Anonymous Method

As the name suggests, an anonymous method in C# is a method without having a name. The Anonymous methods in C# can be defined using the keyword `delegate` and can be assigned to a variable of the delegate type. An anonymous method is an "inline" statement or expression that can be used wherever a delegate type is expected. You can use it to initialize a named delegate or pass it instead of a named delegate type as a method parameter.

To bind a delegate with a method, first, we need to create an instance of a delegate and when we create the instance of a delegate, we need to pass the method name as a parameter to the delegate constructor, and it is the function the delegate will point to. An anonymous method is also related to a delegate. Without binding a named block (function) to a delegate, we can also bind a code block to a delegate means an unnamed code blocked to a delegate which is nothing but an anonymous method in C#.

Example:

```
using System;
namespace DelegateDemo
{
    public class AnonymousMethods
    {
        public delegate string GreetingsDelegate(string name);
        static void Main(string[] args)
        {
            string Message = "Welcome to SSM Infotech Solutions Pvt Ltd";
            GreetingsDelegate gd = delegate(string name)
            {
                return "Hello " + name + " " + Message;
            };
            string GreetingsMessage = gd.Invoke("Dhaval");
            Console.WriteLine(GreetingsMessage);
            Console.ReadKey();
        }
    }
}
// Output:
//    Hello Dhaval Welcome to SSM Infotech Solutions Pvt Ltd
```

Limitations: An anonymous method in C# *cannot contain any jump statement* like goto, break or continue. It *cannot access the ref or out* parameter of an outer method. The Anonymous methods *cannot have or access the unsafe code.*

Points to remember:

- The anonymous methods *can be defined using the delegate keyword.*
- An anonymous method *must be assigned to a delegate type.*
- This method *can access outer variables or functions.*
- An anonymous method *can be passed as a parameter.*
- This method *can be used as event handlers.*

Example:

```
using System;
using System.Collections;
namespace AnonymousMethodExample
{
    public class Program
    {
        public static void Main()
        {
            //Step2: Create a collection of List of Employees
            List<Employee> listEmployees = new List<Employee>()
            {
                new Employee{ ID = 101, Name = "Rahul", Gender = "Male", Salary =
100000},
                new Employee{ ID = 102, Name = "Priyanka", Gender = "Female",
Salary = 200000},
                new Employee{ ID = 103, Name = "Vivek", Gender = "Male", Salary =
300000},
                new Employee{ ID = 104, Name = "Kinjal", Gender = "Female",
Salary = 400000},
                new Employee{ ID = 104, Name = "Jigar", Gender = "Male", Salary =
500000},
            };
            //Step3: An anonymous method is being passed as an argument to
            // the Find() method of List Collection.
            Employee employee = listEmployees.Find(
                                    delegate (Employee x)
                                    {
                                        return x.ID == 103;
                                    }
                                );
            Console.WriteLine(@"ID : {0}, Name : {1}, Gender : {2}, Salary : {3}",
                employee.ID, employee.Name, employee.Gender, employee.Salary);
            Console.ReadKey();
        }
    }
    // Step1: Create a class called Employee with ID, Name, Gender and Salary
    //Properties
    public class Employee
    {
        public int ID { get; set; }
        public string Name { get; set; }
        public string Gender { get; set; }
        public double Salary { get; set; }
    }
}
// Output:
//    ID : 103, Name : Vivek, Gender : Male, Salary : 300000
```