C4: C# Object-Oriented Programming

This course bonus helps you learn in a more meaningful manner.

Section 2: Understanding C# Class

ACTIVITY: Assignment 1

Goal:
Check your understanding of C# Class and C# Objects by writing C# code for the given problem statements.

Duration:
5 mins

Software needed:
Visual Studio Code, Visual Studio, Visual Studio for Mac, or any other IDE that supports C#.

Problem Statements:

1. For the CMS system, define the Course type in a new file Course.cs, in CMS.UI.Models assembly.
2. Create an object instance named computerScience of type Course class in CMS.Application assembly.
3. Create CourseSubject type in a new file CourseSubject.cs in CMS.UI.Models assembly.

Additional Resources:

- Use the C# Extension to right-click the folder and select the New C# Class option.
- You can complete the above assignment in the “CMS” project under the “Section02-Class” folder.
- In case you haven’t done yet, download the source code or clone the GitHub repository as mentioned in the “Download Source Code” lecture in Section 1.

Evaluate:

- Execute your program and check the output mentioned in the problem statement above (if applicable).
- You can cross-refer my solution for this assignment provided in the downloaded source code under the Assignment1Solution folder within the “Section02-Class” folder.

Section 3: Understanding C# Class Fields

ACTIVITY: Assignment 1

Goal:

Check your understanding of C# Class Fields by writing C# code for problem statements.

Duration:

10 mins

Software needed:

Visual Studio Code, Visual Studio, Visual Studio for Mac, or any other IDE that supports C#.

Problem Statements:

1. Add the following fields for the Course class: create an instance of the Course class, assign the field values, and print the same to the console.
    1. CourseId of type integer with public access.
    2. CourseName of type string with public access.
1. For the CourseSubject class, add the following fields:
    1. Id of type integer with public access
    2. SubjectName of type string with public access
2. Add a static field to store MaxSubjects in the Course class and assign a default value of 8.

Additional Resources:

- You can complete the above assignment in the **CMS** project under the **Section03-ClassFields** folder.
- In case you haven't done yet, download the source code or clone the GitHub repository as mentioned in the **Download Source Code** lecture in Section 1.

Evaluate:

- You can cross-refer my solution for this assignment provided in the downloaded source code under the Assignment1Solution folder within the **Section03-ClassFields** folder.

Section 4: Understanding C# Class Methods

ACTIVITY: Assignment 1

Goal:
Check your understanding of C# Class Methods by writing C# code for the problem statements.

Duration:
15 mins

Software needed:
Visual Studio Code, Visual Studio, Visual Studio for Mac, or any other IDE that supports C#.

Problem Statements:

1. For the Course class, add the following:
    a. A private field “subjects” of type List<CourseSubject> to store the list of subjects for the course.
    b. Expose a public property Subjects with only a getter that returns this “subjects” list.
    c. AddSubject method, which accepts CourseSubject as a parameter that returns none. It should add the subject to the subjects list.
    d. RemoveSubject method, which accepts CourseSubject as a parameter and returns none. It should remove the course subject from the subjects list.
2. For the Course class, add an overloaded AddSubject() method that accepts a list of CourseSubjects and adds them to the “subjects” collection.

Additional Resources:

- You can complete the above assignment in the “CMS” project under the “Section04-ClassMethods” folder.
- In case you haven’t done yet, download the source code or clone the GitHub repository as mentioned in the “Download Source Code” lecture in Section 1.

Evaluate:

- You can cross-refer my solution for this assignment provided in the downloaded source code under the Assignment1Solution folder within the “Section04-ClassMethods” folder.

Section 5: Understanding C# Interfaces

ACTIVITY: Assignment 1

Goal:
Check your understanding of C# Interfaces by writing C# code for problem statements.

Duration:
15 mins

Software needed:
Visual Studio Code, Visual Studio, Visual Studio for Mac, or any other IDE that supports C#.

Problem Statements:
For the Student class, do the following:

1. Make it implement the IStudent interface.
2. Move FirstName and LastName fields to the IStudent interface as properties with a getter and a setter.
3. Implicitly implement FirstName and LastName properties in the Student class.
4. Provide default implementation for the GetFullName method in the IStudent interface that returns FirstName and LastName together. Once done, delete the existing implementation for that method from the Student class.
5. Create a new instance of IStudent in CMS.Application project, assign values for FirstName and LastName, then print the result of GetFullName to the console.
6. Create a static method WhoAmI() in the IStudent interface that returns “Student” as the value. Access this static method from the console application and print the result.

Additional Resources:

- You can directly complete the above assignment in the “CMS” project under the “Section05-Interface” folder.
- In case you haven’t done yet, download the source code or clone the GitHub repository as mentioned in the “Download Source Code” lecture in Section 1.

Evaluate:

- You can cross-refer my solution for this assignment provided in the downloaded source code under the Assignment1Solution folder within the “Section05-Interface” folder.

Section 6: Understanding C# Inheritance

ACTIVITY: Assignment 1

Goal:
Check your understanding of Inheritance by writing C# code for the given problem statements.

Duration:
15 mins

Software needed:
Visual Studio Code, Visual Studio, Visual Studio for Mac, or any other IDE that supports C#.

Problem Statements:
For the Staff class, do the following:

1. Make it derive from the Person class and remove the properties from the Staff class that is already defined in the Person class.
2. Add *WorkingHoursPerWeek* as an additional property to the Staff class.
3. Add the Staff(string firstName, string LastName) constructor and pass the values to the corresponding Person class constructor. Also, assign 40 hours to the *WorkingHoursPerWeek* property in this constructor.
4. Add an appropriate access modifier to the CalculateFees method so that it is accessible by any derived type of Staff class. Similarly, make the UpdateInfo method accessible by any derived type of Staff class but within CMS.UI.Display assembly only.
5. In the console application, CMS.Application, create an instance of Staff class initialized with "Mary Smith" name and assign it to Person instance *person2*. Next, print its *WorkingHoursPerWeek* value to the console only if person2 is of type Staff.

Additional Resources:

- You can complete the above assignment in the "CMS" project under the "Section06-Inheritance" folder.
- In case you haven't done yet, download the source code or clone the GitHub repository as mentioned in the "Download Source Code" lecture in Section 1.

Evaluate:

- You can cross-refer my solution for this assignment provided in the downloaded source code under the Assignment1Solution folder within the "Section06-Inheritance" folder.

Section 7: Understanding C# Polymorphism

ACTIVITY: Assignment 1

Goal:
Check your understanding of Polymorphism by writing C# code for the problem statements.

Duration:
15 mins

Software needed:
Visual Studio Code, Visual Studio, Visual Studio for Mac, or any other IDE that supports C#.

Problem Statements:
For the ElectronicsCourse class, do the following:

1. Make it derive from the Course class and remove the members from it that are already defined in the Course class.
2. Add the *AddSubject* method that hides the corresponding implementation from the Course class.
3. Add virtual to the *RemoveSubject* method in the Course class and override the same in the ElectronicsCourse class.
4. Create an electronicsCourse instance for ElectronicsCourse class in the console application. Call *AddSubject* with a new course subject instance with id as 401 and subjectName as "Basics of Electrical Science". Then call the *RemoveSubject* method to remove the previously added course subject. Run the program and check whether the appropriate method from a base class or derived class gets triggered.

Additional Resources:

- You can complete the above assignment in the "CMS" project under the "Section07-Polymorphism" folder.
- In case you haven't done yet, download the source code or clone the GitHub repository as mentioned in the "Download Source Code" lecture in Section 1.

Evaluate:

- You can cross-refer my solution for this assignment provided in the downloaded source code under the Assignment1Solution folder within the "Section07-Polymorphism" folder.