Inheritance

```
namespace Inheritance
{
    public class Employee
    {
        public string FirstName;
        public string LastName;
        public string Email;

        public Employee()
        {
            Console.WriteLine("Parent Class is called...");
        }
        public Employee(string firstName, string lastName, string email)
        {
            FirstName = firstName;
            LastName= lastName;
            Email = email;
        }
        public void ShowFullName()
        {
            Console.WriteLine(FirstName + " " + LastName);
        }
    }
    public class FullTimeEmployee : Employee
    {
        public float YearlySalery;
        public FullTimeEmployee()
        {
            Console.WriteLine("Child class FTE is called...");
        }
    }

    public class PartTimeEmployee : Employee
    {
        public float HourlySalery;
        public PartTimeEmployee() :
base("Rohit","Tekchandani","tekchandanirohit859@gmail.com") // To call other
contructor of base class
        {
            Console.WriteLine("Child class PTE is called...");
        }
    }
    internal class Program
    {
        static void Main(string[] args)
        {
            FullTimeEmployee FTE = new FullTimeEmployee();
            PartTimeEmployee PTE = new PartTimeEmployee();
            FTE.FirstName = "Mohit";
            FTE.LastName = "Panjabi";
            FTE.Email = "panjabimohit@gmail.com";
            FTE.YearlySalery = 56994;
            PTE.HourlySalery = 66;
            FTE.ShowFullName();
            PTE.ShowFullName();
            Console.ReadLine();
        }
    }
}
```

```
3544] Inheritance.exe
C:\Users\Admin\Desktop\CS_syntex\Inh
Parent Class is called...
Child class FTE is called...
Child class PTE is called...
Mohit Panjabi
Rohit Tekchandani
```

Why inheritance

```
public class FullTimeEmployee
{
    string FirstName;
    string LastName;
    string Email;
    float YearlySalary;

    public void PrintFullName()
    {

    }
}
```

```
public class PartTimeEmployee
{
    string FirstName;
    string LastName;
    string Email;
    float HourlyRate;

    public void PrintFullName()
    {

    }
}
```

A lot of code between these 2 classes is duplicated

Using inheritance

```
public class Employee
{
    string FirstName;
    string LastName;
    string Email;

    public void PrintFullName()
    {

    }
}
```

Move all the common code into base Employee class

```
public class FullTimeEmployee
{
    float YearlySalary;
}
```

```
public class PartTimeEmployee
{
    float HourlyRate;
}
```

FullTime and PartTime employee specific code in the respective derived classes

Why Inheritance

Pillars of Object Oriented Programming

1. Inheritance
2. Encapsulation
3. Abstraction
4. Polymorphism

1. Inheritance is one of the primary pillars of object oriented programming.
2. It allows code reuse.
3. Code reuse can reduce time and errors.

Note: You will specify all the common fields, properties, methods in the base class, which allows reusability. In the derived class you will only have fields, properties and methods, that are specific to them.

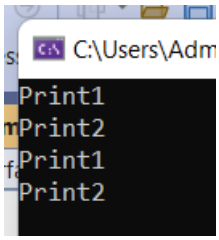
Inheritance Syntax

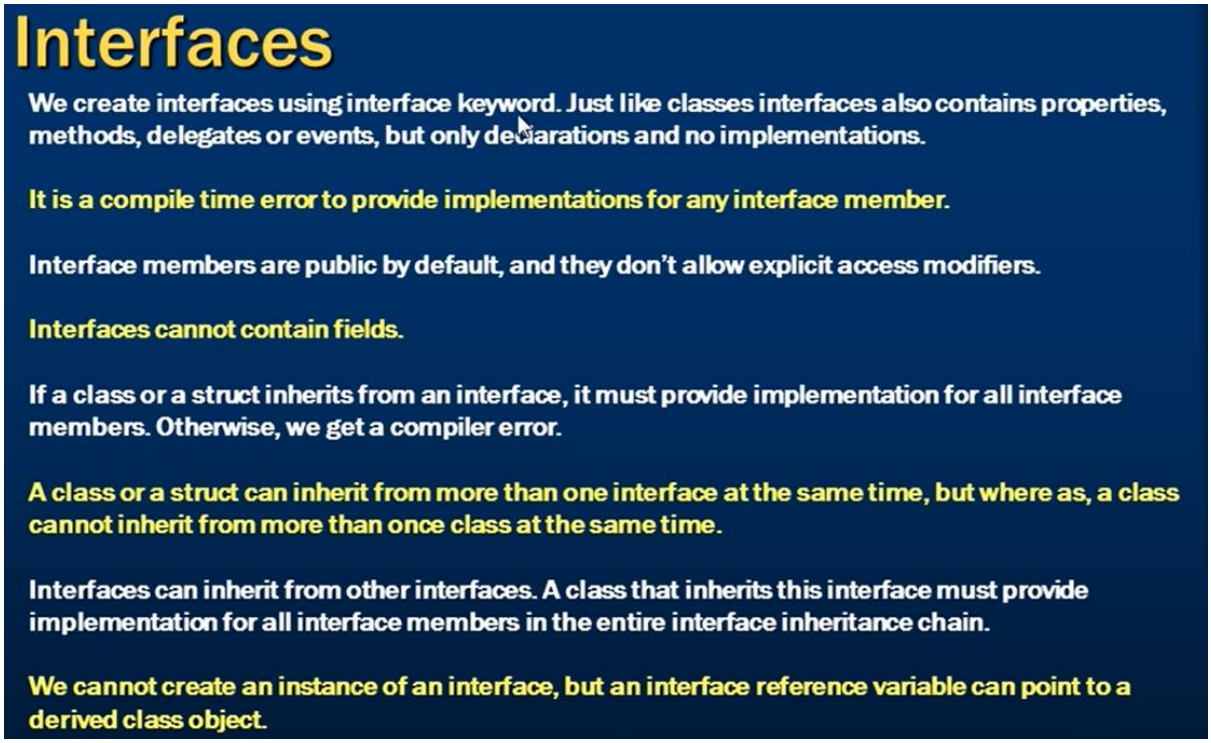
```
public class ParentClass
{
    // Parent Class Implementation
}

public class DerivedClass : ParentClass
{
    // ChildClass Implementation
}
```

1. In this example DerivedClass inherits from ParentClass.
2. C# supports only single class inheritance.
3. C# supports multiple interface inheritance.
4. Child class is a specialization of base class.
5. Base classes are automatically instantiated before derived classes.
6. Parent Class constructor executes before Child Class constructor.

Interface

```
namespace Interface
{
    interface Icustomer1
    {
        void print1();
    }

    interface Icustomer2
    {
        void print2();
    }

    public class Customer : Icustomer1,Icustomer2
    {
        public void print1()
        {
            Console.WriteLine("Print1");
        }
        public void print2()
        {
            Console.WriteLine("Print2");
        }
    }
    internal class Program
    {
        static void Main(string[] args)
        {
            Customer C1 = new Customer();
            Icustomer1 C2 = new Customer();
            Icustomer2 C3 = new Customer();

            C1.print1();
            C1.print2();
            C2.print1();
            C3.print2();

            Console.ReadLine();
        }
    }
}
```





Introduction

Sample Programe:

```
// Namespace Declaration
using System;

class Pragim
{
    public static void Main()
    {
        // Write to console
        Console.WriteLine ("Welcome to PRAGIM Technologies!");
    }
}
```