

ASSIGNMENT - 20

C# Programing : delegates, nullable types

C# programing

-By

Prudhvi.

Nations Benefits.

Q. Write the points discussed about delegates in the class

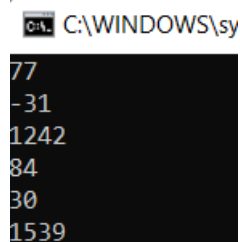
- A. A delegate is like a function pointer
- B. using delegates we can call (or) point to one or more methods
- C. when declaring a delegate, return type and parameters must match with the methods you want to point using the delegate.
- D. Benefit of delegate is that, using single call from delegate, all your methods pointing to delegate will be called.

Q. Write C# code to illustrate the usage of delegates.

CODE

```
class Program
{
    public delegate void arthm(int a, int b);
    public static void add(int a, int b)
    {
        Console.WriteLine(a+b);
    }
    public static void sub(int a, int b)
    {
        Console.WriteLine(a-b);
    }
    public static void mul(int a, int b)
    {
        Console.WriteLine(a*b);
    }
    static void Main(string[] args)
    {
        arthm arth = new arthm(add);
        arth += sub;
        arth += mul;
        // 23,54
        arth(23, 54);
        //57,27
        arth(57, 27);
    }
}
```

OUTPUT



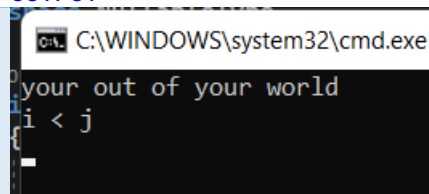
```
C:\WINDOWS\system32\cmd.exe
77
-31
1242
84
30
1539
```

Q. WACP to illustrate nullable types

```
internal class Program
{
    static void Main(string[] args)
    {
        int? a = null;
        int b = 10;
        if(a.HasValue)
            Console.WriteLine(a);
        else
            Console.WriteLine("your out of your world");
        if (Nullable.Compare<int>(a, b) < 0)
            Console.WriteLine("i < j");
        else if (Nullable.Compare<int>(a, b) > 0)
            Console.WriteLine("i > j");
        else
            Console.WriteLine("i = j");

        Console.ReadLine();
    }
}
```

OUTPUT



```
C:\WINDOWS\system32\cmd.exe
your out of your world
i < j
-
```

Q.What are nullable types in C#

nullable reference types, which complement reference types the same way nullable value types complement value types. You declare a variable to be a nullable reference type by appending a ? to the type. For example, string? represents a nullable string. You can use these new types to more clearly express your design intent: some variables must always have a value, others may be missing a value.

Q. out and ref method Parameters.

```
public class MethodPara
{
    public void SimpelOut(out int a)
    {
        a = 10;
    }
    public void simpelref(ref int b)
    {
        b = 12;
    }
}
```

```

    }
    internal class Program
    {
        static void Main(string[] args)
        {
            MethodPara methodPara = new MethodPara();
            int a;
            int b=17;
            methodPara.SimpelOut(out a);
            methodPara.simpelref(ref b);

            Console.WriteLine(b);
        }
    }
}

```

Q. write a C# program to illustrate the same.

CODE

```

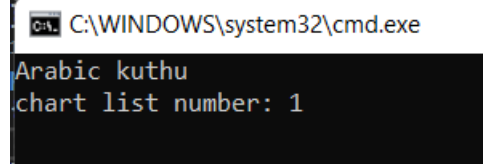
public class MethodPara
{
    public void SimpelOut(out string a)
    {
        a = "Arabic kuthu";
    }
    public void simpelref(ref int b)
    {
        b = 12;
    }
}
internal class Program
{
    static void Main(string[] args)
    {
        MethodPara methodPara = new MethodPara();
        string ac;
        int b=17;

        methodPara.SimpelOut(out ac);
        methodPara.simpelref(ref b);

        Console.WriteLine(ac);
        Console.WriteLine("chart list number: "+b);
    }
}

```

OUTPUT



```

C:\WINDOWS\system32\cmd.exe
Arabic kuthu
chart list number: 1

```

Ref: The ref keyword passes arguments by reference. It means any changes made to this argument in the method will be reflected in that variable when control returns to the calling method.

Out: The out keyword passes arguments by reference. This is very similar to the ref keyword