# Assignment 2 and 3: Classes in C# - School Task Management System

Objective:

The aim of this assignment is to guide students in the application of object-oriented programming principles by developing a School Task Management System in C#. Each class must be meticulously designed, incorporating proper encapsulation, and relationship management. Provided class diagram must be adhered to exactly.

# **Please observe the General assignment requirements outlined in the document on e-centennial**

Also make sure you are not using string.Join() method() and that your code does not include any comments other then your name and student id. You code should be readable without any comments. Any method with comments will be marked with 0.

## TaskManager Class

A screenshot of a computer

Description automatically generated

[10] Design a TaskManager class responsible for managing tasks. Implement all and only the members in the class diagram.

[2] Include a static list of tasks (List<Task>) to store all created tasks.

[10] Implement a static constructor to initialize the task list.

[5] Provide a property Tasks to allow external access to the list.

[10] Implement a method CreateTask(string) that creates a new task, adds it to the list, and returns the task instance.

[10] Create a method AllTasksToString() to generate a formatted string displaying all tasks.

## Course Class

A screenshot of a computer

Description automatically generated

[10] Design a Course class to represent a course. Implement all and only the members in the class diagram.

[8] Include private fields for course-related information: name, code, semester, id.

[4] Include private lists for evaluations (List<Evaluation>) and tasks (List<Task>).

[35] Implement public properties for Code, Semester, Id, and Name with appropriate accessors to the fields.

[10] Implement public read only property Grade that returns the sum of products of weight and grade for each evaluation.

[5] Create a property Evaluations to allow external access to the list of evaluations.

[5] Create a property Tasks to allow external access to the list of tasks.

[10] Implement a method AddEvaluation(EvaluationType type, byte weight, string name) to add an evaluation to the course.

[10] Implement a method AddTask(string description) to add a task to the course’s list of tasks.

[10] Create a method TasksToString() to generate a formatted string displaying all tasks associated with the course.

[10] Create the ToString method to provide a formatted string representation of the course (do not include the tasks).

## Evaluation Class

A screenshot of a computer

Description automatically generated

[10] Design an Evaluation class to represent an evaluation within a course. Implement all and only the members in the class diagram.

[18] Include private fields for id, type, name, dueDate, weight, grade, course, tasks, and textFile.

[45] Implement properties for Type, Name, DueDate, Weight, Grade, Tasks, Course, TextFile, and Id to provide external access to fields.

[10] Create a constructor that takes a Course, EvaluationType, and weight as parameters to initialize the evaluation object. The value for its course field after the object is created is the reference to course that has the evaluation.

[10] Implement a method AddTask(string description) to add tasks to the evaluation’s list of tasks.

[10] Create a method TasksToString() to generate a formatted string displaying all tasks associated with the evaluation.

[10] Override the ToString method to provide a formatted string representation of the evaluation course (do not include the tasks).

## MyDay Class

A screenshot of a computer

Description automatically generated

[10] Design a MyDay class to represent a day and manage daily tasks. Implement all and only the members in the class diagram.

[4] Include private fields for todaysTasks (List<Task>) and date (DateTime).

[4] Implement a private static instance currentDay to keep the only instance of the class.

[10] Create a private constructor that initializes the date to the current date and the list of tasks to an empty list.

[10] Implement a public method NewDay() to retrieve the current day instance or create a new one if it's a new day.

[10] Include a property TodaysTasks for external access to the list of tasks.

[10] Implement a method AddDayTask(Task task) to add tasks to the current day.

[10] Implement the ToString method to provide a formatted string representation of tasks for the current day.

## Task Struct:

A screenshot of a task

Description automatically generated

[10] Design a Task struct to represent an individual task. Implement all and only the members in the diagram.

[10] Include public fields for Description, DueDate, and private field for Done.

[10] Implement a constructor that takes a description as a parameter and initializes DueDate to DateTime.MaxValue and Done to false.

[5] Include a property Done to indicate the completion status of the task.

[10] Override the ToString method to provide a formatted string representation of the task.

## EvaluationType enumeration

A screenshot of a computer

Description automatically generated

[10] Design a EvaluationType enumeration to represent possible types of evaluations. Implement all and only the members in the diagram.

## Test harness

[100] Demonstrate the functionality of the Task Management System in the Main method by doing the following:

1. Create 2 courses and output them to the console.
2. Add 2 assignments, quiz, and test to the first course.
3. Add 3 assignments, and test to the second course.
4. Output both courses to the console
5. Set Due date for the first assignment of the first course.
6. Set grade for the first evaluation of the first course.
7. Set grade for the second evaluation of the first course.
8. Output the first course to the console.
9. Add the task to the first course.
10. Set a due date for created task.
11. Add another task to the first course.
12. Output the tasks for the course.
13. Add a new task to the second evaluation of the first course.
14. Output the tasks for the second evaluation of the first course.
15. Create new MyDay.
16. Mark one of the tasks done.
17. Add any two tasks you created to the day.
18. Output tasks in MyDay.

## Submission Guidelines:

Submit your solution as compressed solution folder in Lab 23 drop box by the decline in the drop box.

Include a brief written explanation of design choices, challenges faced, and lessons learned during implementation. Submit the explanation as a word document. Place the word document in the solution folder together with the solution file.