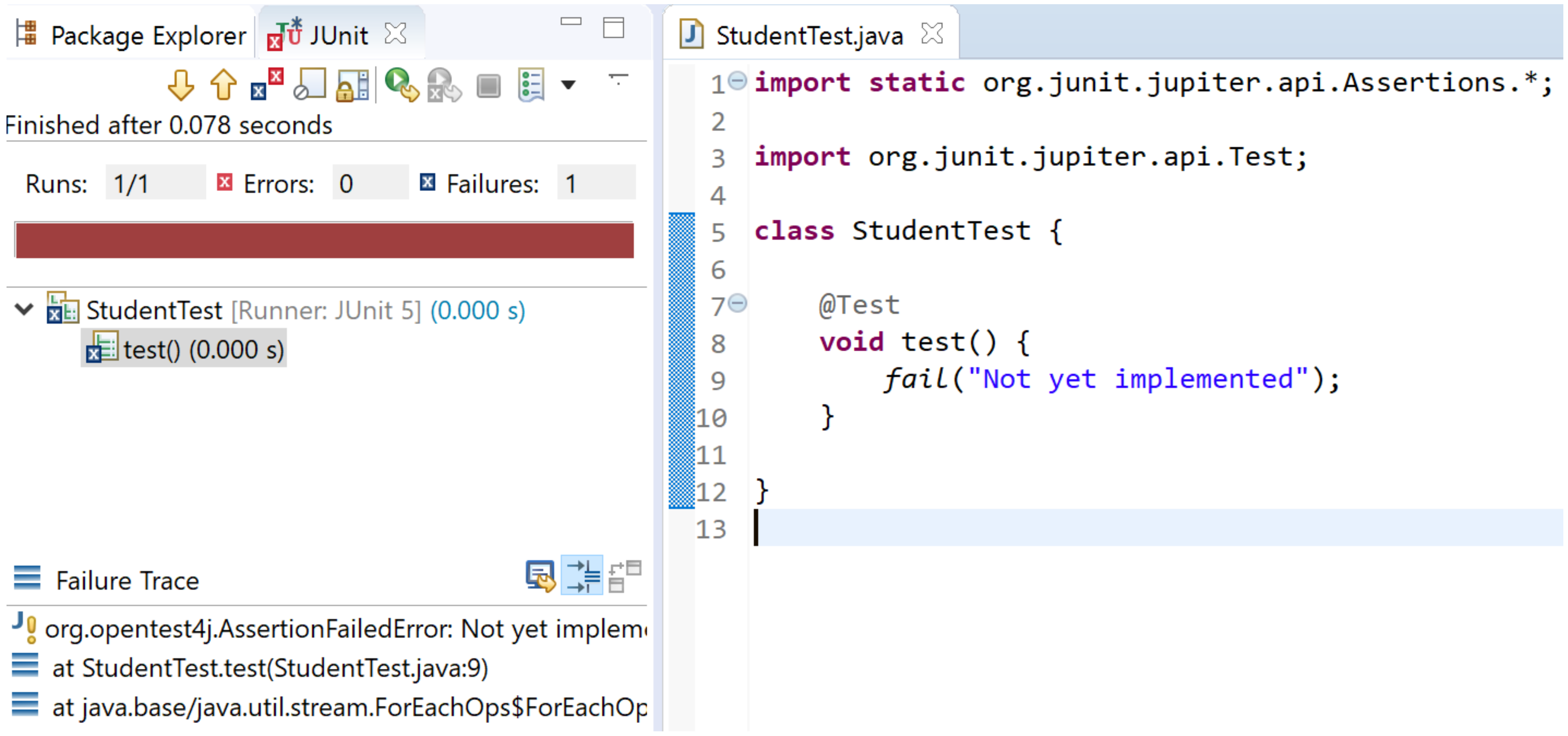
Add Library

Select the library type to add.

4. Choose the version JUnit 5

Create a test class

Create a new JUnit test class (right click the project->New->Junit Test Case)

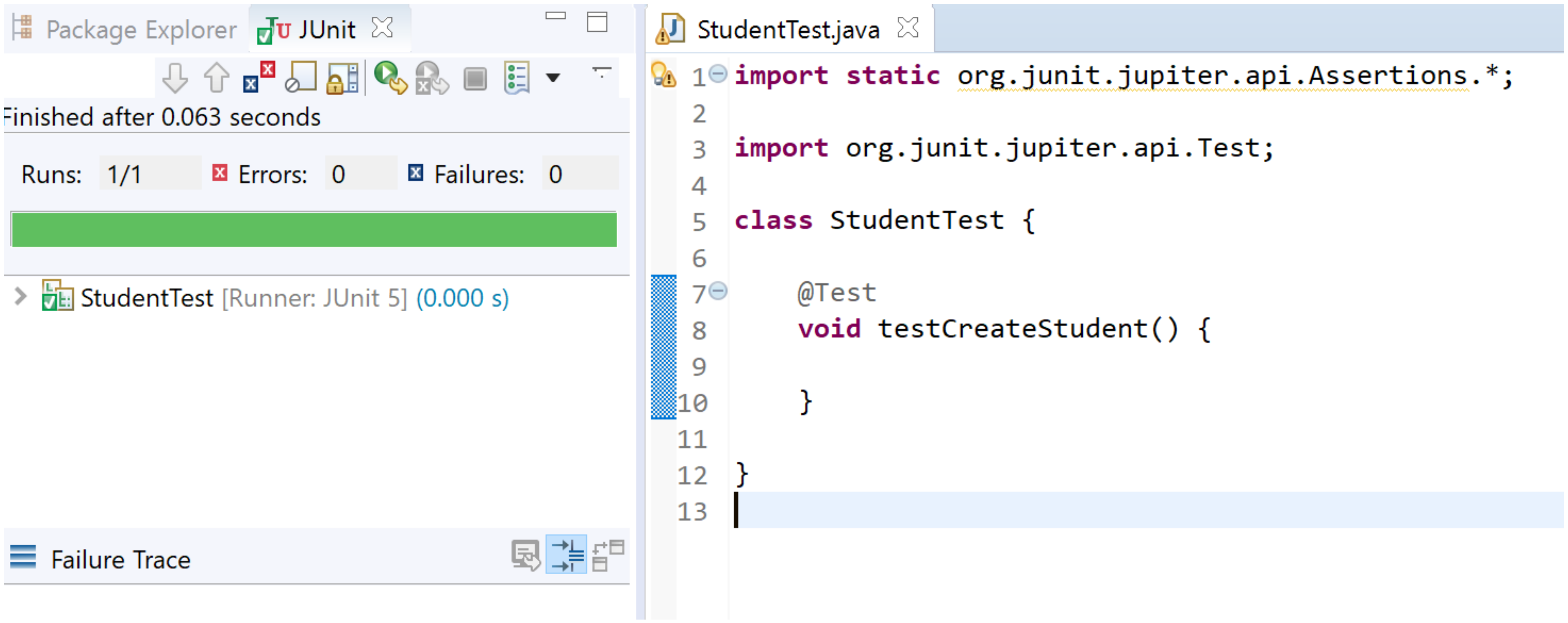


```
Finished after 0.078 seconds
Runs: 1/1   Errors: 0   Failures: 1

StudentTest [Runner: JUnit 5] (0.000 s)
    test() (0.000 s)

Failure Trace
org.opentest4j.AssertionFailedError: Not yet implem
at StudentTest.test(StudentTest.java:9)
at java.base/java.util.stream.ForEachOps$ForEachOp
```

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

import org.junit.jupiter.api.Test;

class StudentTest {

    @Test
    void test() {
        fail("Not yet implemented");
    }

}
```

show test failure - red bar

Empty test



Finished after 0.063 seconds

Runs: 1/1 Errors: 0 Failures: 0

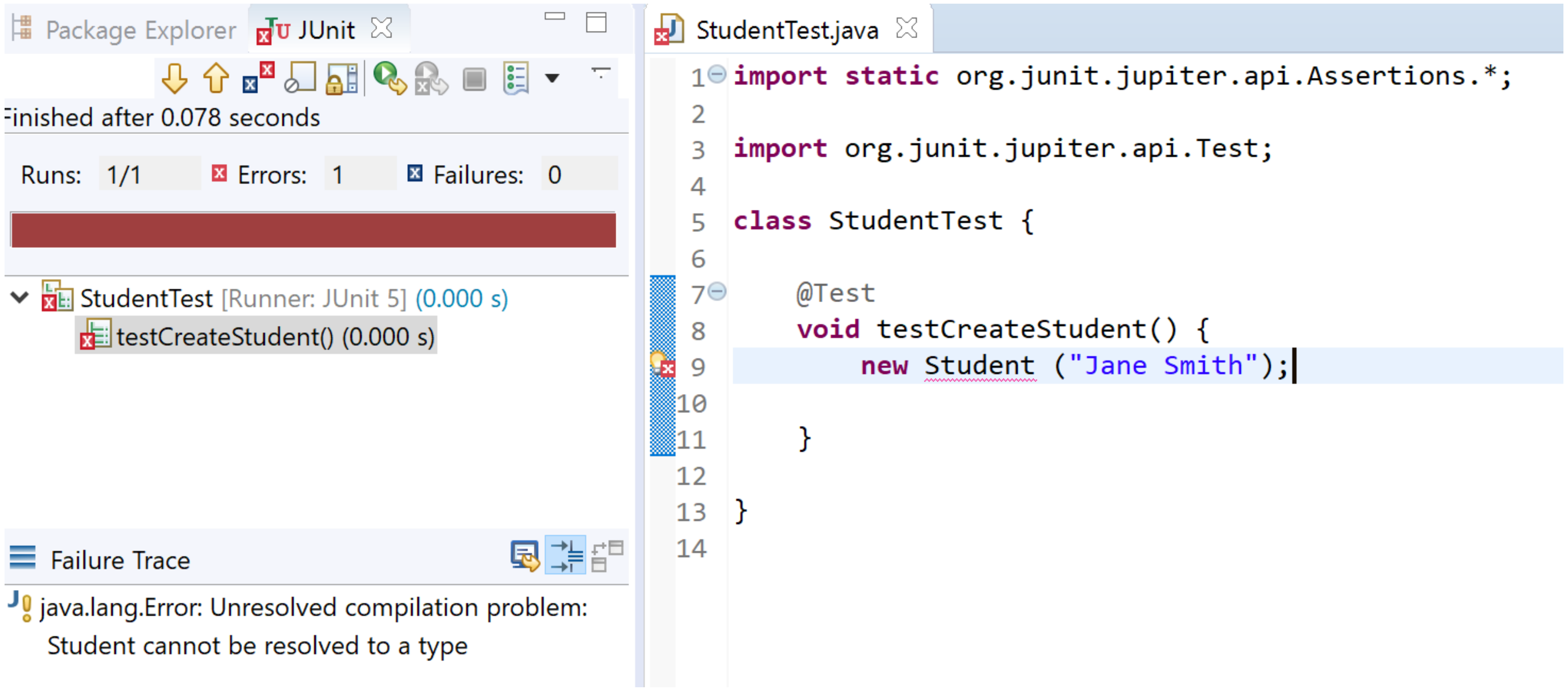
StudentTest [Runner: JUnit 5] (0.000 s)

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

import org.junit.jupiter.api.Test;

class StudentTest {

    @Test
    void testCreateStudent() {

    }

}
```

JUnit test success – green bar

An empty test method will always pass

no object - failure

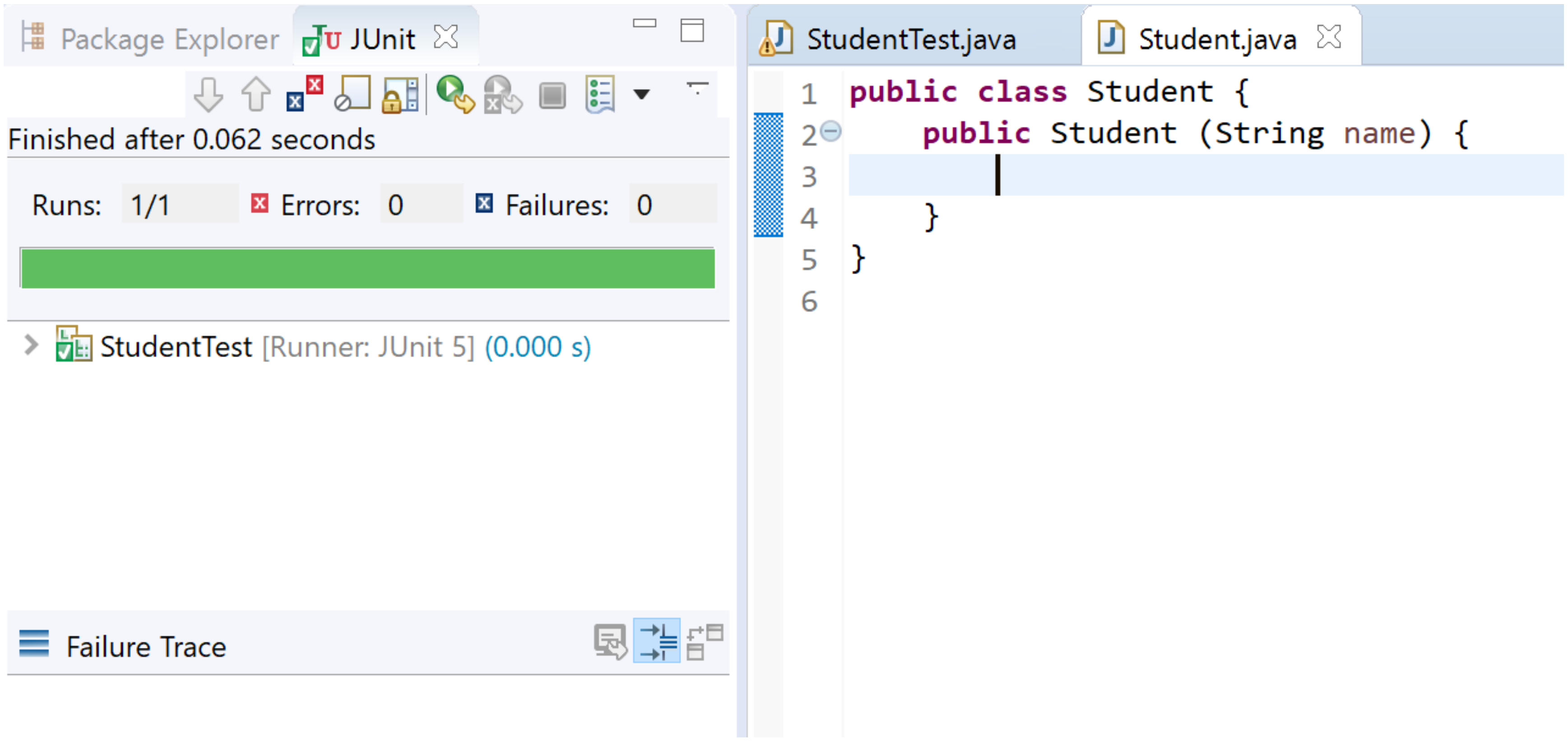


```
Finished after 0.078 seconds
Runs: 1/1   Errors: 1   Failures: 0

StudentTest [Runner: JUnit 5] (0.000 s)
    testCreateStudent() (0.000 s)

java.lang.Error: Unresolved compilation problem:
    Student cannot be resolved to a type
```

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

import org.junit.jupiter.api.Test;

class StudentTest {

    @Test
    void testCreateStudent() {
        new Student ("Jane Smith");

    }

}
```

Now it is time to create **`Student`** class

Adding student class - success



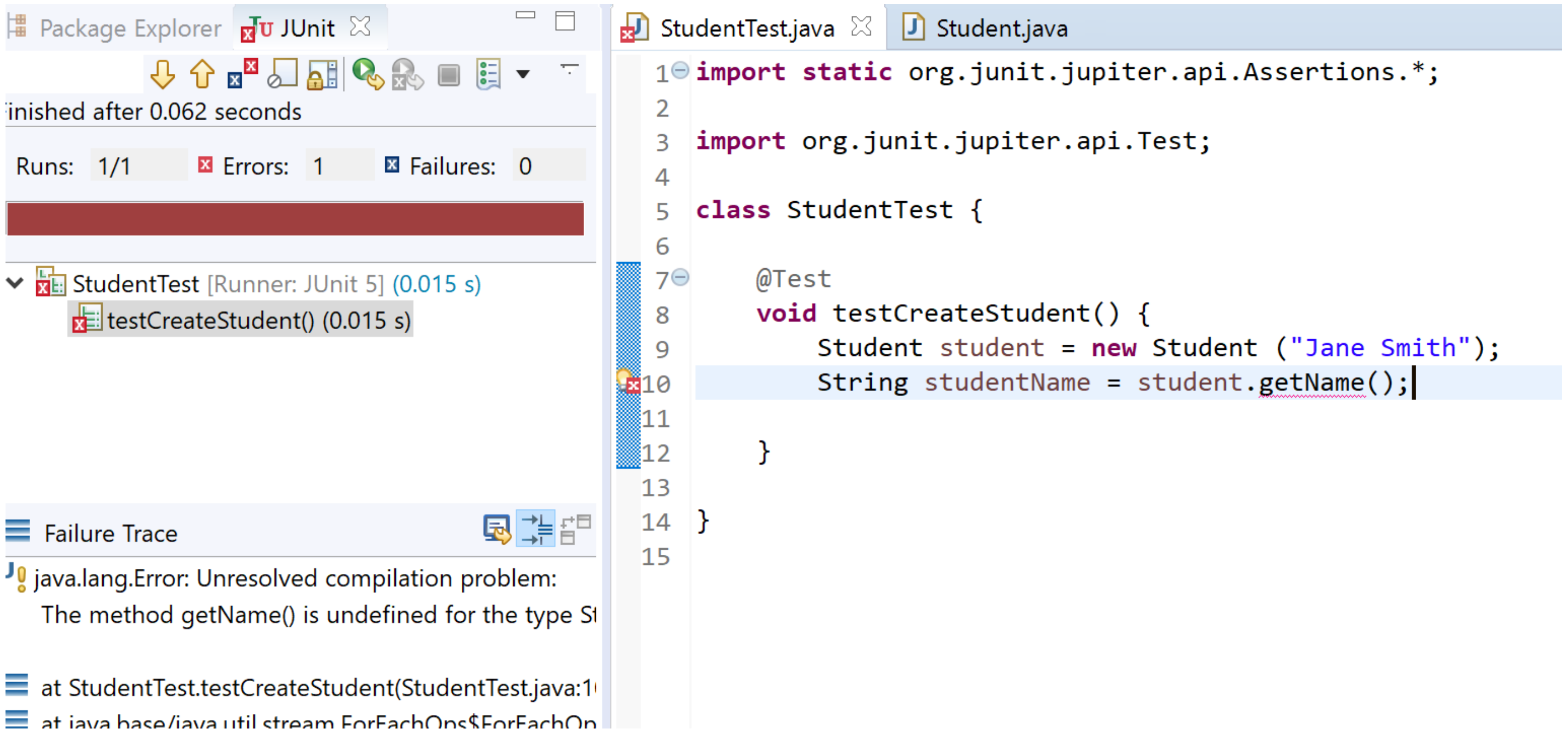
Finished after 0.062 seconds

Runs: 1/1 Errors: 0 Failures: 0

StudentTest [Runner: JUnit 5] (0.000 s)

```
public class Student {
    public Student (String name) {

    }
}

```

Undefined method - failure



Package Explorer JUnit

inished after 0.062 seconds

Runs: 1/1 Errors: 1 Failures: 0

StudentTest [Runner: JUnit 5] (0.015 s)
testCreateStudent() (0.015 s)

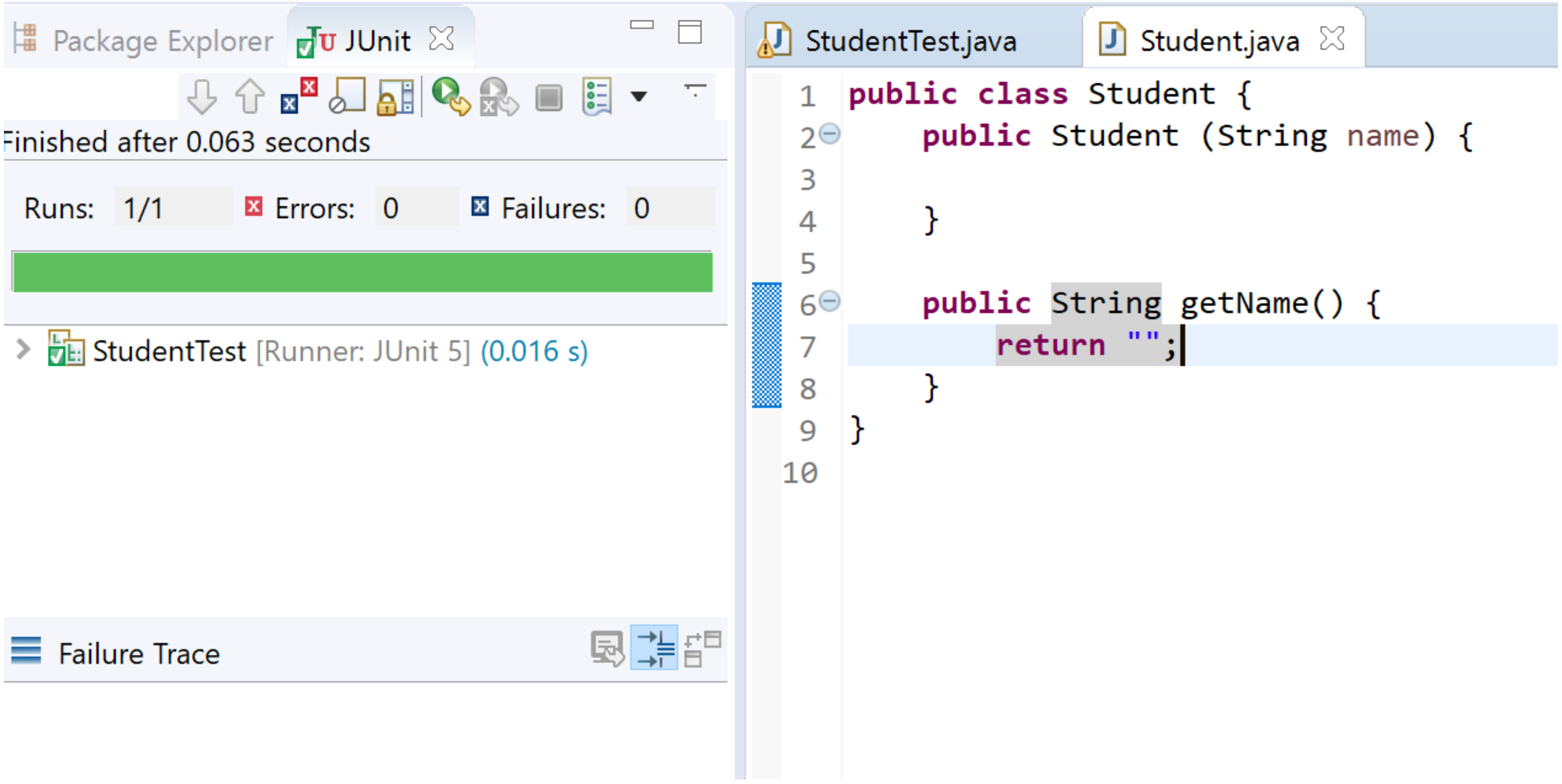
Failure Trace

```
java.lang.Error: Unresolved compilation problem:
  The method getName() is undefined for the type St

at StudentTest.testCreateStudent(StudentTest.java:1
at java.base/java.util.stream.ForEachOps$ForEachOp
```

StudentTest.java Student.java

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

import org.junit.jupiter.api.Test;

class StudentTest {

    @Test
    void testCreateStudent() {
        Student student = new Student ("Jane Smith");
        String studentName = student.getName();

    }

}

```

Add getName() method - success



Finished after 0.063 seconds

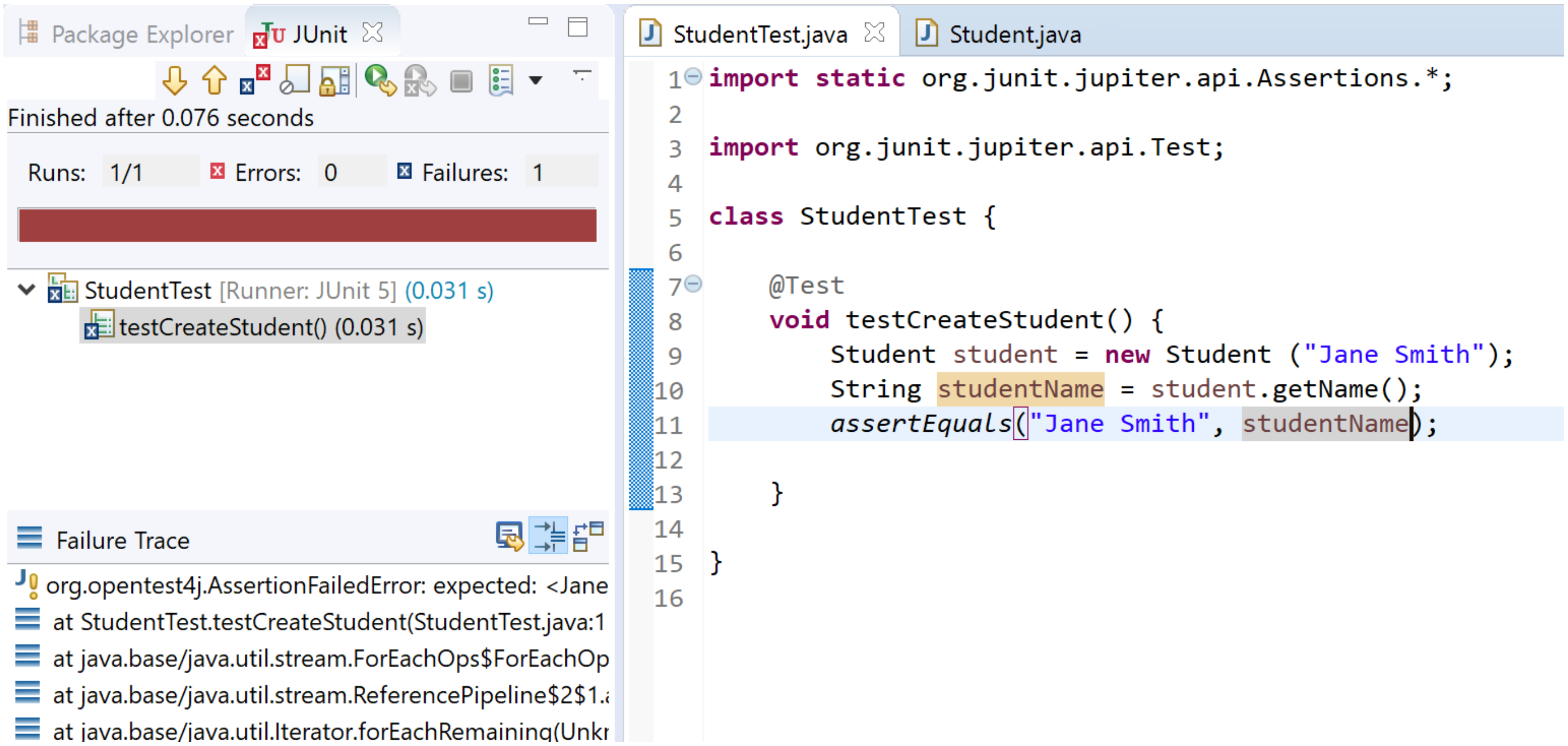
Runs: 1/1 Errors: 0 Failures: 0

StudentTest [Runner: JUnit 5] (0.016 s)

Failure Trace

```
public class Student {
    public Student (String name) {

    }

    public String getName() {
        return "";
    }
}

```

Assertion failure



```
Finished after 0.076 seconds
Runs: 1/1   Errors: 0   Failures: 1

StudentTest [Runner: JUnit 5] (0.031 s)
    testCreateStudent() (0.031 s)
```

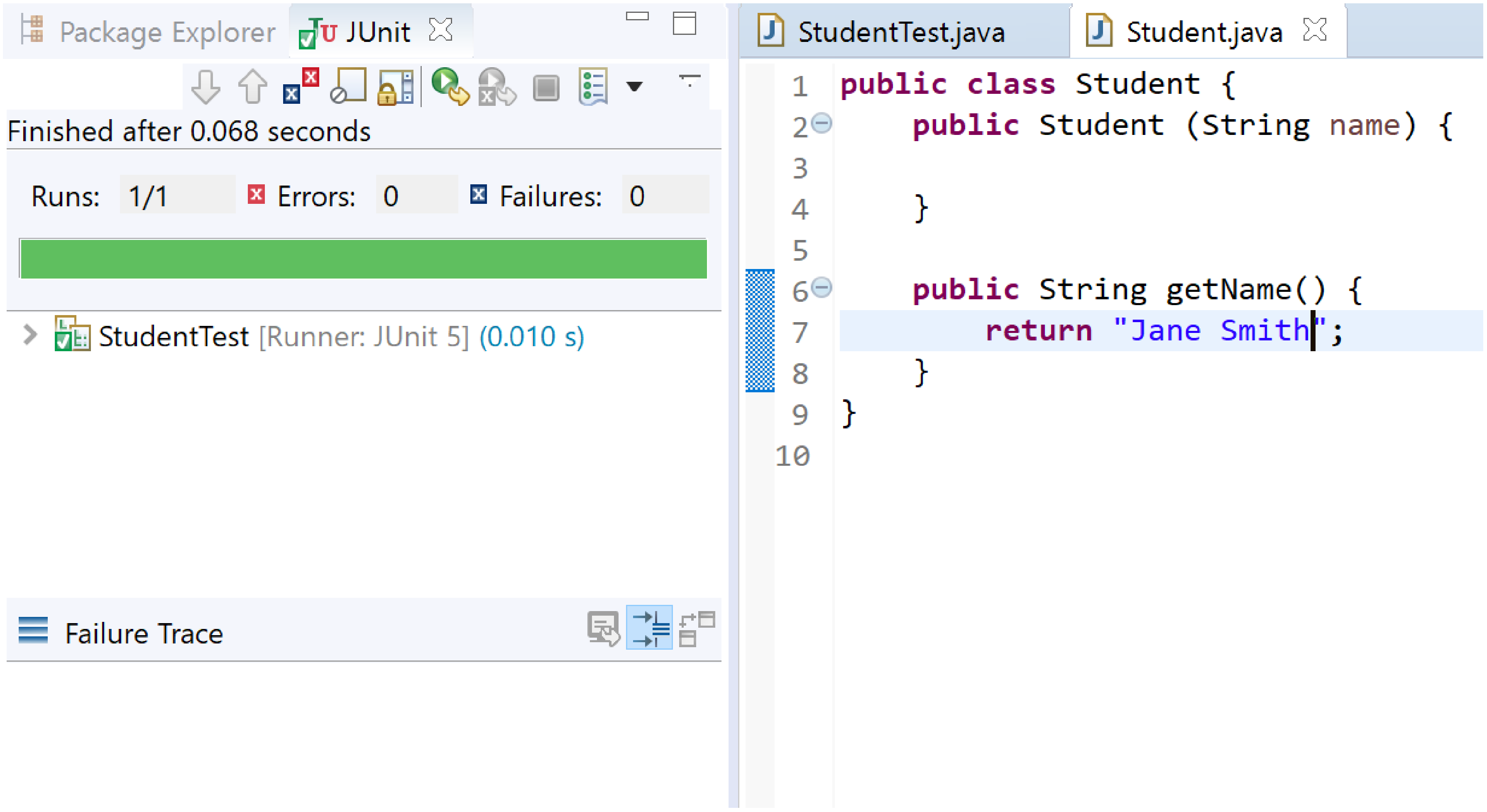
```
org.opentest4j.AssertionFailedError: expected: <Jane
at StudentTest.testCreateStudent(StudentTest.java:1
at java.base/java.util.stream.ForEachOps$ForEachOp
at java.base/java.util.stream.ReferencePipeline$2$1.
at java.base/java.util.Iterator.forEachRemaining(Unkr
```

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

import org.junit.jupiter.api.Test;

class StudentTest {

    @Test
    void testCreateStudent() {
        Student student = new Student ("Jane Smith");
        String studentName = student.getName();
        assertEquals("Jane Smith", studentName);

    }

}
```

`assertEquals("Jane Smith", studentName);`
An assertion that the first argument is the same as the second argument.

Queen Mary University of London

Assertion success



Package Explorer JUnit

Finished after 0.068 seconds

Runs: 1/1 Errors: 0 Failures: 0

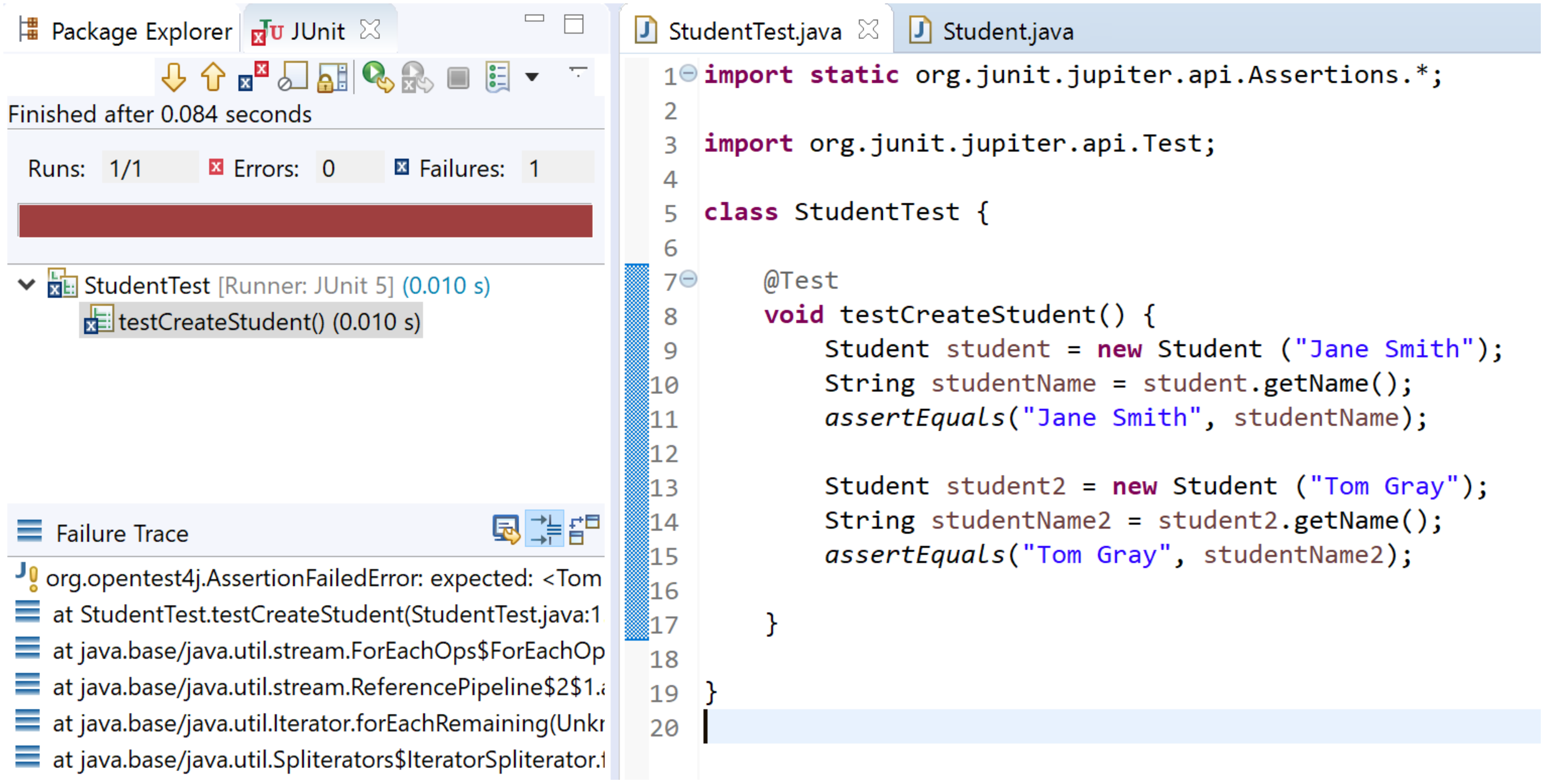
StudentTest [Runner: JUnit 5] (0.010 s)

Failure Trace

StudentTest.java Student.java

```java
public class Student {
    public Student (String name) {

    }

    public String getName() {
        return "Jane Smith";
    }
}
```

Assertion failure



Finished after 0.084 seconds

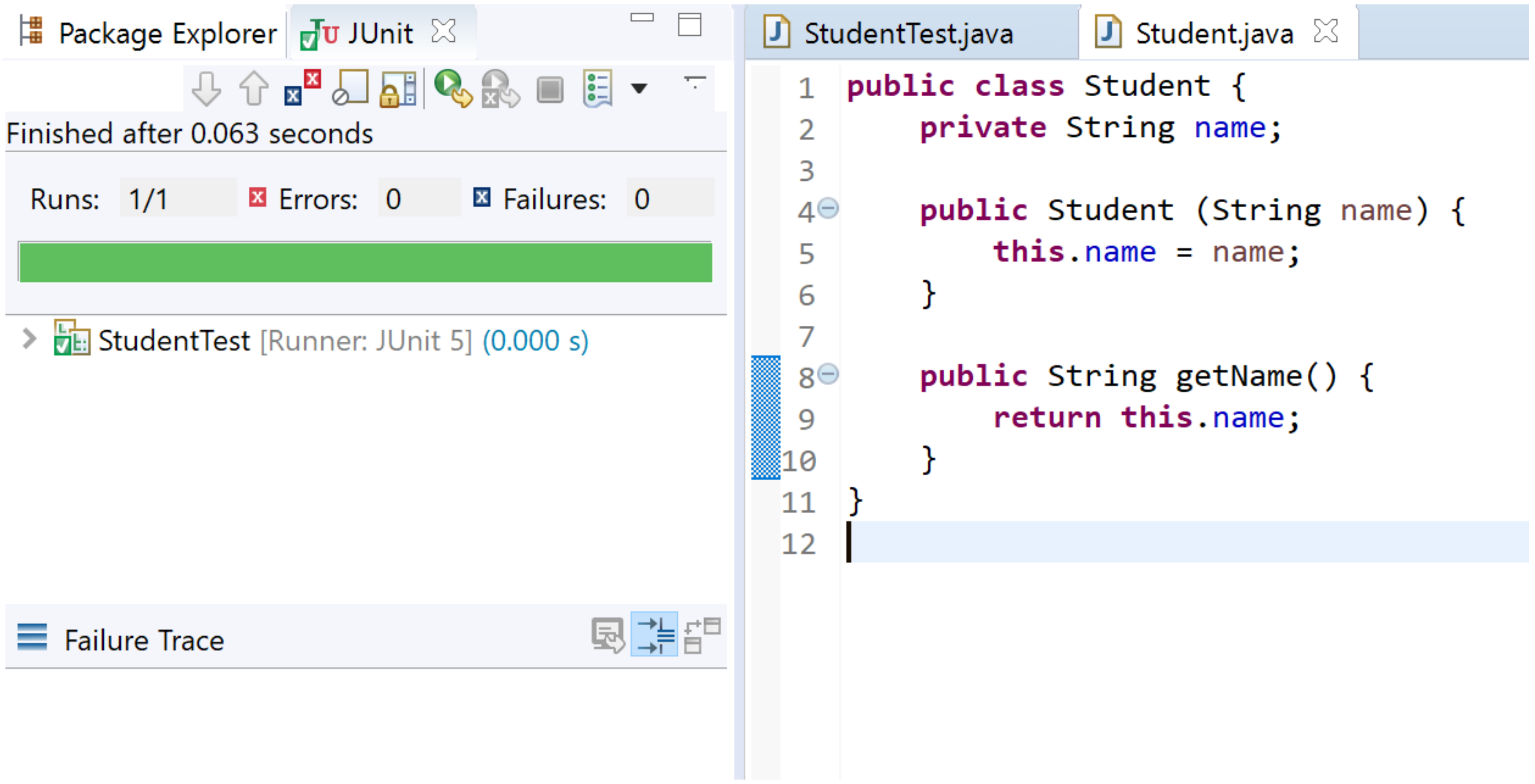
Runs: 1/1 Errors: 0 Failures: 1

StudentTest [Runner: JUnit 5] (0.010 s)
testCreateStudent() (0.010 s)

```
org.opentest4j.AssertionFailedError: expected: <Tom
at StudentTest.testCreateStudent(StudentTest.java:1
at java.base/java.util.stream.ForEachOps$ForEachOp
at java.base/java.util.stream.ReferencePipeline$2$1.
at java.base/java.util.Iterator.forEachRemaining(Unkr
at java.base/java.util.Spliterators$IteratorSpliterator.
```

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

import org.junit.jupiter.api.Test;

class StudentTest {

    @Test
    void testCreateStudent() {
        Student student = new Student ("Jane Smith");
        String studentName = student.getName();
        assertEquals("Jane Smith", studentName);

        Student student2 = new Student ("Tom Gray");
        String studentName2 = student2.getName();
        assertEquals("Tom Gray", studentName2);

    }

}
```

Assertion success



Finished after 0.063 seconds

Runs: 1/1 Errors: 0 Failures: 0

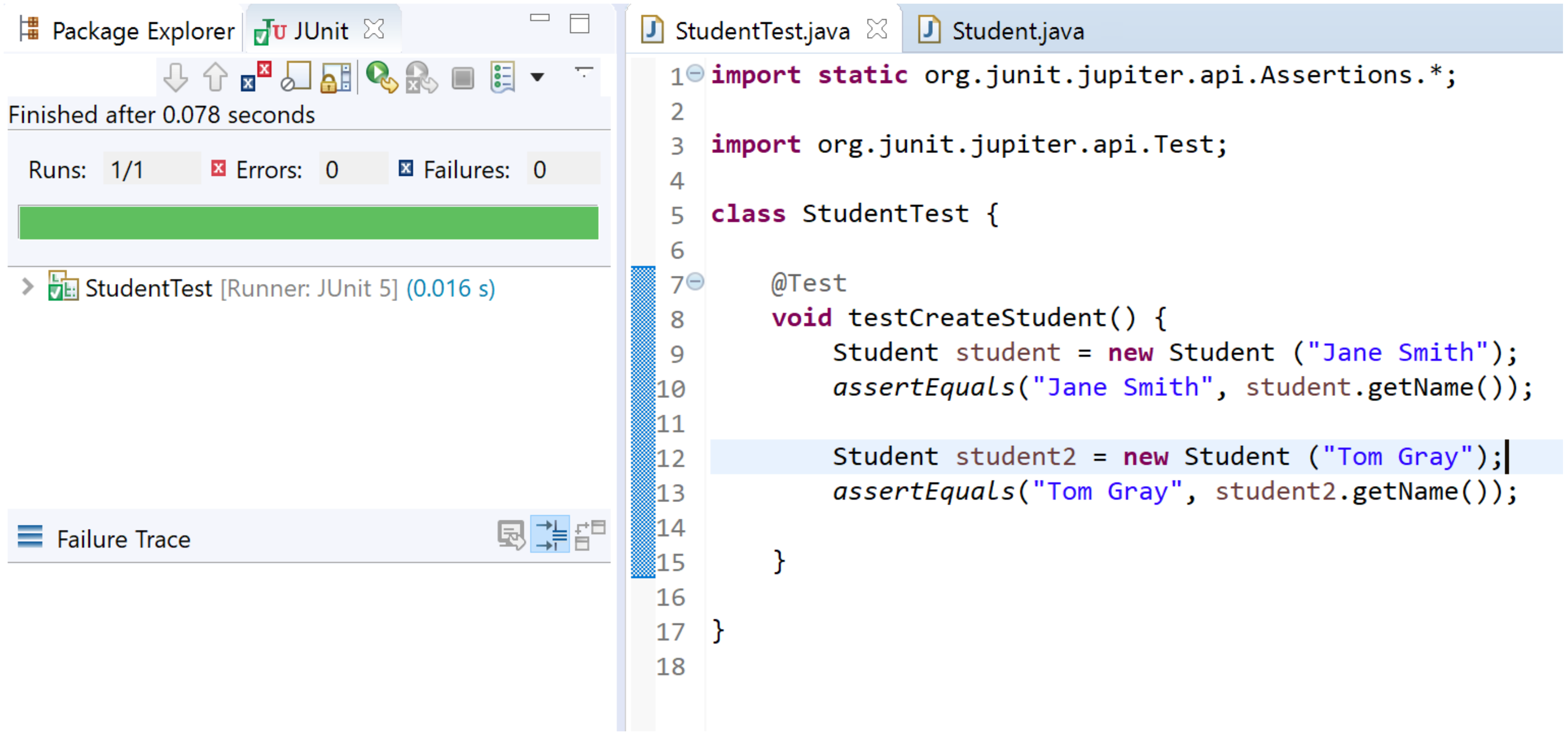
StudentTest [Runner: JUnit 5] (0.000 s)

```java
public class Student {
    private String name;

    public Student (String name) {
        this.name = name;
    }

    public String getName() {
        return this.name;
    }
}
```

Refactoring

- Your primary job is to get the code to work.
- Your second job is to ensure that the code stays clean.
  - No duplicate code in the system
  - The code is clean and expressive, clearly stating the intent of the code
- Frequently review your code
  - Refactoring

Refactoring our test code



Finished after 0.078 seconds

Runs: 1/1 Errors: 0 Failures: 0

StudentTest [Runner: JUnit 5] (0.016 s)

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

import org.junit.jupiter.api.Test;

class StudentTest {

    @Test
    void testCreateStudent() {
        Student student = new Student ("Jane Smith");
        assertEquals("Jane Smith", student.getName());

        Student student2 = new Student ("Tom Gray");
        assertEquals("Tom Gray", student2.getName());

    }

}
```

Remove unnecessary local variables

Useful methods

- assertTrue(condition)
- assertFalse(condition)
- assertNull(object)
- assertNotNull(object)
- assertEquals(expected, actual)
- assertArrayEquals(expected, actual)
- assertEquals(expected[i],actual[i])
- assertArrayEquals(expected[i],actual[i])
- fail(message)

https://junit.org/junit5/docs/5.0.1/api/org/junit/jupiter/api/Assertions.html

Small cycle

- Write a small test to assert some piece of functionality.
- Demonstrate that the test fails.
- Write a small bit of code to make this test pass.
- Refactor both the test and code, eliminating duplicate concepts and ensuring that the code is expressive.

Summary

- Test Driven Development in Agile Process
- Using JUnit

References

- Chapters 8 – “Software Engineering” textbook by Ian Sommerville
- Agile Java™: Crafting Code with Test-Driven Development, Jeff Langr