

Lambda Expression in C#

A lambda expression is an anonymous function that you can use to create delegates or expression tree types. By using lambda expressions, you can write local functions that can be passed as arguments or returned as the value of function calls. Lambda expressions are particularly helpful for writing LINQ query expressions. Lambda expression you can use in two case one when you want to make tree business logic and second when you want simplify code in delegate. In order to understand lambda function, just have quick look of Delegate in C#

What is delegate in C#?

A delegate in C# is similar to a function pointer in C or C++. Using a delegate allows the programmer to encapsulate a reference to a method inside a delegate object. The delegate object can then be passed to code which can call the referenced method, without having to know at compile time which method will be invoked. Unlike function pointers in C or C++, delegates are object-oriented, type-safe, and secure.

Let's create a simple delegate that will called square function.

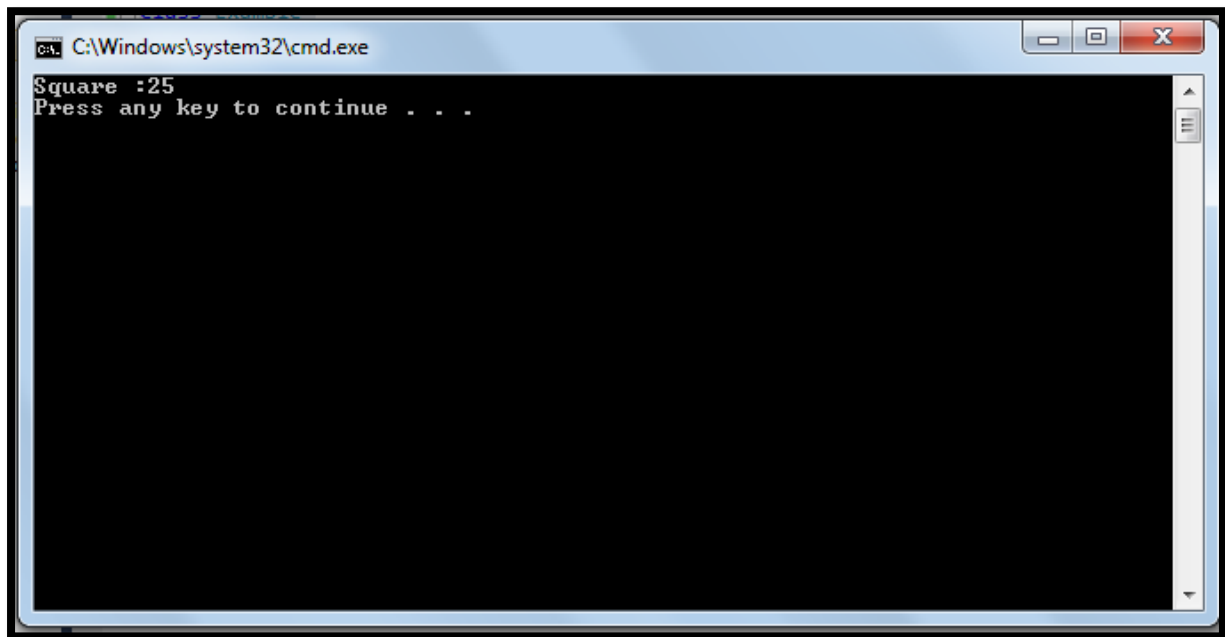
Code:

```
using System;
class Example
{
    delegate int delPointer(int i);
    static int Square(int x)
    {
        return x * x;
    }

    static delPointer obj = Square; //Delegate pointing to Square method..

    public static void Main()
    {
        //Invoking deleaget
        int Result = obj(5);
        Console.WriteLine("Square :" + Result);
    }
}
```

Output:



```
C:\Windows\system32\cmd.exe
Square :25
Press any key to continue . . .
```

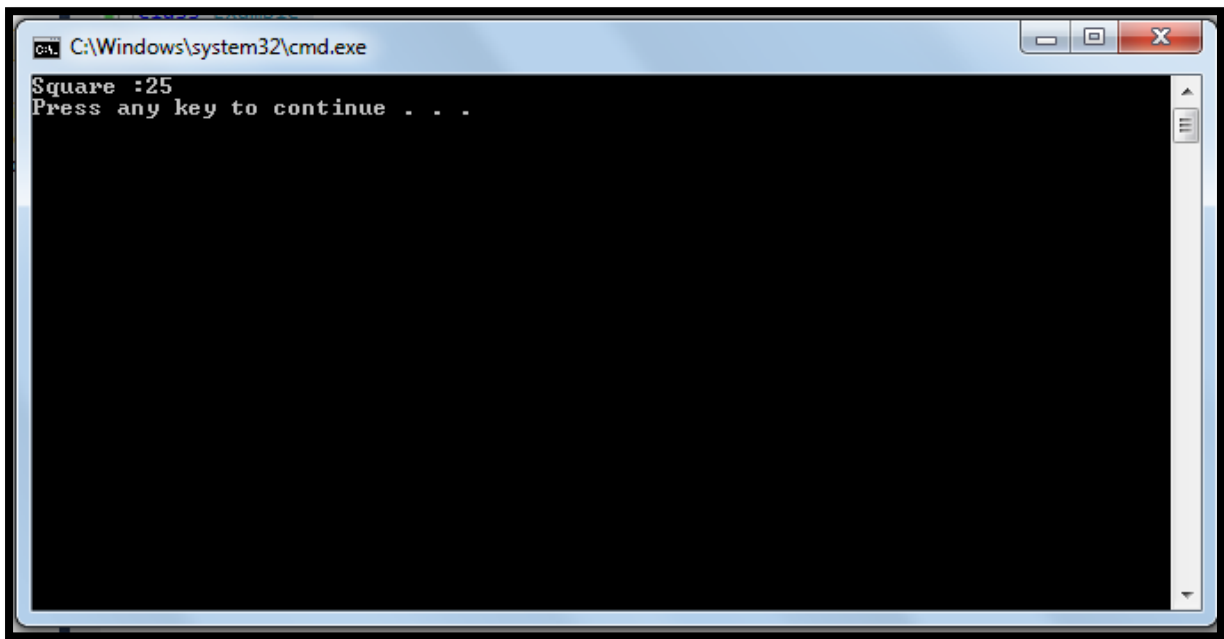
Now let's simplify above code using lambda expression.

To create a lambda expression, you specify input parameters (if any) on the left side of the lambda operator `=>`, and you put the expression or statement block on the other side. For example, the lambda expression `x => x * x` specifies a parameter that's named `x` and returns the value of `x` squared. You can assign this expression to a delegate type, as the following example shows:

Code:

```
using System;
class Example
{
    delegate int del(int i);
    static void Main(string[] args)
    {
        del myDelegate = x => x * x;
        int j = myDelegate(5); //j = 25
        Console.WriteLine("Square:"+j);
    }
}
```

Output:



```
C:\Windows\system32\cmd.exe
Square :25
Press any key to continue . . .
```

The image shows a screenshot of a Windows command prompt window. The title bar at the top reads "C:\Windows\system32\cmd.exe". The main area of the window is black with white text. The text displayed is "Square :25" on the first line and "Press any key to continue . . ." on the second line. The window has standard Windows window controls (minimize, maximize, close) in the top right corner.

Prashant Nimbare

Faculty

NIIT GT