

# 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

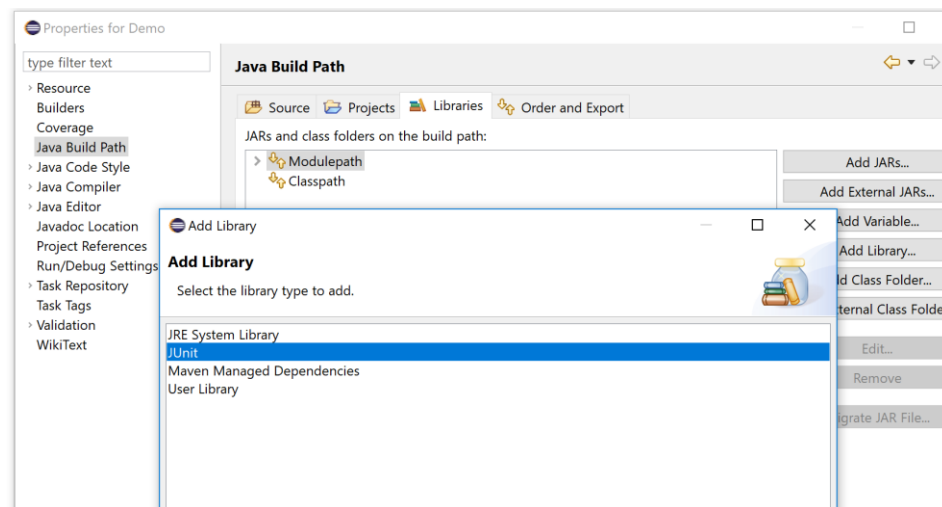
- Junit is a simple unit-testing framework for supporting TDD
- Download JUnit from <https://junit.org/>
- Current version: JUnit 5
- Comes with major IDE, e.g. Eclipse

# 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”



4. Choose the version JUnit 5

# Create a test class

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

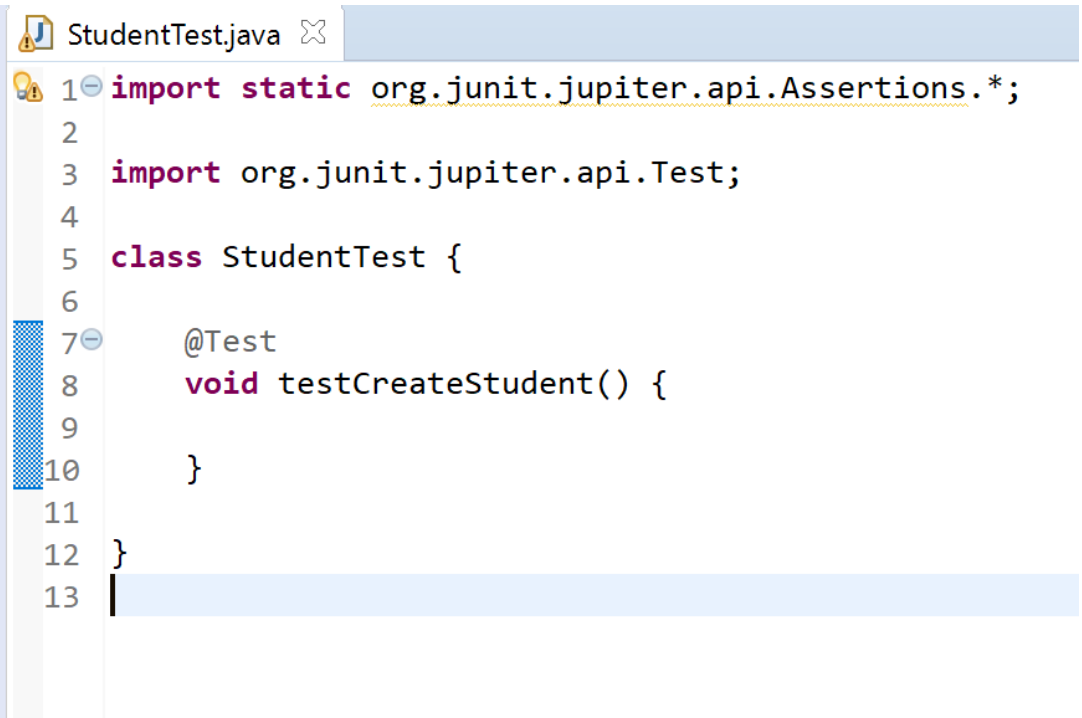
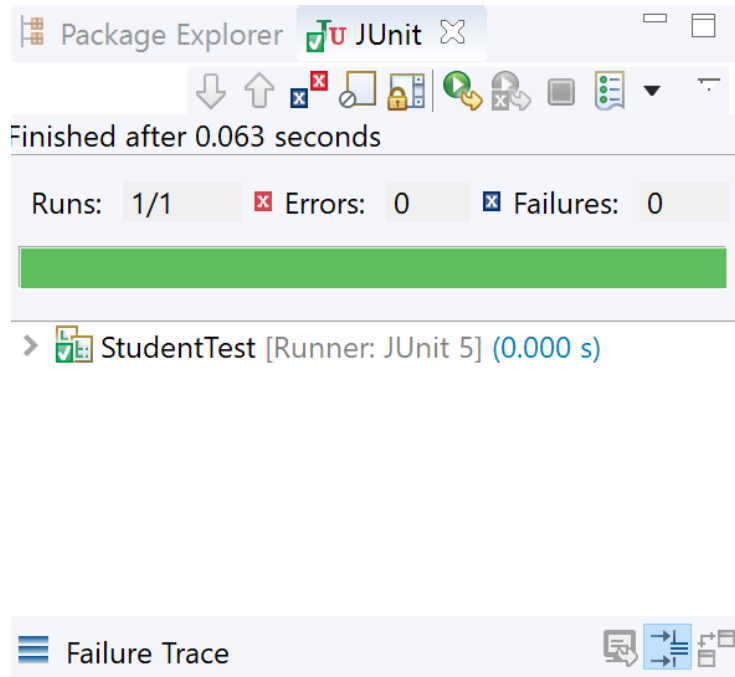
The screenshot shows an IDE interface with two main panels. The left panel displays the 'Package Explorer' and a 'JUnit' runner window. The runner window shows 'Finished after 0.078 seconds' and a summary: 'Runs: 1/1', 'Errors: 0', and 'Failures: 1'. Below this, a red progress bar indicates a failure. The right panel shows the 'StudentTest.java' file with the following code:

```
1 import static org.junit.jupiter.api.Assertions.*;
2
3 import org.junit.jupiter.api.Test;
4
5 class StudentTest {
6
7     @Test
8     void test() {
9         fail("Not yet implemented");
10    }
11
12 }
13
```

At the bottom of the left panel, the 'Failure Trace' is visible, showing the error: 'org.opentest4j.AssertionFailedError: Not yet implemented' at 'StudentTest.test(StudentTest.java:9)'.

**show test failure - red bar**

# Empty test

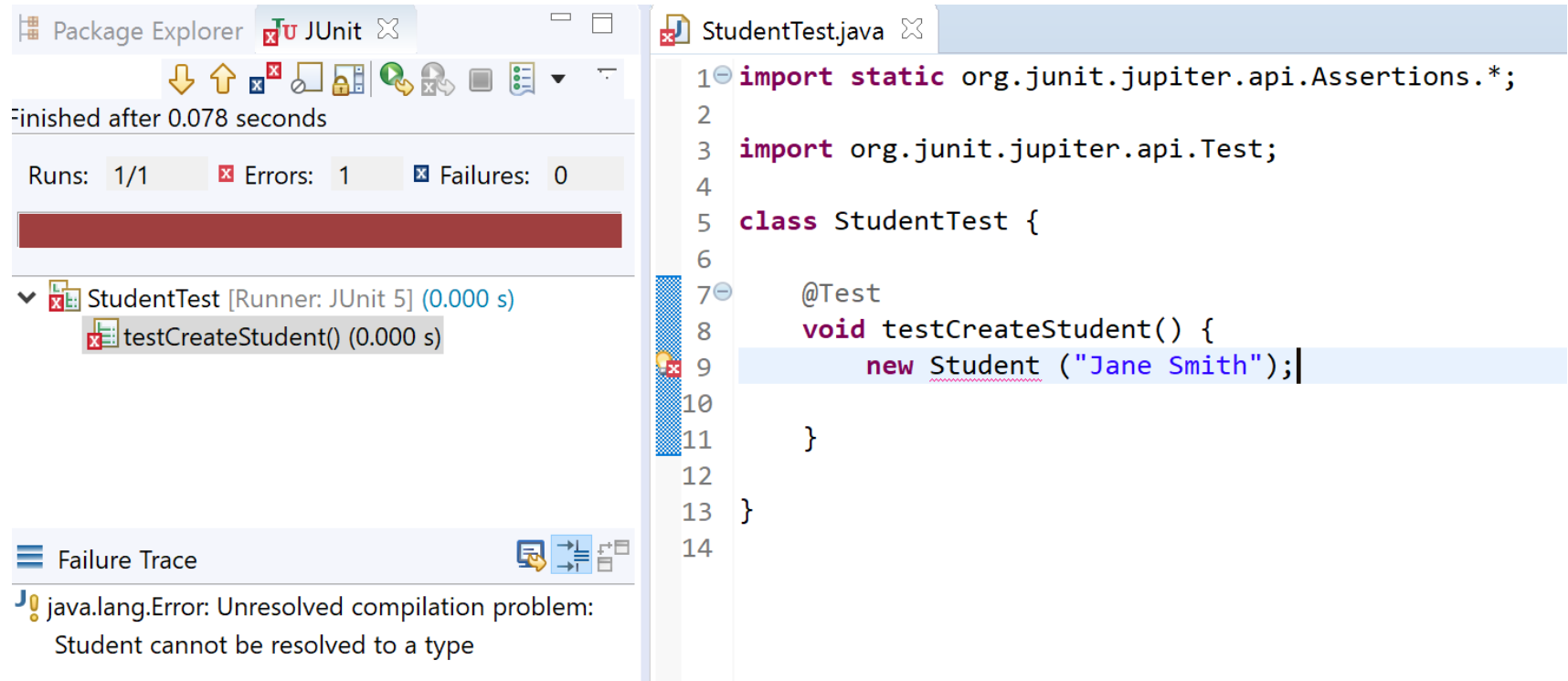


JUnit test success – green bar

An empty test method will always pass



# no object - failure



Package Explorer JUnit

Finished after 0.078 seconds

Runs: 1/1 Errors: 1 Failures: 0

StudentTest [Runner: JUnit 5] (0.000 s)

testCreateStudent() (0.000 s)

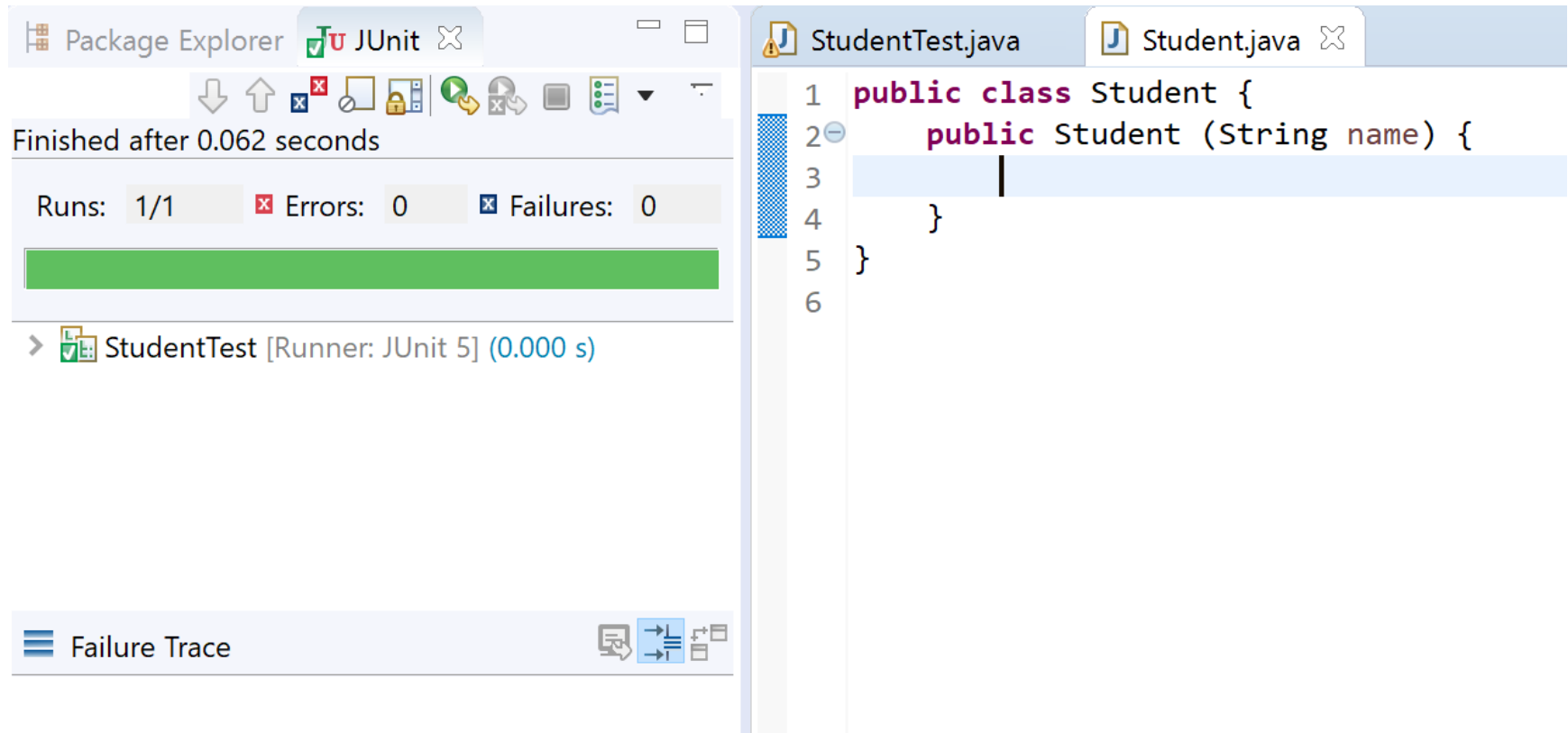
Failure Trace

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

```
1 import static org.junit.jupiter.api.Assertions.*;
2
3 import org.junit.jupiter.api.Test;
4
5 class StudentTest {
6
7     @Test
8     void testCreateStudent() {
9         new Student ("Jane Smith");
10
11     }
12
13 }
14
```

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

# Adding student class - success



The screenshot displays an IDE interface with two main panels. The left panel shows the JUnit test results, and the right panel shows the source code for the `Student` class.

**JUnit Results Panel (Left):**

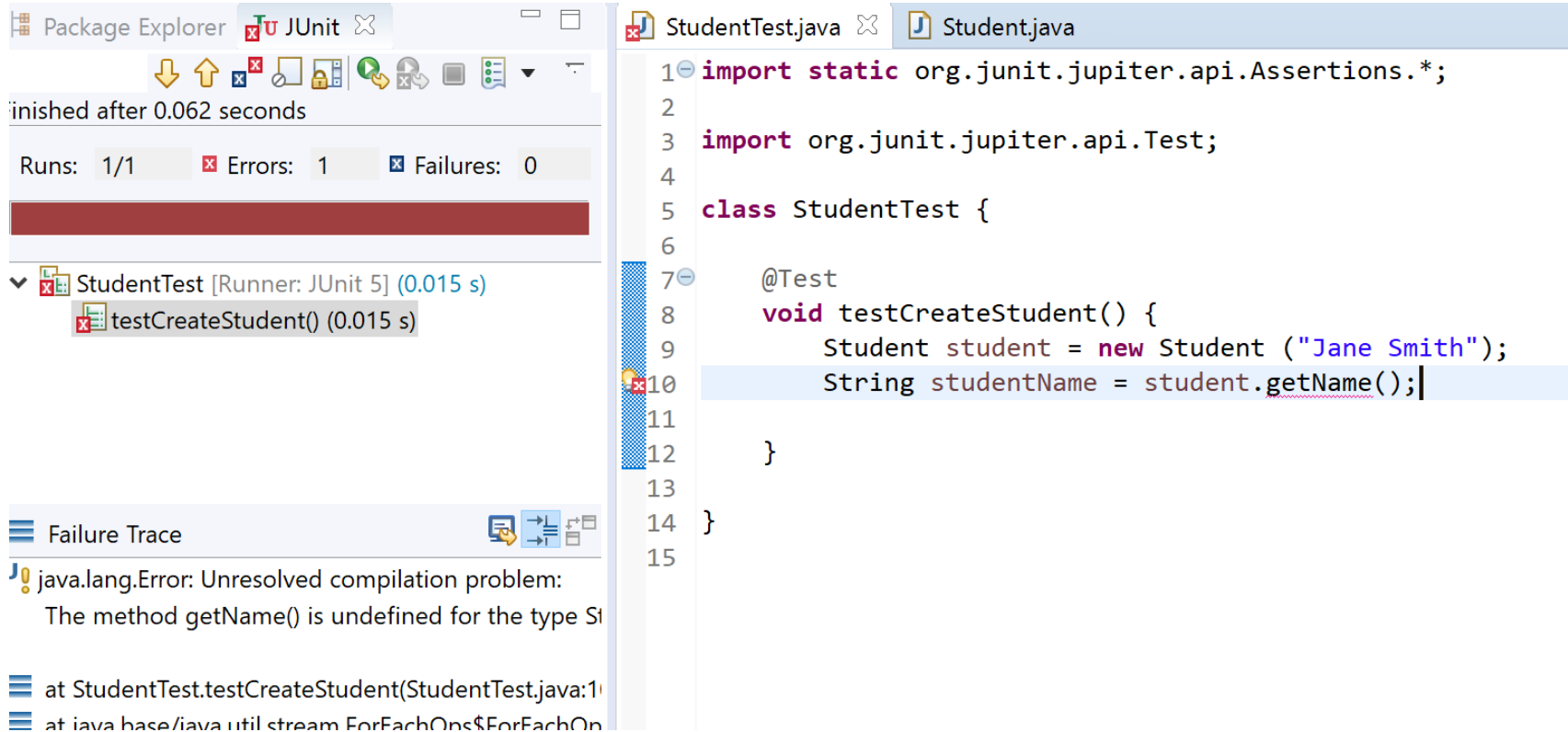
- Package Explorer: JUnit
- Finished after 0.062 seconds
- Runs: 1/1
- Errors: 0
- Failures: 0
- A green progress bar indicates successful completion.
- Test Run: StudentTest [Runner: JUnit 5] (0.000 s)
- Failure Trace: (empty)

**Source Code Panel (Right):**

The code is in `StudentTest.java` and defines the `Student` class:

```
1 public class Student {  
2     public Student (String name) {  
3         |  
4     }  
5 }  
6
```

# Undefined method - failure



The screenshot displays an IDE interface with a JUnit test runner on the left and a Java code editor on the right. The test runner shows a failure for the `testCreateStudent()` method. The error message in the Failure Trace is: `java.lang.Error: Unresolved compilation problem: The method getName() is undefined for the type Student`. The code in the editor shows the `StudentTest` class with a test method `testCreateStudent()` that creates a `Student` object and calls `getName()` on it. The `Student` class is not visible in the Package Explorer, which explains the error.

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)

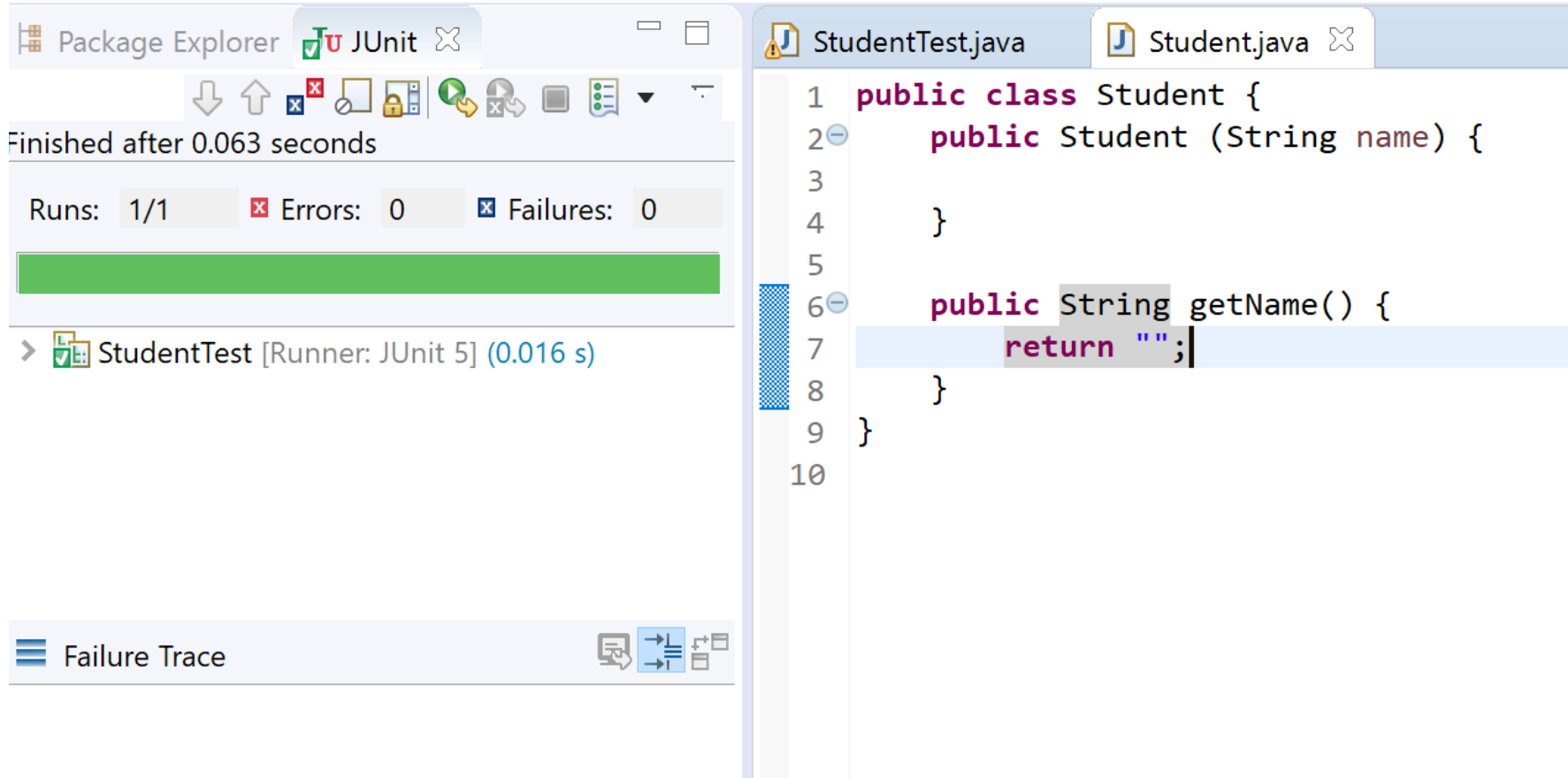
Failure Trace

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

at StudentTest.testCreateStudent(StudentTest.java:10)  
at java.base/java.util.stream.ForEachOps\$ForEachOp

```
1 import static org.junit.jupiter.api.Assertions.*;
2
3 import org.junit.jupiter.api.Test;
4
5 class StudentTest {
6
7     @Test
8     void testCreateStudent() {
9         Student student = new Student ("Jane Smith");
10        String studentName = student.getName();
11
12    }
13
14 }
15
```

# Add getName() method - success



The screenshot displays an IDE interface with two main panels. The left panel shows the JUnit test runner results, indicating a successful run. The right panel shows the source code for the `Student` class, where the `getName()` method has been implemented.

**JUnit Test Results:**

- Package Explorer: JUnit
- Finished after 0.063 seconds
- Runs: 1/1
- Errors: 0
- Failures: 0
- StudentTest [Runner: JUnit 5] (0.016 s)

**Source Code (Student.java):**

```
1 public class Student {
2     public Student (String name) {
3
4     }
5
6     public String getName() {
7         return "";
8     }
9 }
10
```

# Assertion failure

Package Explorer JUnit

Finished after 0.076 seconds

Runs: 1/1 Errors: 0 Failures: 1

StudentTest [Runner: JUnit 5] (0.031 s)

testCreateStudent() (0.031 s)

Failure Trace

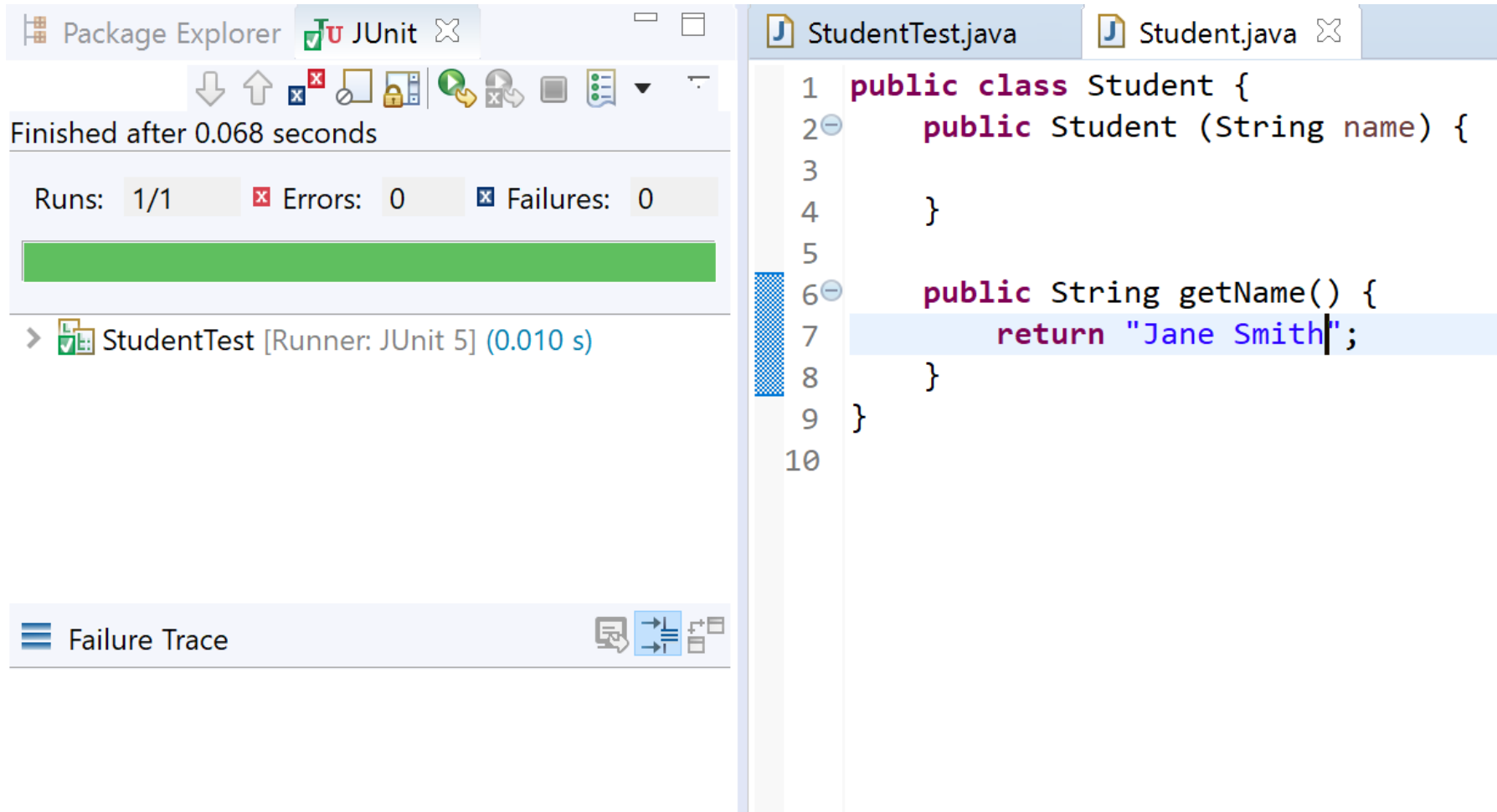
org.opentest4j.AssertionFailedError: expected: <Jane Smith>  
at StudentTest.testCreateStudent(StudentTest.java:11)  
at java.base/java.util.stream.ForEachOps\$ForEachOp\$OfRef.accept(ForEachOps.java:183)  
at java.base/java.util.stream.ReferencePipeline\$3\$1.accept(ReferencePipeline.java:197)  
at java.base/java.util.Iterator.forEachRemaining(Iterator.java:116)  
at java.base/java.util.stream.ReferencePipeline\$2\$1.accept(ReferencePipeline.java:171)  
at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)  
at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)

```
1 import static org.junit.jupiter.api.Assertions.*;
2
3 import org.junit.jupiter.api.Test;
4
5 class StudentTest {
6
7     @Test
8     void testCreateStudent() {
9         Student student = new Student ("Jane Smith");
10        String studentName = student.getName();
11        assertEquals("Jane Smith", studentName);
12    }
13
14 }
15
16 }
```

**`assertEquals("Jane Smith", studentName);`**

An assertion that the first argument is the same as the second argument.

# Assertion success



The screenshot displays an IDE interface with the Package Explorer on the left and the Java editor on the right. The Package Explorer shows the JUnit runner results for StudentTest, indicating a successful run with 1/1 tests passed, 0 errors, and 0 failures. The Java editor shows the source code for StudentTest.java, which includes a public class Student with a constructor and a getName() method. The getName() method returns the string "Jane Smith".

Package Explorer JUnit

Finished after 0.068 seconds

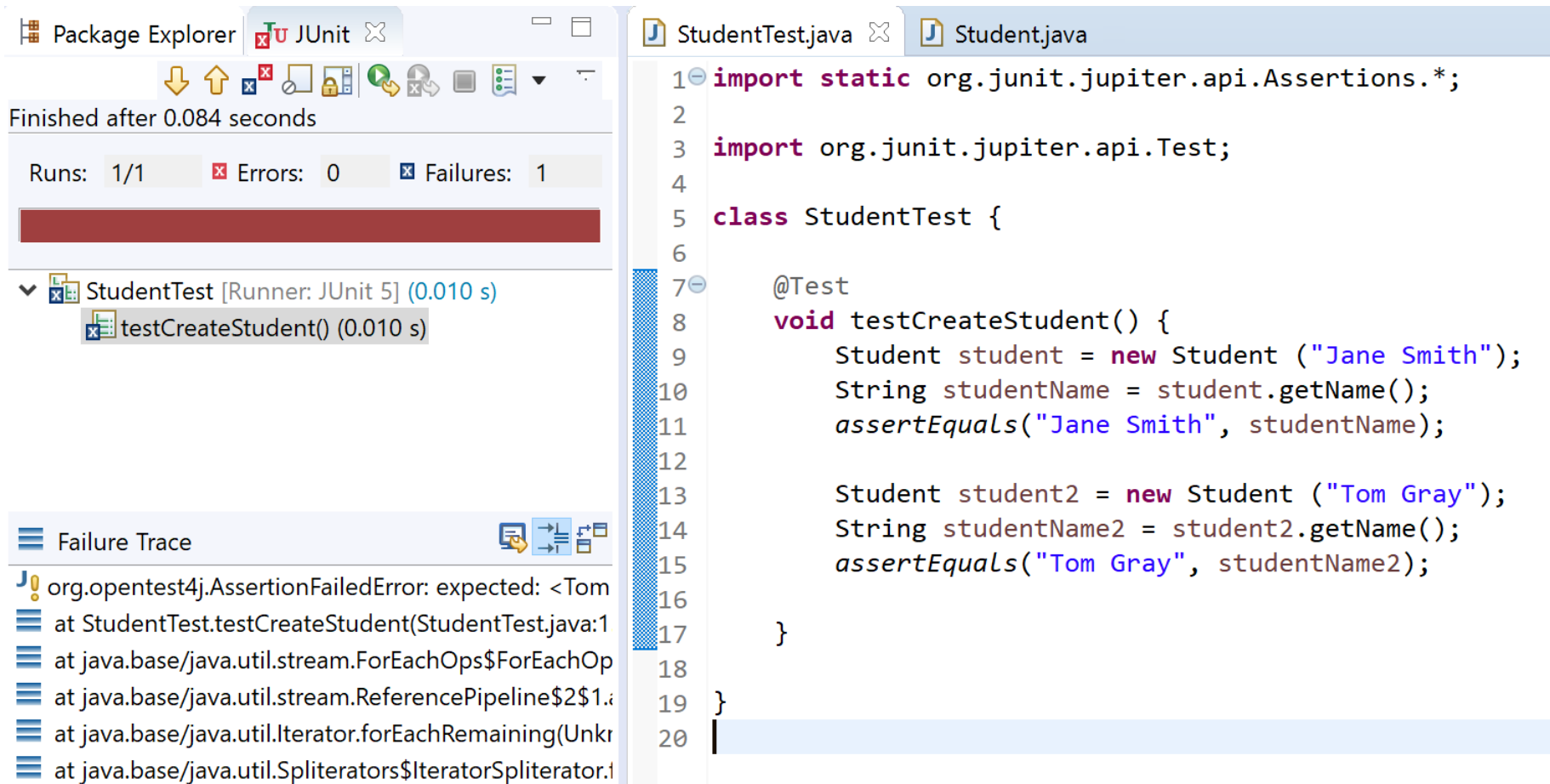
Runs: 1/1 Errors: 0 Failures: 0

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

Failure Trace

```
1 public class Student {
2     public Student (String name) {
3
4     }
5
6     public String getName() {
7         return "Jane Smith";
8     }
9 }
10
```

# Assertion failure



Package Explorer JUnit

Finished after 0.084 seconds

Runs: 1/1 Errors: 0 Failures: 1

StudentTest [Runner: JUnit 5] (0.010 s)

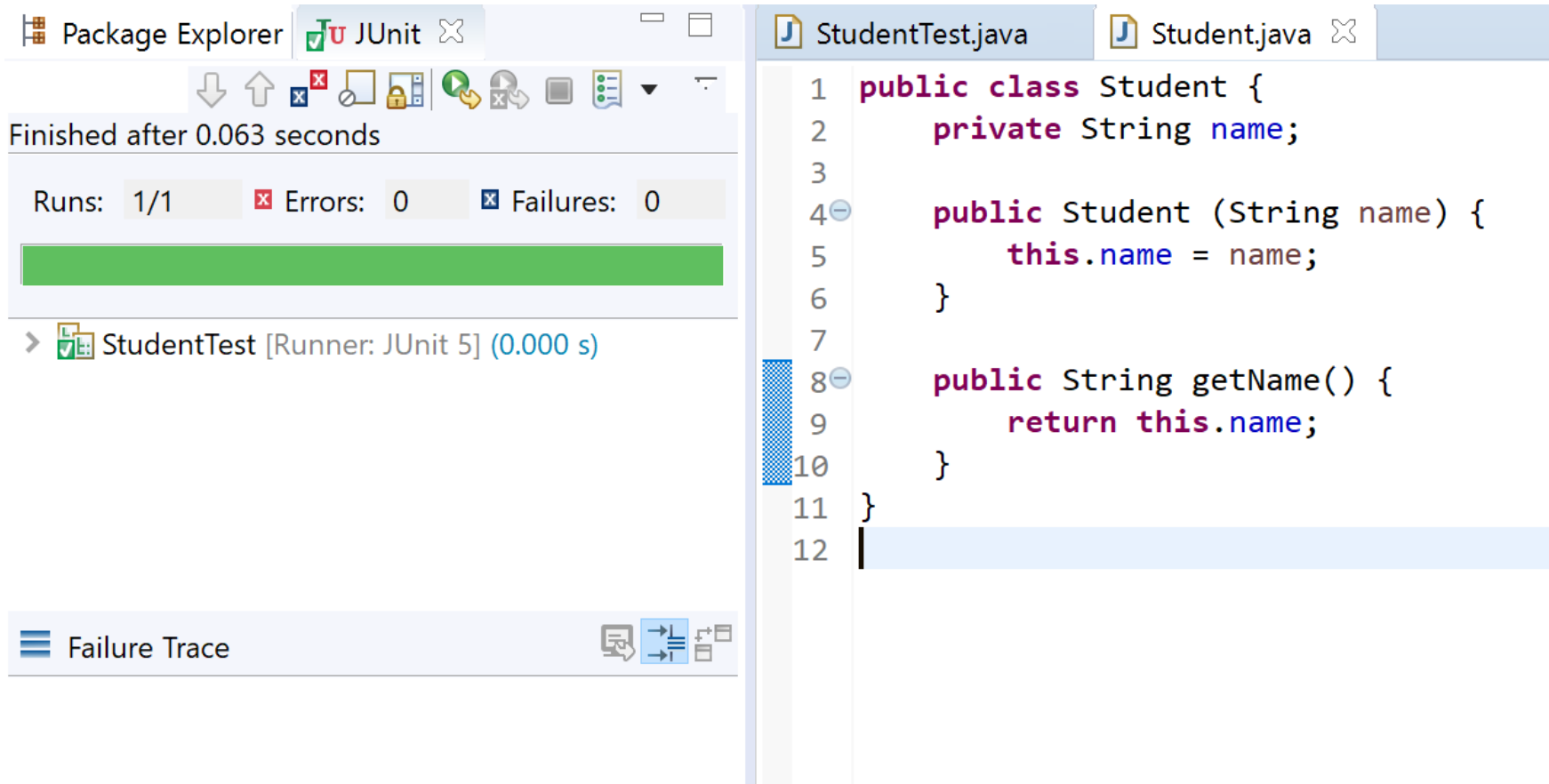
testCreateStudent() (0.010 s)

Failure Trace

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.a  
at java.base/java.util.Iterator.forEachRemaining(Unkr  
at java.base/java.util.Spliterators\$IteratorSpliterator.i

```
1 import static org.junit.jupiter.api.Assertions.*;
2
3 import org.junit.jupiter.api.Test;
4
5 class StudentTest {
6
7     @Test
8     void testCreateStudent() {
9         Student student = new Student ("Jane Smith");
10        String studentName = student.getName();
11        assertEquals("Jane Smith", studentName);
12
13        Student student2 = new Student ("Tom Gray");
14        String studentName2 = student2.getName();
15        assertEquals("Tom Gray", studentName2);
16
17    }
18
19 }
20
```

# Assertion success



The screenshot displays an IDE interface with the Package Explorer on the left and the Java editor on the right. The Package Explorer shows the JUnit runner results for StudentTest, indicating a successful run with no errors or failures. The Java editor shows the source code for StudentTest.java, which defines a Student class with a private String name attribute, a constructor, and a getName() method.

Package Explorer JUnit

Finished after 0.063 seconds

Runs: 1/1 Errors: 0 Failures: 0

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

Failure Trace

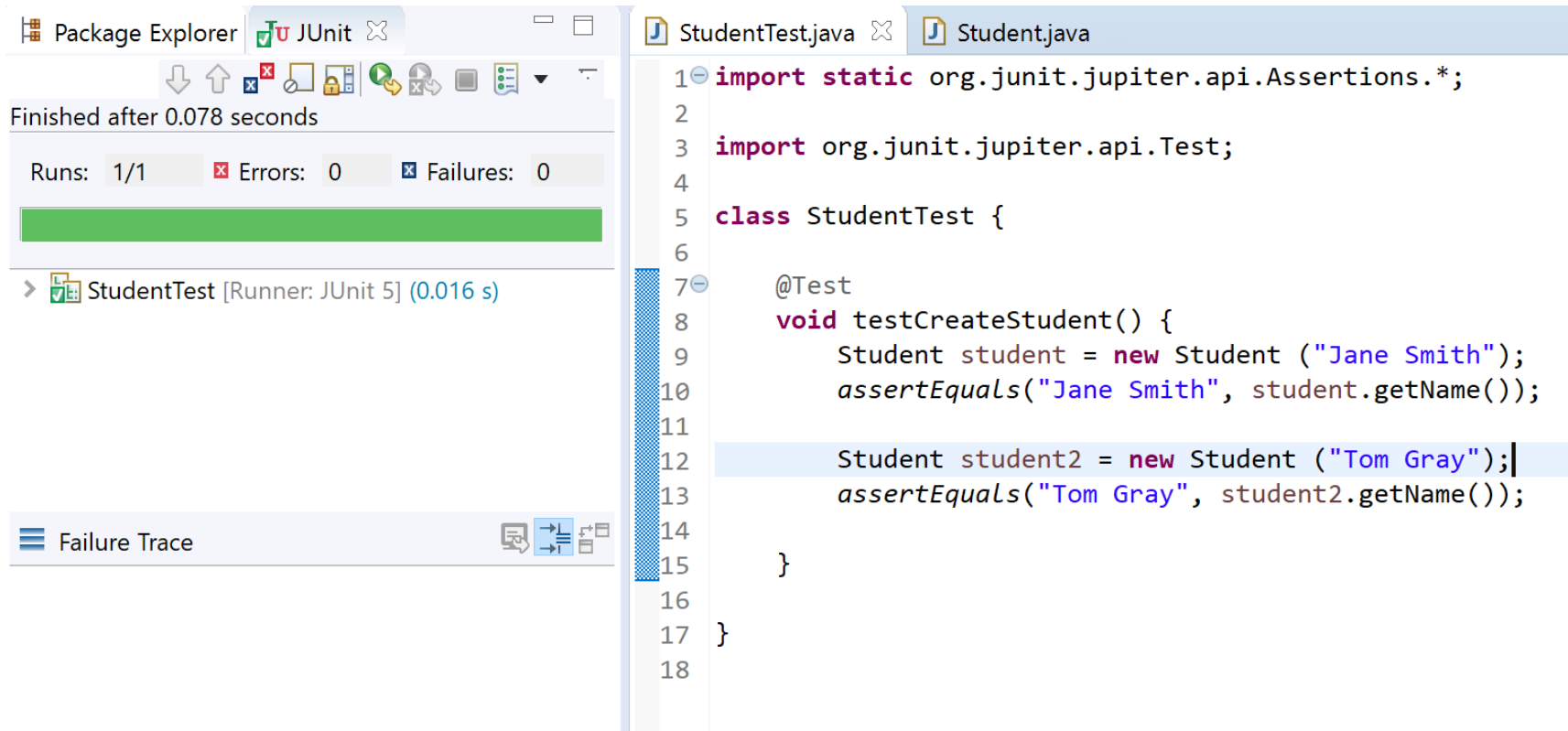
```
1 public class Student {
2     private String name;
3
4     public Student (String name) {
5         this.name = name;
6     }
7
8     public String getName() {
9         return this.name;
10    }
11 }
12
```



# 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



The screenshot displays an IDE interface. On the left, the Package Explorer shows the JUnit tab. Below it, a status bar indicates the test finished after 0.078 seconds, with 1/1 runs, 0 errors, and 0 failures. A green progress bar is shown. Below the progress bar, a link to 'StudentTest [Runner: JUnit 5] (0.016 s)' is visible. At the bottom left, there is a 'Failure Trace' button. On the right, the code editor shows the 'StudentTest.java' file. The code is as follows:

```
1 import static org.junit.jupiter.api.Assertions.*;
2
3 import org.junit.jupiter.api.Test;
4
5 class StudentTest {
6
7     @Test
8     void testCreateStudent() {
9         Student student = new Student ("Jane Smith");
10        assertEquals("Jane Smith", student.getName());
11
12        Student student2 = new Student ("Tom Gray");
13        assertEquals("Tom Gray", student2.getName());
14
15    }
16
17 }
18
```

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