Object-oriented Programming with C#

Tahaluf Training Center 2021

شركة تحالف الإمارات للحلول التقنية ذ.م.م.

TAHALUF AL EMARAT TECHNICAL SOLUTIONS L.L.C.

Day 9

Static Keyword

Use the static modifier to declare a static member, which belongs to the type itself rather than to a specific object. The static modifier can be used to declare static classes fields, methods, properties, operators, events, and constructors.

Day 9

Static Variable

A static variable is declared with the help of static keyword. When a variable is declared as static, then a single copy of the variable is created and shared among all objects at the class level. Static variables are accessed with the name of the class, they do not require any object for access.

Demo

```
public class Product
    {
        public int productId;
        public string productName;
        public double cost;
        public int quantityInStock;
        public static int totalNumProducts;

        //To access the static field inside the class
        public void methodeToAccessStaticFields()
        {
            totalNumProducts = 0;
            //or
            Product.totalNumProducts = 0;
        }
    }
```

Demo

```
static void Main(string[] args)
{
    Product product1 = new Product();
    Product.totalNumProducts++;// 1
    Product product2 = new Product();
    Product.totalNumProducts++;// 2
    Product product3 = new Product();
    Product.totalNumProducts++;// 3

    Console.WriteLine($"Total Number of Products:
    {Product.totalNumProducts}");// 3
    // This number not be stored inside object, it's
    stored in the class memory

    //product1.totalNumProducts++; //Error
    // object dosen't store static fields.
        }
```

Static Variable

Key	Static	Non-Static
Access	A static variable can be accessed by static members as well as non-static member functions.	A non-static variable can not be accessed by static member functions.
Sharing	A static variable acts as a global variable and is shared among all the objects of the class.	A non-static variables are specific to instance object in which they are created.
Memory allocation	Static variables occupies less space and memory allocation happens once.	A non-static variable may occupy more space.
Keyword	A static variable is declared using static keyword.	A normal variable is not required to have any special keyword.

Day 9

Demo

```
public class Product
    {
        public int productId;
        public string productName;
        public double cost;
        public int quantityInStock;
        public static int totalNumProducts;

        //constructor
        public Product()
        {
            Console.WriteLine("Default Constructor");
        }

        //static method: Set method of totalNumProducts
        public static void SetTotalNumProducts(int value)
        {
            totalNumProducts = value;
        }

        //static method: Get method of totalNumProducts
        public static int GetTotalNumProducts()
        {
            return totalNumProducts;
        }
    }
```

Demo

```
static void Main(string[] args)
{
        Product product1 = new Product();

        Product.SetTotalNumProducts(Product.GetTotalNumProducts()
+ 1); //1
       Product product2 = new Product();

        Product.SetTotalNumProducts(Product.GetTotalNumProducts()
+ 1); //2
        Product product3 = new Product();

        Product.SetTotalNumProducts(Product.GetTotalNumProducts()
+ 1); //3

       Console.WriteLine($"Total Number of Products:
{Product.totalNumProducts}");// 3
}
```

Static Constructor

Static Constructor	Instance Constructor
Initialize static fields.	Initialize instance fields.
Execute only once, i.e. when first object is created for the class or when the class is accessed for the first time during the execution of main method.	Execute automatically every time when a new object is created for the class.
“Public” by default. Access modifier can’t be changed.	“Private” by default. We can use any access modifier.

Day 9

1 Static Keyword

2 Static Variable

3 Static Constructor and Method

4 Static Class

5 Boxing and Unboxing

Static Class

A static class is declared with the help of static keyword. A static class can only contain static data members, static methods, and a static constructor. It is not allowed to create objects of the static class. Static classes are sealed, means one cannot inherit a static class from another class.

Differences

Static	Non-Static
Stored in objects.	Stored in class’s memory.
Represents data related to objects.	Represents common data that belongs to all objects.

Differences

Static	Non-Static
Static class is defined using static keyword.	Non-Static class is not defined by using static keyword.
Example : `static class Author{ }`	**Example :** `class Employee { }`

Differences

Static	Non-Static
In static class, you are not allowed to create objects.	In non-static class, you are allowed to create objects using new keyword.
Example : `Author o = new Author(); //Error`	**Example :** `Employee o = new Employee();`

Differences

Static	Non-Static
The data members of static class can be directly accessed by its class name.	The data members of non-static class is not directly accessed by its class name.

Example :
`static class Author{`
`    public static void details()`
`    {`
`        Console.WriteLine("The details of Author is:");`
`    }`
`}`
`Author.details();`
	Example :
	`class Employee`
	`{`
	`    public void greet()`
	`    {`
	`        Console.WriteLine("hello everyone");`
	`    }`
	`}`
	`Employee.greet();//`**Error**

Differences

Static	Non-Static
Static class always contains static members.	Non-static class may contain both static and non-static methods.

Example :
static class Author
{
public static string name;
public static int T_no;
 	
public static void details()
{ /*...*/ }
}
	Example :
	public class Employee4
	{
	public string id;
	public static string depname;
	public static int
	employeeCounter;
	public static int
	AddEmployee() { /*.....*/ }
	}

Differences

Static	Non-Static
Static class does not contain an instance constructor. (just one parameter less constructor)	Non-static class contains an instance constructor.

Example :
Error
	Example :
	`public class Employee4`
	`{`
	`Employee4() { }`
	`public Employee4(string name, string id)`
	`{ this.name = name; this.id = id;`
	`}`

Differences

Static	Non-Static
Static class cannot inherit from another class.	Non-static class can be inherited from another class.
Example : `public static class GFG {` `static void display()` `{ }` `}` `class GFG2 : GFG {` `public static void Main(String[] args)` `{` `}` `} //Error`	**Example :** `public class employee{` `public void display()` `{ }` `}` `class program: employee` `{` `public static void Main(String[] args)` `{` `}` `}`

Differences

Static	**Non-Static**
Static variables are created when the program starts and destroyed when the program stops.	Instance variables are created when an object is created with the use of the keyword 'new' and destroyed when the object is destroyed.

Static	Non-Static
There would only be one copy of each class variable per class, regardless of how many objects are created from it.	In non-static class, you are allowed to create objects using new keyword.

Demo

```
public static class Country
{
    public static string CountryName = “Jordan";
    public static int NoOfStates = 12;

    public static int GetNoOfUnionTerritories()
    {
        return 1;
    }
}
```

Demo

```
static void Main()
{
      //access static fields
       Console.WriteLine(Country.CountryName);
       Console.WriteLine(Country.NoOfStates);
       Console.WriteLine(Country.GetNoOfUnionTerritories());

       Console.ReadKey();
}
```

شركة تحالف الإمارات للحلول التقنية ذ.م.م.
TAHALUF AL EMARAT TECHNICAL SOLUTIONS L.L.C.

Why we use static modifier?

Day 9

Boxing and Unboxing

Boxing and unboxing in C# allows developers to convert .NET data types from value type to reference type and vice versa. Converting a value type to a reference type is called boxing in C# and converting a reference type to a value type is called unboxing in C#.

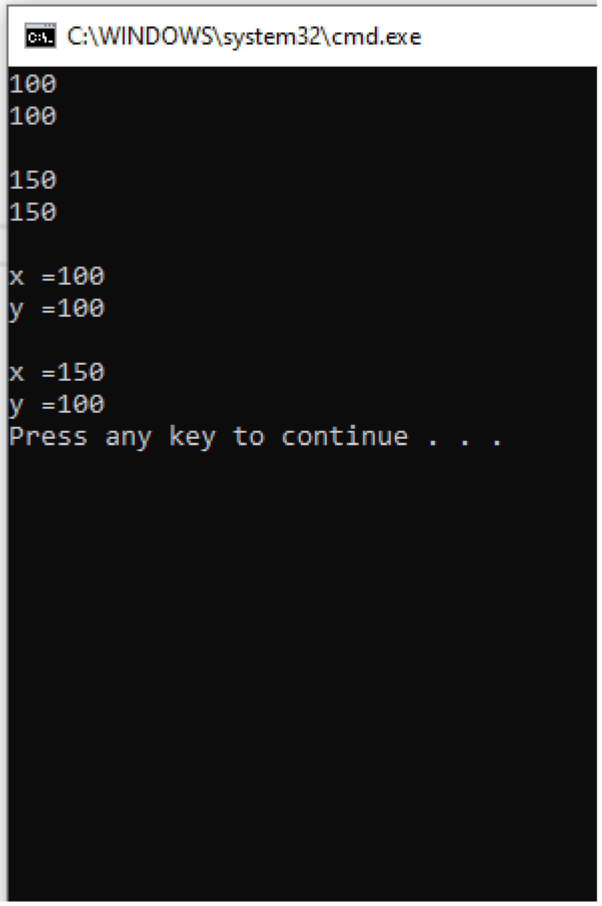
Boxing

```
static void Main()
    {
        //primitive variable
        int x = 10;

        //boxing (value-type to reference-type)
        object obj = x;

        System.Console.WriteLine(x); //Output: 10
        System.Console.WriteLine(obj); //Output: 10
        System.Console.ReadKey();
    }
```

Unboxing

```
static void Main()
        {
            //reference-type variable
            object obj = 10;

            //Unboxing (reference-type to value-type)
            int x = (int)obj;

            System.Console.WriteLine(x); //Output: 10
            System.Console.WriteLine(obj); //Output: 10
            System.Console.ReadKey();
        }
```

```
Book oBook = new Book(100);
Book o2Book = oBook;

Console.WriteLine(oBook.Id);
Console.WriteLine(o2Book.Id);
Console.WriteLine();
oBook.Id = 150;

Console.WriteLine(oBook.Id);
Console.WriteLine(o2Book.Id);
Console.WriteLine();

int x = 100;
int y = x;
Console.WriteLine("x ="+ x);
Console.WriteLine("y =" + y);
Console.WriteLine();
x = 150;
Console.WriteLine("x =" + x);
Console.WriteLine("y =" + y);
```



C:\WINDOWS\system32\cmd.exe

```
100
100

150
150

x =100
y =100

x =150
y =100
Press any key to continue . . .
```