**Saumya Srivastava (** ASET **)**

**INDEX**

* Access modifiers in C#
* Types of Constructor in C#
* Properties initializer in C#
* Keywords in C#

**ACCESS MODIFIERS IN C#**

1. Public : If we declare any field with private ,it can be accessible for all classes.
2. Protected : If we declare any field with protected ,it can be accessed within same class or in inherited class.
3. Private : If we declare any field with private ,it can only be accessed within same class.
4. Internal : Accessible within its own assembly(class library)

**TYPES OF CONSTRUCTOR IN C#**

Constructors are used to create object and gets automatically invoked whenever an instance of the class is created. It does not have return type.

1. Default Constructor : A constructor without any parameters. In this it is not possible to initialize each instance of the class with different value.

public Student()

{

//default constructor

}

1. Copy Constructor : It creates an object by copying variables from another object. It takes reference own its own class.

public Student(School)

{

//copy constructor

}

1. Static Constructor :
   1. It is called before building the first object.
   2. It must be parameterless and can not be overloaded.
   3. You can not access any non – static field inside a static constructor.

public static Student(School)

{

//static constructor

}

1. Private Constructor : We do not create its object. It should not have any public constructor.

private Student()

{

//private constructor

}

**PROPERTIES INITIALIZER IN C#**

1.Automatic property initializer :

* Properties are the fields which have two methods get and set.
* get method: used to read values.
* set method: used to modify the values.

*public string Name { get; set; } //syntax*

2.We can explicitly define the get and set methods.

*private string name;*

*public string \_Name*

*{*

*get => name;*

*set => name = value;*

*}*

* If we want to make the fields read only, we only mention the get method inside the property.

**KEYWORDS IN C#**

* **Virtual Keyword**The Virtual keyword is used for generating a virtual path for its derived classes on implementing method overriding. The Virtual keyword is used within a set with an override keyword. It is used as:

*public virtual void Parent(string name) //Parent Method*

*{*

*Console.WriteLine("hello " + name);*

*}*

* **Override Keyword**  
  The Override keyword is used in the derived class of the base class in order to override the base class method. The Override keyword is used with the virtual keyword.

*public override void Child(string name) //Child Method*

*{*

*Console.WriteLine("Hello " + name);*

*}*

* **New Keyword**  
  The New keyword is also used for polymorphism but in the case of method overriding. In simple words we can say that we are changing what the base class does for the derived class.

*public new void Child(string name) //Child method*

*{*

*Console.WriteLine("Hello new " + name);*

*}*