EIT
Te Whare Takiura o Kahungunu

School of Computing
Te Kura Rorohiko

Bachelor of Computing Systems

ITPR5.500 Introduction to Programming

Assignment 2 - Semester 2 2015

Date due: 6 November 2015 @ 5pm
Weighting: 40%
Marks: 100

Instructions:
Students are to attempt all tasks in the assignment.

Marks will be awarded for logic design, efficient use of code, creativity, and documentation. The completed application will be submitted in soft copy and should include all documentation as specified in this assignment. Keep a backup copy at least until you have your mark.

This assignment must be delivered by 5pm on the due date, and must have been presented during a lab session in the same week.

Assignment 2

MyProjectManager

Problem description

You need to create project manager that consists of tasks and features to manage these tasks. The implementation is completely done in Python, and no other resources are needed.

The MyProject object can contain multiple MyTask objects. MyProject also contains methods for adding, removing, showing, and accessing its MyTask objects. Each MyTask object contains a title, the project member's name who is responsible for this task, and a duration that indicates how many hours this task will take to complete. MyTask objects also contain methods for setting the title, name, and duration, changing the name and duration, and undoing the last modifications to the name and duration.
You need to implement the MyProject object with its MyTask objects, and a simple application that allows the user to interface with MyProject objects. More details per feature are given below.

Requirements

The MyProject object and the contained MyTask objects will be created using classes. The methods in these classes implement their features. These programming concepts will be discussed extensively during class sessions. Many resources are available online as well; docs.python.org is a good starting point.
Below are the details you need to implement at least. You will need to implement more methods to complete all requirements.

The MyProject object contains a list of MyTask objects, a method to add another MyTask object, a method to remove a MyTask object by its title, a method to return the list of task titles, and a method to access a MyTask object by its title. All these features are local to the MyProject object.

> **myTasks**: this is a list only available to the MyProject object, and is initially empty. Its role is to contain MyTask objects.
>
> **add()**: this is a method that requires a MyTask object as a parameter. This method will add the given MyTask object to the myTasks list.
>
> **getTaskTitles()**: this is a method that requires no parameters. The method will return the task titles it currently contains.
>
> **remove()**: this is a method that requires a task title as a parameter. It will then find the MyTask object in myTasks with the given title, and remove it from the list. Implement and comment on a solution in case the title cannot be found when you call this method.
>
> **getMyTask()**: this is a method that requires a task title as a parameter. It will then return the MyTask object in myTasks with the given title. Implement and comment on a solution in case the title cannot be found when you call this method.

A MyTask object contains a task title, an project member's name, and a duration in hours. You may assume that the task title is unique (i.e. there will only be one task with this title). A MyTask object further contains methods to initialise the task, to assign the task to someone else, to change the duration, and to undo the previous changes. All these features are local to a MyTask object.

> **title**: this is a variable that contains the task's title. Initially the title is "<Title of task>".
>
> **memberName**: this is a variable that contains the name of the project member who is responsible for this task. Initially the name is "<Member name>".
>
> **duration**: this is a variable that contains the duration in hours it takes to complete this task. A duration can therefore not be negative, except when it is set initially to -1 meaning "no duration is set".
>
> **setTaskValues()**: this is a method that requires a task title, a project member's name, and a duration as parameters. It will then set title to the given task title, memberName to the given project member's name, and duration to the given duration of the task. You do not need to verify that this title is unique, but you do need to validate the value of duration. Comment on the solution that receives invalid data.
>
> **assignTo()**: this is a method that requires a project member's name and reassigns this task to the new name. It returns true when this is succesfully changed; it returns false if the given project member's name is already registered for this task.
>
> **updateDuration():** this is a method that requires a positive or negative number of hours, and adds or substracts these hours from the current duration. If the updated duration is valid, the method returns true; if the updated duration is not valid, the update is ignored and false is returned.
>
> **undoAssignTo()**: this is a method with no parameters. It will undo the result of the last assignTo() call (it will set the name back to what it was before the last change).*
>
> **undoUpdateDuration():** this is a method with no parameters. It will undo the result of the last updateDuration() call (it will set the duration back to what is was before the last change).*

(*) You only need to implement a solution for the last change, and you do not need to implement a solution that reverts multiple changes. When the name or duration has been changed (again), it can be undone.

The application allows a user to use your MyProject implementation. The user interaction can be completely text based. It is optional to advance this interface to a graphical user interface (GUI). The user can then choose from the features you have in MyProject and MyTask.

Use only the Python interpreter and use only your own code. Do not copy code from someone else or the internet. Evidence of copied code will be treated as plagiarism.

The application must be easy to use in all respects. The program logic should be straightforward and not unnecessarily complex. The code should be tidy and have the proper comments attached to the appropriate code blocks, explaining the idea/design of the code block, and how it relates to other parts of your code. You may assume the reader can read code, so provide value-add in your comments.

Delivery

You need to discuss your progress on this assignment with your lecturer **every week**. This is typically done during the lab sessions. If you cannot make it to a lab session, make sure you still discuss your work with your lecturer in that week. These discussions are part of the marking schedule.
During the lab session in the week this assignment is due, you have to demonstrate your application. The lecturer may then ask you questions about your code. (And you're expected to answer these).

Hand in

- The Python file that contains your assignment implementation.

You must deliver your assignment through the EITOnline course website which contains a link called "Assignment 2 Upload Link". You will upload a single file containing all code as described above. Keep a copy of your solution until you have your mark.

Marking Guide

❑ Material handed in as specified		/ 5
❑ MyProject class implementation		/ 25
❑ *all features implemented according to specification*	*/ 10*	
❑ *code is straightforward, avoiding overhead*	*/ 10*	
❑ *coding conventions applied*	*/ 5*	
❑ MyTask class implementation		/ 25
❑ *all features implemented according to specification*	*/ 10*	
❑ *code is straightforward, avoiding overhead*	*/ 10*	
❑ *coding conventions applied*	*/ 5*	
❑ Application using object features		/ 25
❑ *all features available through the user interface*	*/ 10*	
❑ *code is straightforward, avoiding overhead*	*/ 10*	
❑ *coding conventions applied*	*/ 5*	
❑ Weekly discussions		/ 10
❑ Final demonstration		/ 10

TOTAL _____ / 100