Object Oriented Programming with C#

11/11/2021lec1

Overview of C#

- C# programming language is located inside .NET Framework.
- To develop desktop applications C# programming language has been used.
- .NET Framework has a feature of developing web applications in there ASP.NET and C# are being used.
- In desktop app development visual studio IDE is used.

Overview of IDE

- In visual studio, to come up with a desktop application console application project needs to be selected.
- In console application project,
    - namespace defines the project name.
    - class defines the file that you are currently working on.
    - static void Main (String [ ] args) defines the main method of console application project.

- On top of the control application project there are some library files which are automatically integrated.
- All library files are starting with the keyword “using”.
- On the window solution explorer will visible which contains all details about your console application project.
- In their Programe.cs will be located in that file main method of your application is located.
- File ends up with .cs extension which consider as a class file in C#

25/11/2021lec2

Creating hello world program

- ♥ In C# to display the content on console window there are two available in-built functions.
  - → Console.WriteLine()
  - → Console.Write()

- ♥ The content you want to display on the console window need to be give within the brackets in between “”
- ♥ Both of these functions are performing the same contribution which has done by printf in C programming.

  Ex:
  - → Console.WriteLine(“Hello World”);
  - → Console.Write(“Hello World”);

- ♥ In Console.WriteLine content will print, and curser will move to the next line.
- ♥ In Console.Write content will display and curser will remain in the same line.
- ♥ In Console application projects program will execute and the console window will suddenly disappear. To avoid that one of the following functions could be used.
  1. Console.ReadKey()
  2. Console.ReadLine()

Q] display your name, batch, degree program and age on console application project using 4 separate lines.

```
Console.WriteLine("Chamali");
Console.WriteLine("21.1");
Console.WriteLine("SE");
Console.WriteLine("19");
```

Data types

- Data types define what value a variable can hold.
- In C# there are number of data types which available for different data inputs.
  - → String, int, double, char, bool, float etc.

Declaring variables

- Variables need to be declared under a certain data type. Which means for the variable s that you are declaring should have a specific data type where we can use to store those data in it.

Declaring a string variable

- in OOP programming there's a specific data type known as string which can hold number of characters (words) in any given time.
- String variables will initialize with "" when assigning values.

  Ex: String module = `String module = "object-oriented programming";`

  In above code string variable is having a value at the point of declaration which known as initialization.

Variable declaration examples

```
int age = 23; //store numbers without decimal points
double batch = 21.1;//store numbers with decimal points
char x = 'a'; //store a single character
bool result = true;//store 1 or 0 value (true or false)
```

bool is a data type which can hold either true (1) or false (0)

displaying variables on console application

```
static void Main(string[] args)
        {
            int age = 23;
            Console.WriteLine("" + age);
            Console.WriteLine("Your age is " +age);
            Console.ReadLine();
              }
```

Q] declare a string and float/ double variable and store your name and the batch in those variables. Display the output using console application program.

```
static void Main(string[] args)
        {
            string name = "chamali";
            double batch = 21.1;
            Console.WriteLine("Your name is "+name);//Console. WriteLine(Name);
            Console.WriteLine("Your batch is " + batch);
            Console.ReadLine();
              }
```

02/12/2021lec3

Getting string input from user

- In C# to get user input values Console.ReadLine( ) in-built function has been used.
- What the function does is it will read the line on the console window and taken whatever the value inside the application.
- In C# default data type when taking user input values is considered as string.

  Ex:

```
static void Main(string[] args)
  {
      string name;
      Console.WriteLine("Enter your name: ");
      name = Console.ReadLine();
      Console.WriteLine("Welcome" + name);
      Console.ReadLine();
        }
```

Q] Create a C# console application program to get users first name and last name separately, and then display both of names together

Example
First Name – Alex
Last Name – Hales
output – Alex Hales

```
static void Main(string[] args)
        {
            string fname, lname;
            Console.WriteLine("Enter your first name: ");
            fname = Console.ReadLine();
            Console.WriteLine("Enter you last name: ");
            lname = Console.ReadLine();
            Console.Write(fname+ " " + lname);
            Console.ReadLine();
               }
```

By using + operator we can merge two string variables together. In programming it known as "Concatenation"

Ex: `Console.Write(fname+ " " + lname);`

Getting integer user inputs

- In C# all console inputs will consider as string data inputs.
- Because of that when expecting an integer input the string value need to be convert into integer value. This process is known as "Typecasting".
- To do that, C# provides two in-built functions
    1) int.Parse( )
    2) Convert.ToInt → Convert.ToInt32
- Above functions are capable of converting string value into integer.

Ex:

```
static void Main(string[] args)
 {
     int age;
     Console.WriteLine("Enter your age: ");
     age = int.Parse(Console.ReadLine());
     Console.WriteLine("your age is " + age);
     Console.ReadLine();
              }
```

Q] Create a C# console application program to get two integer inputs from the user and display the answers for four basic arithmetic operations.

```
static void Main(string[] args)
        {
            int num1, num2, addition, subtraction, multiplication, division;
            Console.WriteLine("Enter first number: ");
            num1 = int.Parse(Console.ReadLine());
            Console.WriteLine("Enter second number: ");
            num2 = int.Parse(Console.ReadLine());
            addition = num1 + num2;
            Console.WriteLine("The addition is " + addition);
            subtraction = num1 - num2;
            Console.WriteLine("The subtraction is " + subtraction);
            multiplication = num1 * num2;
            Console.WriteLine("The multiplication is " + multiplication);
            division = num1 / num2;
            Console.WriteLine("The division is " + division);
            Console.ReadLine();
        }
```

Or

```
static void Main(string[] args)
        {
            int num1, num2;
            Console.WriteLine("Enter first number: ");
            num1 = int.Parse(Console.ReadLine());
            Console.WriteLine("Enter second number: ");
            num2 = int.Parse(Console.ReadLine());

            Console.WriteLine("The addition is " + (num1 + num2));
            Console.WriteLine("The subtraction is " + (num1 - num2));
            Console.WriteLine("The multiplication is " + (num1 * num2));
            Console.WriteLine("The division is " + (num1/num2));
            Console.ReadLine();
              }
```

The division should be double. So,

```
static void Main(string[] args)
        {
            int num1, num2;
            Console.WriteLine("Enter first number: ");
            num1 = int.Parse(Console.ReadLine());
            Console.WriteLine("Enter second number: ");
            num2 = int.Parse(Console.ReadLine());

            Console.WriteLine("The addition is " + (num1 + num2));
            Console.WriteLine("The subtraction is " + (num1 - num2));
            Console.WriteLine("The multiplication is " + (num1 * num2));
            Console.WriteLine("The division is " +Convert.ToDouble(num1) / 
Convert.ToDouble(num2));
            Console.ReadLine();
        }
```

Q] write a C# console application program to check weather the user is eligible for voting.

```
static void Main(string[] args)
        {
            int age;
            Console.WriteLine("Enter your age: ");
            age = int.Parse(Console.ReadLine());
            if (age >= 18)
            {
                Console.WriteLine("You are eligible for voting");
            }
            else
            {
                Console.WriteLine("You are not eligible for voting");
            }
            Console.ReadLine();
              }
```

Q] create a console application program to get an integer input from the user. If number is greater than 10 and less than 20, display values from 2 by 2 until 100. If not display a quote user input is not belongs to 10-20.

```
static void Main(string[] args)
        {
            int num;
            Console.Write("Enter a number: ");
            num = int.Parse(Console.ReadLine());
            if (num > 10 && num <20)
            {
                while (num <= 100)
                {
                    Console.WriteLine(num);
                    num += 2;
                }
            }
            else
            {
                Console.WriteLine("user input is not belongs to 10-20");
            }
            Console.ReadLine();
        }
```

09/12/2021lec4

C# classes

- In OOP languages the main idea is code reusability. Which means if we have the existing code, we could be able to use that code in anywhere under the same namespace (project) when required.
- In C# default class file is known as program.cs which contain the main method of your application.
- In C# we are capable of adding number of classes based on the requirement and those class files are capable of containing codes same as program.cs
- A namespace can contain only one main method which need to be in program.cs
- Added classes could contain user define functions based on the preference.
- Class file extension should always be .cs
- A declared class would contain variables and methods based on the requirement.
- Which means there would be n number of variables and n number of methods in a single class where we identify them as member variables and member functions.

- ♥ Same as program.cs C# classes are capable of containing any sort of code such as conditional statements, loop control structures, arithmetic operations, logical operations etc.

Structure of a C# class

```
//member variables
<access specifier> <data type> variable1;
<access specifier> <data type> variable2;


<access specifier> <data type> variableN;
//member methods
<access specifier> <return type> method1(parameter_list) {
        //method body
}
<access specifier> <return type> method2(parameter_list) {
        //method body
}
<access specifier> <return type> method(parameter_list) {
        //method body
}
```

C# access specifiers / modifiers

- ♥ Access specifiers will decide the accessibility level of member variables and member functions in a particular class.
- ♥ In OOP programming there are three main access specifiers,
    - → public
    - → private
    - → protector

public access specifier

- if a variable or a method declared under public access specifier that variable or method can be used by any of the classes in same project (namespace)

  ex: public int Number;

  ```
  public void Myfuction(int Num)
  {
  }
  ```

private access specifier

- if a variable, method is declared under private access specifier that variable or method could be only access by the class which it declares. Which means those items will be only limited to the class itself

  ex: Class

  ```
  private int Sum;
  private void Mymethod;
  {
  }
  ```

Class

```
internal class Class1
    {
        public int Number = 10;
        public void MyMethod()
        {
            Console.WriteLine("my variable is: " + Number);
            Console.ReadKey();
        }
    }
```

Program

```
static void Main(string[] args)
        {
             Class1 objMyclass = new Class1();
            objMyclass.MyMethod();
        }
```

16/12/2021lec5

Accessing the added class from another class

- in C# there's a possibility of adding number of classes to the existing project.
- Once those classes are added to the project those classes need to be called form the main method unless the code that we have written in those classes will not be execute.
- In OOP programming languages above requirement can be fulfilled by using class objects.
- Class object is something like a gateway to created classes and by using class object we can call all public variables and methods.
- Though we have created a class object if the variables and methods are private we won't be able to call them by using class object.

Creating class object

- Class object needs to be created by using the added class name from where the method needs to be called.
- Assume that there's a console application project with a separate class called MyClass with a public void method called Hello. We want to call this function from the main method unless compiler is not aware about this added class.
- Since we need to call the function from the main method, we need to create a class object inside the main method.
- By using the created object public method called hello can be executed.

```
internal class Program
    {
        static void Main(string[] args)
        {
            MyClass obj = new MyClass();
            obj.Hello();
            Console.ReadKey();
        }
          }
```

```
namespace ConsoleApp8
{
    internal class MyClass
    {
        public void Hello()
        {
            Console.WriteLine("Say hello");
        }
    }
}
```

- As the object name you can provide any name unless it is a reserve keyword.
- One class object is enough for the entire class when calling functions.

  Ex: assume that there are ten public methods in a separate class when calling them one created object is enough.

Q1] **Create a C# console application project with a separate class called MyClass. Inside MyClass there is a public void method to check the user inserted integer is an odd number or an even number. User inputs will be taken only from the separate class.**

```
internal class MyClass
    {
        public void CheckOddEven()
        {
            int number;
            Console.WriteLine("enter an integer: ");
            number = int.Parse(Console.ReadLine());
            if (number%2==0)
            {
                Console.WriteLine("Number is even");
            }
            else
            {
                Console.WriteLine("Number is odd");
            }
        }
    }
```

```
internal class Program
    {
        static void Main(string[] args)
        {
            MyClass objMyclass = new MyClass();
            objMyclass.CheckOddEven();
            Console.ReadLine();
        }
    }
```

Q2] Create a C# console application project with a separate class called vehicle. Inside vehicle class there is a public method to get number of wheels which vehicle contained. From the method following outputs should provide.

No of wheels	output
2	Bicycle
3	Three-wheeler
4	Car
No of wheels >= 5	Multi-purpose vehicle

```
internal class vehicle
    {
        public void CheckVehicle()
        {
            int wheels;
            Console.WriteLine("enter the number of
wheels: ");
            wheels = int.Parse(Console.ReadLine());
            if (wheels == 2)
            {
                Console.WriteLine("bicycle");
            }
            else if(wheels == 3)
                    {
                Console.WriteLine("three-wheeler");
            }
            else if(wheels == 4)
                    {
                Console.WriteLine("car");
            }
            else if(wheels >=5)
                Console.WriteLine("multi-purpose
vehicle");
        }
    }
```

```
static void Main(string[] args)
        {
            vehicle objvehicle = new vehicle();
            objvehicle.CheckVehicle();
            Console.ReadLine();
        }
```

Q3] create C# console application project with a separate class called Operations. Inside the operation class there is a public void method with two parameter inputs. Methods will add both parameters together and display the answer as an integer value inside the method itself. Take two user inputs to the method from the main method and call the function properly to get the output.

```
static void Main(string[] args)
        {
            int num1, num2, total;
            Console.WriteLine("enter the first number: ");
            num1 = int.Parse(Console.ReadLine());
            Console.WriteLine("enter the second number: ");
            num2 = int.Parse(Console.ReadLine());

            operations objOperations = new operations();
            objOperations.Summation(num1, num2);

            objOperations.Substraction(num1, num2);

            Console.ReadKey();
        }
```

```
internal class operations
    {
        public void Summation(int num1, int num2)
        {
            Console.WriteLine("summation is: " + (num1 + num2));
        }
        public void Substraction(int num1, int num2)
        {
            Console.WriteLine("substraction is: " + (num1 -
num2));
        }
    }
```

Q4] Modify the Q3 to get the subtraction of user inserted values by adding same type of function to the operations class.

```
static void Main(string[] args)
        {
            int num1, num2, total;
            Console.WriteLine("enter the first number: ");
            num1 = int.Parse(Console.ReadLine());
            Console.WriteLine("enter the second number: ");
            num2 = int.Parse(Console.ReadLine());

            Operations objOperations = new Operations();
            objOperations.Subtraction(num1, num2);

            Console.ReadKey();

        }
```

```
internal class Operations
    {
        public void Subtraction(int num1, int num2)
        {
            Console.WriteLine("subtraction is: " + (num1 - num2));
        }
    }
```

06/01/2022lec6

Q] create a C# console application program with a separate class called “student”. Inside the student class create a method to check student grade once the student mark passed to the method as a parameter.

A → 75-100

B → 65-74

C → 50-64

D → 40-49

F >= 39

```
static void Main(string[] args)
        {
            int stdmark;

            Console.WriteLine("enter your mark: ");
            stdmark = int.Parse(Console.ReadLine());

            student objstudent = new student();
            objstudent.CheckGrades(stdmark);

            Console.ReadLine();

        }
```

```
internal class student
    {
        public void CheckGrades(int marks)
        {
            if(marks>=75 && marks <= 100)
            {
                Console.WriteLine("A");
            }
            else if (marks >= 65 && marks <= 74)
            {
                Console.WriteLine("B");
            }
            else if (marks >= 50 && marks <= 64)
            {
                Console.WriteLine("C");
            }
            else if (marks >= 40 && marks <= 49)
            {
                Console.WriteLine("D");
            }
            else if (marks >= 0 && marks <= 39)
            {
                Console.WriteLine("F");
            }
            else
            {
                Console.WriteLine("invalid input");
            }
        }
    }
```

Q] create a C# console application program with a separate class called calculation. Inside the separate class create a method which accept two integer parameters. Inside the method above two parameters needs to be multiplied and the final answer should return out of the method. Inside the main method final answer should display.

```
static void Main(string[] args)
        {
            int num1, num2, Answer;
            Console.WriteLine("enter the first number: ");
            num1 = int.Parse(Console.ReadLine());
            Console.WriteLine("enter the second number: ");
            num2 = int.Parse(Console.ReadLine());

            calculation objcalculation = new calculation();
            Answer = Objcalculation.mycalculation(num1, num2);
            Console.WriteLine(Answer);

            Console.ReadKey();
        }
```

```
internal class calculation
    {
        public int mycalculation(int num1, int num2)
        {
            int multiplication;
            multiplication = num1 * num2;
            return multiplication;
        }
    }
```

Q] create a C# console application program with a separate class called eligibility. Inside the separate class include a returning function once users age passed as a parameter. Method should check whether user is eligible for voting or not.

```
internal class eligibility
    {
        public string checkeligible(int age)
        {
            if(age >= 18)
            {
                return "you are eligible for voting";
            }
            else
            {
                return "you are nt eligible for voting";
            }
        }
    }
```

```
static void Main(string[] args)
        {
            int age;
            Console.WriteLine("enter your age: ");
            age = int.Parse(Console.ReadLine());

            eligibility objEl = new eligibility();
            string output = objEl.checkeligible(age);
            Console.WriteLine(output);
            Console.ReadLine();
        }
```

Q] create a C# console application project with a separate class called Test. Inside the Test class create an integer variable with value 0. From the main method try to assign value 100 to the same variable. Finally display the value inside the variable from main method.

- When declaring variables if you forgot to mention the access specifier, all those variables will consider as private variables.
- Which means those private variables cannot be access from other classes.

Ex:	int MyNumber;	→ private variable
	private int MyNumber	→ private variable
	public int MyNumber;	→ public variable

- Above concept is applicable for methods to

Ex:	void MyMethod	→ private method
	private void MyMethod	→ private method
	public void MyMethod	→ public method

```
internal class Test
    {
        int MyNumber = 0;
    }
```

13/01/2022lec7

OOP

- OOP languages are based on the concepts of objects. Which means creating number of objects from different classes the code that you have written can be reuse.
- In OOP languages there are 4 basic OOP concepts are being used.
    1) Encapsulation – package the variables and functions into a single unit.
    2) Abstraction – show only the essentials to the outside use.
    3) Inheritance – inherit common variables and functions.
    4) Polymorphism – function the object depending on the data.
- Above mentioned technologies are just concepts which means you can modify above concepts based on the requirement.

Access specifiers

20/02/2022lec8

```
internal class EncapData
    {
        private int Age;
        public void setAge(int age)
        {
            Age = age;
        }
        public string getAge()
        {
            if(Age>=18)
            {
                return "you're eligible for voting";
            }
            else
            {
                return "you aren't eligible for 
voting";
            }
        }
    }
```

```
static void Main(string[] args)
        {
            Console.WriteLine("enter your age: ");
            int UserAge = 
int.Parse(Console.ReadLine());

            EncapData objEncap = new EncapData();
            objEncap.setAge(UserAge);

            Console.WriteLine(objEncap.getAge());
            Console.ReadLine();
        }
```

```
internal class EncapData
    {
        private double rad;
        private double pi = 3.14;
        public void setRad(double radius)
        {
            rad = radius;
        }
        public double getArea()
        {
            return pi * rad * rad;
        }
        public double getCircumstance()
        {
            return 2 * pi * rad;
        }
    }
```

```
static void Main(string[] args)
        {
            Console.WriteLine("enter the raidus value: ");
            double userRadius = double.Parse(Console.ReadLine());

            EncapData objEncap = new EncapData();
            objEncap.setRad(userRadius);

            Console.WriteLine("Area is " + objEncap.getArea());
            Console.WriteLine("Circumference is " +
objEncap.getCircumstance());
            Console.ReadLine();
        }
```

Inheritance

- ♠ Inheritance OOP concept is used to avoid code repetition in applications.
- ♠ In inheritance we can declare classes as base class (parent) and derived class(child).
- ♠ The code which existing in parent class will be automatically apply to the child class when using the inheritance.
- ♠ In inheritance data members which declared as public and protected (access specifiers) will be automatically apply to the derived class or child class.

Initialization of base class and derived class

- ♠ In programming we need to properly identify the parent class and the child class in order to use inheritance.
- ♠ Any class can become a parent class and there are no restrictions.
- ♠ When declaring the child class, child class name should be followed by colon (:) and the base class.
- ♠ Syntax

```
<access-specifier> class <base_class> {
…………                    ← parent class / base class
}

Class <derived_class> : <base_class> {
…………
}                           ← child class / derived class
```

```
static void Main(string[] args)
        {
            DegreeProgram objDegree = new
DegreeProgram();
            objDegree.Details();

            Console.ReadLine();
        }
```

```
internal class Faculty
    {
        public int student =
100;
    }
```

```
internal class DegreeProgram:Faculty
    {
        public void Details()
        {
            Console.WriteLine(student);
        }
    }
```