Task List List

CS 5004 Object Oriented Design

1. **Goals:**

* Explore an implementation of a linked list from scratch
* Use a linked list structure in an assignment
* Get more practice with appropriate class design
* Start your assignment from scratch

1. **In Recitation:**

The module associated with this assignment will take place over two weeks. Instead of having two lab assignments, I’m assigning 1 and another recitation activity. For your recitation, I’d like you to create a UML diagram of this assignment and make a game plan. Have your TA ok this design prior to leaving recitation.

Submit this UML diagram as part of your report and for your recitation check in.

1. **Instructions/Requirements:**

A to-do list is a logical thing to store in a linked list structure as you don’t want to limit the number of tasks it will hold. It’s also more memory efficient as it will release memory when it shrinks.

I’d like you to take what you learned from having to create a linked list and use that to represent a to do list. I’m leaving more of the details up to you on this assignment. Feel free to get creative. Just check the rubric to make sure you line up with the requirements.

**Objective 1** : Create supporting classes

A Task is something that you need to accomplish. A task has a

* int id (if needed)
* String description
* Date date
* Boolean completed
* Enum priority (you determine your priority system)
* You should have the ability to print
* all getters/setters and any other methods needed to support this class, but no more

A Date is a class that has:

* int day
* int month
* int year
* You should have the ability to print
* all getters/setters and any other methods needed to support this class, but no more
* protect all date values through either variable protection or by throwing exceptions for inappropriate values. For example, month should never exceed 12

**Objective 2**: Implementation

Create a to-do list implementation using a linked list structure you create. Through this class you should be able to

* add a task
* remove a task
* change the date on a task
* count all tasks, all completed tasks, all expired tasks
* remove all tasks, all completed tasks, all expired tasks
* print all tasks, all tasks of a specific priority, that have expired

I’m leaving a lot of this implementation up to you. You’ll also need to create whatever class objects you will need to support this list including any interfaces and node class objects. Make sure to consider objective 3 during your creation.

**Objective 3** : Design

* Make sure if this code was refactored to use a different structure other than a linked list this does not affect the other classes.
* Set it up so that date and task are replaceable with different date and task objects if you were to create them
* Abstract out everywhere you can, for example use predicate abstraction to print lists of specific priorities using a lambda expression
* Use recursion not iteration for all list traversal
* Abstract it out to the best of your ability

**Objective 4**: Testing

Choose your testing preference this time. You may use driver testing or JUnit testing, but make sure you do adequately test your implementation. If you use a driver, make sure to test all the functionality of your list and print the results after each manipulation.

Hints: How to tackle the assignment

First create a UML diagram for this project and consider what each piece will do. Then, start by creating empty interfaces and classes for your objects. Now select a method from the ADT and implement it end-to-end. Write tests for it before writing the implementation, and when your implementation passes your tests, move on to the next method.

1. **Extensions:**

Outside academics you will not get specific requirements. Each lab assignment is worth 100 points, but the base requirements will only get you to 85% - 90%. If you want an A, you’ll have to find a way to go above and beyond what is asked. I’ll often make some suggestions to you in this section, but it is entirely up to you what you’d like to add to the assignment. Make sure you know who your grader is and discuss extension expectations with them. You won’t have to do all of the extensions to get 100% credit.

Extension suggestions:

* Use driver and JUnit testing (5 - 10 depending on the quality and amount of testing)
* Add additional functionality not requested (5 - 10 per function depending on quality and complexity of function)
* Have clear well documented JavaDoc comments (1 or 2 points at this point)
* Bring in other design elements not required (5 - 10 points depending on complexity)
* Add the ability to check equality and use this as part of the application when identifying the same task (5 - 10 points)
* Add the ability to compare dates (this is advanced you would need to look into comparable objects and comparators) (5 - 10 points depending on how well its implemented)
* Create a driver that will manipulate the list adding the ability to do live testing. I would use a menu drive command line driver for this. (10 points)
* Test some of the design elements by creating alternate versions of the application that use a different storage mechanism or different date and task implementations (10 - 20 points)

1. **Report:**

Each assignment must include a short report. The generation of this report should take you no more than 15 minutes. This gives you a chance to reflect back on what you learned and it makes grading easier on your grader. For this report, I want the following sections:

1. Reflection (*What did you learn?)*
2. What do you think the advantages are of using a linked list in this application?
3. Do you think this could have been implemented without a linked list? Explain.
4. Extensions (*What extensions are you requesting?)*
5. Grading Statement (*Based on the rubric, what grade do you feel you deserve? Be honest.)*
6. *Academic integrity statement*
7. **Submission:**

Please read carefully. Failure to follow submission instructions can result in a reduced score.

Submit all files on Canvas under the appropriate assignment. Make sure to include the following named as follows:

Submit your files as a single zip file named: “Your Name”\_”Assignment”.zip

Your zip file should contain all files needed to create this project and no .class or metadata files.

Include your report as: “project\_report\_04.pdf”

Submission checklist:

* Did you include adequate comments?
* Did you include comment blocks at the top of each file?
* Did you name your files as requested?
* Does your code compile?
* Did you remove any package lines generated by your IDE?
* Did you take care of any warnings presented by your IDE?

|  |  |  |
| --- | --- | --- |
|  | **Possible** | **Given** |
| Objective 1 : supporting files | | |
| Task matches requirements | 5 | 0 |
| Date matches requirements | 5 | 0 |
| Objective 2 : Implementation | | |
| add a task | 5 | 0 |
| remove a task | 5 | 0 |
| change a task date | 5 | 0 |
| count all tasks, all completed, all expired | 8 | 0 |
| remove all completed, all expired, all | 8 | 0 |
| print all, specific priority, expired | 8 | 0 |
| Objective 3 : Design | | |
| Could be implemented with a different structure | 5 | 0 |
| Date and task replaceable | 8 | 0 |
| Abstraction used where appropriate | 8 | 0 |
| Objective 3 : Testing | | |
| Adequate testing done | 10 | 0 |
| Misc | | |
| Report | 5 | 0 |
| Code Quality (correct indentation, comment blocks, variable naming, etc) | 5 | 0 |
| Not included in total possible: | | |
| Linked list not created from scratch | -80 | 0 |
| Recursion not used | -100 | 0 |
| Does not compile | -100 | 0 |
| Extensions (Not calculated without report) | 10 | 0 |
| Late without pre-approved extension or LE | -100 | 0 |
| Creative or went above and beyond | 10 | 0 |
| Code contains warnings | -20 | 0 |
|  | |  |
| TOTAL POINTS POSSIBLE out of 100 | 90 | 0 |