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

# *Assignment 2 – Linked Lists*

Date assigned: Tuesday, Sept. 27, 2016

Frame design & class diagram due: Tuesday, Oct. 4, 2016

Test cases due: Tuesday, Oct. 11, 2016 in class (no late submissions accepted)

Code and junit tests due: Friday, Oct. 21, 2016

***Objectives:***

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

1. Draw a class diagram.
2. Design junit test cases for a linked list.
3. Create a LinkedList object.
4. Use an iterator to search through a list.
5. Write a serialized object.
6. Read a serialized object.
7. 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 linked list of tasks. A task can either be a homework task or a personal task. Each task has a priority associated with it. The priority can be in the range 1 – 4, where 1 is the highest priority and 4 is the lowest priority. Each task has a due date.

A homework task should contain, at a minimum, the course number, task type (Lab, Assignment, Test, Reading, Essay, Other), task number (i.e. Lab 1, Test 4, etc.), teacher name and optional task description.

A personal task should contain, at a minimum, the task type (Appointment, Bill Payment, Errand, Other), task name (i.e. Dental Teeth Cleaning or Buy mom a birthday card or Go to gym) and optional task location.

Your program should provide the following services:

* a frame to allow all functionality to happen through a drop down menu interface.
* add a new task to the **end** of the list.
* remove a task when it is complete.
* display the task(s) with the earliest due date.
* display all tasks in the order of their due date.
* display the tasks that are high priority (priority 1), ordered by due date
* write the list to a file using serialization, where the filename is provided by the user.
* read the list from a file using serialization, where the filenmae is provided by the user.

Here are two good references for learning about serialization:

<http://www.tutorialspoint.com/java/java_serialization.htm>

<http://www.onjava.com/pub/a/onjava/excerpt/JavaRMI_10/index.html?page=1>

Your design must use the following classes:

* 1. **ToDoListFrame** class that contains the frame. All input and output is handled in the **ToDoListFrame** class. The **ToDoListFrame** should have no knowledge of how the list is stored. You must use a drop down menu.
  2. **Task** class that represents a single task.
  3. **HomeworkTask** class that represents a single homework task.
  4. **PersonalTask** class that represents a single personal task.
  5. I highly recommend having the due dates be objects of the **GregorianCalendar** class.
  6. The **HomeworkTask** 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 **HomeworkTask** objects are considered equal if the course number, teacher, task type and task number are the same.
* Provide an overridden **toString()** method. It should return a **String** with the course number, teacher, task type, task number, priority and date due. If the task description is not null it should be included after the task type.
  1. The **PersonalTask** class must also override the equals(), toString() and compareTo() methods.
* Provide an overridden **equals()** method. Two **PersonalTask** objects are considered equal if the task type and task name are the same.
* Provide an overridden **toString()** method. It should return a **String** with the task type, task name, priority and date due. If the task location is not null it should be included after the task name.
  1. The **Task** class must override the **compareTo()** method.
* Provide the **compareTo(Task** *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. **TaskList** class that is a java collection and uses a **LinkedList** (based on the List ADT) for the tasks. Specifically you should use **java.util.LinkedList** and **java.util.ListIterator.** Whenever you are traversing your **TaskList**, you **must** use an iterator to traverse it.

The **TaskList** class must, at a minimum, provide the following methods, which you should determine the parameters for:

**addTask() -** add a new task to the **end** 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)).

**getTasksByDueDate() -** return all of the tasks in the order they are due

**getHighPriorityTasksByDueDate() -** return all of the high priority tasks in the order they are due

* 1. Serialization must be used for the file reading and writing.
  2. **TaskTest** class that contains the junit test cases for the **Task** class.
  3. **TaskListTest** class that contains the junit test cases for the **TaskList** class.

***To Do:***

***Phase 1:***

1. Sketch a frame design for the program. This can be hand drawn, and either scanned or a legible photo of it can be submitted. Draw a class diagram for the program. Make sure that all classes listed in the problem specification are included (7 in total).

***Phase 2:***

1. Design test cases 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** |
| TaskList | addTask(…)  removeTask(…) | Eva-Steve, Alex B, Coady M, Brian |
| TaskList | findNextDueTasks(…) | Cody B, JJ, Vann, Brae |
| TaskList | getTasksByDueDate(…)  getHighPriorityTasksByDueDate(…) | Nathan, Andrew, Martin, Louis-Phillipe |
| Task, HomeworkTask, PersonalTask | equals(…) | Jeremy, Max, Alex S |
| Task | compareTo(…) | Cody B, JJ, Vann, Brae |
| TaskList | saveToFile(…)  readFromFile(…) | Philip, Chris, Amir |

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 |
| HomeworkTask task1= new Homework("420-B31", "Lab", 1, “High”); | Instantiate a HomeworkTask object | courseNumber = "420-B31"  TaskType = "Lab"  TaskNumber = 1  Task Priority = 1  dateDue = new GregorianCalendar() | A new default HomeworkTask object |

**Phase 2:**

1. Code the **Task**, **HomeworkTask and PersonalTask** classes.
2. Code the Task**Test** class with the JUnit test case to test all of the test cases (not just the one that you wrote).
3. Code the **TaskList** class.
4. Code the **TaskListTest** class with the JUnit test case to test all of the test cases (not just the one that you wrote).
5. Code the 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** |
| Frame Design |  | 10 |
| Class Diagram |  | 10 |
| Test Case assigned to each student |  | 20 |
| Task, HomeworkTask, PersonalTask class code |  | 25 |
| TaskList class code |  | 30 |
| TaskList code for serialization |  | 15 |
| JUnit code – Task and TaskList |  | 25 |
| Frame |  | 25 |
| Correct Program Execution |  | 10 |
| Organization |  | 5 |
| **Total** |  | **175** |

***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 frame design.
4. The java project for the assignment. The project name should be ***username*\_B31\_A02\_Linked\_Lists** .