Name : Priya Gosai

Topics :

Module-3

I. Understanding Classes
II. Depth in Classes
III. Scope & Accessibility Modifiers
IV. Namespaces and .Net Library
V. Creating and adding ref. to assemblies
VI. Working with collections

1. Understanding Classes

Class is like a blue print of the object .we can define the data and functionality their object will have in the class. we can also create our own methods in class ,variables of different types and objects.

1.1 Class type and usage :

There are types of the class like.

1) Abstract Class
2) Sealed Class
3) Static class

Abstract Class:

- We can make a class abstract by using abstract keyword. We can also make any variable or method abstract by using abstract keyword.
- Abstract class does not have any instance method or it cannot be instantiated. It can only inherited and used as a base class. that means we cannot create a member of abstract class.
- Abstract class does not contain the implementation of the abstract method .it allows only declaration of the method. in the child class the implementation of the abstract method is given and if in child class the implementation is not given then it will also become a abstract class. By using override keyword the abstract method's implementation is given in the child class.
- Abstract class may contain the abstract members (methods, properties, indexers) but it is not mandatory.
- It cannot be a sealed class because it is used as a base class only.

Use of the abstract class:

- In the program if we have some same properties for two or more class we can store them together in a single class.
- By doing this we can do code maintainability. in very good way. like if we have more number of classes then doing changes in all classes is very time consuming. instead of this we can make abstract class and define all same members for different class in that and later inherit the abstract class so all the class will get the properties of the base class.

Static class:

- Static class can be created using **static** keyword.
- It can contain only static members within the class. That means we cannot create instance member or methods or constructor. It can contain static constructor which is used for initialize the static members of the class and it will call only once.
- Static class cannot be instantiated or inherited. So we cannot create a object of the static class.
- It is called using the class name followed by the member name.
- It is also a sealed class because It cannot be inherited.

Use of the static class:

- Static class is a container for the static methods and properties. Static methods and static properties are most used static members.
- Static class is used to when there is no need to change the value or properties of the class member. once the class member is define as static it will be constant throughout the program we cannot change the value of static member.
- It will make code cleaner and it is easy to access because it is called using class name directly.
- It will also reduce the memory requirement because they do not require instances created for each new object. That means they consume fewer

resources and no duplication of the same class or member is needed in memory.

Sealed class:

- A Sealed class is a class that prevents inheritance.
- It is created using sealed keyword preceds the class keyword.
- BY making a class sealed class we cannot inherit that class and it cannot become the base class.
- It will prevent the class from inheritance .

Use of the sealed class:

- In our program there will be such critical information or data that we don't want to be inherited in other class.
- For that purpose the concept of the sealed class is used it will not allow the user to inherit the sealed class.
- For example in the systeminfo class there is crutial data and information about operating system and for the security purpose we don't want that class to be inherited that's why sealed class is used.
- So when we designing a class library and want to restrict our classes not to be derived by other developer ,we can make our class sealed. If anyone tries to do so it will give compile time error.
- **Syntax:** sealed class SystemInfo

```
{
    //body of class

}
```

2. Depth in classes:

2.1 Object, properties ,Methods, Events:

A class is a logical collection of the similar types of the objects.

It is one of the most fundamental types in the c#.

It is basically a datastructure that contain methods, objects, properties, fields, functions etc.

Objects and class:

- Object is a instance of a class. Class is a defination of the object but not a object itself.
- Generally A class instance are invoke using new keyword .they provide memory to the instance. it invokes a constructor to initialize it and return a reference object.
- **For example,**
  If you want to create a instance of the class student;
  Student s1 = new student();
  Here we have created a instance of class student and referenced it with s1.

Properties of the class:

- Properties are the special type of class member that provides a flexible mechanism to read, write, or compute the value of a private field.
- Properties allows you to Control the access of the member variable and it's recommended way to access variable from the outside of class.
- It's like a combination of variable and method. it's much like a variable but have logic behind it.
- It cannot take any parameter as an argument but we can process the value before assigning to our returned.
- By using properties we can access private member of a class in other class it's a main function of a properties.

- It define like a variable and has get and set accessors code added.

Types of the properties:

1. *Read / Write properties*: in this we can use both set and get. So we can set the property and also get that.
2. *Read only properties*: in this we can only use get property. the value is already set and cannot be changed. One can only read the value.
3. *Write properties:* in this we can use only set property so we can only set the property and cannot get it .one can only write the value and cannot read that.
4. *Auto Implemented Property:* in this both the properties are used and if we want we can make one of that private *also.*

Methods of class:

Methods are basically a group of statements together perform some specific task and retuen some values. It may take parameters as an argument.

To perform some task by ethod we have to first define a method and then call the method by creating the object of that method.

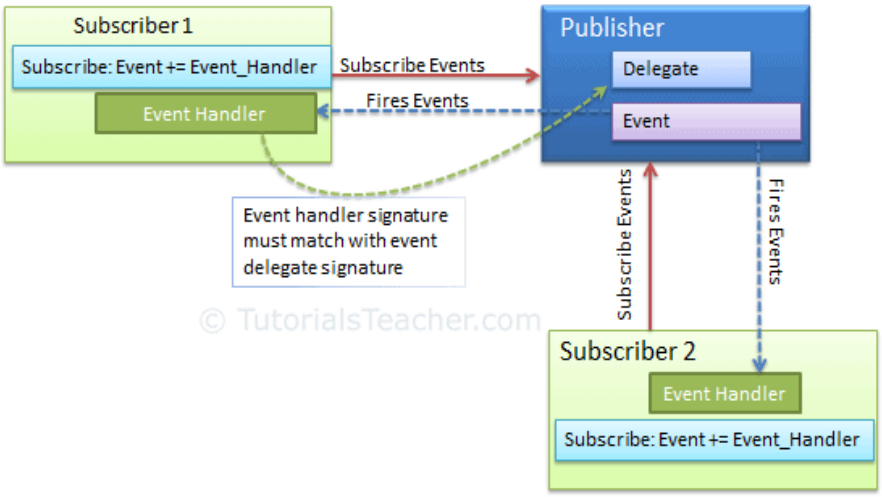
There are many types of methods available in c#:

1. **Instance methods:** instance methods that operates on the instance member of the class and instance can be accessed by this.
   Example : public void sum();
2. **Abstract methods**: It is the method with no Implementation and is implicitly virtual.it can degine only in the abstract class. if class contain at least one abstract method the class will become abstract class.
   Example : public **abstrac**t getId();
3. **Static methods :**in this method we cannot create a instance member , only static members are allowed .it is accessed by class name.created by using static keyword.
   Example : public **static** EmployeeInfo();

4. **Virtual methods**: it make some default functionality .in other words ,virtual methods are being implemented in the base class and can be overridden in the derived class.
   Example : Public **virtual** int Calculate();
5. **partial methods:** A partial method has its signature defined in one part of partial type, and its implementation is given in the another part of the partial type.
   Example : partial Void Onsomethinghappened();

Events :

- Events are encapsulated delegates. It is dependent on delegates.
- We can say events are the notification which is sent by object when some action is performed or occurred.
- Generally events are user actions such as key press,clicls,mouse movements etc.or some occurrence such as system generated notification.
- Events are declared and raised in a class and associated with the event handler using delegates within the same class or some other class.
- In events there are Publishers and subscribers. the class who raise the event is called publisher and the class who receives the notification is called subscriber.
- Typically, A publishers raises an event when some action occurred and the subscriber which are interested in that event to get notify should register that event and handle it.
- An event has no return type and it is always void.
- In The diagram of how event occurred and handled is shown



How to declare an event:

An event can be declare in following steps:

- **Declare a delegate** :public delegate String Someactionoccur(String str);
- **Declare a event with event keyword**.: event Someactionoccur clickevent; in this first step defines a delegate named Someactionaccur and an event clickevent. which invoke the delegate when it is raised.
- Define an Event handler that respond when event raised
- You must have method ready for delegates

3. Scope and Accessibility Modifiers:

Access modifiers and specifiers are keywords to specify the accessibility of a type and its members.

Access modifiers describe as the scope of accessibility of an object and its members. All members have an accessibility level.

We can control the accessibility level and the scope of that object members by using access modifiers.

We are using access modifiers for security of our applications or programs.

Basically c# has a 5 access specifiers or modifiers.

- Private
- Public
- Internal
- Protected

3.1 public, private, protected:

public:

- we can use public keyword with any member of the program. Like class, variable, delegates, conustructor, methods , events etc.
- it will allows you to access that member anywhere in the program.
- Though it is declare as a public with the help of the public keyword it has no limits.
- We can use it in a class assembly even outside if the class assembly.
- Most DLLs are known to be produced by public class and members written in a .cs file
- Example:
  Public void sum() -- Method

Public class student – class
Public int student – constructor
Public String name; -- variable

Private:

- It will limit the accessibility of a member to within the defined type.
- If a variable or a function is being created in a class and they are declared as a private then another class cannot access that.
- It is most restrictive and accessible only within the body of class in which it is declared.
- Example:
  Private String name; -- variable
  Private void sum() -- method

Protected:

- The scope of accessibility in the protected access modifier is limited within the class or struct and the derived class from the class in whch we have declared members as a protected.
- Protected variables and functions in the other class cannot access in the main method because main() function is in a separate class.
- They are access only in the same class or derived class.
- Example:
  Protected String msg; -- variable
  Protected void display() – method

Internal Access Modifiers:

The internal access modifiers can access within the program that contain its declaration and also access only within files in the same assembly level but not from another assembly.

4. Namespace & .Net Library

Namespace:

- A namespace is used to organize your code and is a collection of classes. interfaces, structs, enums and delegates.
- And if we don't want to use namespace in our program we can use fully qualified name.but is is very difficult to write if we have include namespace in our program we can use the methods which is in tha namespace directly.
- In c# there are many built in namespace available in the .Net library .we can use them in our program by including proper namespace in the program.
- They do play an important role in writing cleaner codes and managing larger projects.
- We can access the member of the namespace by (.)Dot operator.
- Namespace can be included in a program using the **using** keyword.
- the syntax is,
  Using Namespace-Name;

  Example : using System;

.Net Library:

- .Net Library includes classes, interfaces , delegates and value type that optimize the development process and provide access to system functionality.
- It provides a huge classes, interfaces , delegates and it is impossible to store them in one directory that's why they stores in the different different class library files and namespaces.

- .NET library includes types that perform the following functions :
    - Represent base Data type and exceptions.
    - Encapsulates data structures.
    - Perform I/O.
    - Supports exception handling.
    - Support for creating web-services
    - Provide data access ,rich client side GUI ,and server –Controlled, client side GUI.
- It provides a rich set of interfaces as well as abstracr and concrete classes. We can use them or in some case you can derive your own classes from that classes
- It includes the System namespace and the core types of the .NET framework.

5. Creating and adding ref. to Assemblies:

- Normally we do not have to add any assembly reference to our current projects because visual studio is able to automatically add the appropriate references to the project .
- But sometimes when you want or need to manually add a reference to an assembly to your project.
- We can add some in built .Net Assembly to our project.
- We can add references to the project by simply following this steps:
    - Go to the solution menu
    - Right click on the references
    - Click on the add
    - Now you can add any references you want to add.
- Once you have selected the assembly ,click the OK to add the reference and close the dialog.
- The add reference dialog allows you to select and add multiple reference at the same time.
- You must use the using statement in the code in order to directly use the custom dll in the coded step.
- We can also include that assembly which are provided by the other developer globally in the our project visual studio also provide that feature that we can direct go to that window and import whichever assembly we want to use.
- The example of how we can add that in our project and also we can add our custom DLL file to the project is given in the demo document.

6. Working with collections:

- Collection is just like array. there is some difference between array and collection but the method of accessibility is same.
- Like arrays collections also has index numbers and other properties.
- It provides a more flexible way to working with group of objects.
- We can say collection is a dynamic array.
- In the array we cannot change the length of array and also we cannot insert a new element in middle of array, same as we cannot delete a number from the middle of array .
- But in the collections we can insert the elements or we can delete that elements and also we can change the length of the array at runtime. all of this is possible in the collections.
- It is a set of a related data type that may not necessarily belong to the same data type that can be set or modified dynamically at runtime.
- We can never insert a value into a middle of array because if we want to do this the array length should be increased.
- In collection there is a autoincreased mechanism that will allows user to change the length of array.
- In collections we can store the different type of data at one place.that means we can store different data type value in a single array and it is possible in the collection in the array we can only store the data of only one data type but in the collection it is possible.
- Accessing the collection is same as the accessing the arrays.
- The example of the collection is given in the demo document.

Collections are generally of 2 types:

Generaic :

- they are founf under the system.Collectio.Generic.

- they are more flexible and are the preffered way to work eith data.
- It inhance code reuse ,type safety, and performance .
- It includes Dictionary<T,T> , List<T>, Queue<T>, SortedList<T>, and stack<T>.

Non-generaic:

- It includes blockingcollection<T>, ConcurrentDictionary<T,T>, ConcurrentQueue<T>, and ConcurrentStack<T>.

Standard:

- it is found in generally under system.Collections .
- it includes Arraylist, Hashtable, queue, and stack.