Overview

The goal of this assignment is to create a simple Java application to manage your “to-do lists”. Every to-do item has a unique title. In addition, your application should support three specific types of to-do items: tasks containing a priority (either “URGENT”, “IMPORTANT”, or “SOMEDAY”); meetings containing a location and time; and deadlines containing a due date. Your to-do items will be stored and managed by a to-do planner. The planner will allow you to add, remove, and retrieve to-do items. You'll compile and run your application using the commands “javac TodoPlannerTextUI.java” and “java TodoPlannerTextUI”. The TodoPlannerTextUI is a helper class that implements a text-based user-interface for your planner application. An example run of your program should look something like this:

Welcome to the Todo Planner!

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

Main Menu:
1. Create a Task
2. Create a Meeting
3. Create a Deadline
4. Remove a Todo Item
5. Display Todo Items
6. Exit
Choice: 1

Please enter a task title: Haircut
Please enter a priority (1-Urgent, 2-Important, 3-Someday): q
Oops, please try again: 2
Added Task:
   Haircut (Priority: Important)

Main Menu:
1. Create a Task
2. Create a Meeting
3. Create a Deadline
4. Remove a Todo Item
5. Display Todo Items
6. Exit
Choice: exit
Oops, please try again: 2

Please enter a meeting title: Group project user stories
Please enter a meeting location: CS Lab
Please enter a meeting time: Jan. 29 at 3:30pm
Added Meeting:
   Group project user stories (Location: CS Lab, Time: Jan. 29 at 3:30)

Main Menu:
1. Create a Task
2. Create a Meeting
3. Create a Deadline
4. Remove a Todo Item
5. Display Todo Items
6. Exit
Choice: 6

Goodbye!

You can treat the time format as a plain string (for meetings and deadlines). If the user enters a to-do item that has the same title as one already entered, then you should tell the user that the title already exists and
prompt for a new title. Display the to-do items in an ordered list, e.g.,

1. Haircut (Priority: Important)
2. Group project user stories (Location: CS Lab, Time: Jan. 29 at 3:30)

For removing an item, first display all the items and then prompt the user for an item number to remove. Your prompts, when appropriate, should check that valid input was entered.

**Instructions**

You will need to create six classes to implement your program: `TodoPlannerTextUI`, `TodoPlanner`, `Todo`, `Task`, `Meeting`, and `Deadline`. The `TodoPlanner` holds and manages a set of to-do items. The `Todo` class holds a title. The `Task`, `Meeting`, and `Deadline` classes each extend `Todo` with additional fields. Finally, the `TodoPlannerTextUI` is the application class (i.e., it holds the `main` method) and should handle all of the user-interface functionality for the `TodoPlanner` class. You should not include any user interface code in any of the classes other than `TodoPlannerTextUI`.

Test your program to ensure it works correctly. Make sure you try a sufficient number of cases, and provide the results of your test runs. Be sure to format and document your source code according to the *The Java Elements of Style* (see the style guide). Create a directory named `hw2`, place your class files in it, and submit this directory using the `submit` command on ada. Hand in hard-copy consisting of a cover sheet, your design document, your source code, and your test runs.

**Extra Credit**

Instead of handling days and times as plain strings, use the `GregorianCalendar` class and associated helper classes (e.g., `DateFormatSymbols`) to represent and display these values. Prompt the user to enter the time using a specific format (e.g., “YYYY/MM/DD HH:MM”), parse their answer to construct a `GregorianCalendar` object, and then “pretty print” this object (e.g., as “Jan 29, 2010 at 3:30pm”) when displaying a meeting or deadline time. The book gives information on how to do most of this as well as the online Java API.