AlmaMate

Destructors

Objectives

- In this session, you will learn to:
  - Implement destructors
  - Identify the life cycle of an object

Implementing Destructors

AlmaMate

- Destructors are special methods that are used to release the instance of a class from memory.
- A class can have only one destructor.
- The purpose of the destructor is to perform the required memory cleanup action.
- The .NET Framework automatically runs the destructor to destroy objects in the memory.

Declaration of Destructors

- A destructor has the same name as its class but is prefixed with a ~, which is the symbol of tilde.
- The following code shows the implementation of a destructor:

```
using System;
namespace Destructors
{
 class Calculator
 {
     static int number1, number2, total;
     public void AddNumber()
     {
         total= number1+number2;
```

Declaration of Destructors (Contd.)

AlmaMate

```
Console.WriteLine("The Result is {0}", total);
}
Calculator () //Constructor
{
     number1=20;
     number2=30;
     total=0;
     Console.WriteLine ("Constructor Invoked");
}
~Calculator () //Destructor
{
     Console.WriteLine ("Destructor Invoked ");
}
```

Declaration of Destructors (Contd.)

```
        static void Main(string[] args)
        {
            Calculator c1=new Calculator ();
            c1.AddNumber();
        }
    }
}
```

Declaration of Destructors (Contd.)

AlmaMate

- The decision to invoke the destructor is made by a component of the CLR known as the garbage collector.
- Garbage collection is a process that automatically frees the memory of objects that are no more in use.
- The garbage collector ensures that:
  - Objects get destroyed.
  - Only unused objects are destroyed.

Declaration of Destructors (Contd.)

AlmaMate

- C# provides the following methods to release the instance of a class from memory:
  - The `Finalize()` method:
    - Is a special method that is called from the class to which it belongs or from the derived classes.
    - Is called after the last reference to an object is released from the memory.
  - The `Dispose()` method:
    - Is called to release a resource, such as a database connection, as soon as the object using such a resource is no longer in use.
    - Implements the `IDisposable` interface.

Identifying the Life Cycle of an Object

AlmaMate

Let us now understand the life cycle of an object with the help of a program.

Identifying the Life Cycle of an Object (Contd.)

AlmaMate

- The following code allows you to determine the life cycle of an object of the `TestCalculator` class:

```
using System;
//Life Cycle of an Object
namespace Objects
{
  class TestCalculator
  {
    TestCalculator()
    {
      Console.WriteLine("Constructor Invoked");
    }
```

Identifying the Life Cycle of an Object (Contd.)

```
~TestCalculator()
{
  Console.WriteLine
  ("Destructor Invoked");
}
public static void Main(string[] args)
{
  Console.WriteLine("Main() Begins");
  TestCalculator Calc1 = new TestCalculator();
  {
```

The destructor of all the object is invoked when the garbage collector is invoked.

The `Calc1` object has function scope. Therefore, its constructor is executed after the execution of `Main()` method begins.

Identifying the Life Cycle of an Object (Contd.)

```
    Console.WriteLine("Inner Block
        Begins ");
       TestCalculator Calc2 = new
       TestCalculator();
    Console.WriteLine("Inner
        Block Ends");
              }
    Console.WriteLine("Main()
        Ends");


          }

         }
}
```

The `Calc2` object has block scope. Therefore, its constructor is executed after the inner block begins.