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

# *Assignment 2 – Linked Lists*

Date assigned: Friday, Sept. 18, 2015

Phase 1 Date Due: Friday, Sept. 25, 2015 in class (no late submissions accepted)

Phase 2 Date Due: Monday, Oct. 5, 2015

***Objectives:***

After completing this assignment, the student should be able to:

1. Draw a class diagram.
2. Design test cases for a linked list.
3. Add to, remove from, find and traverse a singly linked list.
4. Override the **toString()** and **equals()** method of the **Object** class.

***Problem Specifications:***

Develop a program to maintain a To-do List. The To-do List is a **singly linked list** of homework tasks. A homework task can be a Lab, Assignment, Test, Reading, Essay, or Other. Each homework task has a priority associated with it. The priority can be High, Medium or Low. Your program should provide the following services:

* add a new homework task to the **start** of the list
* remove a homework task when it is complete
* provide a list of the homework tasks in the order of the date that they were assigned
* find the homework task(s) with the earliest due date.
* output the homework tasks in the order of their due date.
* output the homework tasks that are high priority, ordered by due date

Your design must use the following classes:

* 1. **Homework** class that represents a single homework task. It should contain at a minimum, the course number, priority(High, Medium, Low), task type(Lab, Assignment, Test, Reading, Essay, Other), task number (i.e. Lab 1, Test 4, etc), optional task description if type is other, date assigned and date due.

I highly recommend having the dates be objects of the **GregorianCalendar** class.

The **Homework** class must override the equals(), toString() and compareTo() methods. You will need these in order to compare the dates.

* Provide an overridden **equals()** method. Two **Homework** objects are considered equal if the course number, task type and task number are the same.
* Provide an overridden **toString()** method. It should return a **String** with the course number, task type, task number, priority, date assigned and date due. If the task description is not null it should be included after the task type.
* Provide the **compareTo(Homework** *obj***)** method. This should return a negative number if the due date is less than *obj* due date; should return a zero if the due date is the same as *obj* due date; should return a positive number if the due date is greater than *obj* due date.
  1. **HomeworkList** class that is a singly linked list of **Homework** objects. (Use the **SinglyLinkedList** class for your list.)
* The **HomeworkList** class must, at a minimum, provide the following methods, which you should determine the parameters for:
* **addTask() -** add a new task to the **start** of the list

**removeTask() -** remove a specific task

**findNextDueTasks() -** find **all** of the task(s) with the earliest due date (there can be more than one with the same due date (i.e. a list of them)).

**showTasksByAssignedDate() -** output all of the tasks in the order of their assigned date

**showTasksByDueDate() -** output all of the tasks in the order they are due

**showHighPriorityTasksByDueDate() -** output all of the high priority tasks in the order they are due

* 1. **SinglyLinkedList** provided from Moodle. \*Note that you should use this class, but not change the code in it.
  2. **SLNode** class provided from Moodle. \*Note that you should use this class, but not change the code in it.
  3. **HomeworkTest** class that contains the junit test cases for the **Homework** class.
  4. **HomeworkListTest** class that contains the junit test cases for the **HomeworkList** class.

***To Do:***

***Phase 1:***

1. Copy the **linkedList** folder from Moodle to your **420-B31\Assignments** folder.
2. Draw a class diagram for the program. Make sure that all classes listed in the problem specification are included (6 in total).
3. Design test cases for the for the following methods. The table below shows the assignment of test cases to each student. On the assignment due date, I will put all of the test cases together and provide them to the class to use to write your junit test cases for the assignment.

|  |  |  |
| --- | --- | --- |
| Class | Method to Test (determine parameters for method) | Student |
| HomeworkList | addTask(…) | Michael, Frank |
| HomeworkList | removeTask(…) | Kevin, Troy |
| HomeworkList | findNextDueTasks(…) | Ryan, Zach |
| HomeworkList | showTasksByAssignedDate(…) | Tyler, Guillaume |
| HomeworkList | showTasksByDueDate(…) | Francis, Nicolas |
| HomeworkList | showHighPriorityTasksByDueDate(…) | Dzmitry, Isaac |
| Homework | equals(…) | Emilie, Louis |
| Homework | compareTo(…) | Mark, Thomas |

The test case format should use the table we learned in class, which is listed below, with an example row for the start of a test case

|  |  |  |  |
| --- | --- | --- | --- |
| Operation | Purpose | Object State | Expected Result |
| Homework task1= new Homework("420-B31", "Lab", 1, “High”); | Instantiate a Homework object | courseNumber = "420-B31"  TaskType = "Lab"  TaskNumber = 1  Task Priority = “High”  dateAssigned = new GregorianCalendar()  dateDue = new GregorianCalendar() | A new default HomeworkList list |

1. Submit the class diagram and test cases in class on Fri. Sept. 25, 2015. We will be discussing the solution in class that day, so no late submissions will be accepted.

**Phase 2:**

1. Code the **Homework** class.
2. Code the **HomeworkTest** class with the JUnit test case to test all of the test cases (not just the one that you wrote).
3. Code the **HomeworkList** class.
4. Code the **HomeworkListTest** class with the JUnit test case to test all of the test cases (not just the one that you wrote).
5. Bonus Points: Code a frame to complete the program to meet the specifications.
6. Revise your class diagram to reflect any changes made during coding.

***Marking Scheme:***

|  |  |  |
| --- | --- | --- |
|  | **Mark** | **Out of** |
| Class Diagram |  | 10 |
| Test Case assigned to each student |  | 20 |
| Homework class code |  | 25 |
| HomeworkList class code |  | 40 |
| JUnit code – Homework and HomeworkList |  | 25 |
| Organization |  | 5 |
| **Total** |  | **125** |
| Bonus: Frame |  | 10 |

***Organization Marks:***

Marks will be given for organization. This includes:

* naming files and folders according to the department standards
* giving meaningful names to variables, classes, objects and methods
* formatting and indenting Java classes using the Eclipse format tool
* submitting the assignment in correctly on **Moodle**
* including all required files in the submitted assignment folder

***To be handed in:***

The following files should be included in a ***username*\_B31\_A02\_Linked\_Lists** folder and uploaded to **Moodle**:

1. The test cases assigned to you.
2. The class diagram for the assignment.
3. The java project for the assignment. The project name should be ***username*\_B31\_A02\_Linked\_Lists** . It should include the **JUnit** tests for the **Homework** and **HomeworkList** classes.