Computer Science 260

Homework 2: Designing a Program

**Due**: Fri. Feb. 15, at the beginning of class.

The following is the functional specification for a program. The purpose of the program is to maintain an automated "to-do" list.

----------------------------------------

**Toodle: An Automated To-Do List**

**Functional Specification**

A to-do list is a collection of tasks. A task has an identifier, a description, a status, and for some tasks, a date. The program will give each task a unique identifier (called the ID) when it is created. The task description is a string. The status of a task is either "incomplete" or "complete". New tasks are automatically given the "incomplete" status.

The task identifier is a positive integer, unique to each task, and is assigned by the program. The user will enter the description of a task when the task is created.

The user can change the status of a task to "complete" after the task is created by giving its ID. In addition to the above attributes, a completed task has a date that is set automatically to the current date when the task is completed.

When the program starts, it will attempt to open a file called "task\_list.txt". This file will contain the following information in this order:

1. the ID of the next task to be created;
2. the total number of tasks stored in the file; and
3. all information about all tasks that have been entered by the user.

Once the file has been opened, the program should read this information. If the file "task\_list.txt" does not exist, the program should execute as if there have been no previous tasks.

Once the program has read the file of tasks, the user will be prompted to enter the number of a command. The available commands and the number of each command are:

* Create a new task (1)
* Print all incomplete tasks. (2)
* Print all tasks. (3)
* Mark a task as complete (4)
* Quit the program (5)
* Print the command menu (6)

When the user selects command 5 (Quit the program), the program should write all of the information in the task list to the file "task\_list.txt" in the correct format so that it can be read the next time the program is run.

The following dialogs show the format of the output that your program must use for each command. User input is shown in bold. (Note, these screens do not necessarily represent a single execution of the program. They are to illustrate program display IO only.)

|  |
| --- |
| Please enter a command (1 – 6; 6 for help): **6**  1 – Create a new task  2 – Print all incomplete tasks  3 – Print all tasks  4 – Mark a task as incomplete  5 – Save the task list and exit the program  6 – Print this menu |
| Please enter a command (1 – 6; 6 for help): **1**  Enter task description: **Do laundry**  -- Task Created: ID is 17 –-  Please enter a command (1 – 6; 6 for help): **2**  ID Task  ------ ----  17 Do laundry  0 Study for Calculus Exam  3 Call Mom – ask for money  5 See doctor about growth |
| Please enter a command (1 – 6; 6 for help): **3**  ID Task Status Date  ------ ---- ------ ----  17 Do laundry Incomplete  0 Study for Calculus Exam Incomplete  3 Call Mom, ask for money Incomplete  4 Binge watch new show Complete 2/ 1/2019  11 Read chap 3 for 291 Complete 1/31/2019  12 Buy new shirt for interview Complete 2/ 2/2019  5 See doctor about growth on toe Incomplete |
| Please enter a command (1 – 6; 6 for help): **4**  Enter ID number of task: **5**  --- Task 5 marked as completed. --- |
| Please enter a command (1 – 6; 6 for help): **3**  ID Task Status Date  ------ ---- ------ ----  17 Do laundry Incomplete  0 Study for Calculus Exam Incomplete  3 Call Mom, ask for money Incomplete  4 Binge watch new show Complete 2/ 1/2019  11 Read chap 3 for 291 Complete 1/31/2019  12 Buy new shirt for interview Complete 2/ 2/2019  5 See doctor about growth on toe Complete 2/11/2019 |
| Please enter a command (1 – 6): **5**  Saving task list ...  Goodbye |

**What to do:** Your job in this homework is to turn this functional specification into an object-oriented design. Design the program in three steps, as discussed in class:

1. Start by identifying the classes that will be used by the program. Remember that you are not worried about the implementation here – try to identify everything in the spec that might be represented by a class.

You do not have to include any classes that the program uses that come from the java.lang or java.util packages. However, if your design uses any other built-in classes, they should be included in your design.

1. Take the classes you identified in part 1, and then identify the public and protected methods of each class. You do not have to list any methods for the classes in your design that come from the Java API. Do not list any private methods either, since these will not be known until the implementation.
2. Identify the public instance and class variables for each class. Again, you do not have to list any variables from classes in the Java API.

After you have identified the classes, methods and the variables of each class, identify the relationships between the classes in your design. Use only the dependency, aggregation, inheritance, and interface implementation relationships.

**What to turn in:** When you have completed your design, use the Dia tool to create a neat UML class diagram that represents your design. Print out a hardcopy of your diagram to submit.