Introduction

After a week of rigorous coding, Welcome back!

You have learned all about the C# in the previous labs and manuals. Let's move on to the next, new, and interesting concepts.

Students, Object-Oriented Programming is different from Procedural programming as it is about creating objects that contain both data and methods.

Let's do some coding.

Class Declaration

We have learned in the previous manual about the basic code that visual studio provides and programmers start the work from the main function directly.

Syntax:

```
class class_name
{
        // class_members
}
```

This code is written outside the "main function" and inside the "class program" and it creates a new class in the program. To understand this concept, try writing the following program.

Task: Write a program that creates a new class of students.

Solution:

Write the following code before the main function of the code and execute the program by clicking on the start button.

Code:

```
1   using ...
6
7   namespace Test
8   {
        0 references
9       class Program
10      {
11          class students
12          {
13              public string name;
14              public int roll_no;
15              public float cgpa;
16          }
            0 references
17          static void Main(string[] args)
18          {
19              Console.Read();
20          }
21      }
22  }
```

The code will generate a new class of students where each student would have the following properties.

- string type Name
- int type Roll Number
- float type CGPA

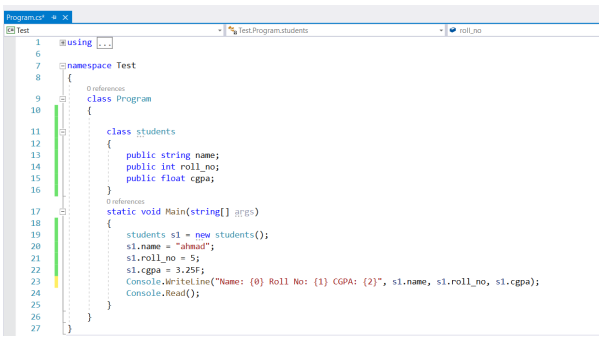
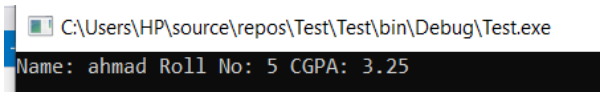
It can include many other properties however, for simplicity these work with these three characteristics at the time.

Now, in order to create a "new object" of class students, we will declare a class type object in the main function.

students s1 = new students();

This line will "create a **new object** of class **students**" having the above-defined properties. To understand this concept, try assigning values to the s1 variable.

Task: Write the code to assign values to a class object.
Solution:
Write the following code into the main function of the code and execute the program by clicking on the start button.

Code:	**Output:**
	

```
using ...

namespace Test
{
    class Program
    {

        class students
        {
            public string name;
            public int roll_no;
            public float cgpa;
        }
        static void Main(string[] args)
        {
            students s1 = new students();
            s1.name = "ahmad";
            s1.roll_no = 5;
            s1.cgpa = 3.25f;
            Console.WriteLine("Name: {0} Roll No: {1} CGPA: {2}", s1.name, s1.roll_no, s1.cgpa);
            Console.Read();
        }
    }
}
```

```
C:\Users\HP\source\repos\Test\Test\bin\Debug\Test.exe
Name: ahmad Roll No: 5 CGPA: 3.25
```

Multiple Class Objects

Just like we learned to use **Structs** to create multiple objects of "User-Defined DataType", Similarly, the class is also a "User Defined DataType" that is used to create multiple objects having the same properties but different values. Let's try to understand this concept through coding.

Task: Write the code to create multiple class objects and assign values to all of them.
Solution:
Write the following code into the main function of the code and execute the program by clicking on the start button.

Code:	**Output:**

```
namespace Test
{
    0 references
    class Program
    {
        4 references
        class students
        {
            public string name;
            public int roll_no;
            public float cgpa;
        }
        0 references
        static void Main(string[] args)
        {
            // first Object
            students s1 = new students();
            s1.name = "ahmad";
            s1.roll_no = 5;
            s1.cgpa = 3.25f;
            Console.WriteLine("Name: {0} Roll No: {1} CGPA: {2}", s1.name, s1.roll_no, s1.cgpa);
            // Second Object
            students s2 = new students();
            s2.name = "bilal";
            s2.roll_no = 6;
            s2.cgpa = 3.75f;
            Console.WriteLine("Name: {0} Roll No: {1} CGPA: {2}", s2.name, s2.roll_no, s2.cgpa);
            Console.Read();
        }
    }
}
```

```
C:\Users\HP\source\repos\Test\Test\bin\Debug\Test.exe
Name: ahmad Roll No: 5 CGPA: 3.25
Name: bilal Roll No: 6 CGPA: 3.75
```

Observe that each object possesses the same properties however, we have assigned different values to the same variables. The output reflects that all variables belong to a separate “class object” and therefore can be assigned new values in other class objects.

We access the variables by using the (**dot .**) operator in front of the class object name. For example, to print the name of the s1 student on the console, we will use the following code.

string name = s1.name;
Console.WriteLine(“Name: {0}”, name);

Taking input from User in Class Object

Taking input in class object variables is the same as taking input in any other variables in C#.

Look at the following code snippet to have a clear understanding of this concept.

Task: Write the code to create a class object and take input user name, roll number, and CGPA from the user and store them in the class object.

Solution

Write the following code on your computer and execute the program by clicking on the start button.

Code:	Output:

```
namespace Test
{
    0 references
    class Program
    {
        2 references
        class students
        {
            public string name;
            public int roll_no;
            public float cgpa;
        }
        0 references
        static void Main(string[] args)
        {
            // first Object
            students s1 = new students();
            Console.WriteLine("Enter Name: ");
            s1.name = Console.ReadLine();
            Console.WriteLine("Enter Roll No: ");
            s1.roll_no = int.Parse(Console.ReadLine());
            Console.WriteLine("Enter CGPA: ");
            s1.cgpa = float.Parse(Console.ReadLine());
            Console.WriteLine("Name: {0} Roll No: {1} CGPA: {2}", s1.name, s1.roll_no, s1.cgpa);

            Console.Read();
        }
    }
}
```

```
C:\Users\HP\source\repos\Test\Test\bin\Debug\Test.exe
Enter Name:
IRZAM
Enter Roll No:
12
Enter CGPA:
3.3
Name: IRZAM Roll No: 12 CGPA: 3.3
```

Student Management System with Class

Task: Write a program that shows three menu options

1. Add Student.
2. Show Students.
3. Top Students.

- Add Student allows users to add a student's information that includes RollNo, Name, GPA, isHostelide, Department.
- Show Student displays all the added students on the screen.
- Top Student lists the information of the top 3 students.

Solution:

Sr. #	Action	Description
1.	students.cs* Program.cs Test 1 using System; 2 using System.Collections.Generic; 3 using System.Linq; 4 using System.Text; 5 using System.Threading.Tasks; 6 7 namespace Test.BL 8 { 10 references 9 class students 10 { 11 public string name; 12 public int roll_no; 13 public float cgpa; 14 public char isHostelide; 15 public string department; 16 } 17 }	Create a separate class in a new folder named "BL", with public data members. **Note:** The objective of OOP is that we want to keep our code and class data separate. Therefore, this is the preferred practice that we will adopt in the coming lectures.

2.	1 reference `static char menu()` `{` `  Console.Clear();` `  char choice;` `  Console.WriteLine("Press1 for Adding a Student: ");` `  Console.WriteLine("Press2 for View Student: ");` `  Console.WriteLine("Press3 for Top three students: ");` `  Console.WriteLine("Press4 to exit: ");` `  choice = char.Parse(Console.ReadLine());` `  return choice;` `}`	Creates the main menu function.
3.	1 reference `static students addStudent()` `{` `  Console.Clear();` `  students s1 = new students();` `  Console.WriteLine("Enter Name: ");` `  s1.name = Console.ReadLine();` `  Console.WriteLine("Enter Roll No: ");` `  s1.roll_no = int.Parse(Console.ReadLine());` `  Console.WriteLine("Enter CGPA: ");` `  s1.cgpa = float.Parse(Console.ReadLine());` `  Console.WriteLine("Enter Department: ");` `  s1.department = Console.ReadLine();` `  Console.WriteLine("Is Hostelide (y || n): ");` `  s1.isHostelide = char.Parse(Console.ReadLine());` `  return s1;` `}`	1. Creates a function that takes the following inputs from the user • Name • Roll Number • CGPA • Department • IsHostelide 2. Creates a class “students” type object and stores the input in that object 3. Returns this object to the main function so it can be stored into the main array inside the main function.
4.	`static void viewStudent(students[] s, int count)` `{` `  Console.Clear();` `  for (int i =0; i < count; i++)` `  {` `  Console.WriteLine("Name: {0} Roll No: {1} CGPA: {2} Department: {3} IsHostelide: {4}", s[i].name, s[i].roll_no, s[i].cgpa, s[i].department, s[i].isHostelide);` `  }` `  Console.WriteLine("Press any key to continue.. ");` `  Console.ReadKey();` `}`	Receives complete arrays of students and prints the information in line by line manner.

5.	`static void topStudent(students[] s, int count)` `{` `  Console.Clear();` `  if (count == 0)` `  {` `  Console.WriteLine("No Record Present");` `  }` `  else if (count == 1)` `  {` `  viewStudent(s, 1);` `  }` `  else if (count == 2)` `  {` `  for (int x = 0; x < 2; x++)` `  {` `  int index = largest(s, x, count);` `  students temp = s[index];` `  s[index] = s[x];` `  s[x] = temp;` `  }` `  viewStudent(s, 2);` `  }` `  else` `  {` `  for (int x = 0; x < 3; x++)` `  {` `  int index = largest(s, x, count);` `  students temp = s[index];` `  s[index] = s[x];` `  s[x] = temp;` `  }` `  viewStudent(s, 3);` `  }` `}`	Prints the first three students from the list.
6.	2 references `static int largest(students[] s, int start, int end)` `{` `  int index = start;` `  float large = s[start].cgpa;` `  for (int x = start; x < end; x++)` `  {` `  if (large < s[x].cgpa)` `  {` `  large = s[x].cgpa;` `  index = x;` `  }` `  }` `  return index;` `}`	Finds and returns the index of the largest item from the array.

7.	``` static void Main(string[] args) { students[] s = new students[10]; char option; int count = 0; do { option = menu(); if (option == '1') { s[count] = addStudent(); count = count + 1; } else if (option == '2') { viewStudent(s, count); } else if (option == '3') { topStudent(s, count); } else if (option == '4') { break; } else { Console.WriteLine("Invalid Choice"); } } while (option != '4'); Console.WriteLine("Press Enter to Exit.."); Console.Read(); } ```	Invokes the respective functionality according to input provided by the user.

Challenge # 1:

Task: Write a program that shows three menu options

1. Add Products.
2. Show Products.
3. Total Store Worth.

- Add Product allows the user to add product information that includes ID, Name, price, Category, BrandName, Country.
- Show Product display all the added products on the screen.
- Total Store Worth calculates the sum of the price of all the products.

Challenge # 2:

Task Covert the signUp/signIn application that you developed in the previous lab by using the class concepts.

Make a class named Credentials with two attributes namely

- Username
- Password

The data should be loaded from the file and loaded into the attributes of the class.

Good Luck and Best Wishes !!

Happy Coding ahead :)