CS2212

Introduction to Software Engineering

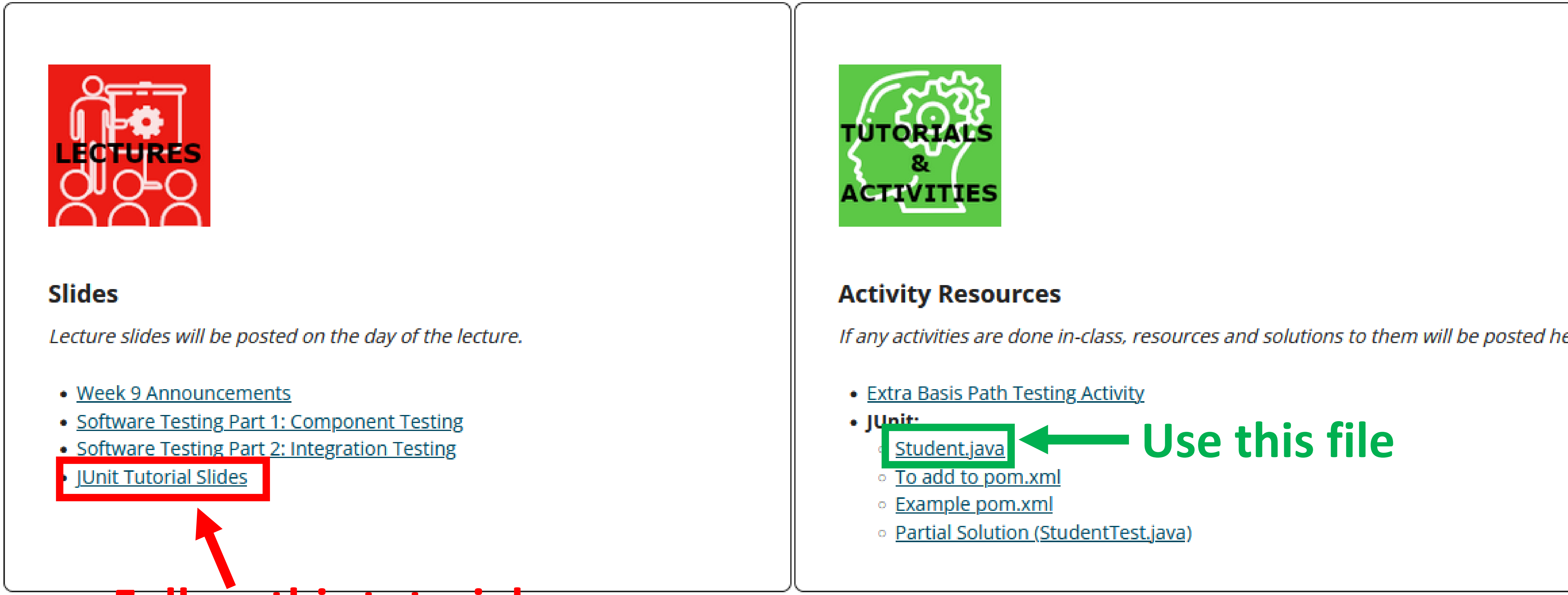
Junit Tutorial

Ask Questions Live

cs1.ca/ask

JUnit Tutorial

On week 9 page on OWL



Slides

Lecture slides will be posted on the day of the lecture.

- Week 9 Announcements
- Software Testing Part 1: Component Testing
- Software Testing Part 2: Integration Testing
- JUnit Tutorial Slides

Activity Resources

If any activities are done in-class, resources and solutions to them will be posted he

- Extra Basis Path Testing Activity
- JUnit:
  - Student.java
  - To add to pom.xml
  - Example pom.xml
  - Partial Solution (StudentTest.java)

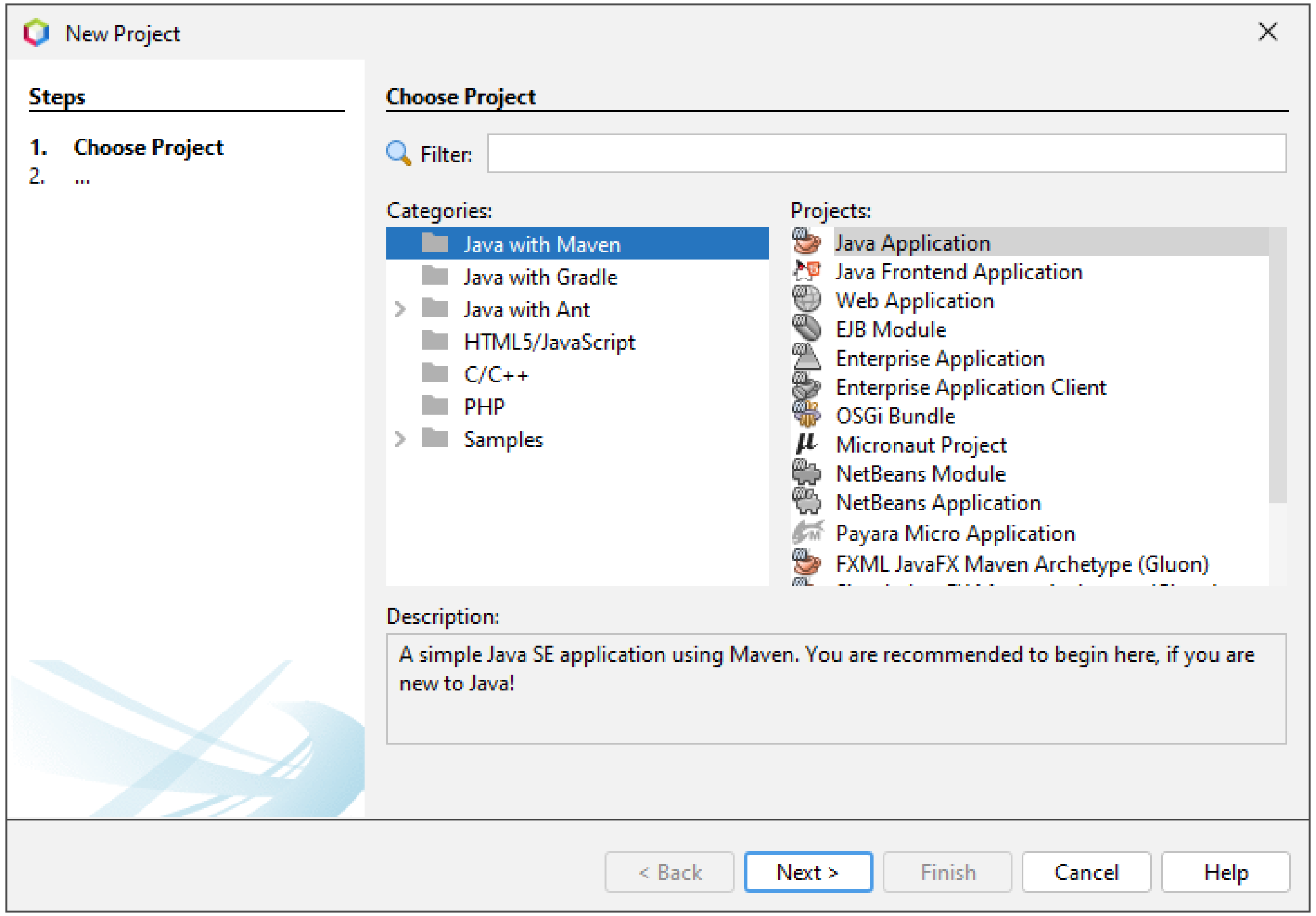
Use this file

Follow this tutorial

Get a Project Setup

Let's start by creating a new project in NetBeans:

1. Create a basic Java Application:


New Project

Steps

1. Choose Project
2. ...

Choose Project

Filter:

Categories:

- Java with Maven
- Java with Gradle
- Java with Ant
- HTML5/JavaScript
- C/C++
- PHP
- Samples

Projects:

- Java Application
- Java Frontend Application
- Web Application
- EJB Module
- Enterprise Application
- Enterprise Application Client
- OSGi Bundle
- Micronaut Project
- NetBeans Module
- NetBeans Application
- Payara Micro Application
- FXML JavaFX Maven Archetype (Gluon)

Description:

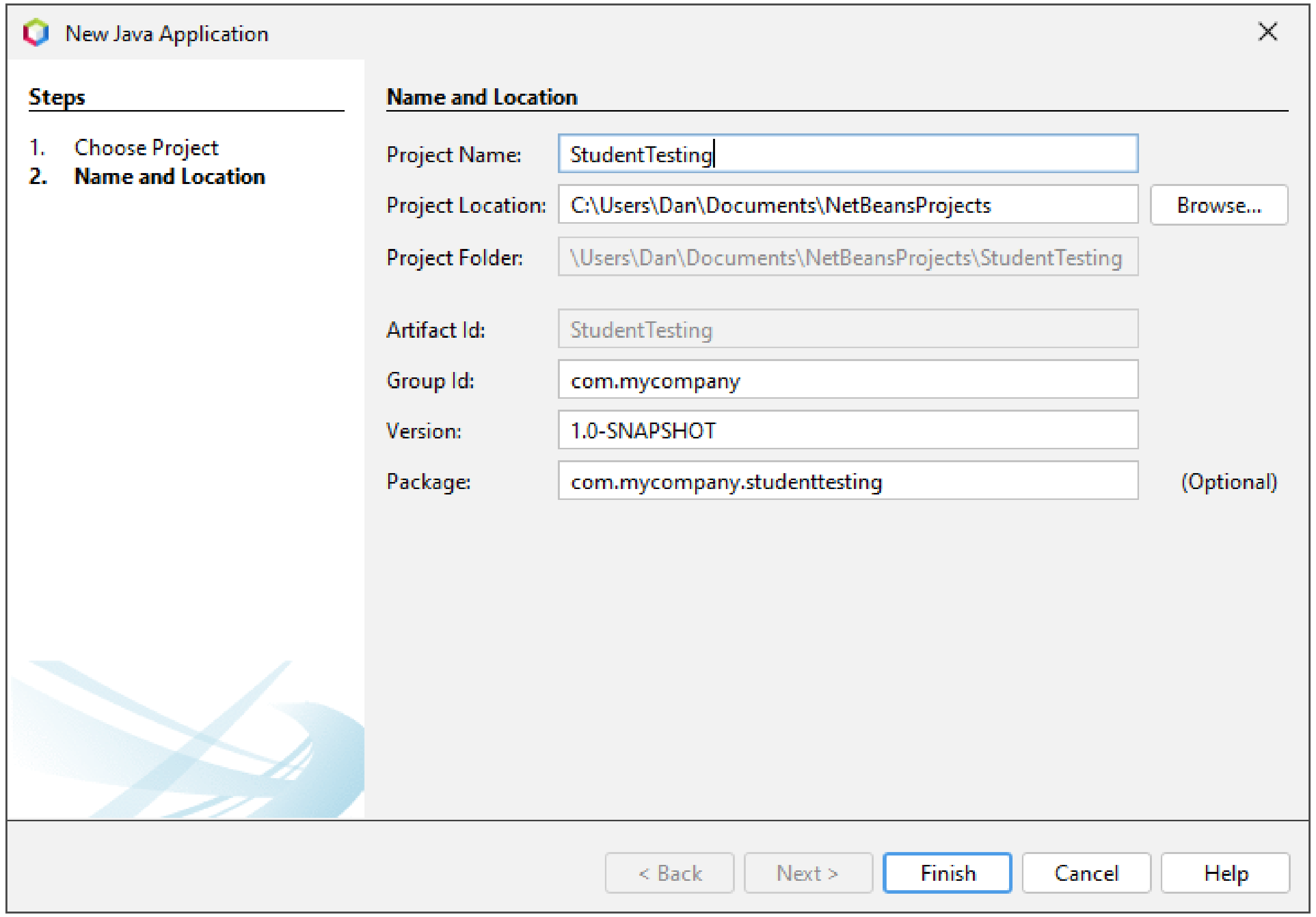
A simple Java SE application using Maven. You are recommended to begin here, if you are new to Java!

< Back | Next > | Finish | Cancel | Help

Get a Project Setup

Let's start by creating a new project in NetBeans:

2. Name it StudentTesting, make sure the Group Id is `com.mycompany` package is `com.mycompany.studenttesting`, the other settings can remain as defaults:



New Java Application

Steps

1. Choose Project
2. **Name and Location**

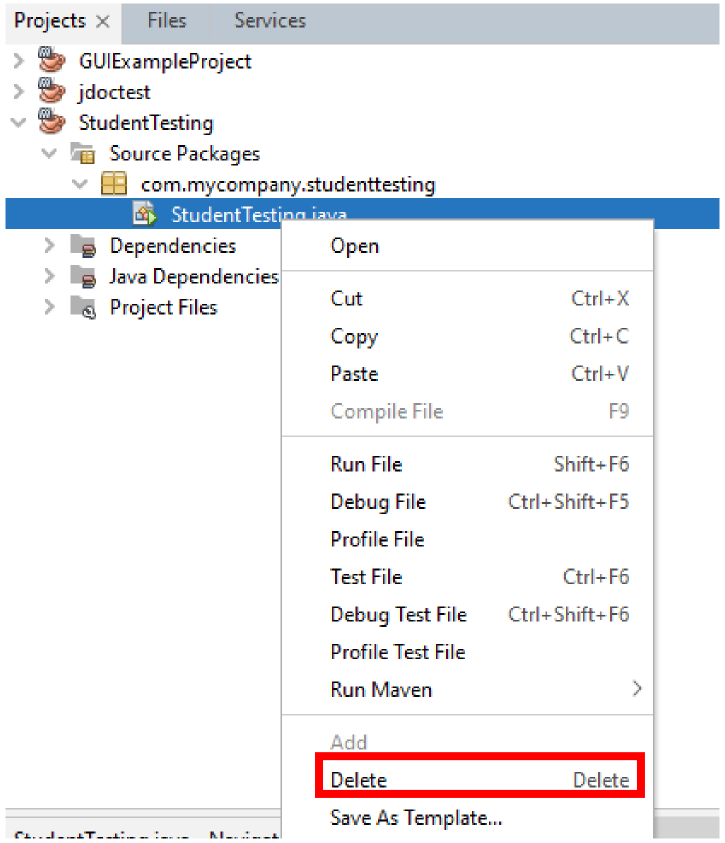
Name and Location

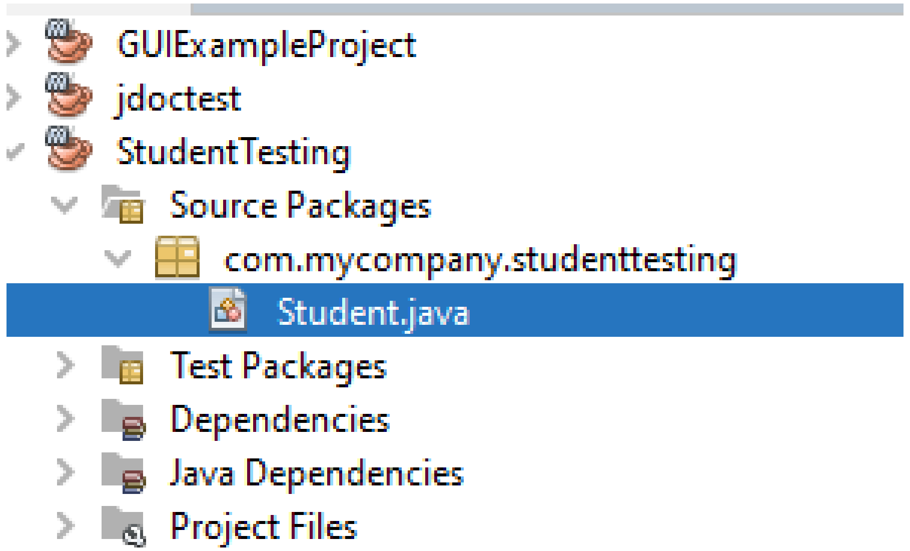
Project Name:	StudentTesting
Project Location:	C:\Users\Dan\Documents\NetBeansProjects
Project Folder:	\Users\Dan\Documents\NetBeansProjects\StudentTesting
Artifact Id:	StudentTesting
Group Id:	com.mycompany
Version:	1.0-SNAPSHOT
Package:	com.mycompany.studenttesting (Optional)

Get a Project Setup

Let's start by creating a new project in NetBeans:

3. Delete the StudentTesting.java file, and add the Student.java file from this weeks page on OWL (can simply drag and drop Student.java into the com.mycompany.studenttesting package):

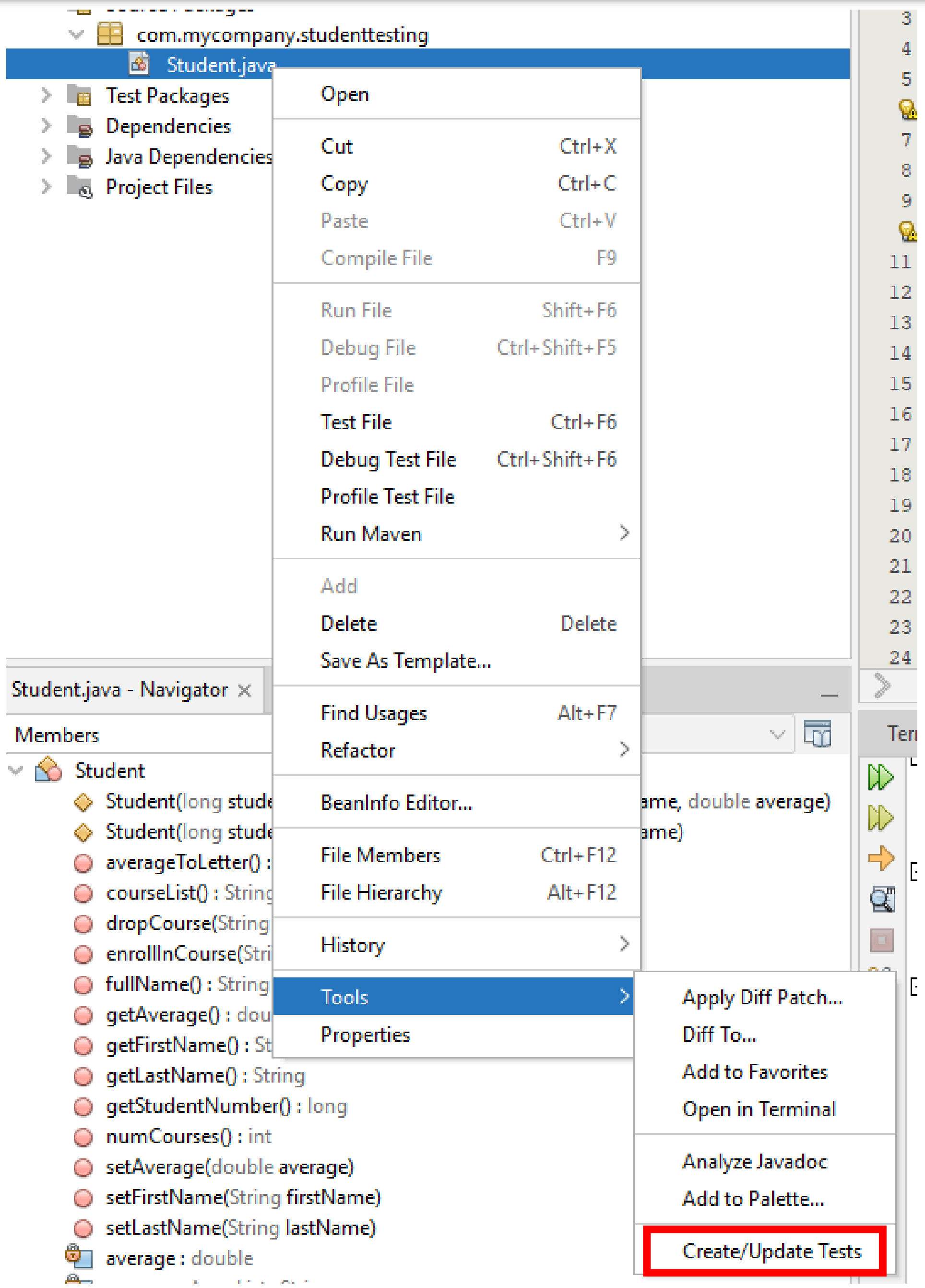




Create a Test

Let's create a test for the Student class.

1. Right click on Student.java and select Tools -> Create/Update Tests.

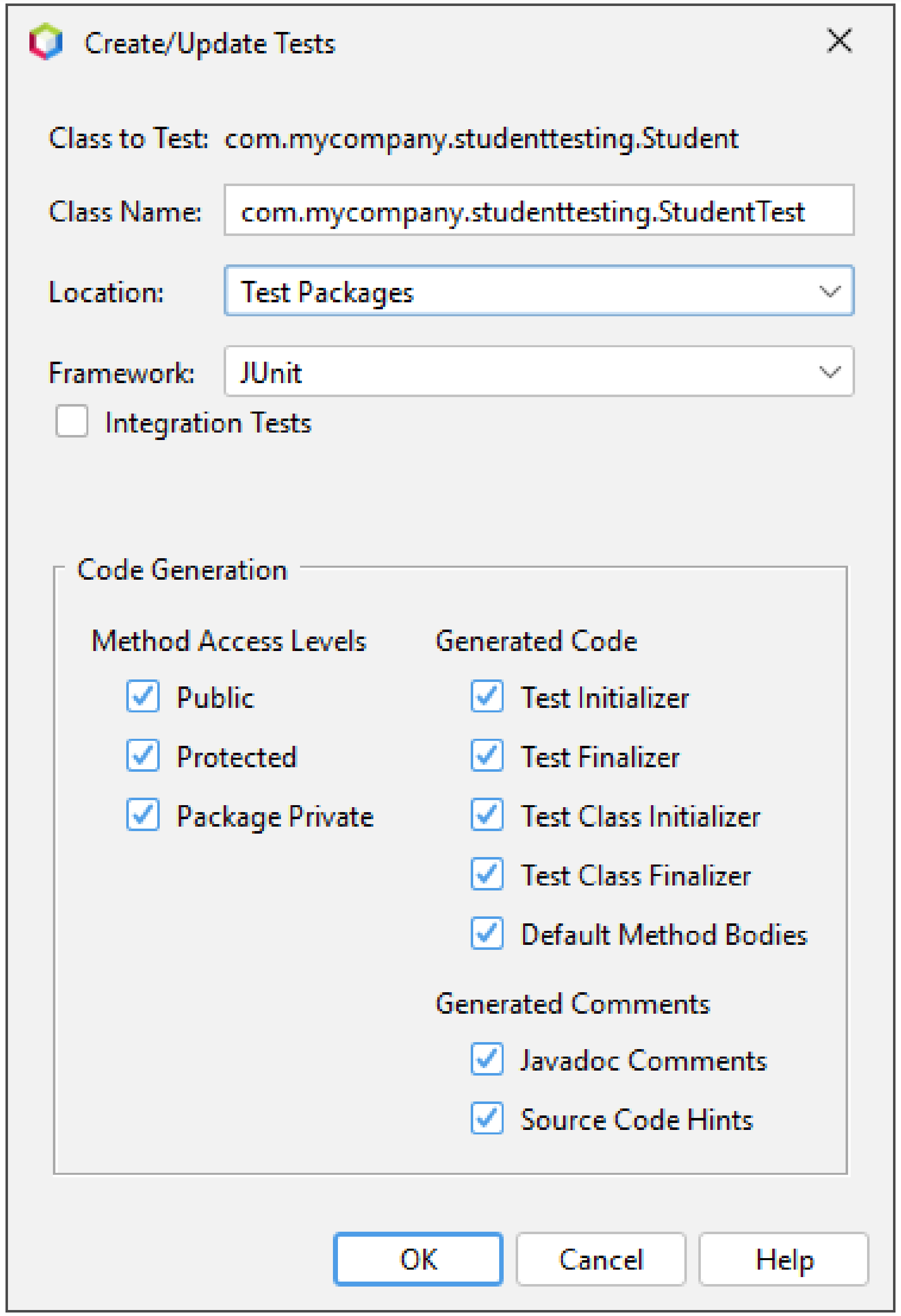


Create a Test

Let's create a test for the Student class.

2. In the Create/Update Tests window make sure that JUint is selected as the testing framework and that "integration tests" is unchecked.

3. Make sure all the check boxes in Code Generation are checked and click the Ok button.


Create/Update Tests

Class to Test: com.mycompany.studenttesting.Student

Class Name: com.mycompany.studenttesting.StudentTest

Location: Test Packages

Framework: JUnit

[ ] Integration Tests

Code Generation

Method Access Levels
- [✓] Public
- [✓] Protected
- [✓] Package Private

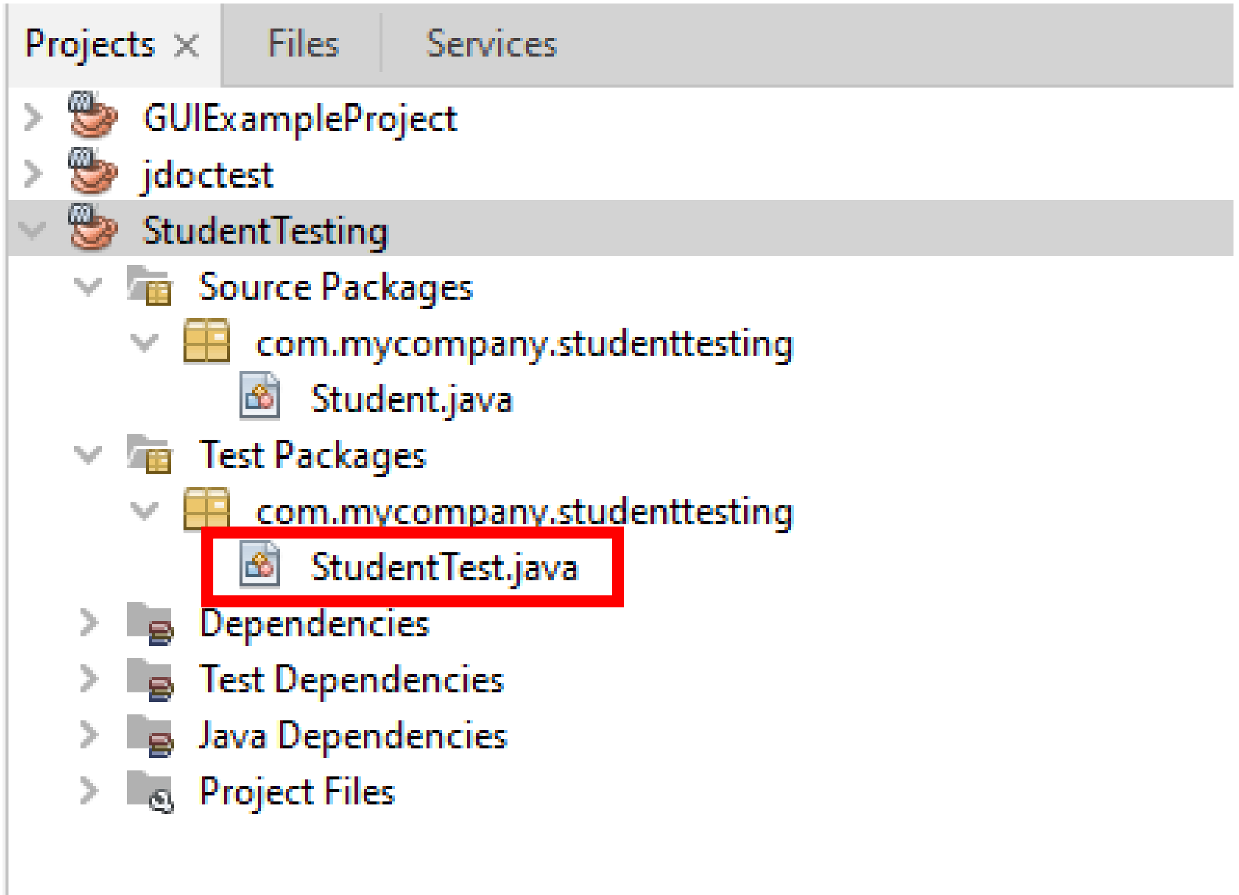
Generated Code
- [✓] Test Initializer
- [✓] Test Finalizer
- [✓] Test Class Initializer
- [✓] Test Class Finalizer
- [✓] Default Method Bodies

Generated Comments
- [✓] Javadoc Comments
- [✓] Source Code Hints

Create a Test

Let’s create a test for the Student class.

4. NetBeans will have created a new StudentTest class located in the Test Packages. This class will have the start of tests for each method in the Student class.



Test Class

```
package com.mycompany.studenttesting;

import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.AfterAll;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

/**
 *
 * @author Dan
 */
public class StudentTest {

    public StudentTest() {
    }

    @BeforeAll
    public static void setUpClass() {
```

```
 */
public class StudentTest {

    public StudentTest() {
    }

    @BeforeAll
    public static void setUpClass() {
    }

    @AfterAll
    public static void tearDownClass() {
    }

    @BeforeEach
    public void setUp() {
    }

    @AfterEach
    public void tearDown() {
    }

    /**
     * Test of averageToLetter method, of class Student.
     */
```

These methods are our test fixtures.

They set up and tear down anything we need for testing before and after our tests are run.

For example, they might set up test files, database connections, or objects used by all tests.

```
  */
public class StudentTest {

    public StudentTest() {
    }

    @BeforeAll
    public static void setUpClass() {
    }

    @AfterAll
    public static void tearDownClass() {
    }

    @BeforeEach
    public void setUp() {
    }

    @AfterEach
    public void tearDown() {
    }

    /**
     * Test of averageToLetter method, of class Student.
     */
```

Methods annotated with **@BeforeAll** are run **before** any tests.

Used to setup anything needed for testing beforehand. For example, setting up database connections, filling files with test data, etc.

@BeforeAll is analogous to @BeforeClass in JUnit 4

```
 */
public class StudentTest {

    public StudentTest() {
    }

    @BeforeAll
    public static void setUpClass() {
    }

    @AfterAll
    public static void tearDownClass() {
    }

    @BeforeEach
    public void setUp() {
    }

    @AfterEach
    public void tearDown() {
    }

    /**
     * Test of averageToLetter method, of class Student.
     */
```

Methods annotated with **@AfterAll** are run **after** all tests have been completed.

Used to tear down anything needed to be closed properly before testing is complete. For example, closing database connections, removing test files, etc.

@AfterAll is analogous to @AfterClass in JUnit 4

```
 */
public class StudentTest {

    public StudentTest() {
    }

    @BeforeAll
    public static void setUpClass() {
    }

    @AfterAll
    public static void tearDownClass() {
    }

    @BeforeEach
    public void setUp() {
    }

    @AfterEach
    public void tearDown() {
    }

    /**
     * Test of averageToLetter method, of class Student.
     */
```

Methods annotated with **@BeforeEach** are run once **before** each test.

Used to setup anything that needs to be reset before each and every test.

@BeforeEach is analogous to @Before in JUnit 4

```
 */
public class StudentTest {

    public StudentTest() {
    }

    @BeforeAll
    public static void setUpClass() {
    }

    @AfterAll
    public static void tearDownClass() {
    }

    @BeforeEach
    public void setUp() {
    }

    @AfterEach
    public void tearDown() {
    }

    /**
     * Test of averageToLetter method, of class Student.
     */
```

Methods annotated with **@AfterEach** are run once **after** each test.

Used to teardown anything that needs to be reset after each and every test.

@AfterEach is analogous to @After in JUnit 4

```
 */
public class StudentTest {

    public StudentTest() {
    }

    @BeforeAll
    public static void setUpClass() {
    }

    @AfterAll
    public static void tearDownClass() {
    }

    @BeforeEach
    public void setUp() {
    }

    @AfterEach
    public void tearDown() {
    }

    /**
     * Test of averageToLetter method, of class Student.
     */
```

These methods are optional and can be removed if not needed in your testing or simply left blank like this.

```
 */
public class StudentTest {

    public StudentTest() {
    }

    @BeforeAll
    public static void setUpClass() {
    }

    @AfterAll
    public static void tearDownClass() {
    }

    @BeforeEach
    public void setUp() {
    }

    @AfterEach
    public void tearDown() {
    }

    /**
     * Test of averageToLetter method, of class Student.
     */
```

For now, let's add a System.out.println line to each so we can understand when they are run.

```
 */
public class StudentTest {

    public StudentTest() {
    }

    @BeforeAll
    public static void setUpClass() {
        System.out.println("setUpClass()");
    }

    @AfterAll
    public static void tearDownClass() {
        System.out.println("tearDownClass()");
    }

    @BeforeEach
    public void setUp() {
        System.out.println("setUp()");
    }

    @AfterEach
    public void tearDown() {
        System.out.println("tearDown()");
    }
```

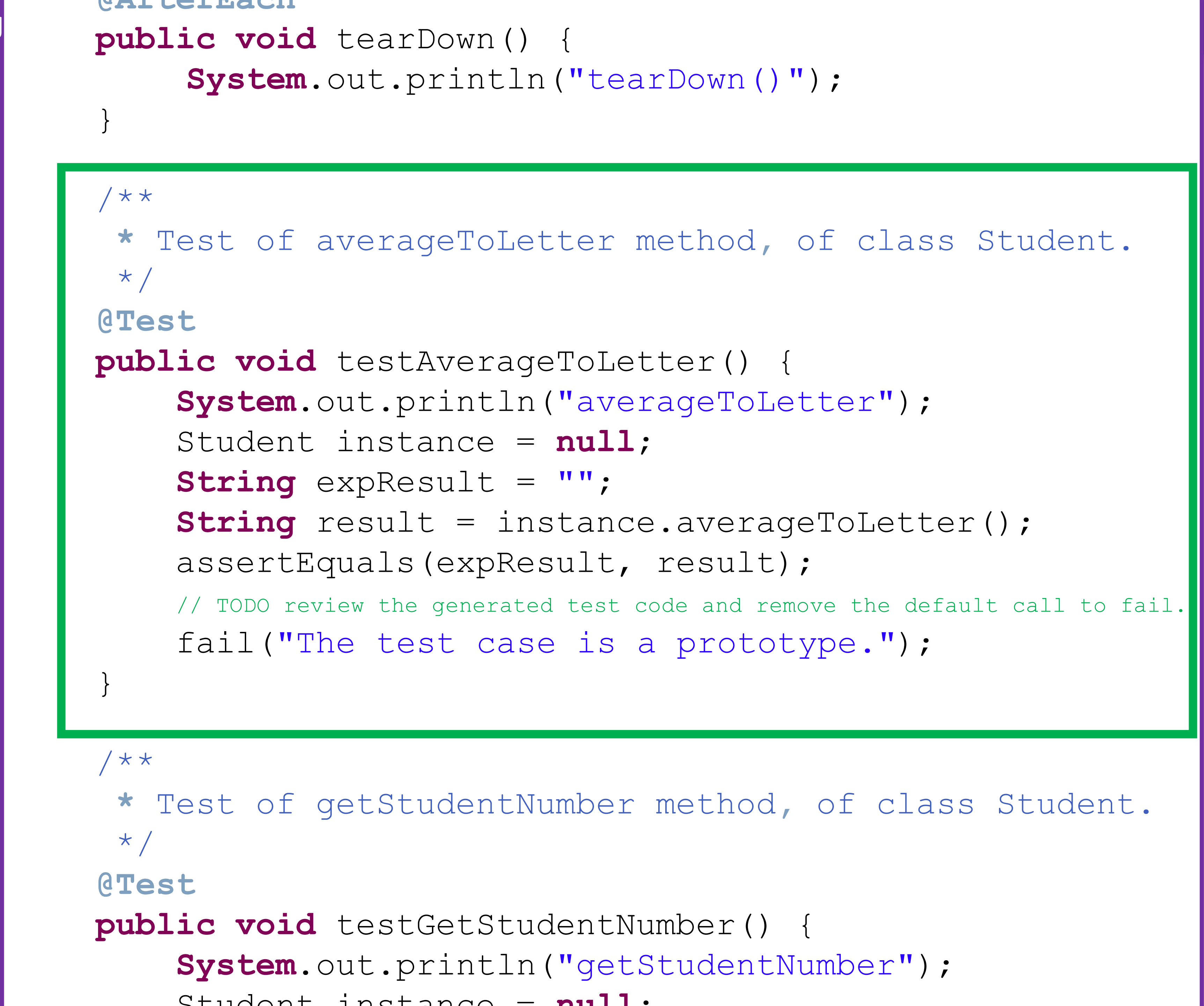
For now, let’s add a System.out.println line to each so we can understand when they are run.

```java
@AfterEach
public void tearDown() {
     System.out.println("tearDown()");
}

/**
 * Test of averageToLetter method, of class Student.
 */
@Test
public void testAverageToLetter() {
    System.out.println("averageToLetter");
    Student instance = null;
    String expResult = "";
    String result = instance.averageToLetter();
    assertEquals(expResult, result);
    // TODO review the generated test code and remove the default call to fail.
    fail("The test case is a prototype.");
}
/**
 * Test of getStudentNumber method, of class Student.
 */
@Test
public void testGetStudentNumber() {
    System.out.println("getStudentNumber");
    Student instance = null;
```

Methods with the **@Test** annotation are our individual tests.

One for test for each method in the Student class was generated for us.

These methods can (and should) be documented using JavaDoc comments to explain what the test does.



```
@AfterEach
public void tearDown() {
     System.out.println("tearDown()");
}

/**
 * Test of averageToLetter method, of class Student.
 */
@Test
public void testAverageToLetter() {
    System.out.println("averageToLetter");
    Student instance = null;
    String expResult = "";
    String result = instance.averageToLetter();
    assertEquals(expResult, result);
    // TODO review the generated test code and remove the default call to fail.
    fail("The test case is a prototype.");
}
/**
 * Test of getStudentNumber method, of class Student.
 */
@Test
public void testGetStudentNumber() {
    System.out.println("getStudentNumber");
    Student instance = null;
```

The automatically generated code is just a place holder that needs to be replaced or modified.

For example:
You would setup instance to equal an instance of the Student class (rather than just null) and set expResult to the expected result from calling the averageToLetter() method.

```
    public void testGetStudentNumber() {
        System.out.println("getStudentNumber");
        Student instance = null;
        long expResult = 0L;
        long result = instance.getStudentNumber();
        assertEquals(expResult, result);
        // TODO review the generated test code and remove the default call to fail.
        fail("The test case is a prototype.");
    }

    /**
     * Test of getFirstName method, of class Student.
     */
    @Test
    public void testGetFirstName() {
        System.out.println("getFirstName");
        Student instance = null;
        String expResult = "";
        String result = instance.getFirstName();
        assertEquals(expResult, result);
        // TODO review the generated test code and remove the default call to fail.
        fail("The test case is a prototype.");
    }

    /**
```

Remainder of the file is automatically generated tests. Once for each method in the Student class.

```
        assertArrayEquals(expResult, result);
        // TODO review the generated test code and remove the default call to fail.
        fail("The test case is a prototype.");
    }

    /**
     * Test of fullName method, of class Student.
     */
    @Test
    public void testFullName() {
        System.out.println("fullName");
        Student instance = null;
        String expResult = "";
        String result = instance.fullName();
        assertEquals(expResult, result);
        // TODO review the generated test code and remove the default call to fail.
        fail("The test case is a prototype.");
    }

}
```

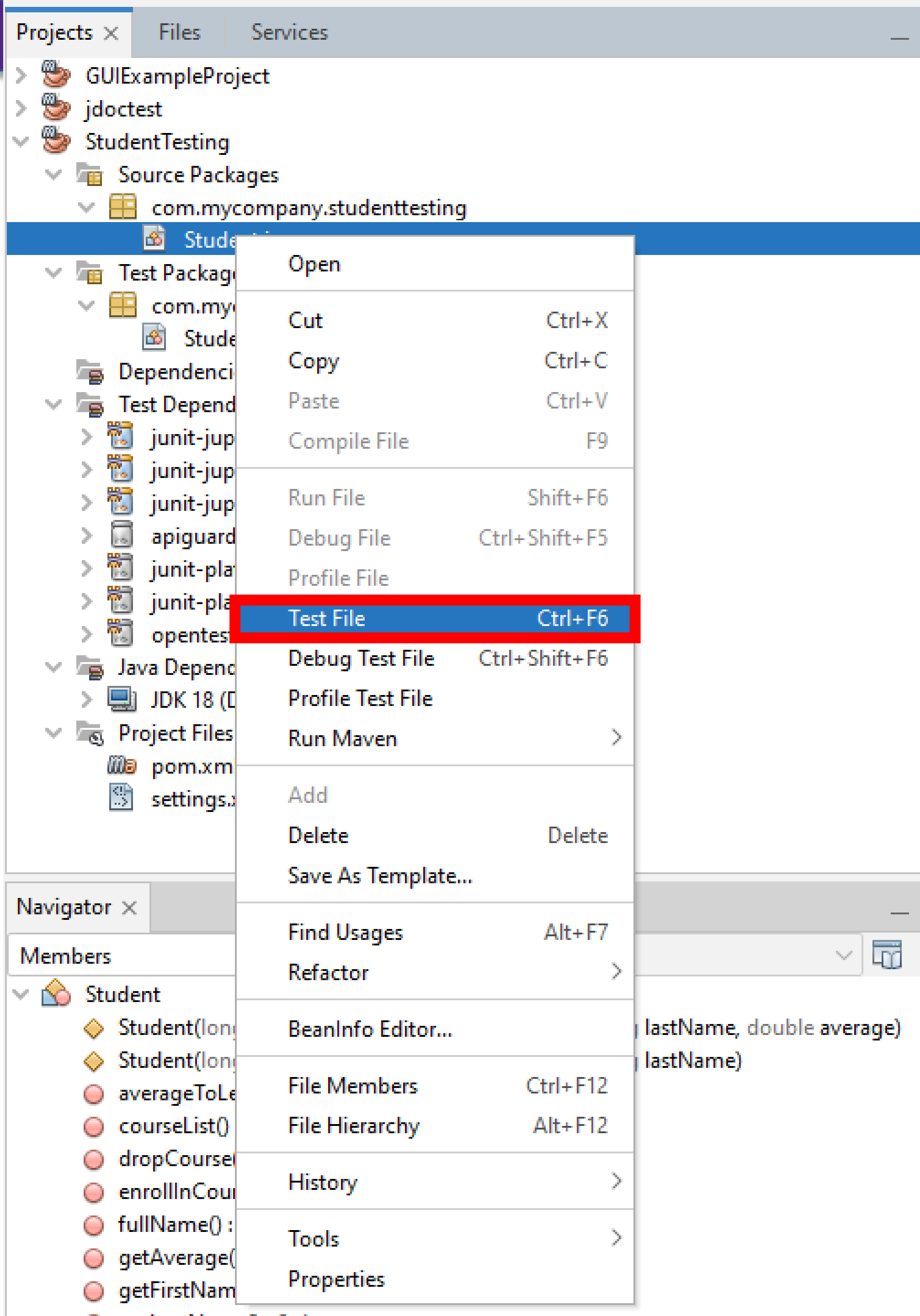
Remainder of the file is automatically generated tests. Once for each method in the Student class.

```java
        String[] result = instance.courseList();
        assertArrayEquals(expResult, result);
        // TODO review the generated test code and remove the default call to fail.
        fail("The test case is a prototype.");
    }

    /**
     * Test of fullName method, of class Student.
     */
    @Test
    public void testFullName() {
        System.out.println("fullName");
        Student instance = null;
        String expResult = "";
        String result = instance.fullName();
        assertEquals(expResult, result);
        // TODO review the generated test code and remove the default call to fail.
        fail("The test case is a prototype.");
    }

}
```

For now, lets leave these tests as they are and try running them.

Running Tests

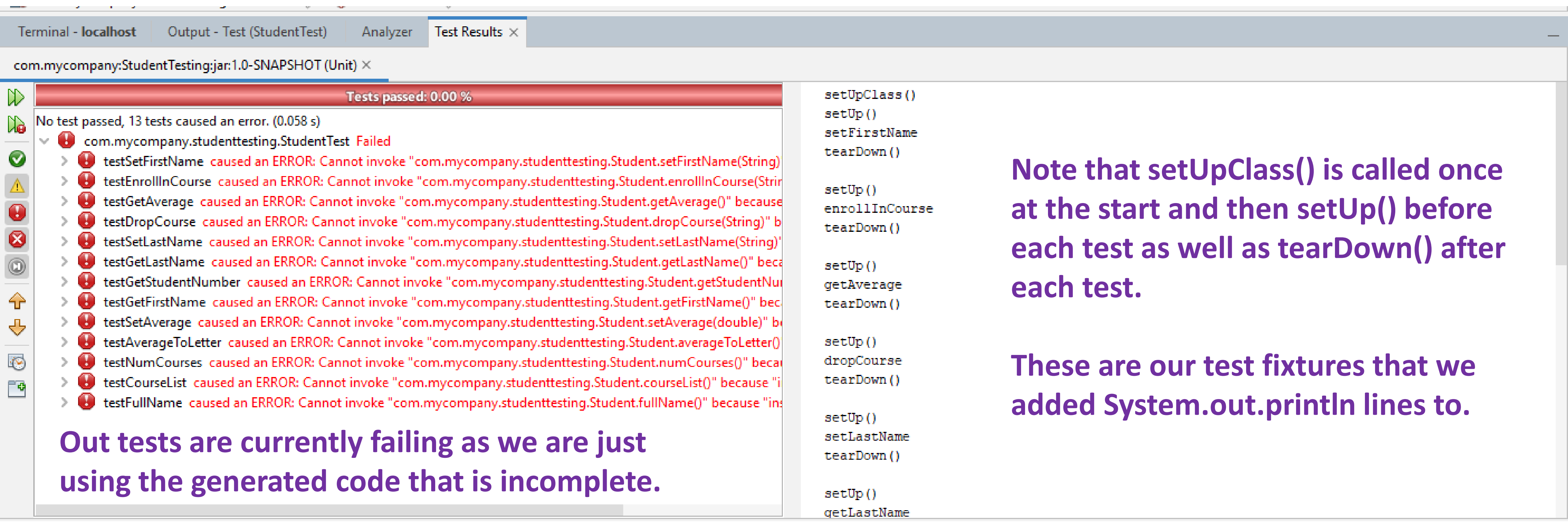
- To run a test in NetBeans, simply right click on the class you want to test (Student.java in this case) and select “Test File”.
- Alternatively, you can right click on the test file (StudentTest.Java) and select “Test File”.
- Both methods will run our tests on the Student calass.



Running Tests

- If everything worked correctly we should get the following output:

Test Results **Standard Output**



Terminal - localhost | Output - Test (StudentTest) | Analyzer | Test Results

com.mycompany:StudentTesting:jar:1.0-SNAPSHOT (Unit)

```
Tests passed: 0.00 %
No test passed, 13 tests caused an error. (0.058 s)
com.mycompany.studenttesting.StudentTest Failed
  testSetFirstName caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.setFirstName(String)
  testEnrollInCourse caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.enrollInCourse(Strin
  testGetAverage caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.getAverage()" because
  testDropCourse caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.dropCourse(String)" b
  testSetLastName caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.setLastName(String)"
  testGetLastName caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.getLastName()" beca
  testGetStudentNumber caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.getStudentNu
  testGetFirstName caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.getFirstName()" bec
  testSetAverage caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.setAverage(double)" b
  testAverageToLetter caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.averageToLetter()
  testNumCourses caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.numCourses()" becau
  testCourseList caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.courseList()" because "i
  testFullName caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.fullName()" because "ins
```

```
setUpClass()
setUp()
setFirstName
tearDown()

setUp()
enrollInCourse
tearDown()

setUp()
getAverage
tearDown()

setUp()
dropCourse
tearDown()

setUp()
setLastName
tearDown()

setUp()
getLastName
```

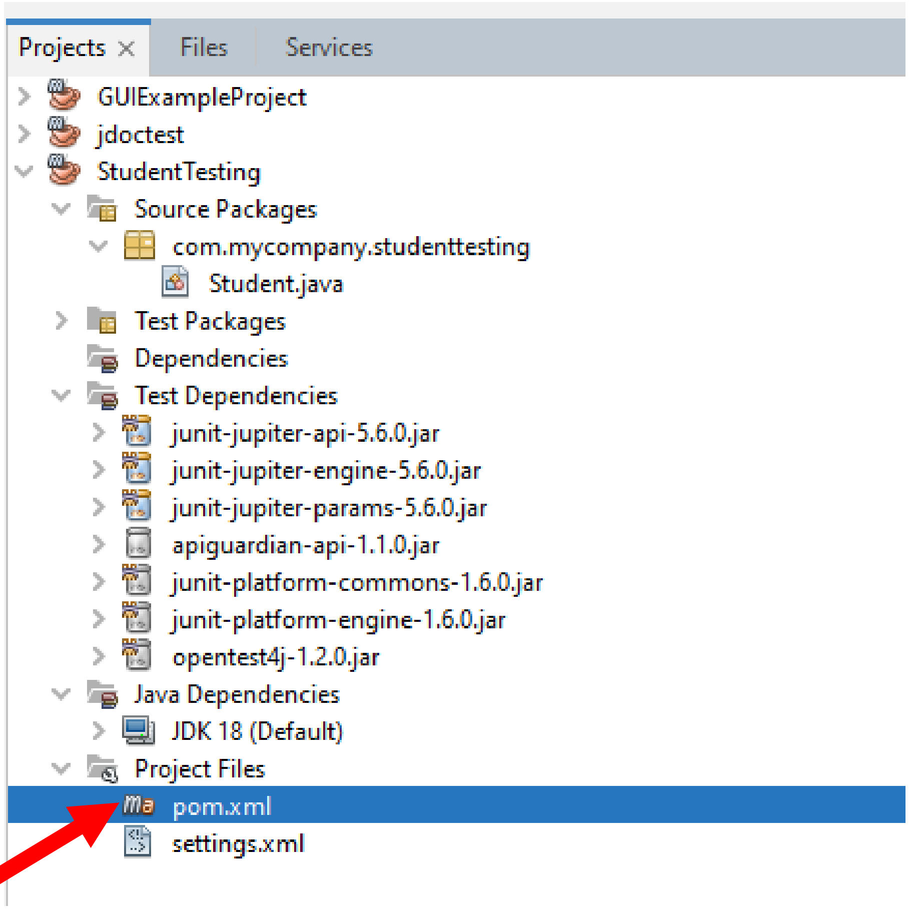
Out tests are currently failing as we are just using the generated code that is incomplete.

Note that setUpClass() is called once at the start and then setUp() before each test as well as tearDown() after each test.

These are our test fixtures that we added System.out.println lines to.

Setting Up Maven to Use JUnit

- If you are not getting output from the fixture methods (the setUp() and tearDown lines in the output), the issue is likely that Surefire's POJO tests rather than JUnit are being used to run the tests.
- This is not a big deal for simple tests, but causes issues when we want to use features such as @BeforeAll and @BeforeEach.
- To correct this we need to edit the pom.xml file in our NetBeans project.
- Open pom.xml in your NetBeans IDE.



```
Projects ×   Files   Services
> GUIExampleProject
> jdoctest
v StudentTesting
  v Source Packages
    v com.mycompany.studenttesting
        Student.java
  > Test Packages
    Dependencies
  v Test Dependencies
    > junit-jupiter-api-5.6.0.jar
    > junit-jupiter-engine-5.6.0.jar
    > junit-jupiter-params-5.6.0.jar
    > apiguardian-api-1.1.0.jar
    > junit-platform-commons-1.6.0.jar
    > junit-platform-engine-1.6.0.jar
    > opentest4j-1.2.0.jar
  v Java Dependencies
    > JDK 18 (Default)
  v Project Files
      pom.xml
      settings.xml
```

Setting Up Maven to Use JUnit

- Add the following to the end of pom.xml right before **</project>:**

```
<build>
  <plugins>
      <plugin>
         <artifactId>maven-surefire-plugin</artifactId>
         <version>2.19.1</version>
         <dependencies>
            <dependency>
               <groupId>org.junit.platform</groupId>
                <artifactId>junit-platform-surefire-provider</artifactId>
                <version>1.1.0</version>
            </dependency>
         </dependencies>
      </plugin>
   </plugins>
</build>
```

Lets Write the testGetStudentNumber Test

Automatically Generated Code

```
/**
  * Test of getStudentNumber method, of class Student.
  */
 @Test
 public void testGetStudentNumber() {
     System.out.println("getStudentNumber");
     Student instance = null;
     long expResult = 0L;
     long result = instance.getStudentNumber();
     assertEquals(expResult, result);
     // TODO review the generated test code and remove the default call to fail
     fail("The test case is a prototype.");
 }
```

Lets Write the testGetStudentNumber Test

Automatically Generated Code

```
/**
 * Test of getStudentNumber method, of class Student.
 */
@Test
public void testGetStudentNumber() {
    System.out.println("getStudentNumber");
    Student instance = null;
    long expResult = 0L;
    long result = instance.getStudentNumber();
    assertEquals(expResult, result);
    // TODO review the generated test code and remove the default call to fail
    fail("The test case is a prototype.");
}
```

Prints out "getStudentNumber".

Lets Write the testGetStudentNumber Test

Automatically Generated Code

```
/**
  * Test of getStudentNumber method, of class Student.
  */
 @Test
 public void testGetStudentNumber() {
     System.out.println("getStudentNumber");
     Student instance = null;
     long expResult = 0L;
     long result = instance.getStudentNumber();
     assertEquals(expResult, result);
     // TODO review the generated test code and remove the default call to fail
     fail("The test case is a prototype.");
 }
```

Sets up an instance of Student to test. Right now this is just being set to null, we need to update this line to create a student object.

Lets Write the testGetStudentNumber Test

```
/**
  * Test of getStudentNumber method, of class Student.
  */
 @Test
 public void testGetStudentNumber() {
     System.out.println("getStudentNumber");
     Student instance = new Student(1234567, "Jhon", "Doe");
     long expResult = 0L;
     long result = instance.getStudentNumber();
     assertEquals(expResult, result);
     // TODO review the generated test code and remove the default call to fail
     fail("The test case is a prototype.");
 }
```

Update the line to create a new instance of student.

Lets Write the testGetStudentNumber Test

```
/**
  * Test of getStudentNumber method, of class Student.
  */
 @Test
 public void testGetStudentNumber() {
     System.out.println("getStudentNumber");
     Student instance = new Student(1234567, "Jhon", "Doe");
     long expResult = 0L;
                                     tNumber();

                                     t code and remove the default call to fail
                                     pe.");
 }
```

The next line is the result we expect to get from getStudentNumber for this student.

This needs to be updated for the student we just created.

Lets Write the testGetStudentNumber Test

```
/**
  * Test of getStudentNumber method, of class Student.
  */
 @Test
 public void testGetStudentNumber() {
     System.out.println("getStudentNumber");
     Student instance = new Student(1234567, "Jhon", "Doe");
     long expResult = 1234567;
```

In this case we are expecting a student ID of 1234567.

So update expResult to be equal to 1234567.

```
                                                        ;

                                                        d remove the default call to fail
     fail("The test case is a prototype.");
 }
```

Lets Write the testGetStudentNumber Test

```
/**
  * Test of getStudentNumber method, of class Student.
  */
 @Test
 public void testGetStudentNumber() {
     System.out.println("getStudentNumber");
     Student instance = new Student(1234567, "Jhon", "Doe");
     long expResult = 1234567;
     long result = instance.getStudentNumber();
                                              d remove the default call to fail
     fail("The test case is a prototype.");
 }
```

This line calls the method we are testing and stores the value returned in result.

Lets Write the testGetStudentNumber Test

```
/**
  * Test of getStudentNumber method, of class Student.
  */
 @Test
 public void testGetStudentNumber() {
     System.out.println("getStudentNumber");
     Student instance = new Student(1234567, "Jhon", "Doe");
     long expResult = 1234567;
     long result = instance.getStudentNumber();
     assertEquals(expResult, result);

 }
```

This method checks that the given values are equal.

If they are not equal the test is marked as failed and execution of the test stops.

If they are equal execution of the test continues. If a test makes it to the end of the method without failing an assert or the fail method being called, the test is considered to be passed.

JUnit Assertions

Many more at:
https://junit.org/junit5/docs/current/api/org.junit.jupiter.api/org/junit/jupiter/api/Assertions.html

Method Name	Input	Description
assertEquals	Two values, expected and actual.	*Assert* that expected and actual are equal.
assertNotEquals	Two values, unexpected and actual.	*Assert* that unexpected and actual are not equal.
assertArrayEquals	Two arrays, expected and actual.	*Assert* that expected and actual arrays are equal. If both are null, they are considered equal.
assertTrue	A Boolean expression.	*Assert* that the supplied condition is true.
assertFalse	A Boolean expression.	*Assert* that the supplied condition is false.
assertThrows	An expected exception type and executable (e.g. calling a method).	*Assert* that execution of the supplied executable throws an exception of the expected type and return the exception. If no exception is thrown, or if an exception of a different type is thrown, this method will fail.
assertTimeout	A duration (time limit) and an executable (e.g. a method call).	*Assert* that execution of the supplied executable completes before the given timeout is exceeded.
assertNull	A value.	*Assert* that the value is null.
assertNotNull	A value.	*Assert* that the value is null.
fail	An optional message.	*Fail* the test with the given failure message if provided.

Lets Write the testGetStudentNumber Test

```
/**
  * Test of getStudentNumber method, of class Student.
  */
 @Test
 public void testGetStudentNumber() {
```

This caused the test to fail with the message "The test case is a prototype".

This was added automatically to the test to remind you that this is just an automatically generated placed holder and we need to add in our own code.

Since we just did that, remove these lines.

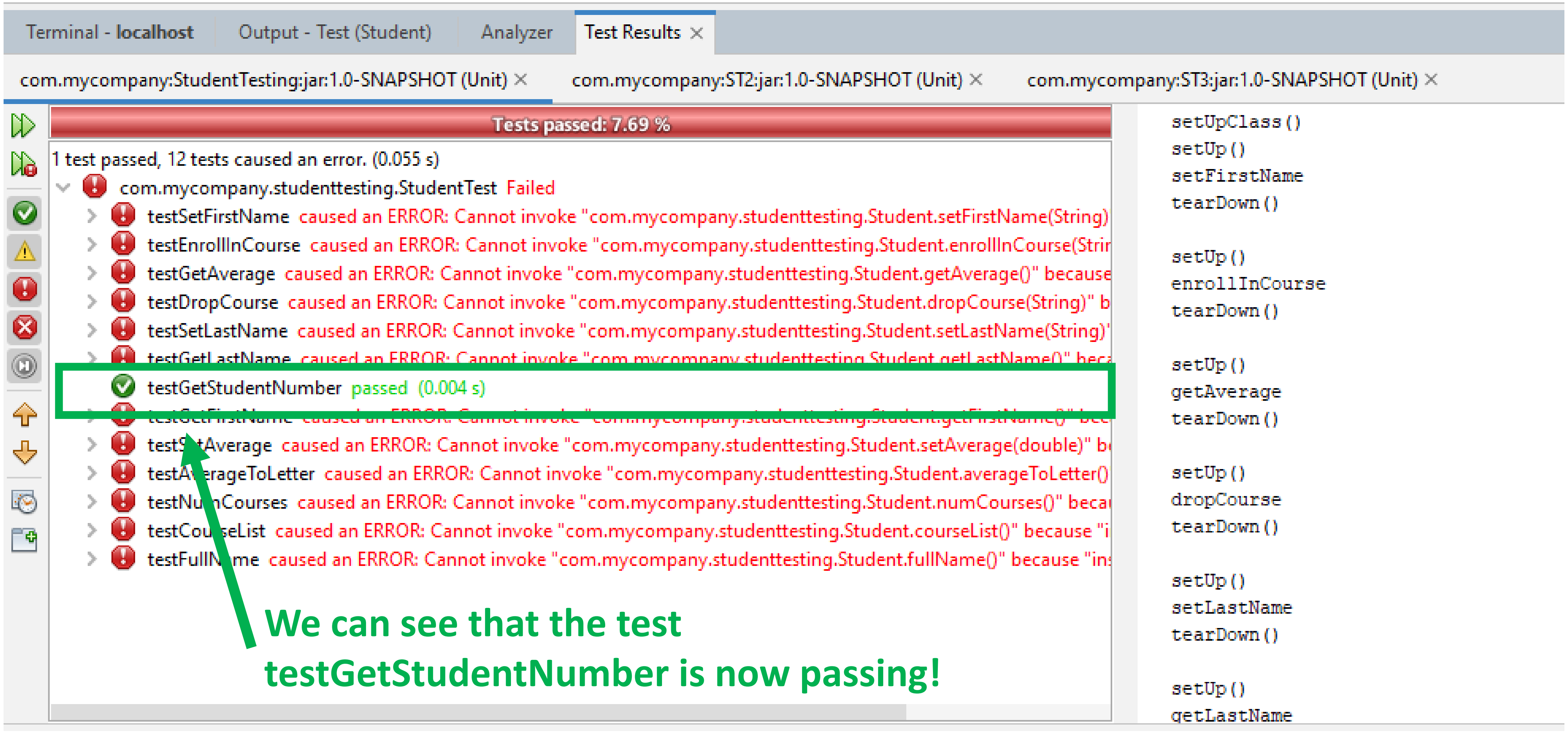
```
     assertEquals(expResult, result);
     // TODO review the generated test code and remove the default call to fail
     fail("The test case is a prototype.");
 }
```

Lets Write the testGetStudentNumber Test

```
/**
  * Test of getStudentNumber method, of class Student.
  */
 @Test
 public void testGetStudentNumber() {
     System.out.println("getStudentNumber");
     Student instance = new Student(1234567, "Jhon", "Doe");
     long expResult = 1234567;
     long result = instance.getStudentNumber();
     assertEquals(expResult, result);
 }
```

Let’s now try rerunning the tests.
Right click on Student.java and select “Test File”.

Lets Write the testGetStudentNumber Test



Terminal - **localhost** | Output - Test (Student) | Analyzer | Test Results ×

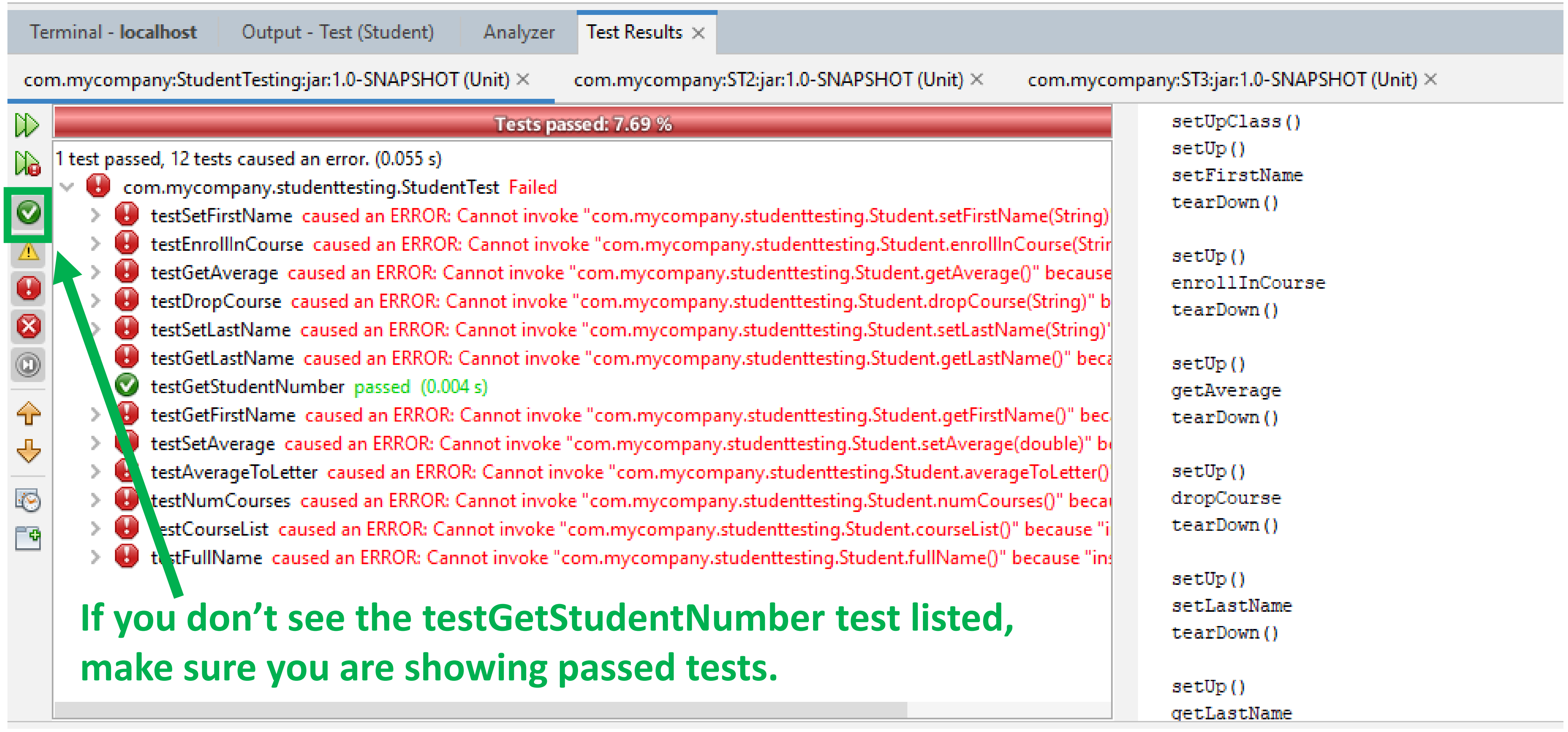
com.mycompany:StudentTesting:jar:1.0-SNAPSHOT (Unit) × | com.mycompany:ST2:jar:1.0-SNAPSHOT (Unit) × | com.mycompany:ST3:jar:1.0-SNAPSHOT (Unit) ×

```
Tests passed: 7.69 %
1 test passed, 12 tests caused an error. (0.055 s)
com.mycompany.studenttesting.StudentTest Failed
  testSetFirstName caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.setFirstName(String)
  testEnrollInCourse caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.enrollInCourse(Strin
  testGetAverage caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.getAverage()" because
  testDropCourse caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.dropCourse(String)" b
  testSetLastName caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.setLastName(String)"
  testGetLastName caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.getLastName()" beca
  testGetStudentNumber passed (0.004 s)
  testGetFirstName caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.getFirstName()" bec
  testSetAverage caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.setAverage(double)" b
  testAverageToLetter caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.averageToLetter()
  testNumCourses caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.numCourses()" becau
  testCourseList caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.courseList()" because "i
  testFullName caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.fullName()" because "ins
```

```
setUpClass()
setUp()
setFirstName
tearDown()

setUp()
enrollInCourse
tearDown()

setUp()
getAverage
tearDown()

setUp()
dropCourse
tearDown()

setUp()
setLastName
tearDown()

setUp()
getLastName
```

We can see that the test testGetStudentNumber is now passing!

Lets Write the testGetStudentNumber Test


Terminal - localhost
Output - Test (Student)
Analyzer
Test Results
com.mycompany:StudentTesting:jar:1.0-SNAPSHOT (Unit)
com.mycompany:ST2:jar:1.0-SNAPSHOT (Unit)
com.mycompany:ST3:jar:1.0-SNAPSHOT (Unit)
Tests passed: 7.69 %

```
1 test passed, 12 tests caused an error. (0.055 s)
com.mycompany.studenttesting.StudentTest Failed
  testSetFirstName caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.setFirstName(String)
  testEnrollInCourse caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.enrollInCourse(Strir
  testGetAverage caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.getAverage()" because
  testDropCourse caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.dropCourse(String)" b
  testSetLastName caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.setLastName(String)'
  testGetLastName caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.getLastName()" beca
  testGetStudentNumber passed (0.004 s)
  testGetFirstName caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.getFirstName()" bec
  testSetAverage caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.setAverage(double)" b
  testAverageToLetter caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.averageToLetter()
  testNumCourses caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.numCourses()" becau
  testCourseList caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.courseList()" because "i
  testFullName caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.fullName()" because "ins
```

```
setUpClass()
setUp()
setFirstName
tearDown()

setUp()
enrollInCourse
tearDown()

setUp()
getAverage
tearDown()

setUp()
dropCourse
tearDown()

setUp()
setLastName
tearDown()

setUp()
getLastName
```

If you don’t see the testGetStudentNumber test listed, make sure you are showing passed tests.

Now Lets Fix the testSetFirstName Test

Automatically Generated Code

```
@Test
public void testSetFirstName() {
    System.out.println("setFirstName");
    String firstName = "";
    Student instance = null;
    instance.setFirstName(firstName);
    // TODO review the generated test code and remove the default call to fai
    fail("The test case is a prototype.");
}
```

Now Lets Fix the testSetFirstName Test

Automatically Generated Code

```
@Test
public void testSetFirstName() {
    System.out.println("setFir
    String firstName = "";
    Student instance = null;
    instance.setFirstName(firstName);
    // TODO review the generated test code and remove the default call to fai
    fail("The test case is a prototype.");
}
```

Start by setting a first name to test.
For now, any valid value is fine.

Now Lets Fix the testSetFirstName Test

Automatically Generated Code

```
@Test
public void testSetFirstName() {
    System.out.println("setFir
    String firstName = "Dan";
    Student instance = null;
    instance.setFirstName(firstName);
    // TODO review the generated test code and remove the default call to fai
    fail("The test case is a prototype.");
}
```

Start by setting a first name to test. For now, any valid value is fine.

Now Lets Fix the testSetFirstName Test

Automatically Generated Code

```
@Test
public void testSetFirstName() {
    System.out.println("setFirstName")
    String firstName = "Dan";
    Student instance = null;
    instance.setFirstName(firstName);
    // TODO review the generated test code and remove the default call to fai
    fail("The test case is a prototype.");
}
```

Now we need a make a valid Student object.

Make sure to use a different first name as we want to test that it was updated correctly.

Now Lets Fix the testSetFirstName Test

```
@Test
public void testSetFirstName() {
    System.out.println("setFirstName");
    String firstName = "Dan";
    Student instance = new Student(1234567, "Jhon", "Doe");
    instance.setFirstName(firstName);
    // TODO review the generated test code and remove the default call to fai
    fail("The test case is a prototype.");
}
```

Now we need a make a valid Student object.

Make sure to use a different first name as we want to test that it was updated correctly.

Now Lets Fix the testSetFirstName Test

```
@Test
public void testSetFirstName() {
    System.out.println("setFirstName");
    String firstName = "Dan";
    Student instance = new Student(1234567, "Jhon", "Doe");
    instance.setFirstName(firstName);
                                      de and remove the default call to fai
                                      ;
}
```

This line calls the method we are testing with the value we set for firstName.

Now Lets Fix the testSetFirstName Test

```
@Test
public void testSetFirstName() {
    System.out.println("setFirstName");
    String firstName = "Dan";
    Student instance = new Student(1234567, "Jhon", "Doe");
    instance.setFirstName(firstName);
    // TODO review the generated test code and remove the default call to fai
    fail("The test case is a prototype.");
}
```

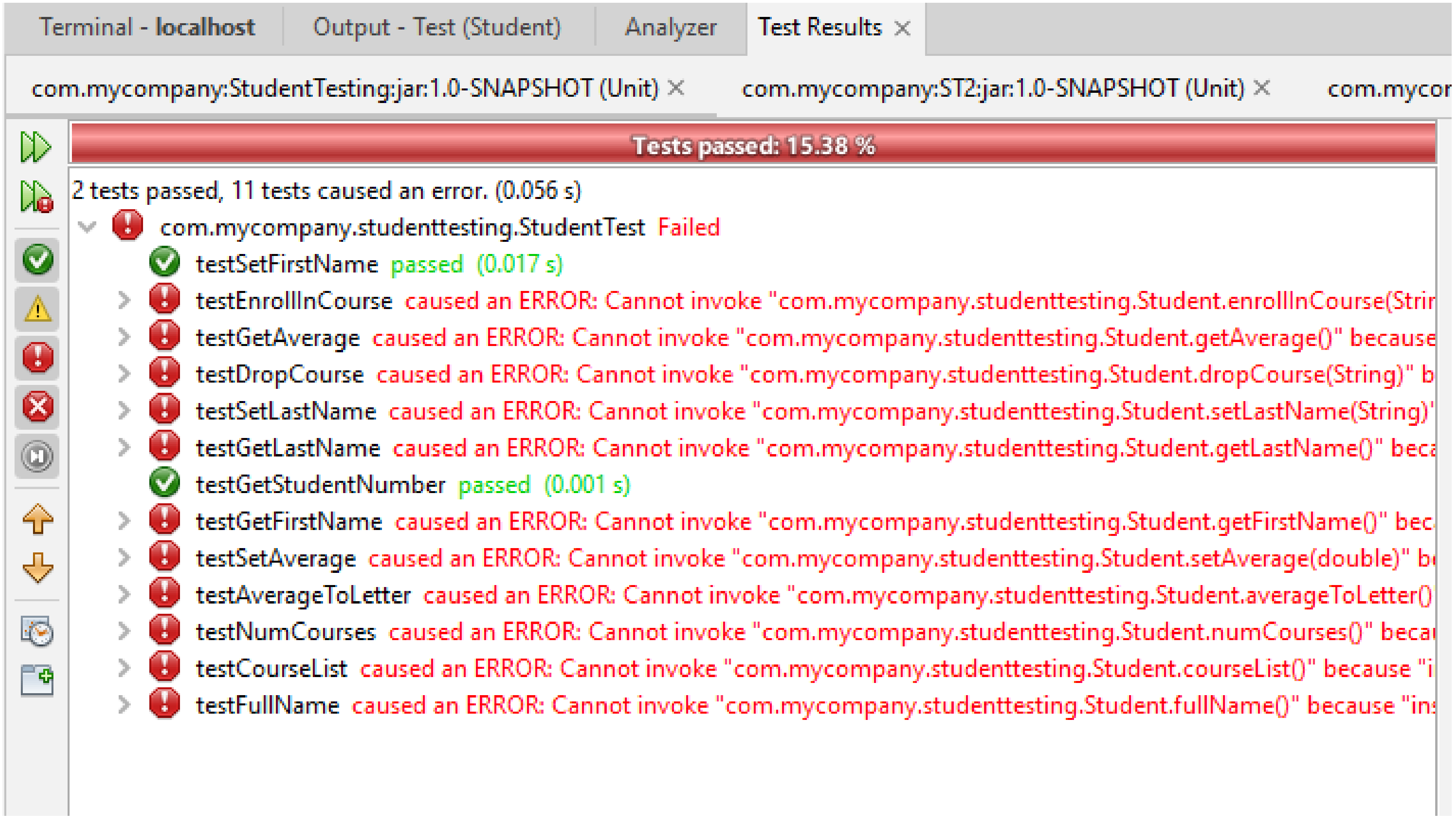
Now need to replace these lines with an assertation that checks if the student's first name was set correctly.

Now Lets Fix the testSetFirstName Test

```
@Test
public void testSetFirstName() {
    System.out.println("setFirstName");
    String firstName = "Dan";
    Student instance = new Student(1234567, "Jhon", "Doe");
    instance.setFirstName(firstName);
    String result = instance.getFirstName();
    assertEquals(firstName, result);
}
```

We now call getFirstName to return the updated first name from Student and use assertEquals to check that it is the result we expected.

Now Lets Fix the testSetFirstName Test



```
com.mycompany:StudentTesting:jar:1.0-SNAPSHOT (Unit)   com.mycompany:ST2:jar:1.0-SNAPSHOT (Unit)   com.mycor

Tests passed: 15.38 %
2 tests passed, 11 tests caused an error. (0.056 s)
com.mycompany.studenttesting.StudentTest Failed
    testSetFirstName passed (0.017 s)
    testEnrollInCourse caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.enrollInCourse(Strir
    testGetAverage caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.getAverage()" because
    testDropCourse caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.dropCourse(String)" b
    testSetLastName caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.setLastName(String)"
    testGetLastName caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.getLastName()" beca
    testGetStudentNumber passed (0.001 s)
    testGetFirstName caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.getFirstName()" bec
    testSetAverage caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.setAverage(double)" b
    testAverageToLetter caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.averageToLetter()
    testNumCourses caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.numCourses()" becau
    testCourseList caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.courseList()" because "i
    testFullName caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.fullName()" because "in
```

A More Complex Example: testAverageToLetter

Automatically Generated Code

```
@Test
public void testAverageToLetter() {
        System.out.println("averageToLetter");
        Student instance = null;
        String expResult = "";
        String result = instance.averageToLetter();
        assertEquals(expResult, result);
        // TODO review the generated test code and remove the default call to
        fail("The test case is a prototype.");
}
```

The averageToLetter method in Student converts their average into a letter grade.

As there are multiple inputs to test, we should break this into multiple tests.

A good guideline is that each test should only fail for one reason. Having only one assert per test is a good way to enforce this.

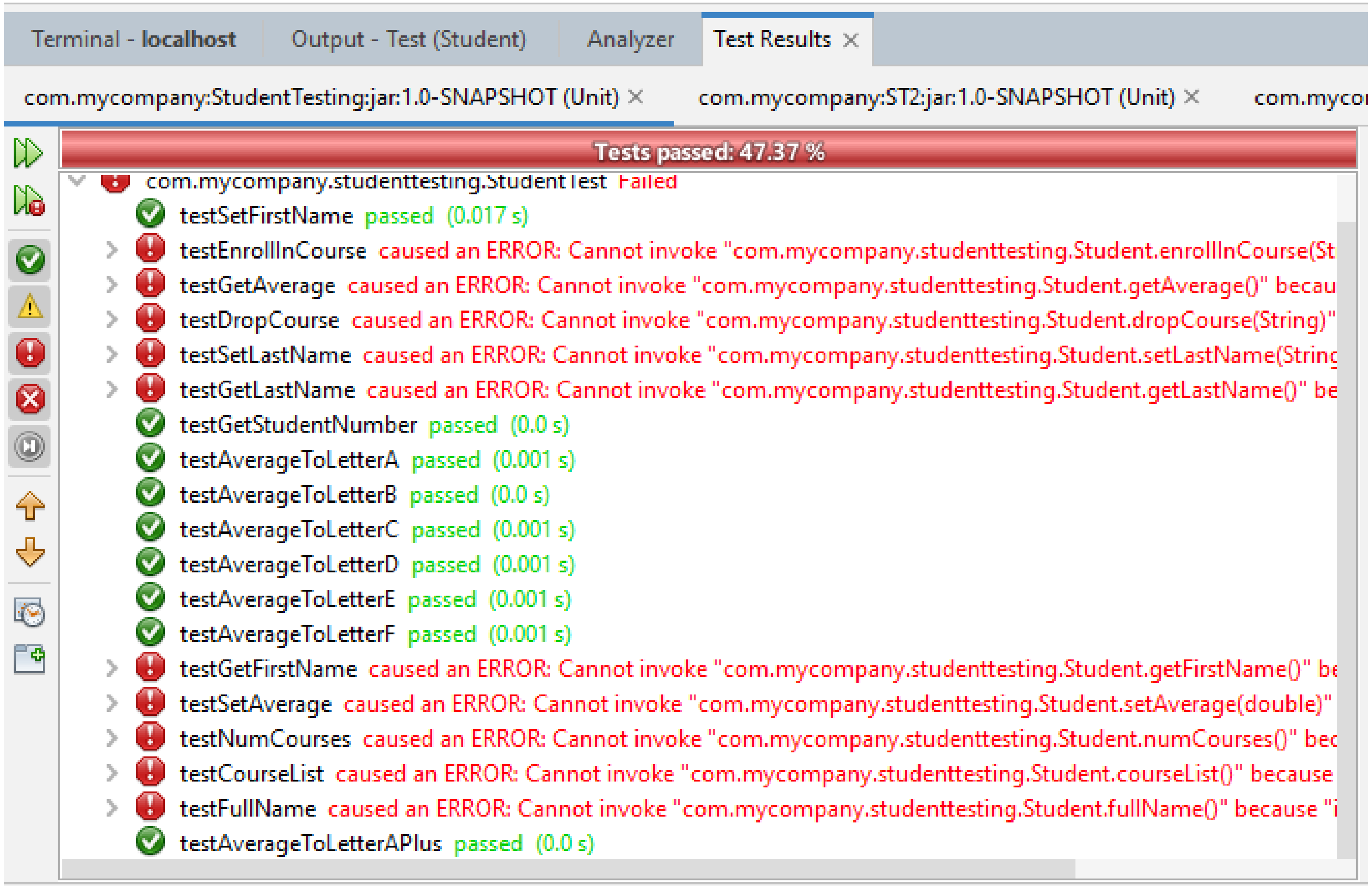
A More Complex Example: testAverageToLetter

```
    @Test
    public void testAverageToLetterAPlus() {
        System.out.println("averageToLetterAPlus");
        Student instance = new Student(1234, "Joe", "Bloggs", 90);
        String expResult = "A+";
        String result = instance.averageToLetter();
        assertEquals(expResult, result);
    }

    @Test
    public void testAverageToLetterA() {
        System.out.println("averageToLetterAPlus");
        Student instance = new Student(1234, "Joe", "Bloggs", 80);
        String expResult = "A";
        String result = instance.averageToLetter();
        assertEquals(expResult, result);
    }

    @Test
    public void testAverageToLetterB() {
        System.out.println("averageToLetterAPlus");
```

```
@Test
public void testAverageToLetterB() {
    System.out.println("averageToLetterAPlus");
    Student instance = new Student(1234, "Joe", "Bloggs", 70);
    String expResult = "B";
    String result = instance.averageToLetter();
    assertEquals(expResult, result);
}

@Test
public void testAverageToLetterC() {
    System.out.println("averageToLetterAPlus");
    Student instance = new Student(1234, "Joe", "Bloggs", 60);
    String expResult = "C";
    String result = instance.averageToLetter();
    assertEquals(expResult, result);
}

@Test
public void testAverageToLetterD() {
    System.out.println("averageToLetterAPlus");
    Student instance = new Student(1234, "Joe", "Bloggs", 50);
    String expResult = "D";
    String result = instance.averageToLetter();
    assertEquals(expResult, result);
```

```
        System.out.println("averageToLetterAPlus");
        Student instance = new Student(1234, "Joe", "Bloggs", 50);
        String expResult = "D";
        String result = instance.averageToLetter();
        assertEquals(expResult, result);
    }

    @Test
    public void testAverageToLetterE() {
        System.out.println("averageToLetterAPlus");
        Student instance = new Student(1234, "Joe", "Bloggs", 40);
        String expResult = "E";
        String result = instance.averageToLetter();
        assertEquals(expResult, result);
    }

    @Test
    public void testAverageToLetterF() {
        System.out.println("averageToLetterAPlus");
        Student instance = new Student(1234, "Joe", "Bloggs", 39.99);
        String expResult = "F";
        String result = instance.averageToLetter();
        assertEquals(expResult, result);
    }
```

A More Complex Example: testAverageToLetter


Terminal - localhost
Output - Test (Student)
Analyzer
Test Results
com.mycompany:StudentTesting:jar:1.0-SNAPSHOT (Unit)
com.mycompany:ST2:jar:1.0-SNAPSHOT (Unit)
com.myco

```
Tests passed: 47.37 %
com.mycompany.studenttesting.StudentTest Failed
  testSetFirstName passed (0.017 s)
  testEnrollInCourse caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.enrollInCourse(St
  testGetAverage caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.getAverage()" becau
  testDropCourse caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.dropCourse(String)"
  testSetLastName caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.setLastName(String
  testGetLastName caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.getLastName()" be
  testGetStudentNumber passed (0.0 s)
  testAverageToLetterA passed (0.001 s)
  testAverageToLetterB passed (0.0 s)
  testAverageToLetterC passed (0.001 s)
  testAverageToLetterD passed (0.001 s)
  testAverageToLetterE passed (0.001 s)
  testAverageToLetterF passed (0.001 s)
  testGetFirstName caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.getFirstName()" be
  testSetAverage caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.setAverage(double)"
  testNumCourses caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.numCourses()" bec
  testCourseList caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.courseList()" because
  testFullName caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.fullName()" because "i
  testAverageToLetterAPlus passed (0.0 s)
```

Testing For an Exception: testSetAverage

Automatically Generated Code

```
@Test
public void testSetAverage() {
    System.out.println("setAverage");
    double average = 0.0;
    Student instance = null;
    instance.setAverage(average);
    // TODO review the generated test code and remove the default call to fai
    fail("The test case is a prototype.");
}
```

In this case the setAverage method can throw an IllegalArgumentException exception if the given average is over 100 or under 0.

We want to create tests that check that violating either bound will cause this exception, as well as test valid input (at least 3 tests in total).

```
    @Test
    public void testSetAverageValid() {
        System.out.println("setAverage");
        double average = 82.95;
        Student instance = new Student(1234, "Joe", "Bloggs", 32.12);
        instance.setAverage(average);
        double result = instance.getAverage();
        assertEquals(average, result);
    }

    @Test
    public void testSetAverageLowerBound() {
        System.out.println("setAverage");
        double average = Math.nextDown(0);
        System.out.println("Testing average of " + average);
        Student instance = new Student(1234, "Joe", "Bloggs", 32.12);
        assertThrows(IllegalArgumentException.class, ()->{instance.setAverage(average);});
    }

    @Test
    public void testSetAverageUpperBound() {
        System.out.println("setAverage");
        double average = Math.nextUp(100);
        System.out.println("Testing average of " + average);
        Student instance = new Student(1234, "Joe", "Bloggs", 32.12);
        assertThrows(IllegalArgumentException.class, ()->{instance.setAverage(average);});
    }
```

```java
@Test
public void testSetAverageValid() {
    System.out.println("setAverage");
    double average = 82.95;
    Student instance = new Student(1234, "Joe", "Bloggs", 32.12);
    instance.setAverage(average);
    double result = instance.getAverage();
    assertEquals(average, result);
}
```

Check valid input (in this case setting an average of 82.95.

Could also do tests that check the boundaries of valid inputs (e.g. 0 and 100).

```java
    double average = Math.nextDown(0);
    System.out.println("Testing average of " + average);
    Student instance = new Student(1234, "Joe", "Bloggs", 32.12);
    assertThrows(IllegalArgumentException.class, ()->{instance.setAverage(average);});
}

@Test
public void testSetAverageUpperBound() {
    System.out.println("setAverage");
    double average = Math.nextUp(100);
    System.out.println("Testing average of " + average);
    Student instance = new Student(1234, "Joe", "Bloggs", 32.12);
    assertThrows(IllegalArgumentException.class, ()->{instance.setAverage(average);});
}
```

```
    @Test
    public void testSetAverageValid() {
        System.out.println("setAverage");
        double average = 82.95;
        Student instance = new Student(1234, "Joe", "Bloggs", 32.12);
        instance.setAverage(average);
        double result = instance.getAverage();
        assertEquals(average, result);
```

Checking that a value outside of the lower bounds causes a IllegalArgumentException.

```
    @Test
    public void testSetAverageLowerBound() {
        System.out.println("setAverage");
        double average = Math.nextDown(0);
        System.out.println("Testing average of " + average);
        Student instance = new Student(1234, "Joe", "Bloggs", 32.12);
        assertThrows(IllegalArgumentException.class, ()->{instance.setAverage(average);});
    }

    @Test
    public void testSetAverageUpperBound() {
        System.out.println("setAverage");
        double average = Math.nextUp(100);
        System.out.println("Testing average of " + average);
        Student instance = new Student(1234, "Joe", "Bloggs", 32.12);
        assertThrows(IllegalArgumentException.class, ()->{instance.setAverage(average);});
    }
    @Test
    public void testSetAverageValid() {
        System.out.println("setAverage");
        double average = 82.95;
        Student instance = new Student(1234, "Joe", "Bloggs", 32.12);
        instance.setAverage(average);
        double result = instance.getAverage();
        assertEquals(average, result);
    }
```

What value is just bellow zero (0) for a floating point?

Is it -1? -0.1, -0.01, -0.001….?

```
    @Test
    public vo
        System.out.println("setAverage");
        double average = Math.nextDown(0);
        System.out.println("Testing average of " + average);
        Student instance = new Student(1234, "Joe", "Bloggs", 32.12);
        assertThrows(IllegalArgumentException.class, ()->{instance.setAverage(average);});
    }

    @Test
    public void testSetAverageUpperBound() {
        System.out.println("setAverage");
        double average = Math.nextUp(100);
        System.out.println("Testing average of " + average);
        Student instance = new Student(1234, "Joe", "Bloggs", 32.12);
        assertThrows(IllegalArgumentException.class, ()->{instance.setAverage(average);});
    }
```

```java
    @Test
    public void testSetAverageValid() {
        System.out.println("setAverage");
        double average = 82.95;
        Student instance = new Student(1234, "Joe", "Bloggs", 32.12);
        instance.setAverage(average);
        double result = instance.getAverage();
        assertEquals(average, result);
    }

    @Test
    public vo
        System.out.println("setAverage");
        double average = Math.nextDown(0);
        System.out.println("Testing average of " + average);
        Student instance = new Student(1234, "Joe", "Bloggs", 32.12);
        assertThrows(IllegalArgumentException.class, ()->{instance.setAverage(average);});
    }

    @Test
    public void testSetAverageUpperBound() {
        System.out.println("setAverage");
        double average = Math.nextUp(100);
        System.out.println("Testing average of " + average);
        Student instance = new Student(1234, "Joe", "Bloggs", 32.12);
        assertThrows(IllegalArgumentException.class, ()->{instance.setAverage(average);});
    }
```

We can use Math.nextDown(0) to find the next floating point number after zero.

In this case it was -1.401298464324817E-45

```
@Test
public void testSetAverageValid() {
    System.out.println("setAverage");
    double average = 82.95;
```

Checks that our call to setAverage throws an IllegalArgumentException.

Need to wrap the call in:

```
() -> {
    ...your code to test here...
}
```

This is needed as assertThrows takes a executable block or lambda expression.

```
    Student instance = new Student(1234, "Joe", "Bloggs", 32.12);
    assertThrows(IllegalArgumentException.class, ()->{instance.setAverage(average);});
}

@Test
public void testSetAverageUpperBound() {
    System.out.println("setAverage");
    double average = Math.nextUp(100);
    System.out.println("Testing average of " + average);
    Student instance = new Student(1234, "Joe", "Bloggs", 32.12);
    assertThrows(IllegalArgumentException.class, ()->{instance.setAverage(average);});
}
```

```
    @Test
    public void testSetAverageValid() {
        System.out.println("setAverage");
        double average = 82.95;
        Student instance = new Student(1234, "Joe", "Bloggs", 32.12);
        instance.setAverage(average);
        double result = instance.getAverage();
        assertEquals(average, result);
    }

    @Test
    public void testSetAverageLowerBound() {
        System.out.println("setAverage");
        double average = Math.nextDown(0);
```

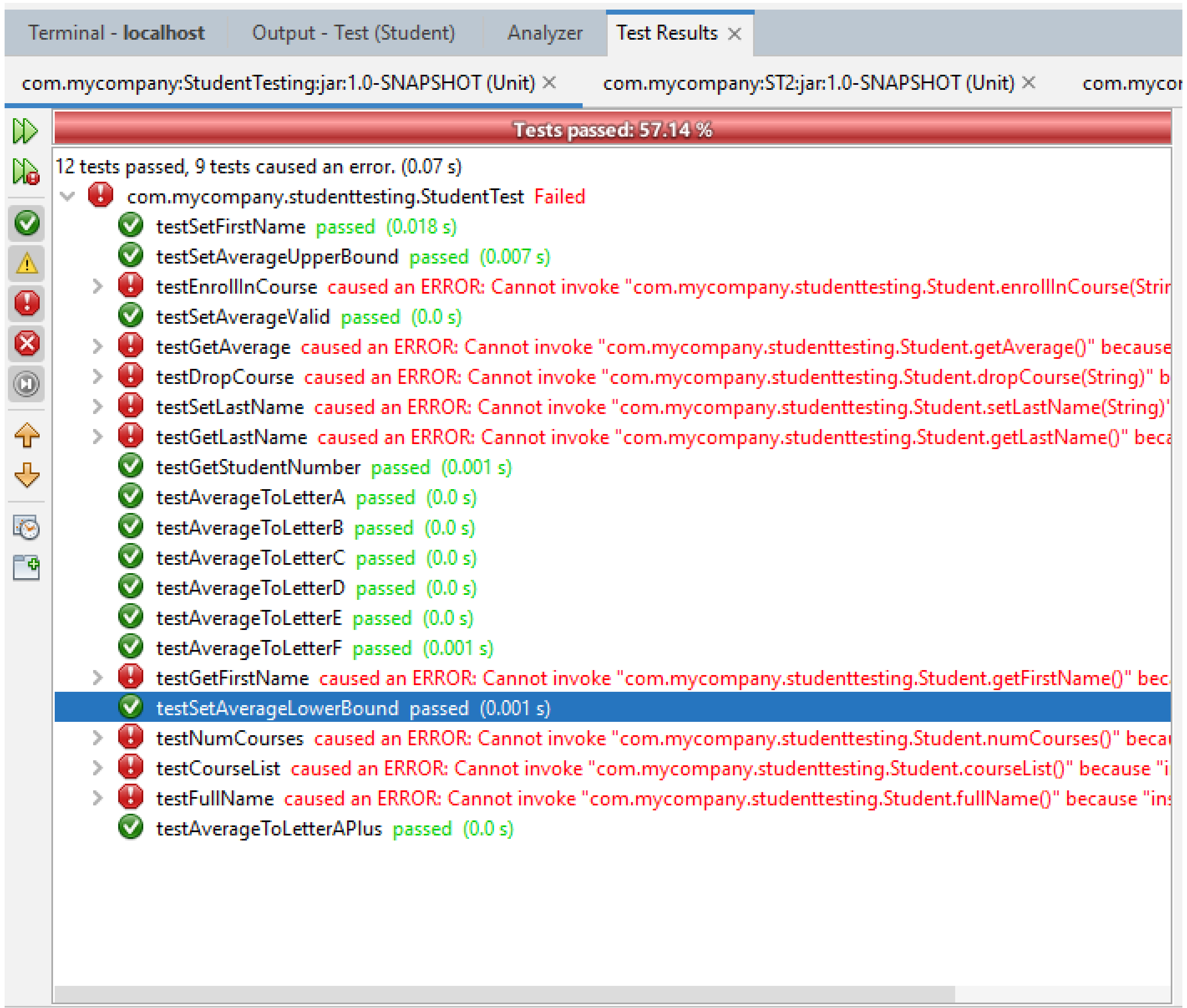
Same idea as testSetAverageLowerBounds but now we are testing the upper bounds with Math.nextUp(100) which finds the floating point number after 100.

In this case it is 100.00000762939453

```
ge(average);});
```

```
    @Test
    public void testSetAverageUpperBound() {
        System.out.println("setAverage");
        double average = Math.nextUp(100);
        System.out.println("Testing average of " + average);
        Student instance = new Student(1234, "Joe", "Bloggs", 32.12);
        assertThrows(IllegalArgumentException.class, ()->{instance.setAverage(average);});
    }
```

Testing For an Exception: testSetAverage



Terminal - **localhost** | Output - Test (Student) | Analyzer | Test Results

com.mycompany:StudentTesting:jar:1.0-SNAPSHOT (Unit) | com.mycompany:ST2:jar:1.0-SNAPSHOT (Unit) | com.mycor

```
Tests passed: 57.14 %
12 tests passed, 9 tests caused an error. (0.07 s)
com.mycompany.studenttesting.StudentTest Failed
    testSetFirstName passed (0.018 s)
    testSetAverageUpperBound passed (0.007 s)
    testEnrollInCourse caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.enrollInCourse(Strin
    testSetAverageValid passed (0.0 s)
    testGetAverage caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.getAverage()" because
    testDropCourse caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.dropCourse(String)" b
    testSetLastName caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.setLastName(String)"
    testGetLastName caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.getLastName()" beca
    testGetStudentNumber passed (0.001 s)
    testAverageToLetterA passed (0.0 s)
    testAverageToLetterB passed (0.0 s)
    testAverageToLetterC passed (0.0 s)
    testAverageToLetterD passed (0.0 s)
    testAverageToLetterE passed (0.0 s)
    testAverageToLetterF passed (0.001 s)
    testGetFirstName caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.getFirstName()" bec
    testSetAverageLowerBound passed (0.001 s)
    testNumCourses caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.numCourses()" becau
    testCourseList caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.courseList()" because "i
    testFullName caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.fullName()" because "ins
    testAverageToLetterAPlus passed (0.0 s)
```

Testing Arrays: testEnrollInCourse()

Automatically Generated Code

```
@Test
public void testEnrollInCourse() {
    System.out.println("enrollInCourse");
    String course = "";
    Student instance = null;
    instance.enrollInCourse(course);
    // TODO review the generated test code and remove the default call to fail.
    fail("The test case is a prototype.");
}
```

We now want to test the enrollInCourse method that adds the course to a list that can be retrieved with the courseList method.

enrollInCourse should throw an exception if the student is already enrolled in a course.

```java
@Test
public void testEnrollInCourseOneValue() {
    System.out.println("enrollInCourse");
    String courses[] = {"CS1032"};
    Student instance = new Student(1234, "Joe", "Bloggs");
    instance.enrollInCourse(courses[0]);
    String result[] = instance.courseList();
    assertArrayEquals(courses, result);
}
```

Testing enrollInCourse by adding a single value, “CS1032”.

```java
public void testEnrollInCourseTwoValues() {
    System.out.println("enrollInCourse");
    String courses[] = {"CS1032", "CS2212"};
    Student instance = new Student(1234, "Joe", "Bloggs");
    instance.enrollInCourse(courses[0]);
    instance.enrollInCourse(courses[1]);
    String result[] = instance.courseList();
    assertArrayEquals(courses, result);
}

@Test
public void testEnrollInCourseManyValues() {
    System.out.println("enrollInCourse");
    String courses[] = {"CS1032", "CS2212", "CS2211", "CS1026", "CS1027", "CS2034"};
    Student instance = new Student(1234, "Joe", "Bloggs");
    for(String course: courses) {
        instance.enrollInCourse(course);
```

```java
@Test
public void testEnrollInCourseOneValue() {
    System.out.println("enrollInCourse");
    String courses[] = {"CS1032"};
    Student instance = new Student(1234, "Joe", "Bloggs");
    instance.enrollInCourse(courses[0]);
    String result[] = instance.courseList();
    assertArrayEquals(courses, result);
}
```

We call courseList to get an array of courses the student is enrolled in. This should now only contain “CS1032”.

```java
    System.out.println("enrollInCourse");
    String courses[] = {"CS1032", "CS2212"};
    Student instance = new Student(1234, "Joe", "Bloggs");
    instance.enrollInCourse(courses[0]);
    instance.enrollInCourse(courses[1]);
    String result[] = instance.courseList();
    assertArrayEquals(courses, result);
}

@Test
public void testEnrollInCourseManyValues() {
    System.out.println("enrollInCourse");
    String courses[] = {"CS1032", "CS2212", "CS2211", "CS1026", "CS1027", "CS2034"};
    Student instance = new Student(1234, "Joe", "Bloggs");
    for(String course: courses) {
```

```
    @Test
    public void testEnrollInCourseOneValue() {
        System.out.println("enrollInCourse");
        String courses[] = {"CS1032"};
        Student instance = new Student(1234, "Joe", "Bloggs");
        instance.enrollInCourse(courses[0]);
        String result[] = instance.courseList();
        assertArrayEquals(courses, result);
    }
```

assertArrayEquals allows us to check that the array returned is equal to the array we excepted.

```
    @
    p
        System.out.println("enrollInCourse");
        String courses[] = {"CS1032", "CS2212"};
        Student instance = new Student(1234, "Joe", "Bloggs");
        instance.enrollInCourse(courses[0]);
        instance.enrollInCourse(courses[1]);
        String result[] = instance.courseList();
        assertArrayEquals(courses, result);
    }

    @Test
    public void testEnrollInCourseManyValues() {
        System.out.println("enrollInCourse");
        String courses[] = {"CS1032", "CS2212", "CS2211", "CS1026", "CS1027", "CS2034"};
        Student instance = new Student(1234, "Joe", "Bloggs");
        for(String course: courses) {
            instance.enrollInCourse(course);
        assertArrayEquals(courses, result);
    }

    @Test
    public void testEnrollInCourseTwoValues() {
        System.out.println("enrollInCourse");
        String courses[] = {"CS1032", "CS2212"};
        Student instance = new Student(1234, "Joe", "Bloggs");
        instance.enrollInCourse(courses[0]);
        instance.enrollInCourse(courses[1]);
        String result[] = instance.courseList();
        assertArrayEquals(courses, result);
    }

    @Test
    public void testEnrollInCourseManyValues() {
        System.out.println("enrollInCourse");
        String courses[] = {"CS1032", "CS2212", "CS2211", "CS1026", "CS1027", "CS2034"};
        Student instance = new Student(1234, "Joe", "Bloggs");
        for(String course: courses) {
            instance.enrollInCourse(course);
        }
        String result[] = instance.courseList();
        assertArrayEquals(courses, result);
    }

    @Test
    public void testEnrollInCourseException() {
        assertArrayEquals(courses, result);
    }

    @Test
    public void testEnrollInCourseTwoValues() {
        System.out.println("enrollInCourse");
        String courses[] = {"CS1032", "CS2212"};
        Student instance = new Student(1234, "Joe", "Bloggs");
        instance.enrollInCourse(courses[0]);
        instance.enrollInCourse(courses[1]);
        String result[] = instance.courseList();
        assertArrayEquals(courses, result);
    }
```

Same idea as before, but now testing adding two values, CS1032 and CS2212.

```
        String courses[] = {"CS1032", "CS2212", "CS2211", "CS1026", "CS1027", "CS2034"};
        Student instance = new Student(1234, "Joe", "Bloggs");
        for(String course: courses) {
            instance.enrollInCourse(course);
        }
        String result[] = instance.courseList();
        assertArrayEquals(courses, result);
    }

    @Test
    public void testEnrollInCourseException() {
        assertArrayEquals(courses, result);
    }

    @Test
    public void testEnrollInCourseTwoValues() {
        System.out.println("enrollInCourse");
        String courses[] = {"CS1032", "CS2212"};
        Student instance = new Student(1234, "Joe", "Bloggs");
        instance.enrollInCourse(courses[0]);
        instance.enrollInCourse(courses[1]);
        String result[] = instance.courseList();
        assert
    }
```

Now testing with many values.

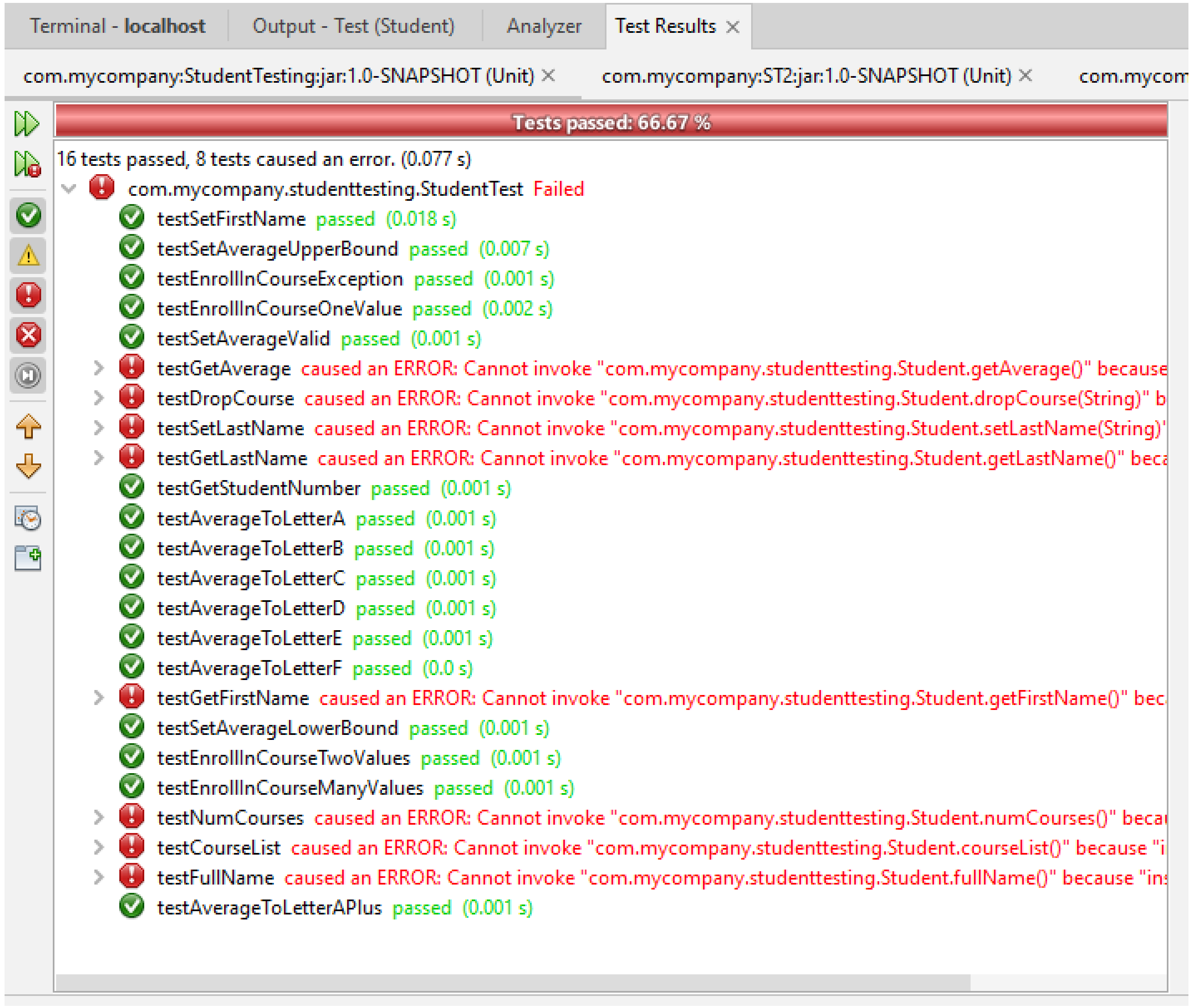
A for each loop is used to add each value from courses individually.

```
    @Test
    public void testEnrollInCourseManyValues() {
        System.out.println("enrollInCourse");
        String courses[] = {"CS1032", "CS2212", "CS2211", "CS1026", "CS1027", "CS2034"};
        Student instance = new Student(1234, "Joe", "Bloggs");
        for(String course: courses) {
            instance.enrollInCourse(course);
        }
        String result[] = instance.courseList();
        assertArrayEquals(courses, result);
    }

    @Test
    public void testEnrollInCourseException() {
```

```java
        assertArrayEquals(courses, result);
    }

    @Test
    public void testEnrollInCourseManyValues() {
        System.out.println("enrollInCourse");
        String courses[] = {"CS1032", "CS2212", "CS2211", "CS1026", "CS1027", "CS2034"};
        Student instance = new Student(1234, "Joe", "Bloggs");
        for(String course: courses) {
            instance.enrollInCourse(course);
        }
        String
        assert
    }

    @Test
    public void testEnrollInCourseException() {
        System.out.println("enrollInCourse");
        Student instance = new Student(1234, "Joe", "Bloggs");
        instance.enrollInCourse("CS2212");
        assertThrows(IllegalArgumentException.class, ()->{instance.enrollInCourse("CS2212");});
    }
```

Lastly check that enrollInCourse will throw an exception if the same course is added. In this case we try adding CS2212 twice and assert that an IllegalArgumentException is thrown on the second call.

Testing For an Exception: testSetAverage


Terminal - localhost
Output - Test (Student)
Analyzer
Test Results
com.mycompany:StudentTesting:jar:1.0-SNAPSHOT (Unit)
com.mycompany:ST2:jar:1.0-SNAPSHOT (Unit)
com.mycom

```
Tests passed: 66.67 %
16 tests passed, 8 tests caused an error. (0.077 s)
com.mycompany.studenttesting.StudentTest  Failed
    testSetFirstName  passed  (0.018 s)
    testSetAverageUpperBound  passed  (0.007 s)
    testEnrollInCourseException  passed  (0.001 s)
    testEnrollInCourseOneValue  passed  (0.002 s)
    testSetAverageValid  passed  (0.001 s)
    testGetAverage  caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.getAverage()" because
    testDropCourse  caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.dropCourse(String)" b
    testSetLastName  caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.setLastName(String)"
    testGetLastName  caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.getLastName()" beca
    testGetStudentNumber  passed  (0.001 s)
    testAverageToLetterA  passed  (0.001 s)
    testAverageToLetterB  passed  (0.001 s)
    testAverageToLetterC  passed  (0.001 s)
    testAverageToLetterD  passed  (0.001 s)
    testAverageToLetterE  passed  (0.001 s)
    testAverageToLetterF  passed  (0.0 s)
    testGetFirstName  caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.getFirstName()" bec
    testSetAverageLowerBound  passed  (0.001 s)
    testEnrollInCourseTwoValues  passed  (0.001 s)
    testEnrollInCourseManyValues  passed  (0.001 s)
    testNumCourses  caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.numCourses()" becau
    testCourseList  caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.courseList()" because "i
    testFullName  caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.fullName()" because "ins
    testAverageToLetterAPlus  passed  (0.001 s)
```

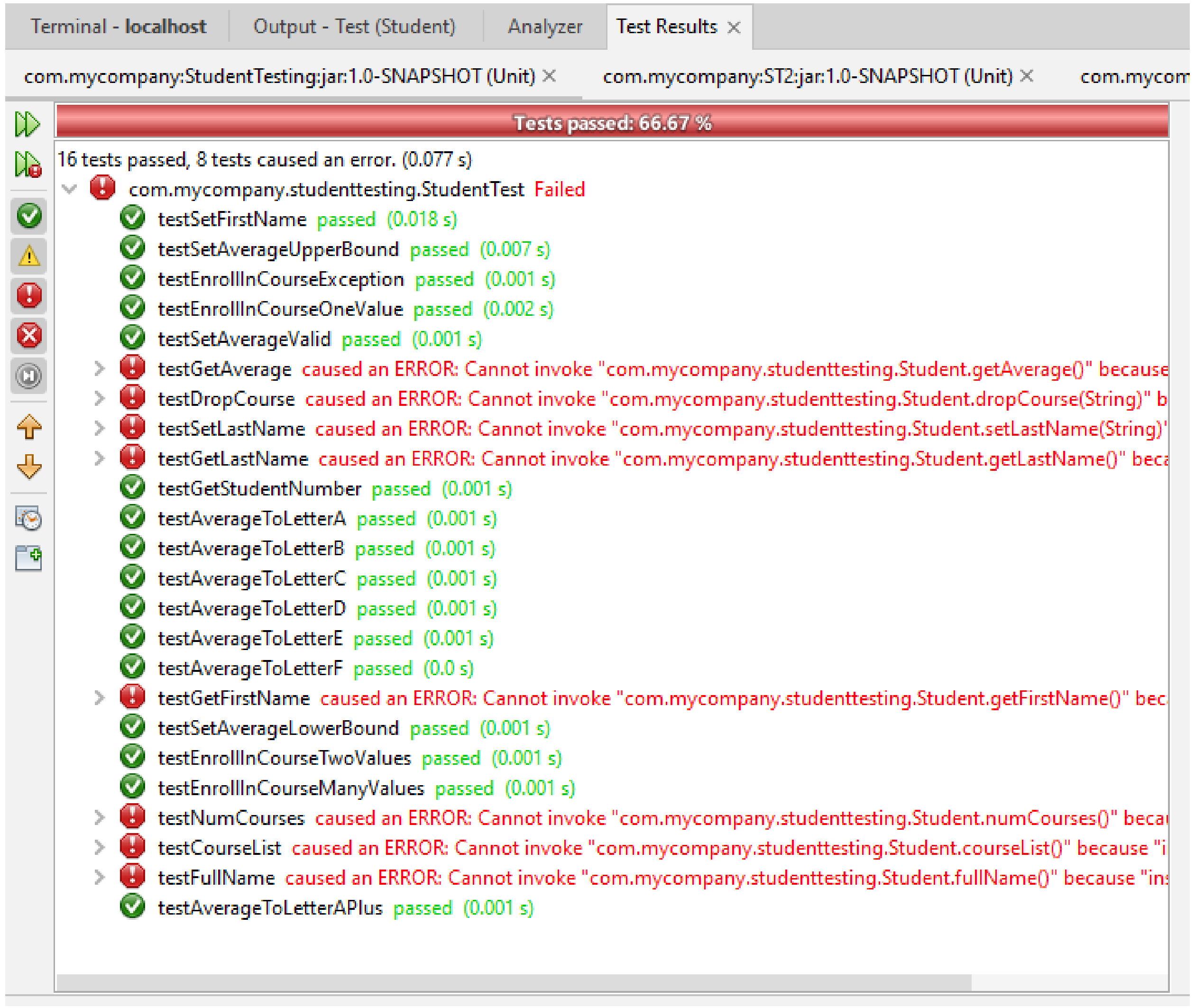
Activity

We still have some tests that are failing as we have not implemented them yet.

Individually or as a group try to create tests for the remaining methods.

Start with **testSetLastName**, **testGetLastName**, and **testFullName** as these are easier tests to implement.

If you have time remaining, try implementing the other tests (note that in some cases you need to add extra testing methods).


```
Terminal - localhost   Output - Test (Student)   Analyzer   Test Results ×
com.mycompany:StudentTesting:jar:1.0-SNAPSHOT (Unit) ×   com.mycompany:ST2:jar:1.0-SNAPSHOT (Unit) ×   com.mycom

Tests passed: 66.67 %
16 tests passed, 8 tests caused an error. (0.077 s)
com.mycompany.studenttesting.StudentTest Failed
    testSetFirstName passed (0.018 s)
    testSetAverageUpperBound passed (0.007 s)
    testEnrollInCourseException passed (0.001 s)
    testEnrollInCourseOneValue passed (0.002 s)
    testSetAverageValid passed (0.001 s)
    testGetAverage caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.getAverage()" because
    testDropCourse caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.dropCourse(String)" b
    testSetLastName caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.setLastName(String)"
    testGetLastName caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.getLastName()" beca
    testGetStudentNumber passed (0.001 s)
    testAverageToLetterA passed (0.001 s)
    testAverageToLetterB passed (0.001 s)
    testAverageToLetterC passed (0.001 s)
    testAverageToLetterD passed (0.001 s)
    testAverageToLetterE passed (0.001 s)
    testAverageToLetterF passed (0.0 s)
    testGetFirstName caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.getFirstName()" bec
    testSetAverageLowerBound passed (0.001 s)
    testEnrollInCourseTwoValues passed (0.001 s)
    testEnrollInCourseManyValues passed (0.001 s)
    testNumCourses caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.numCourses()" beca
    testCourseList caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.courseList()" because "i
    testFullName caused an ERROR: Cannot invoke "com.mycompany.studenttesting.Student.fullName()" because "ins
    testAverageToLetterAPlus passed (0.001 s)
```

You can find a copy of the current version of *StudentTest.java* on OWL with the tests created in the slides.