**CSCI 2120**

**Homework Two – Expanding class “Task”**

This semester we are going to be building a task-tracking program called “TaskMaster”. Generally, with any non-trivial coding project, once you’ve mapped out the classes and objects you’re going to need to model your problem, coding typically begins with those low level classes that are simply “acted upon” (i.e. their methods are invoked elsewhere in the code, but they do not, themselves, invoke methods of any of your other classes). These classes are referred to as “pure servers” – they are acted upon, but do not act upon others…

Your second assignment, is to write a more flexible version of the idea of a Task, using inheritance.

**Submission:**

Create a new repository on gitlab.cs.uno.edu (see rubric) for these assignments for this course. Add your instructor as a developer to your repository. Create a new directory within your repository. Name this directory HW2. As you write the program, add, commit, and push your files to the remote repository on the GitLab server. When you have finished the assignment, make sure you’ve uploaded the most up to date version of your files (check the website to see).

**Assignment Statement:**

In the writing of your Task class, it occurs to you that at some point it would be good to be able to represent the idea that some tasks depend on other, sub-tasks, to be completed in order for the overall task to be complete. For example, the Task “Passing 2120” would require successful completion of Tasks “HW1”, “HW2” etc.

We’ll model this idea using inheritance. Turn Task into an abstract class that will serve as a building point for classes “SimpleTask” – which is basically a concrete version of “Task” and “CompoundTask” – a CompoundTask will “have” as components, other “Task” objects (which may themselves be either SimpleTasks or CompoundTasks. For now, let’s set an upper limit of 10 Tasks that a CompoundTask may be composed of – use an array for this.

**Task, SimpleTask, and CompoundTask implementation: (50%)**

* The “start time” and date of a compound task should be the earliest start time and date of its sub-tasks
* The “end time” from the previous assignment is really the concept of a “due date” – when they task SHOULD be finished (I may not have conveyed this well in the first assignment). That “due date” or should be the latest end time and date of its sub-tasks
* Now that you’ve changed “end date and time” to “due date and time” add another date/time combo representing the ACTUAL date/time of completion.
* Make sure you have a setter method to mark a Task as having been completed, passing in the actual completion date and completion time.
* A CompoundTask cannot be marked completed until its sub-tasks have been marked completed. Build in mechanisms to allow for completion marking of sub-tasks, and for checking that all sub-tasks have been completed before a CompoundTask can be marked completed.

**JavaDoc:** I expect extensive JavaDoc comments on this HW, with any necessary preconditions and postconditions explicitly stated (20%)

**Testing:** A JUnit Tester for SimpleTask and a JUnit Tester for CompoundTask are required and will constitute a large portion of your grade. (30%)

**Due Date**

You need to complete this assignment and have it pushed to the remote repository on the Git server by the date/time specified on Moodle.

There should be a README.txt file stating exactly how to compile, run, and test your code. In general IF there are bonuses, and you chose to do them, they should be separate (in subdirectories labeled bonus1, etc.) and fully complete implementations beyond the required one (in other words, start working from a fully implemented original copy in the subdirectory to do the bonus).