Inheritance:

Why ? For reusability , Save efforts , cost , time

Class Employee

{

}

Class PattimeEmployee {}

Class FullTime{}

// Private members are never inherited

Class A

{

Int z;

}

Class B : A

{

Int y;

}

Class B will have 1 variable : y

Class A

{

Public Int z;

}

Class B : A

{

Int y;

}

Class B will have 2 variable : y,z

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp3

{

class Student

{

int rn;

string name;

public void GetDetails()

{

Console.WriteLine("Enter RollNo");

rn = Convert.ToByte(Console.ReadLine());

Console.WriteLine("Enter Name");

name = Console.ReadLine();

}

public void DisplayDetails()

{

Console.WriteLine("RollNo is " + rn);

Console.WriteLine("Name is " + name);

}

}

class ParttimeStudent : Student

{

string course;

public void GetPartimeStudentDetails()

{

Console.WriteLine("Enter Course");

course = Console.ReadLine();

}

public void DisplayParttimeStudentDetails()

{

Console.WriteLine("Course is " + course);

}

}

}

Method Overriding : Redefining methods of base / parent class in child classes

Here , signature should be same in parent & child classes.

Its is only possible with Inheritance

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp3

{

class Student

{

int rn;

string name;

public void GetDetails()

{

Console.WriteLine("Enter RollNo");

rn = Convert.ToByte(Console.ReadLine());

Console.WriteLine("Enter Name");

name = Console.ReadLine();

}

public void DisplayDetails()

{

Console.WriteLine("RollNo is " + rn);

Console.WriteLine("Name is " + name);

}

}

class ParttimeStudent : Student

{

string course;

public void GetDetails()

{

base.GetDetails();

Console.WriteLine("Enter Course");

course = Console.ReadLine();

}

public void DisplayDetails()

{

base.DisplayDetails();

Console.WriteLine("Course is " + course);

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp3

{

class Program

{

static void Main(string[] args)

{

ParttimeStudent parttime = new ParttimeStudent();

parttime.GetDetails();

// parttime.GetPartimeStudentDetails();

parttime.DisplayDetails();

// parttime.DisplayParttimeStudentDetails();

}

}

}

CONSTRUCTORS ARE NEVER INHERIETED

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp3

{

class Student

{

int rn;

string name;

public Student() { }

public Student(int rn , string name)

{

this.rn = rn;

this.name = name;

}

public void GetDetails()

{

Console.WriteLine("Enter RollNo");

rn = Convert.ToByte(Console.ReadLine());

Console.WriteLine("Enter Name");

name = Console.ReadLine();

}

public void DisplayDetails()

{

Console.WriteLine("RollNo is " + rn);

Console.WriteLine("Name is " + name);

}

}

class ParttimeStudent : Student

{

string course;

**public ParttimeStudent() : base () { }**

**public ParttimeStudent (int rn , string name ,**

**string course)**

**: base(rn , name)**

**{**

**this.course = course;**

**}**

public void GetDetails()

{

base.GetDetails();

Console.WriteLine("Enter Course");

course = Console.ReadLine();

}

public void DisplayDetails()

{

base.DisplayDetails();

Console.WriteLine("Course is " + course);

}

}

}

Protected Variables

Private variables > NEVER INHERITED

Public Variables > INHERITED , THEY ARE ACCESSIBLE OUTSIDE THE CLASS

Protected Variables : INHERITED , BUT ARE NOT ACCESSIBLE OUSIDE THE CLASS

Internal : Within the Assembly