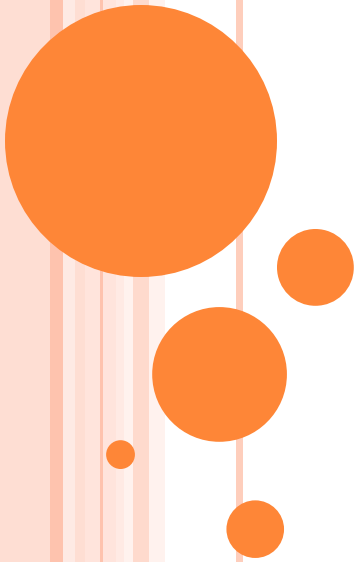
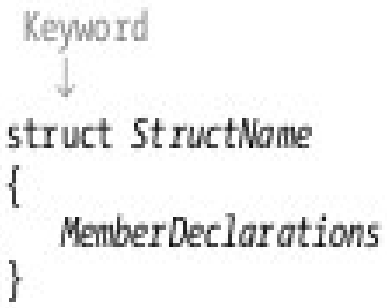


STRUCTS AND ENUMERATION

Asma Kanwal
Lecturer
Department of Computer Science
GC University, Lahore

STRCUTS

- Classes are reference types, and structs are value types.
- Structs are implicitly sealed, which means they cannot be derived from.



STRCUTS EXAMPLE

```
struct Point
  {
    public int X;
    public int Y;
  }

  class Program
  {
    static void Main()
    {
      Point first, second, third;

      first.X  = 10; first.Y = 10;
      second.X = 20; second.Y = 20;
      third.X  = first.X + second.X;
      third.Y  = first.Y + second.Y;

      Console.WriteLine("first:   {0}, {1}", first.X,  first.Y);
      Console.WriteLine("second:  {0}, {1}", second.X, second.Y);
      Console.WriteLine("third:   {0}, {1}", third.X,  third.Y);
    }
  }
```

STRUCTS WITH CONSTRCUTOR

```
struct Simple
  {
    public int X;
    public int Y;

    public Simple(int a, int b)          // Constructor with parameters
    {
      X = a;
      Y = b;
    }
  }

  class Program
  {
    static void Main()
    {          Call implicit constructor
                                          ↓
      Simple s1 = new Simple();
      Simple s2 = new Simple(5, 10);
                                          ↑
                 Call constructor
      Console.WriteLine("{0},{1}", s1.X, s1.Y);
      Console.WriteLine("{0},{1}", s2.X, s2.Y);
```

STRUCTS FIELD INITIALZATION

```
struct Simple
 {

    public int x = 0;            // Compile error
    public int y = 10;           // Compile error
 }
```

STRUCTS (IMPLICIT SEALED)

As structs are non inheritable so following modifiers are not allowed with structs

protected

internal

abstract

virtual

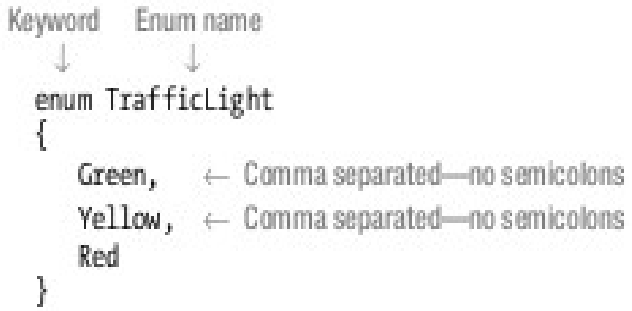
Structs As Return Values and Parameters

- **Return value :** When a struct is a return value, a copy is created and returned from the function member.
- **Value parameter:** When a struct is used as a value parameter, a copy of the actual parameter struct is created. The copy is used in the execution of the method.
- **ref and out parameters:** If you use a struct as a ref or out parameter, a reference to the struct is passed into the method so that the data members can be changed.

ENUMERATION

An enumeration, or enum, is a programmer-defined type, such as a class or a struct.

- Like structs, enums are value types and therefore store their data directly, rather than separately, with a reference and data.
- Enums have only one type of member: named constants with integer values.



ENUMERATION

Every enum type has an underlying integer type, which by default is int.

- Each enum member is assigned a constant value of the underlying type.
- By default, the compiler assigns 0 to the first member and assigns each subsequent member the value one more than the previous member.

EXAMPLE

```
enum TrafficLight
  {
     Green,
     Yellow,
     Red
  }
  Class Program {
                static void Main( ){
  TrafficLight t1 = TrafficLight.Green;
  TrafficLight t2 = TrafficLight.Yellow;
  TrafficLight t3 = TrafficLight.Red;

  Console.WriteLine("{0},\t{1}",   t1, (int) t1);
  Console.WriteLine("{0},\t{1}",   t2, (int) t2);
  Console.WriteLine("{0},\t{1}\n", t3, (int) t3);
}
}
```

Output:

Green, 0
Yellow, 1
Red,   2