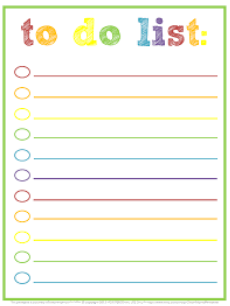
CS161: Introduction to Computer Science
Homework Assignment 7
Due: Wednesday 4/3 by 11:59pm

To Do Lists

In this assignment, you will implement a To-Do List application that allows the user to continuously (1) add items to the to do list, (2) remove an item from the to do list, (3) list all items in the to do list and (4) mark an item on the list as completed. Your program should loop continuously until the user chooses to quit. See the next page for an example of how your program should run.



I recommend using 3 Java classes for this assignment:

- A `ToDoItem` class that encapsulates all information related to a particular item on the list – e.g., it might contain a piece of text (the actual to do item), as well as keep track of whether the item has been completed or not.
- A `ToDoList` class that contains the actual list
- A `Controller` class that contains a `Scanner` and continuously prints the menu, reads in the user's choice, and calls the appropriate methods in the `ToDoList` class.

You will recognize this as being very similar to our `CoinGame` lab and very similar to the `BookCase` example. This is one of the more sophisticated assignments that you have done that must pull together your knowledge of writing classes, while loops, conditionals, and ArrayLists.

For your `ToDoList` class, I recommend having the following methods:

- An `addItem(String message)` – this method adds a new item to the list
- A `removeItem(int index)` – this method removes an item from the list
- A `markAsCompleted(int index)` – this method marks a given item as completed
- A `toString()` – this method returns a `String` representation of the list

When I run your program, I will enter invalid positions (e.g. I might try to remove the item at position -1 from the list). Your code should check that any integer input is valid. If it is not valid, then you should simply ignore the input and re-print the menu.

If the `ToDoItem` class seems unnecessary, you can get rid of it! In particular, you can keep track of all of that information inside the `ToDoList` class.

The `Controller` class should have the `main()` method and it must use a do-while loop and a switch statement to handle the input/output. So, to summarize, your program should:

1. Continuously print a menu that allows the user to add an item, remove an item, list all items, and mark an item as completed
2. Your `Controller` class should use a do-while loop and a switch statement
3. Your code should check for invalid integer inputs (e.g. if someone tries to remove the item at position -1 from the list)

Other than that, you are free to design your program as you wish.

Example of how your program should run:

```
========= To Do List =======
1. Add an item
2. Remove an item
3. List all items
4. Mark as completed
5. Quit
Choose an option (1-5): 1


Enter the item:
Finish cs homework

[Item 0] o Finish cs homework


========= To Do List =======
1. Add an item
2. Remove an item
3. List all items
4. Mark as completed
5. Quit
Choose an option (1-5): 1


Enter the item:
Lit. review for history paper

[Item 0] o Finish cs homework
[Item 1] o Lit. review for history paper


========= To Do List =======
1. Add an item
2. Remove an item
3. List all items
4. Mark as completed
5. Quit
Choose an option (1-5): 1


Enter the item:
Post response to psych reading

[Item 0] o Finish cs homework
[Item 1] o Lit. review for history paper
[Item 2] o Post response to psych reading


========= To Do List =======
1. Add an item
2. Remove an item
3. List all items
4. Mark as completed
```

```
5. Quit
Choose an option (1-5): 4


Enter the number of the item completed: 0

[Item 0] x Finish cs homework
[Item 1] o Lit. review for history paper
[Item 2] o Post response to psych reading


========= To Do List =======
1. Add an item
2. Remove an item
3. List all items
4. Mark as completed
5. Quit
Choose an option (1-5): 4


Enter the number of the item completed: 2

[Item 0] x Finish cs homework
[Item 1] o Lit. review for history paper
[Item 2] x Post response to psych reading


========= To Do List =======
1. Add an item
2. Remove an item
3. List all items
4. Mark as completed
5. Quit
Choose an option (1-5): 2


Enter the number of the item to delete: 1

[Item 0] x Finish cs homework
[Item 1] x Post response to psych reading


========= To Do List =======
1. Add an item
2. Remove an item
3. List all items
4. Mark as completed
5. Quit
Choose an option (1-5): 5

Goodbye!
```

(Notice that I used 'o' and 'x' to indicate whether an item has been completed or not.)

Extensions

Looking for additional challenges? Here are some interesting ways that you could extend this application:

1. Allow the user to specify a priority of low, medium, or high for each item on the list.
2. Add methods to the `ToDoList` class that allow the user to promote or demote an item on the list – i.e. to move an item up by one or down by one on the list. This would allow the user to order the list according to importance
3. Add methods to the `ToDoList` class that let the user print portions of the list – e.g., print all items that have high priority or print all items whose note contains a certain word. For example, I might want to print all the items on the list that contain the word "cs161" to see what I need to do for this course.
4. A surprisingly difficult thing to do is to remove all completed tasks from the list. This would require a loop to iterate over all items in the list. However, as you're iterating over the list you are also changing the list (i.e. removing items from the list). This is known as *concurrent modification* – you are concurrently traversing and deleting from the list. If you're interested in how to implement this functionality, let me know and I can give you some hints.

Style Guide

Before you submit your assignment, go through the checklist below and make sure your code conforms to the following style guide.

Checklist

- ☐ All unused variables are deleted
- ☐ All instance variables are used in more than one method (if not, make them local)
- ☐ Javadoc comment for all classes
- ☐ All methods have Javadoc comments (with appropriate `@param` and `@return` tags)
- ☐ All numbers have been replaced with constants (i.e. no magic numbers)
- ☐ Proper capitalization of variables, methods, and classes
- ☐ Use white space to separate different sections of your code

Submitting your homework assignment

You should submit your `hw7` folder with your Java classes inside. Please rename your folder with both of your first and last names *before* you zip it.