# *Programming III (420-B31-HR)*

# *Lab 4 –Testing; Singly Linked Lists*

Date assigned: Tuesday, September 13, 2016

**Objectives:**

1. Review how to write test cases
2. Learn to:
   1. use JUnit for unit testing
   2. use a singly linked list

**Reference:**

PowerPoint Slides from Moodle: *Testing.pptx*

PowerPoint Slides from Moodle: *SinglyLinkedLists.pptx*

**To be handed in:**

1. The Lab 4 Quiz should be completed on **Moodle**.
2. Your ***username\_*L04\_Testing\_SinglyLinkedLists** folder should be zipped and uploaded to **Moodle**. Show the teacher your completed lab to get it marked.

# Unit Testing with JUnit

**Objectives**: Learn to test method calls with JUnit.

**To Do:**

**To Start:**

1. Copy the **B31\_L04\_ Testing\_SinglyLinkedLists** folder from the **Moodle page** for the course. Rename it to ***username*\_B31\_L04\_ Testing\_SinglyLinkedLists**.
2. Start **Eclipse** and use your **420-B31\Labs** folder as your workspace.
3. Create a new **Java Project** called ***username*\_B31\_L04\_ Testing\_SinglyLinkedLists**. (Use the **JavaSE-1.8** execution environment JRE.)

**To Do:**

The class diagram for the **Temperature** class is shown here:



A series of test cases have been created to test the methods of the **Temperature** class (see the appendix.) We are going to use **JUnit** to test the test cases. The first test case we are going to test is:

**Test Case 1** Instantiation of a Temperature object using default values for the attributes.

|  |  |  |  |
| --- | --- | --- | --- |
| Operation | Purpose | Object State | Expected Result |
| Temperature t1 = new Temperature() | To create a temperature object using the default values. | units = 'C'  temperature = 0.0 | A new Temperature object with default values for the attributes. |
| t1.getTemperature() | To verify instantiation and accessor method |  | 0.0 |
| t1.getUnits() | To verify instantiation and accessor method |  | 'C' |

## First we have to create a subclass of the **JUnit** **TestCase** class:

### Select **File**-> **New**. Select **JUnit** **Test Case** . Make sure that **New Junit 4 test** is checked and that the source folder is your ***username\_B31\_L04*\_ Testing\_SinglyLinkedLists** **/src** folder. Complete the rest of the window as follows:

**Package:** b31\_l04 \_testing

**Name**: TemperatureTest

**Class under test**: b31\_l04\_ testing.Temperature

### Click **Finish**. If you get a message stating that Junit 4 is not on the build path, accept the default action.

## Comment out the testTemperature method that is created by default.

## Add a test method to the **TemperatureTest** class as follows:

**public void testCase1()**

**{**

**// Test the instantiation of a Temperature object using default values**

**// for the attributes.**

**Temperature t1 = new Temperature();**

**// verify that the temperature has been correctly set**

**assertEquals("Test Case 1: getTemperature()", 0.0,**

**t1.getTemperature(),0.000001);**

**// verify that the units have been correctly set**

**assertEquals("Test Case 1: getUnits ()", 'C', t1.getUnits());**

**} // testCase1()**

To see what **assert()** methods are available and what they do, refer to the documentation at the junit.org site <http://junit.sourceforge.net/javadoc/org/junit/Assert.html>

## Run **TemperatureTest**. At the left of the screen the JUnit pane will be display the results of the test cases:

*The green bar of happiness – all the test cases passed.*

## Refer to the appendix to code the JUnit test methods for test cases 2 and 3. The name of each JUnit test method must begin with **test**. Run **TemperatureTest** to test each one as you add it.

Now we are going to write the JUnit test method for test case 4.

**Test Case 4** Instantiation of a Temperature object using illegal values for the attributes.

| Operation | Purpose | Object State | Expected Result |
| --- | --- | --- | --- |
| Temperature t4 = new Temperature(24.0,'A') | To create a temperature object using an illegal value for units. |  | Illegal argument exception |

## Code the **testCase4()** method as follows:

**public void testCase4()**

**{**

**// Test the instantiation of a Temperature object using illegal values**

**// for the attributes.**

**try**

**{**

**// Try to instantiate a Temperature object with an invalid unit type**

**Temperature t4 = new Temperature(24.0, 'A');**

**// If the program gets here, the exception wasn't thrown**

**fail("Test case 4 - IllegalArgumentException was not thrown.");**

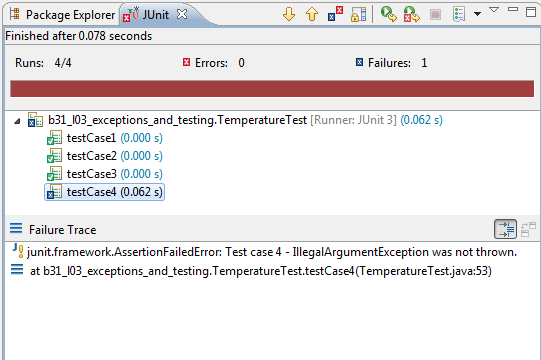
**}**

**catch (IllegalArgumentException e)**

**{**

**assertTrue(true); // It worked**

**}**

** } // testCase4()**

## Run **TemperatureTest**. The output will now look like:

*The red bar of sadness – at least one of the test cases failed.*

There is an error. The first line of the error message contains the message you provided in the **testCase4()** method.

## Correct the **setUnits()** method in the **Temperature** class so that it throws an exception as documented. Run **TemperatureTest** again. If the bar is still red, there is still something wrong.

## Once you get a green bar, add test methods for the rest of the test cases. There are other errors in the Temperature class. Correct them as you encounter them. Test and correct until you get the green bar again.

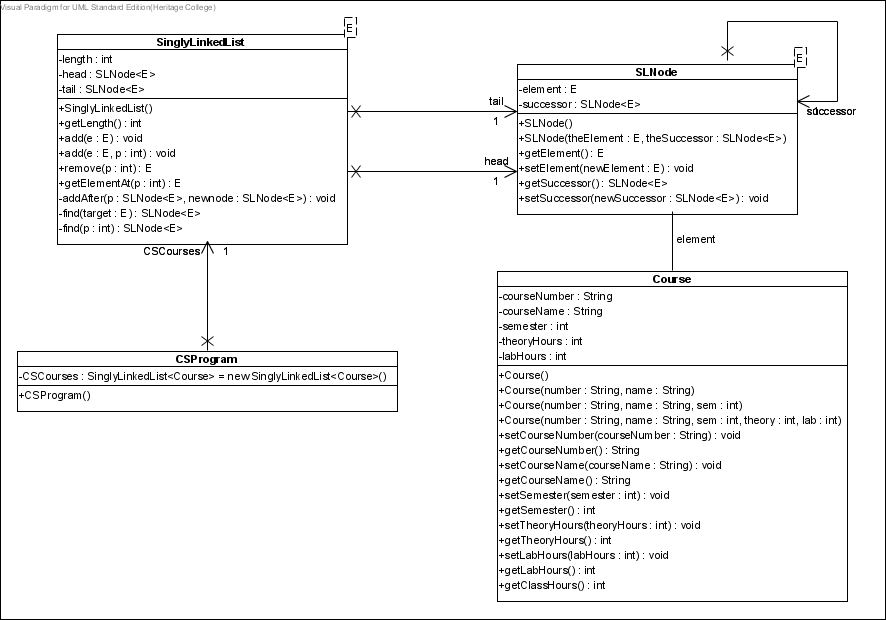
## The template for a test case is in a Word document called ***username*\_B31\_L04\_Answers.docx**. Use it create a test case 9, which is similar to test case 8, to test the **getFahrenheit()** method.

## Add a method to your JUnit test case to test the test case you created in the previous step.

# Singly Linked Lists

**Objectives:** Learn how to use a singly linked list.

**To Do:**

The **CSProgram** class contains a singly linked list of **Course** objects representing the courses in the Computer Science program. The constructor has been coded to add all of the courses. The class diagram is shown here:

## Open the **CSProgram** class.

What is the course number of the first course added to the list? 420-B31

What is the course number of the last course added to the list? 420-E71

## Run the **SinglyLinkedListDemo** program. It creates a **CSProgram** object and executes the **listCourses()** method.

What is the course number of the first course in the list? 420-E71

What is the course number of the last course in the list? 420-B31

Where is an element added to the list – at the beginning or at the end of the list? To the beginning

## Create a method in the **CSProgram** class called **getHours(int)** that accepts a semester number and returns the number of weekly classroom (lecture + lab) hours in that semester. Your method should throw an **IllegalArgumentException** if the semester is not between 1 and 6. The Javadoc documentation for the **SinglyLinkedList** class is in the **javadoc** folder in ***username*\_B31\_ L04\_ Testing\_SinglyLinkedLists** folder

## Complete test cases 2 and 4 in the **getHours()** test plan in ***username*\_B31\_L04\_Answers.docx**.

## Create a JUnit class to test your **getHours()** method using all the test cases in the test plan.

## Override the **Object** **equals()** method in the **Course** class. Two courses are considered equal if their course numbers are the same.

## Create a JUnit class to test your **equals()** method using the test cases in Appendix I.

## Add a method to the **SinglyLinkedList** class to replace a node with a target object. The method signature should be **replace(E target, E replacement)**. The method should return true if the replacement was successful and false if not. The pseudocode for the method is:

## Let cursor reference the node whose element field is to be replaced.

## Replace cursor’s element field with the new element.

## Add a **boolean** method called **reviseProgram()** to the **CSProgram** class to make the changes described in the table below. The method should return true if successful and false if not. (Note that reviseProgram should not pass in parameters – it should deal specifically with the courses listed below.)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Action** | **Old Course** | | | | | **New Course** | | | | |
| **Course**  **Num** | **Name** | **Sem** | **Theory** | **Lab** | **Course**  **Num** | **Name** | **Sem** | **Theory** | **Lab** |
| Add |  |  |  |  |  | 420-C50 | Web Programming V | 5 | 2 | 3 |
| Remove | 420-B50 | Programming V | 5 | 2 | 2 |  |  |  |  |  |
| Replace | 420-B40 | Programming IV | 4 | 2 | 2 | 420-B41 | Programming IV | 5 | 2 | 3 |

## Add a method to your JUnit class that tests **CSProgram** to test your **reviseProgram()** method. Use the **getHours()** method to see if the number of hours in 4th semester has decreased to 15 and the number of hours in the 5th semester has increased to 26.

# Moodle Quiz

1. Log on to **Moodle**, go to the **Programming III** course page and complete the **Lab 4 Quiz** by September 16.

**Marking Scheme**

|  |  |  |
| --- | --- | --- |
|  | **Mark** | **Out of** |
| **Part A: Junit** |  |  |
| TemperatureTest Class |  | 17 |
| Temperature Class |  | 4 |
| Test Case 9 |  | 8 |
| **Part B: Singly Linked Lists** |  |  |
| CSProgram   * getHours() method |  | 11 |
| * reviseProgram() method |  | 19 |
| getHours() test plan |  | 5 |
| SinglyLinkedList replace() |  | 9 |
| CSProgram junit test case   * getHours() method |  | 10 |
| * reviseProgram() method |  | 5 |
| Course equals() method |  | 5 |
| CourseTest |  | 9 |
| **Total** |  | **102** |