**Destructors In C# Programming**

A destructor is a special method which has the same name as the class but starts with the character ~ before the class name and immediately de-allocates memory of objects that are no longer required.

Following are the features of destructors:

* Destructors cannot be overloaded or inherited.
* Destructors cannot be explicitly invoked.
* Destructors cannot specify access modifiers and cannot take parameters.

A destructor is a special method which has the same name as the class but starts with the character ~ before the class name and immediately de-allocates memory of objects that are no longer required.

A C# class can also have a destructor (only one is allowed per class), which is a special method and also has the same name as the class but prefixed with a special symbol ~.

A destructor of an object is executed when the object is no longer required in order to de-allocate memory of the object.

**Following are the features of destructors:**

* Destructors cannot be overloaded or inherited.
* Destructors cannot be explicitly invoked.
* Destructors cannot specify access modifiers and cannot take parameters.

**The following code demonstrates the use of destructors:**

using System;

class Employee

{

privateint \_empId;

private string \_empName;

privateint \_age;

private double \_salary;

Employee(int id, string name, int age, double sal)

{

Console.WriteLine(“Constructor for Employee called”);

\_empId = id;

\_empName = name;

\_age = age;

\_salary = sal;

}

~Employee()

using System;

class Employee

{

privateint \_empId;

private string \_empName;

privateint \_age;

private double \_salary;

Employee(int id, string name, int age, double sal)

{

Console.WriteLine(“Constructor for Employee called”);

}

static void Main(string[] args)

{

Employee objEmp = new Employee(1, “John”, 45, 35000);

Console.WriteLine(“Employee ID: “ + objEmp.\_empId);

Console.WriteLine(“Employee Name: “ + objEmp.\_empName);

Console.WriteLine(“Age: “ + objEmp.\_age);

Console.WriteLine(“Salary: “ + objEmp.\_salary);

}

}

**In Above Code,**

* The destructor ~Employee is created having the same name as that of the class and the constructor.
* The destructor is automatically called when the object objEmp is no longer needed to be used.
* However, you have no control on when the destructor is going to be executed.

**Source Code Of Destructors In C#**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DESTRUCTORS

{

class person

{

public string Name;

public int Age;

public person(string Name, int Age)

{

this.Name = Name;

this.Age = Age;

}

public string getName()

{

return this.Name;

}

public int getAge()

{

return this.Age;

}

~person()

{

Console.WriteLine("Destructor has been invoked !!");

}

}

class Program

{

static void Main(string[] args)

{

person Ali = new person("Ali",22);

person Anas = new person("Anas", 23);

Console.WriteLine(Ali.getName());

Console.WriteLine(Ali.getAge());

Console.WriteLine("--------------");

Console.WriteLine(Anas.getName());

Console.WriteLine(Anas.getAge());

//Console.ReadLine();

}

}

}