# Assignment 4 and 5: Serialization and Specification in C# - School Task Management System

Objective:

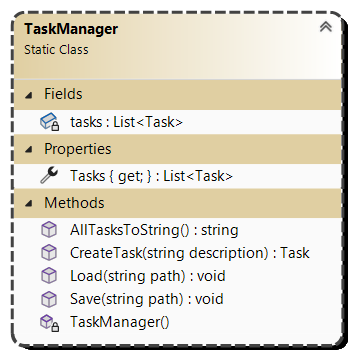
# The objective of this assignment is to extend the existing School Task Management System to include new features and improvements, promoting a more robust and versatile system. The modifications focus on introducing an Assignment and Quiz class, refining existing classes, and incorporating JSON serialization for persistence. The class library will be created as well.

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

## Class Library

[100] Move all the classes to a class library (.NET Framework) except Program class that contains main() method. Name the class library in the following format YourFirstName\_StudentId\_Lib ( for example: jake\_300123456\_lib).

## TaskManager Class

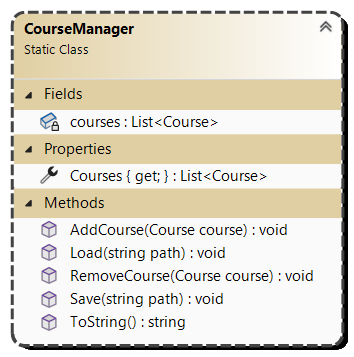


[10] Modify a TaskManager class that is responsible for **managing all** tasks. Implement all and only the members in the class diagram.

[30] Introduce new methods (Save and Load) for saving and loading tasks into JSON to implement persistence capabilities.

The other members are not changed.

## CourseManager Class



[10] Design a static ClassManager class responsible for managing courses. Implement all and only the members in the class diagram.

[2] Include a static list of courses (List<Course>) to store courses.

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

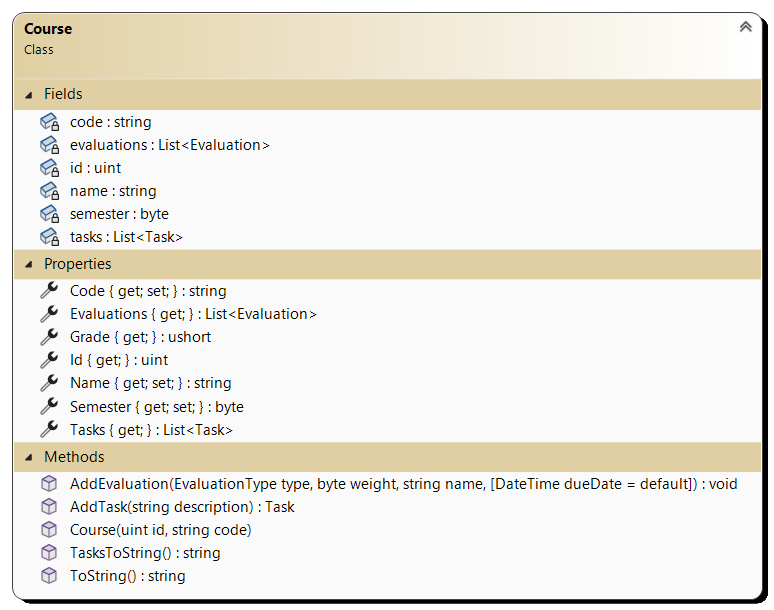
[10] Implement a method AddCourse(Course course) that adds a new course to the list.

[10] Implement a method RemoveCourse(Course course) that removes a course from the list.

[10] Create a method ToString() to generate a formatted string containing all courses.

[30] Implement new methods for saving and loading courses using JSON for implement data persistence.

## Course Class



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

[8] Make sure that fields task and evaluations are included in JSON sterilization and corresponding properties are excluded.

[10] Modify method AddEvaluation(EvaluationType type, byte weight, string name) to include optional parameter dueDate (DateTime dueDate = default). The method will do the following:

a. If the evaluation type is assignment, it will create an instance of Assignment class.

b. If the evaluation type is quiz it will create an instance of Quiz class.

c. If it is any other evaluation type it created an instance of Evaluation class.

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

## Evaluation Class

A screenshot of a computer

Description automatically generated

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

[4] Make *type* field read only.

[4] Make Type property read only.

[2] To avoid circular reference when serializing, exclude Course property from serialization.

[20] Add property EvaluationText that return the text in the file specified by textFile field.

## Assignment Class

## 

[10] Design Assignment class, derived from the existing Evaluation class. Implement all and only the members in the class diagram.

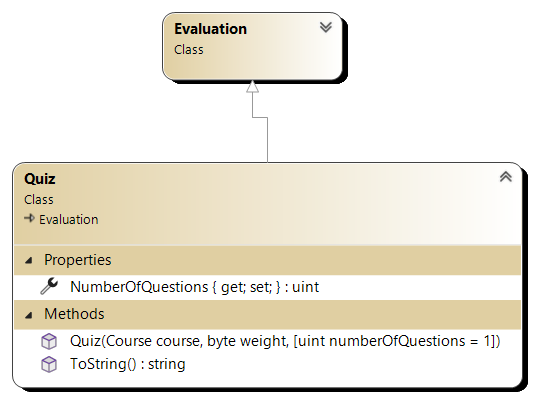
[3] Introduce a property, IsGroupAssignment, indicating whether the assignment is for a group or an individual.

[15] The constructor Assignment(Course course, byte weight, DateTime dueDate=default, bool isGroupAssignment=false) has two optional parameters and it sets the values to the appropriate properties.

[15] Method AddTask(string description) creates and returns a task with description provided. It also sets a due date to be the same as the due date of the Assignment object that it is added to.

[10] ToString() method returns the string representing Evaluation and includes if it is group assignment or not. (see the output for format)

## Quiz Class



[10] Design Quiz class, derived from the existing Evaluation class. Implement all and only the members in the class diagram.

[3] Introduce a property, NumberOfQuestions, representing number of questions on a quiz.

[15] The constructor Quiz(Course course, byte weight, uint numberOfQuestions=1) has one optional parameter and it sets the value to the appropriate property.

[10] ToString() method returns the string representing Evaluation and includes the number of questions. (see the output for format)

## MyDay Class

A screenshot of a computer

Description automatically generated

This type has not changed.

## Task Class:

A screenshot of a task

Description automatically generated

[10] Modify Task struct to be a class. This class represents an individual task. Implement all and only the members in the diagram.

[8] Include public properties for Description, DueDate, IsDone, and Id. Id has a private setter.

[10] Implement a constructor Task(string description) that takes a description as a parameter and initializes DueDate to DateTime.MinValue, IsDone to false, Id to a new Guid and sets description to provided value.

## EvaluationType enumeration

A screenshot of a computer

Description automatically generated

This type has not changed.

## Test harness

Create Console app(.NET Framework) that is utilising the class app that is created.

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

1. 1. Create two courses

2. Add courses to the course manager

3. Write content of course manager to the console

4. Add 2 assignments to the first course with due dates in 14 and 7 days respectively

5. Add quiz, and test to the first course

6. Add test to the first course

7. Add 2 assignments to the second course with due dates in 10, 5, and 6 days respectively

8. Add test to the first course

9. Set TextFile for the first assignment of the first course to 'Evaluations\COMP123\Assignmnet1.txt'

10. Write content of the evaluation text for the first assignment of the first course

11. Write content of course manager to the console

12. Add grade to the first assignment of the first course

13. Write content of course manager to the console

14. Add grade to the second assignment of the first course

15. Write content of course manager to the console

16. Add task 'Read chapter 2' to the first course with due date in 6 days

17. Add task 'Read chapter 3' to the first course

18. Output tasks for the first course to the console

19. Add task 'Start assignment' to the first assignment of the first course

20. Add task 'Write main() method' to the first assignment of the first course and mark it done

21. Output all task for the first evaluation of the first course to the console

22. Create MyDay and add two tasks

23. Output the MyDay to the console

24. Save content of the course manager to courses.json

25. Save content of the task manager to tasks.json

26. Load content to the course manager from courses.json

27. Load content to the task manager from tasks.json

28. Output a separator

29. Write content of course manager to the console

30. Write all tasks in the task manager to the console

The output for above test harness should look similar to the following one provided in output45.txt file.

## Submission Guidelines:

Submit your solution as compressed solution folder in Lab 45 drop box by the deadline 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.