**OOP**

**Practical 5, Week 5**

**Submission**

1. Your submission should contain two files. One of these files is **PDF** document with screenshots of the implementation (Java code) and testing only. Another file is **ZIP** file with the Java project.
2. You must save the files with name

**{YourStudentNumber}-Pratical5.pdf;**

**{YourStudentNumber}-Pratical5.zip;**

For example: 202107081314-Pratical5.pdf, 202107081314-Pratical5.zip

1. You must upload from the student website: student.zy.cdut.edu.cn

**Marking scheme**

You will gain up to 5 marks for the completion of the exercise.

The markers will use the following marking scheme for each exercise.

|  |  |
| --- | --- |
| Rubric | marks |
| No attempt has been made to answer the question. No implementation at all, or completely inappropriate considerations | 0 |
| Some attempt has been made to design the algorithm, and some considerations shown. No effort to implement a working solution and test it. | 1 |
| Incomplete design/programming, but significant effort has gone into it. Some consideration and implementation of the result, but very limited, some of the rules have not been properly implemented, no testing. | 2 |
| Mostly complete design but the implementation does not match the design or does not follow a correct standard. The program works but does not use loops and string methods properly. | 3 |
| Complete design with fair to good implementation, and some testing shown. | 4 |
| Excellent design and implementation of the whole problem, including testing and implementation. | 5 |

**OOP**

# Week 5, Assessed exercises

# Introduction

The project you'll work on is called *Task List*. It’s a simulation of a mobile phone application that helps you keep track of what things you have to do.

When the application is running, you can:

* add tasks to the list
* prioritize more important items
* remove tasks once they are done

For this app, we provide the GUI (graphical user interface) code, which you don't need to know anything about.

You will be writing the code that manipulates the list of tasks.

# Assessed Exercise 5-1: Creating a project and adding some existing code

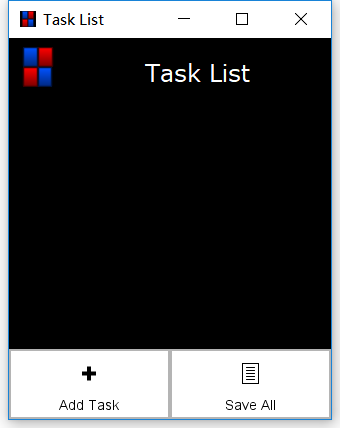
Download the files (*OOP Practical 5 student-start-version.zip*) for this project from the module website. Make a new project using the files you downloaded by **carefully** following these instructions:

1. Create a new Java Application. Make sure that the box marked “Create Main Class” is **not** ticked.
2. Open up Windows Explorer and browse to the folder in which you created your project. You should find a subfolder **src**. Place the files that you downloaded into this subfolder

You should now have the project ready to go. The program files are as follows:

* TaskListGUI.java – the main file, which you don't need to change it
* **ListsManager.java** – where you will write the code for the lists

Run the program.



One thing to note about the tasks in this application, is that tasks come in two priorities: **high priority** for important tasks, and **low priority** for other tasks. Important tasks are highlighted with red and displayed further up the screen, whilst the other tasks are coloured with blue and displayed lower down.

For example, here the important tasks are

* pick up a delivery from Dongyuan
* finish off program for OOP

and the other tasks are

* return books to the Eastern library
* submit quizzes

# Assessed Exercise 5-2: Initialising the lists

In the ListsManager class, each task is simply represented as a String, and there are actually *two* lists of Strings, called highPriority and lowPriority.

In the **Constructor** method of the **ListsManager** class, initialise both the **highPriority** and **lowPriority** lists to empty lists.

[Hint: look at the lecture notes to see how you can initialise lists.]

# Assessed Exercise 5-3: Adding a Task

When running the application, if you click the Add menu option at the bottom of the screen, you should notice a textbox appears, in which you can type a description of a task.

To get this to work, implement the **addTask** method by filling in its method body. This should take the given item and add it to the end of the list of *low priority* tasks.

If you add a new task that contains character “**#**”, please remove the character “**#**” and add this task to the end of the **highPriority** list.

[Hint: try to use method *contains* from class String]

# Assessed Exercise 5-4: Removing Items

When running the application, if you try clicking on a task, a menu of options pops up. The first option is the Done menu option, which will remove a task from the list.

To implement this, you need to fill in the body of the **removeLowPriorityTask** method, which should remove the item at the given index position for the list of low-priority tasks.

Similarly, implement the **removeHighPriorityTask** method, which does the same on the other list. At this point, you will be able to check that the **removeLowPriorityTask** method works, but will have to leave checking the other method until after the next exercise, when you can make some tasks high priority.

# Assessed Exercise 5-5: Changing a Task’s Priority

Another of the options when clicking on a task is the option to change the priority of a task. The idea is that changing the priority of a *low-priority* task will remove it from the low-priority list and add it to the end of the high-priority list. However, changing the priority of a *high-priority* task will add it to the *beginning* of the list of low-priority tasks.

Implement this by filling in the body of the **changePriority** method.

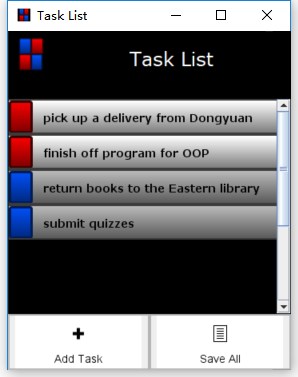
# Assessed Exercise 5-6: Making a Task More Important

The remaining option when clicking on one of the important tasks is to make the task more prominent by moving it one up the list of important tasks. So, implement the method **promote** to do just this. When this option is chosen for the first of the important tasks, there should be no effect (since the task can’t actually go higher up!)

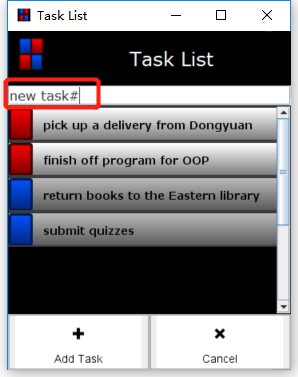
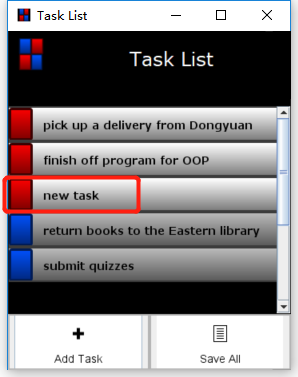
Check that your code works.

**Sample Testing Case with Four Steps and Result**

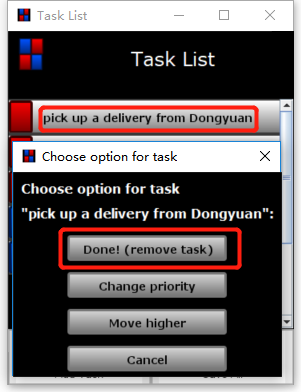
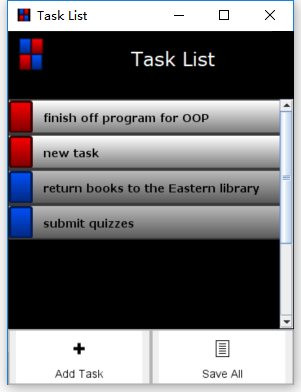
1. **Add four tasks (Task can be added only after you completed Assessed Exercise 5-1, 5-2, 5-3)**



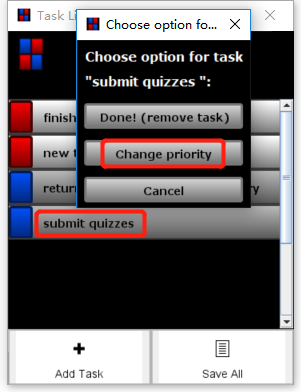
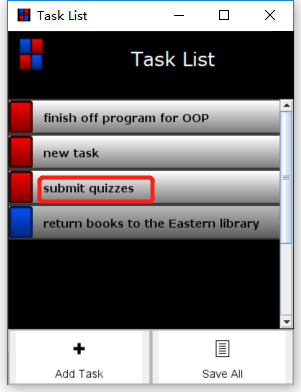
1. **Add one task with character “#”**

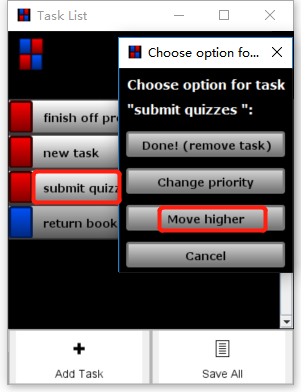
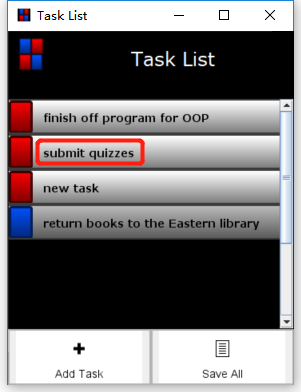
1. **Remove item**

1. **Changing a task’s priority**

1. **Making a task more important**

Complete the implementation and testing.

1. Implementation

**(Please show your design with some comments in your program and paste the source code of ListsManager.java here with screenshots)**

1. Testing

**Testing 1 (Same sample test case with the four steps)**

**Testing 2 (Your own different test case-the first step is same, other three steps are different)**