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}-Practical5.pdf;

   {YourStudentNumber}-Practical5.zip;

   For example: 202107081314-Practical5.pdf, 202107081314-Practical5.zip
3. 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:

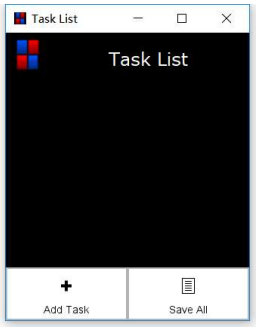
1. Create a new Java Application. Make sure that the box marked "Create Main Class" is **not** ticked.

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.



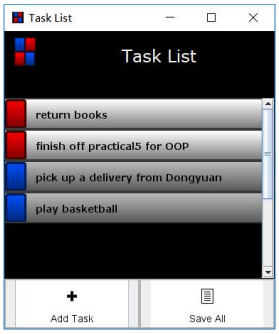
One thing to note about the tasks in this application, is that tasks come in two priorities: **high priority** for important tasks, and **low priority** for other tasks. Important tasks are highlighted with red and displayed further up the screen, whilst the other tasks are coloured with blue and displayed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

For example, here the important tasks are

return books
finish off practical5 for OOP

and the other tasks are

pick up a delivery from Dongyuan
play basketball



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.

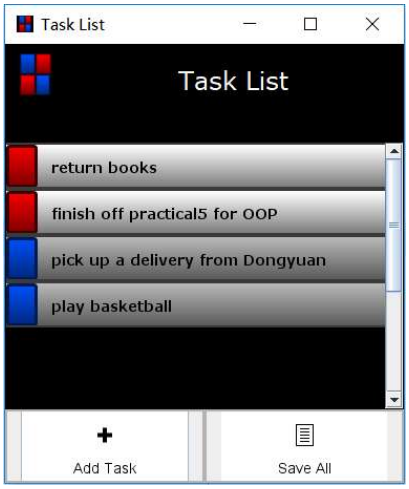
Implement this by filling in the body of the **changePriority** method.

Assessed Exercise 5-6: Making a Task More Important

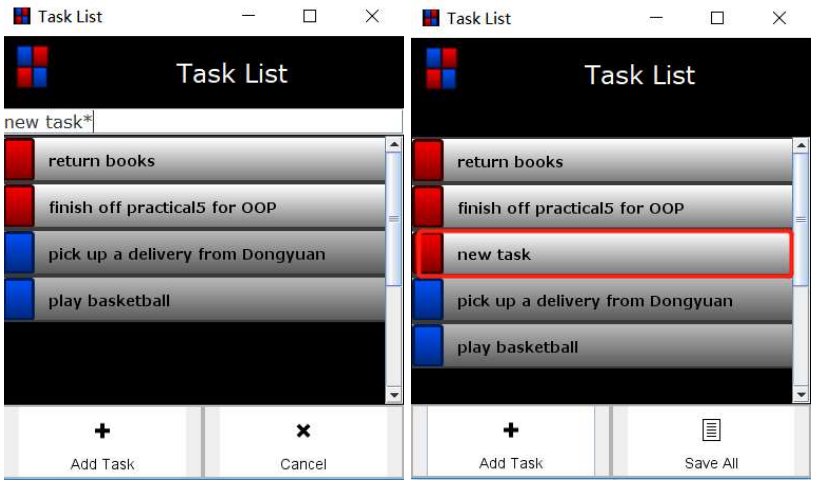
The remaining option when clicking on one of the important tasks is to make the task more prominent by moving it one up the list of important tasks. So, implement the method **promote** to do just this. When this option is chosen for the first of the important tasks, there should be no effect (since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

Sample Testing Case with Four Steps and Result

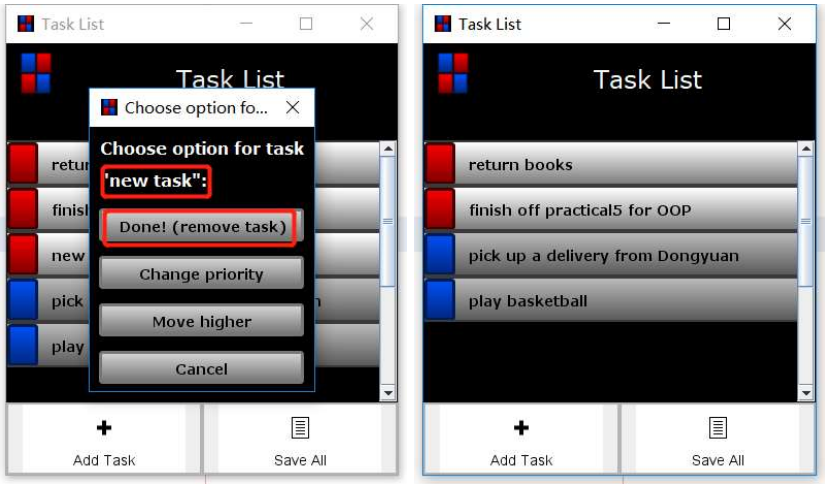
（1） Add four tasks (Task can be added only after you completed Assessed Exercise 5-1, 5-2, 5-3)



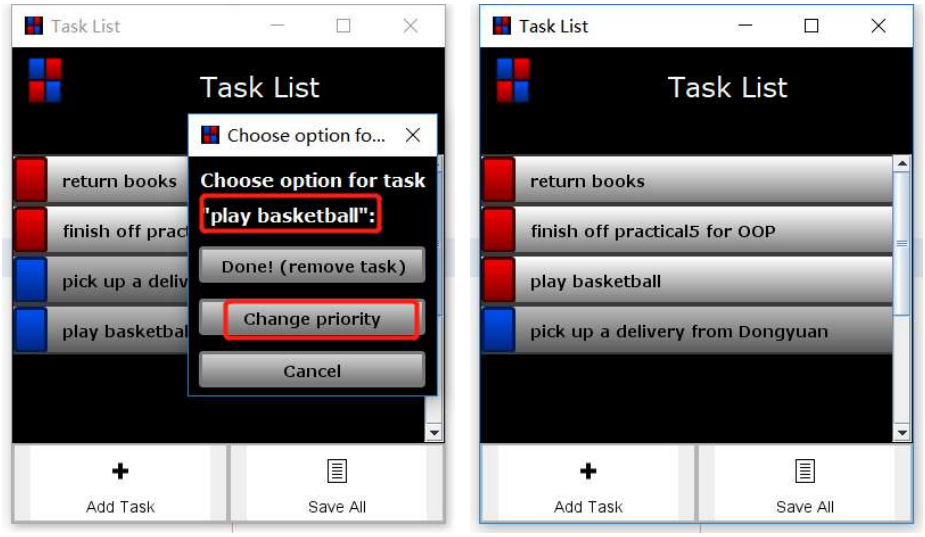
（2） Add one task with character “*”



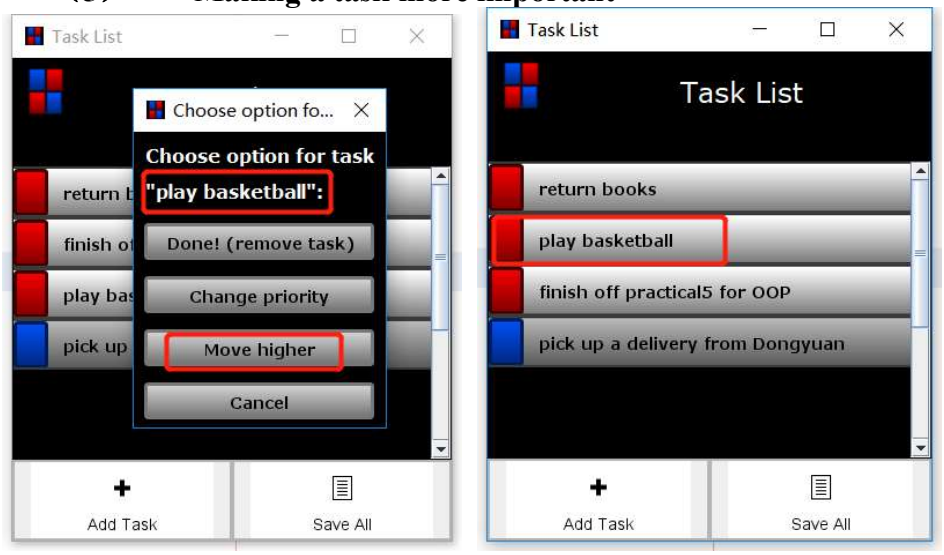
（3） Remove item



（4） Changing a task’s priority



（5） 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)

```java
/.../

import java.util.*;

2 个用法
public class ListsManager {

    //
    14 个
    public ArrayList <String> highPriority;
    10 个用法
    public ArrayList <String> lowPriority;

    1 个用法
    public TaskListGUI controller; // a reference to the main program, with the GUI

    /* Constructs a ListManager object. */
    1 个用法
    public ListsManager(TaskListGUI ref) {
        // (you should not change the code in this method)
        controller = ref;
        highPriority = new ArrayList<>();
        lowPriority = new ArrayList<>();
    /* Initialises both lists, to initially be empty. */
    }

    /* Remove the specified char from the string without replace method. */
    1 个用法
    public String deleteChar(String sourceString,String deleteChar) {
        String newString = "";
        // Creates an empty string to hold the string with the specified char removed.
        for (int i = 0;i < sourceString.length();i++) {
            char c = sourceString.charAt(i);
            // Iterate over each char in the sourceString.
            if (deleteChar.equals(c)) {
                // Do nothing to discard the char equal to deleteChar in the sourceString.
            }else {
                newString += c;
                // Add a char to newString that is not equal to deleteChar.
```

```
38              }
39          }return newString;
40      }
41
42      /* Adds the given task to the end of the low-priority list.
43       */
        1 个用法
44  @   public void addTask(String task) {
45          if (task.contains("*")) {
46              String newTask = deleteChar(task, deleteChar: "*");
47              // Remove the * from the sourceString task.
48              highPriority.add(newTask);
49              // Add the task without the * to the string list highPriority.
50              }else {
51              lowPriority.add(task);
52              // Add the task without the * to the string list lowPriority.
53          }
54      }
55
56      /* Removes the given task to the end of the high-priority list.
57       * The index integer gives the current position of the task in
58       * the relevant list of tasks. (You can assume that index is a
59       * valid position in the relevant list.)
60       */
        2 个用法
61      public void removeLowPriorityTask(int index) {
62          lowPriority.remove(index);
63          // Removes the task with the specified index from lowPriority.
64      }
65
66
67      /* Removes the given task from the high-priority list.
68       * The index integer gives the current position of the task in
69       * the relevant list of tasks. (You can assume that index is a
70       * valid position in the relevant list.)
71       */
        2 个用法
```

```
    public void removeHighPriorityTask(int index) {
        highPriority.remove(index);
        // Removes the task with the specified index from highPriority.
    }

    /* Changes the priority of the given task.
     * The given boolean important indicates the previous priority
     * of the task. e.g. if important is true, then the task was
     * a high-priority task.
     * The index integer gives the current position of the task in
     * the relevant list of tasks. (You can assume that index is a
     * valid position in the relevant list.)
     */
    2 个用法
    public void changePriority(boolean important, int index) {
        if (important == true) {
            String deleteString = highPriority.get(index);
            // Create a new string variable to store the string at the index specified in the string list highPriority.
            removeHighPriorityTask(index);
            // Start by removing the operand string from highPriority.
            lowPriority.add( index: 0,deleteString);
            //  Adds the operand string to the beginning of the string list lowPriority.
        }else {
            String deleteString = lowPriority.get(index);
            // Create a new string variable to store the string at the index specified in the string list lowPriority.
            removeLowPriorityTask(index);
            // Start by removing the operand string from lowPriority.
            highPriority.add(deleteString);
            // Adds the operand string to the beginning of the string list lowPriority.
        }
    }

    /* Produces a string that can be used to save the task lists. */
    public String toString() {
        String myHighPriorityTaskLists = highPriority.toString();
        String myLowPriorityTaskLists = lowPriority.toString();
        String myTaskLists = myHighPriorityTaskLists + myLowPriorityTaskLists;
        return myTaskLists;  // dummy return value (before this method is implemented)
    }

    /* Pushes an important task one higher up.
     * The index gives the current position of the task in the
     * highPriority list. (You can assume that index is a
     * valid position in the highPriority list.)
     */
    1 个用法
    public void promote(int index) {
        if (index > 0) {
            String s = highPriority.get(index);
            // Create a string s to hold the index task (the action object) specified in highPriority.
            highPriority.remove(index);
            // Remove the operation object from highPriority.
            highPriority.add( index: index - 1,s);
            // Adds the operation object to a higher position of highPriority.
        }
    }

    1 个用法
    void initialiseLists() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Too
    }
}
```

/* U08223 Week 1
 *
 * This class provides methods to manage the two lists of tasks.
 * Your code will complete the methods below.
 */

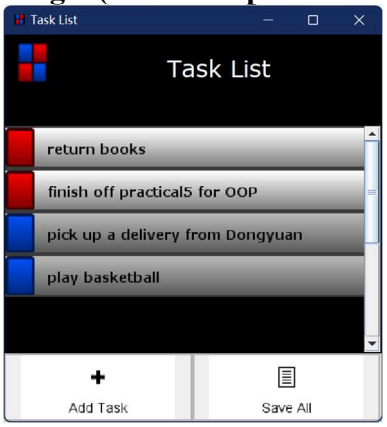
```
import java.util.*;

public class ListsManager {

   // the two lists of tasks
                        public ArrayList <String> highPriority;
   public ArrayList <String> lowPriority;

   public TaskListGUI controller; // a reference to the main program, with the GUI

   /* Constructs a ListManager object. */
   public ListsManager(TaskListGUI ref) {
      // (you should not change the code in this method)
      controller = ref;
      highPriority = new ArrayList<>();
      lowPriority = new ArrayList<>();
   /* Initialises both lists, to initially be empty. */
   }

   /* Remove the specified char from the string without replace method. */
   public String deleteChar(String sourceString,char deleteChar) {
      String newString = "";
      // Creates an empty string to hold the string with the specified char removed.
      for (int i = 0;i < sourceString.length();i++) {
         char c = sourceString.charAt(i);
         // Iterate over each char in the sourceString.
         if (Objects.equals(deleteChar, c)) {
            // Do nothing to discard the char equal to deleteChar in the sourceString.
         }else {
            newString += c;
            // Add a char to newString that is not equal to deleteChar.
         }
      }return newString;
   }

   /* Adds the given task to the end of the low-priority list.
    */
   public void addTask(String task) {
      if (task.contains("*")) {
         String newTask = deleteChar(task,'*');
         // Remove the * from the sourceString task.
         highPriority.add(newTask);
         // Add the task without the * to the string list highPriority.
         }else {
         lowPriority.add(task);
         // Add the task without the * to the string list lowPriority.
```

```
    }
  }

  /* Removes the given task to the end of the high-priority list.
   * The index integer gives the current position of the task in
   * the relevant list of tasks. (You can assume that index is a
   * valid position in the relevant list.)
   */
  public void removeLowPriorityTask(int index) {
    lowPriority.remove(index);
    // Removes the task with the specified index from lowPriority.
  }


  /* Removes the given task from the high-priority list.
   * The index integer gives the current position of the task in
   * the relevant list of tasks. (You can assume that index is a
   * valid position in the relevant list.)
   */
  public void removeHighPriorityTask(int index) {
    highPriority.remove(index);
    // Removes the task with the specified index from highPriority.
  }

  /* Changes the priority of the given task.
   * The given boolean important indicates the previous priority
   * of the task. e.g. if important is true, then the task was
   * a high-priority task.
   * The index integer gives the current position of the task in
   * the relevant list of tasks. (You can assume that index is a
   * valid position in the relevant list.)
   */
  public void changePriority(boolean important, int index) {
    if (important == true) {
      String deleteString = highPriority.get(index);
      // Create a new string variable to store the string at the index specified in the string
list highPriority.
      removeHighPriorityTask(index);
      // Start by removing the operand string from highPriority.
      lowPriority.add(0,deleteString);
      //  Adds the operand string to the beginning of the string list lowPriority.
    }else {
      String deleteString = lowPriority.get(index);
      // Create a new string variable to store the string at the index specified in the string
list lowPriority.
      removeLowPriorityTask(index);
      // Start by removing the operand string from lowPriority.
```

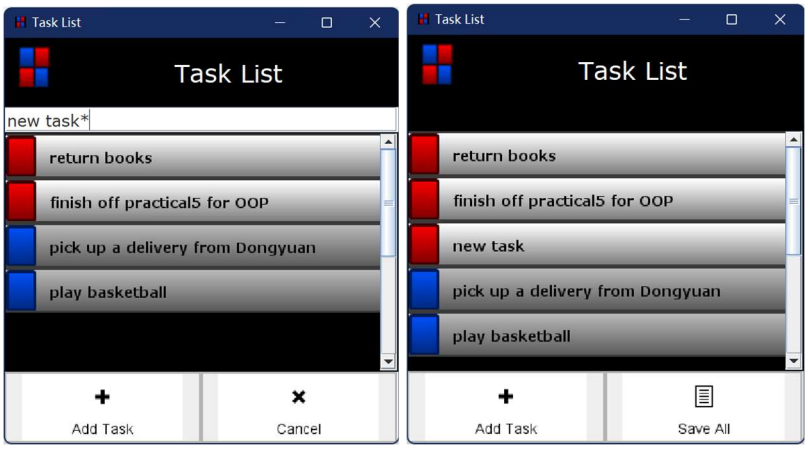
```
            highPriority.add(deleteString);
            // Adds the operand string to the beginning of the string list lowPriority.
        }
    }

    /* Produces a string that can be used to save the task lists. */
    public String toString() {
        String myHighPriorityTaskLists = highPriority.toString();
        String myLowPriorityTaskLists = lowPriority.toString();
        String myTaskLists = myHighPriorityTaskLists + myLowPriorityTaskLists;
        return myTaskLists;  // dummy return value (before this method is implemented)
    }

    /* Pushes an important task one higher up.
     * The index gives the current position of the task in the
     * highPriority list. (You can assume that index is a
     * valid position in the highPriority list.)
     */
    public void promote(int index) {
        if (index > 0) {
            String s = highPriority.get(index);
            // Create a string s to hold the index task (the action object) specified in highPriority.
            highPriority.remove(index);
            // Remove the operation object from highPriority.
            highPriority.add(index - 1,s);
            // Adds the operation object to a higher position of highPriority.
        }
    }

    void initialiseLists() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body
of generated methods, choose Tools | Templates.
    }
}
```

（2） Testing

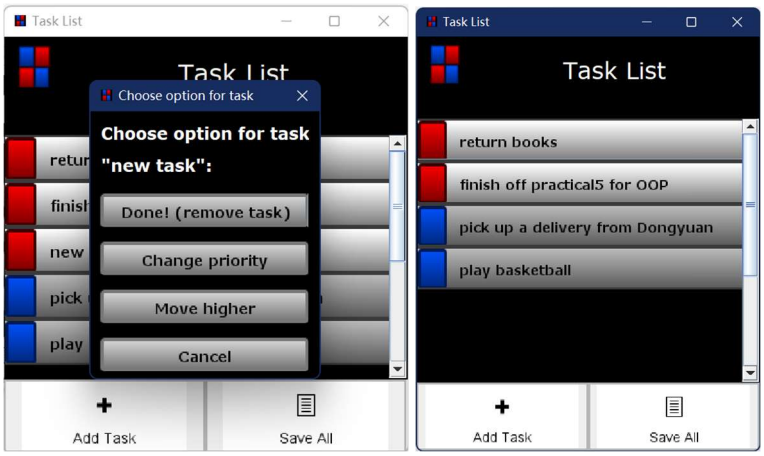
Testing 1 (Same sample test case with the four steps)

1. 

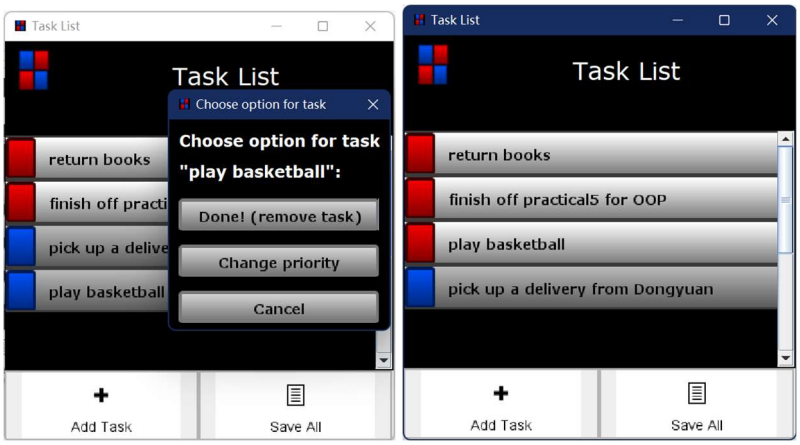
2.


Task List
Task List
new task*
return books
finish off practical5 for OOP
pick up a delivery from Dongyuan
play basketball
Add Task
Cancel
Task List
Task List
return books
finish off practical5 for OOP
new task
pick up a delivery from Dongyuan
play basketball
Add Task
Save All

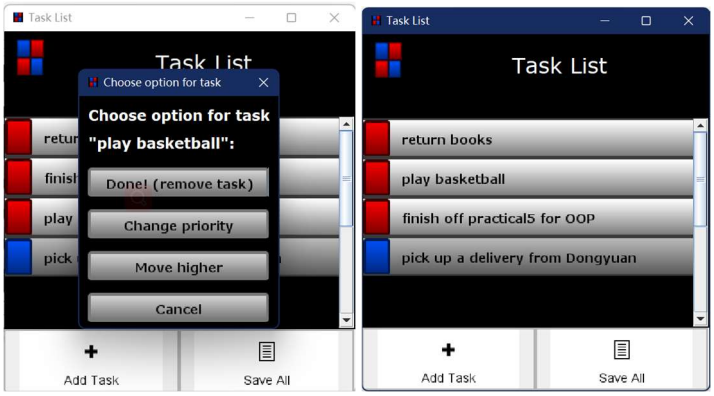
3.


Task List
Choose option for task
Choose option for task "new task":
Done! (remove task)
Change priority
Move higher
Cancel
Add Task
Save All
Task List
Task List
return books
finish off practical5 for OOP
pick up a delivery from Dongyuan
play basketball
Add Task
Save All

4.


Task List
Task List
Choose option for task
Choose option for task "play basketball":
Done! (remove task)
Change priority
Cancel
Add Task
Save All
Task List
Task List
return books
finish off practical5 for OOP
play basketball
pick up a delivery from Dongyuan
Add Task
Save All

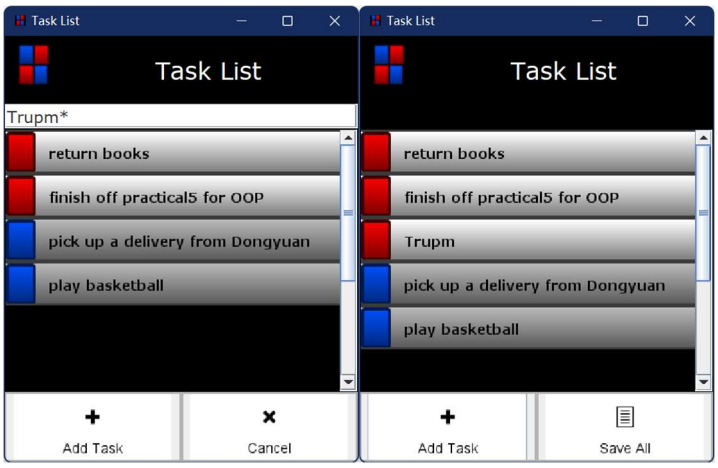
5.


Task List
Choose option for task
Choose option for task "play basketball":
Done! (remove task)
Change priority
Move higher
Cancel
Add Task
Save All
Task List
Task List
return books
play basketball
finish off practical5 for OOP
pick up a delivery from Dongyuan
Add Task
Save All

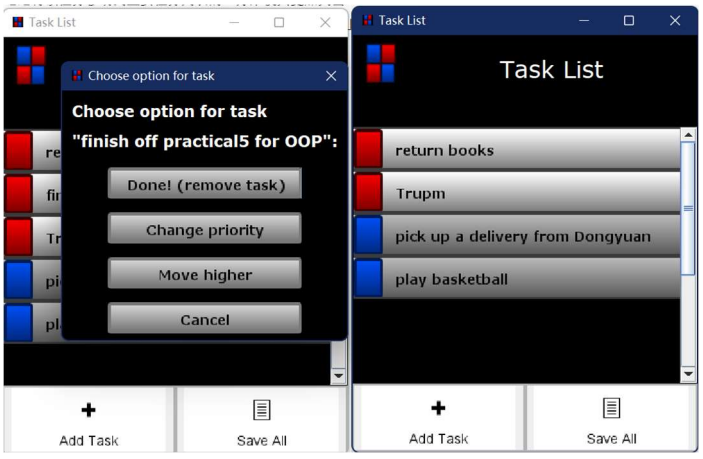
Testing 2 (Your own different test case-the first step is same, other three steps are different)

1.

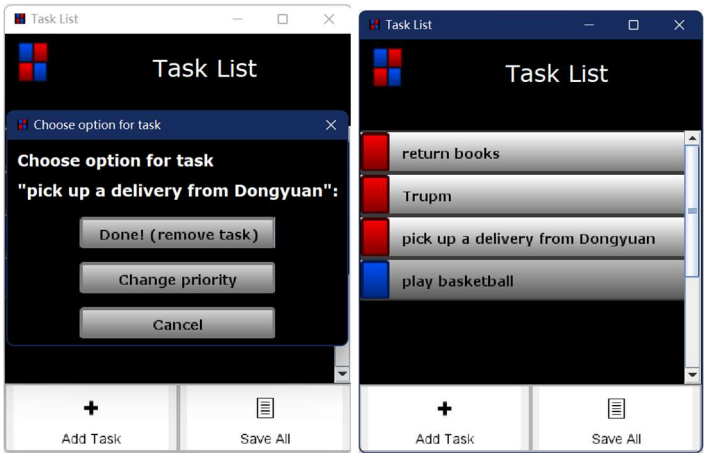
2.



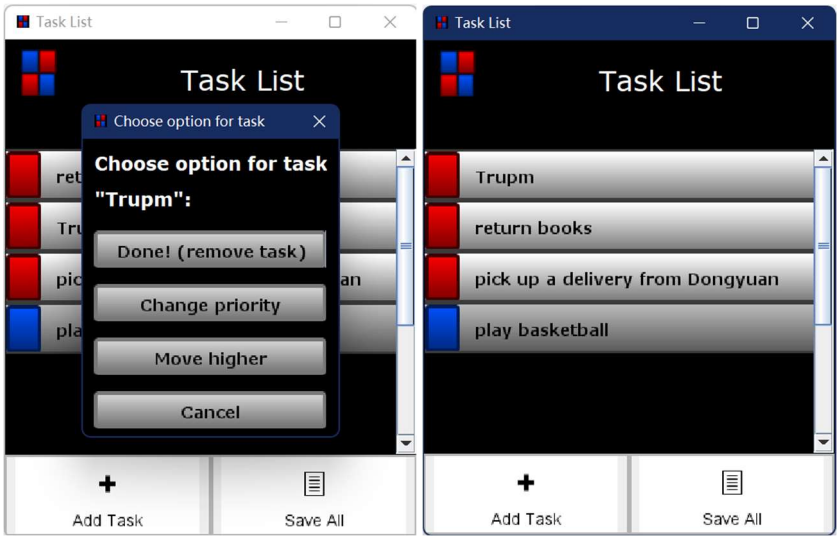
3.



4.


Task List
Task List
Choose option for task
Choose option for task
"pick up a delivery from Dongyuan":
Done! (remove task)
Change priority
Cancel
Add Task
Save All
Task List
Task List
return books
Trupm
pick up a delivery from Dongyuan
play basketball
Add Task
Save All

5.


Task List
Task List
Choose option for task
Choose option for task
"Trupm":
Done! (remove task)
Change priority
Move higher
Cancel
Add Task
Save All
Task List
Task List
Trupm
return books
pick up a delivery from Dongyuan
play basketball
Add Task
Save All