Namespace : Collection of logically related classes

We can store class with same name in one project in different namespace

We should store logically related classes in one namespace

using System;

using System.Collections;

// namespace

namespace A1

{

class student

{

}

}

namespace A2

{

class student

{

}

}

namespace Payroll

{

class Employee

{

}

class Salary

{

}

}

class Program

{

static void Main()

{

A1.student s = new A1.student();

A2.student s1 = new A2.student();

}

}

Readonly > Readonly variable are different for all the objects. There is a separate copy of this variable for different objects. We can not change its value. We can assign it a value either while declaring or within any constructor.

Static > Static variable are shared by all the objects. There is a single copy of this variable. We can change its value

Const > Const variable are shared by all the objects. There is a single copy of this variable. We can not change its value. When we declare it, it is mandatory to assign a value to this variable

using System;

using System.Collections.Generic;

using System.Linq;

using System.Security.AccessControl;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp2

{

class Student

{

readonly string city="Delhi";

readonly int rn;

string name;

string address;

static string batch;

const string course ="DotNet";

public Student() { rn = 9; }

public Student(int rn , string name, string address)

{ this.rn = int.Parse(Console.ReadLine());

this.rn = rn;

this.name = name;

this.address = address;

}

public void GetStudentDetails()

{

//Console.WriteLine("Enter RollNo");

//rn = int.Parse(Console.ReadLine());

// course = "Java";

batch = "B001";

// rn = 90;

}

public void DisplayStudentDetails()

{

Console.WriteLine("ROllNo " + rn);

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

Console.WriteLine("Batch is " + batch);

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

}

}

}

using ConsoleApp2;

using System;

using System.Collections;

// namespace

namespace A1

{

class student

{

}

}

namespace A2

{

class student

{

}

}

namespace Payroll

{

class Employee

{

}

class Salary

{

}

}

class Program

{

static void Main()

{

A1.student s = new A1.student();

A2.student s1 = new A2.student();

Student student = new Student();

student.GetStudentDetails();

student.DisplayStudentDetails();

Student student1 = new Student(1, "Ajay", "Delhi");

Student student2 = new Student(2, "Ajay", "Delhi");

}

}

Inheritance : Deriving Features of one class to other

Advantage : Reusabilty , save time , efforts

Parent class / Base Class / Super class is the class from where you are inheriting

Child Class / Derived Class / Sub class is the class into which you inherit the features

Class Base {}

Class Child : Base {}

Single A -- > B

Multilevel > A -> B > C

Hierarchial > A > B , A>C

Multiple > A + B > C Not allowed in C# because of Diamond Problem > because of ambiguity

Hybrid > Not allowed

Employees > Part time / Full Time

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.CompilerServices;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp2

{

class Employee

{

int id;

string name;

string address;

string dept;

public Employee() { }

public Employee(int id, string name, string address, string dept)

{

this.id = id;

this.name = name;

this.address = address;

this.dept = dept;

}

//public Employee(int id, string name, string address, string dept)

//{

// this.id = id;

// this.name = name;

// this.address = address;

// this.dept = dept;

//}

public void GetEmployeeDetails()

{

Console.WriteLine("Enter Employee ID");

id = int.Parse(Console.ReadLine());

Console.WriteLine("Enter Name");

name = Console.ReadLine();

Console.WriteLine("Enter Address");

address = Console.ReadLine();

Console.WriteLine("Enter Department");

dept = Console.ReadLine();

}

public void DisplayEmployeeDetails()

{

Console.WriteLine("ID is " + id);

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

Console.WriteLine("Address is " + address);

Console.WriteLine("Deprtament is " + dept);

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp2

{

class PartTimeEmployee : Employee

{

string projectName;

string projectduration;

//public PartTimeEmployee(string projectName,

// string projectduration) : base() {

// this.projectName = projectName;

// this.projectduration = projectduration;

//}

public PartTimeEmployee(int id, string name, string address,

string dept, string projectName,

string projectduration) : base(id, name, address, dept)

{

this.projectName = projectName;

this.projectduration = projectduration;

}

public void GetPartimeEMployeeDetails()

{

Console.WriteLine("Enter Project Name");

projectName = Console.ReadLine();

Console.WriteLine("Enter Project Duration");

projectduration = Console.ReadLine();

}

public void DisplayPartTimeEMployeeDetails()

{

Console.WriteLine("Project Name is " + projectName );

Console.WriteLine("Project durtion is " + projectduration);

}

}

}

----------\

class Program

{

static void Main()

{

// Employee emp = new Employee();

Console.WriteLine("EMployee Class");

Employee emp1 = new Employee(1, "Ajay", "Delhi", "HR");

//emp1.GetEmployeeDetails();

emp1.DisplayEmployeeDetails();

Console.WriteLine("Parttime Employee Class");

PartTimeEmployee pt = new PartTimeEmployee(2, "Depak", "Delhi", "ACcts", "SalesRepo", "2 Months");

pt.DisplayEmployeeDetails();

pt.DisplayPartTimeEMployeeDetails();

}

Method Overriding means : Redefining methods of base class in child class

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp2

{

class PartTimeEmployee : Employee

{

string projectName;

string projectduration;

//public PartTimeEmployee(string projectName,

// string projectduration) : base() {

// this.projectName = projectName;

// this.projectduration = projectduration;

//}

public PartTimeEmployee(int id, string name, string address,

string dept, string projectName,

string projectduration) : base(id, name, address, dept)

{

this.projectName = projectName;

this.projectduration = projectduration;

}

//public void GetPartimeEMployeeDetails()

//{

// Console.WriteLine("Enter Project Name");

// projectName = Console.ReadLine();

// Console.WriteLine("Enter Project Duration");

// projectduration = Console.ReadLine();

//}

public void GetEmployeeDetails()

{

base.GetEmployeeDetails();

Console.WriteLine("Enter Project Name");

projectName = Console.ReadLine();

Console.WriteLine("Enter Project Duration");

projectduration = Console.ReadLine();

}

//public void DisplayPartTimeEMployeeDetails()

//{

// Console.WriteLine("Project Name is " + projectName );

// Console.WriteLine("Project durtion is " + projectduration);

//}

public void DisplayEmployeeDetails()

{

base.DisplayEmployeeDetails();

Console.WriteLine("Project Name is " + projectName);

Console.WriteLine("Project durtion is " + projectduration);

}

}

}

Access specifiers

Private : Which are available within the class , they are not inherited

Public : Which are available outside the class , they are inherited

Student rn name

Sports : sports name, score

Test : subject marks

Result : Print the final result Final Score= marks 1+ marks2 + score

Result r = new Result();

r.

protected : inherited in the child class but not accessible outside the clas

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

using System.Resources;

using System.Text;

using System.Threading.Tasks;

namespace PrintResult

{

class Student

{

int rn;

string name;

public void GetDetails()

{

Console.WriteLine("Enter RollNo");

rn = int.Parse(Console.ReadLine());

Console.WriteLine("ENter Name");

name = Console.ReadLine();

}

public void DisplayDetails()

{

Console.WriteLine("Roll No is" + rn);

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

}

}

class Sports : Student

{

string sportsName;

protected int score;

public void GetSportsDetails()

{

Console.WriteLine("ENter Sports Name");

sportsName = Console.ReadLine();

Console.WriteLine("ENter Score");

score = int.Parse(Console.ReadLine());

}

public void DisplaySportsDetails()

{

Console.WriteLine("Sports Name is " + sportsName);

Console.WriteLine("Score is " + score);

}

}

class Test : Sports

{

protected int marks1, marks2;

public int marks3, marks4;

public void GetMarks()

{

Console.WriteLine("ENter Marks1");

marks1 = int.Parse(Console.ReadLine());

Console.WriteLine("ENter Marks2");

marks2 = int.Parse(Console.ReadLine());

}

public void DisplayMarks()

{

Console.WriteLine("Marks 1 are " + marks1);

Console.WriteLine("Marks 2 are " + marks2);

}

}

class Result : Test

{

//Student s = new Student();

//Sports s1 = new Sports();

//Test t = new Test();

int finalscore;

public void DisplatFinalResult()

{

finalscore = marks1 + marks2 + score;

Console.WriteLine("Final Score is " + finalscore);

}

}

class a

{

Test t = new Test();

public void Get()

{

}

}

}

Class Library : It is a file which only contains methods > We get a dll from here

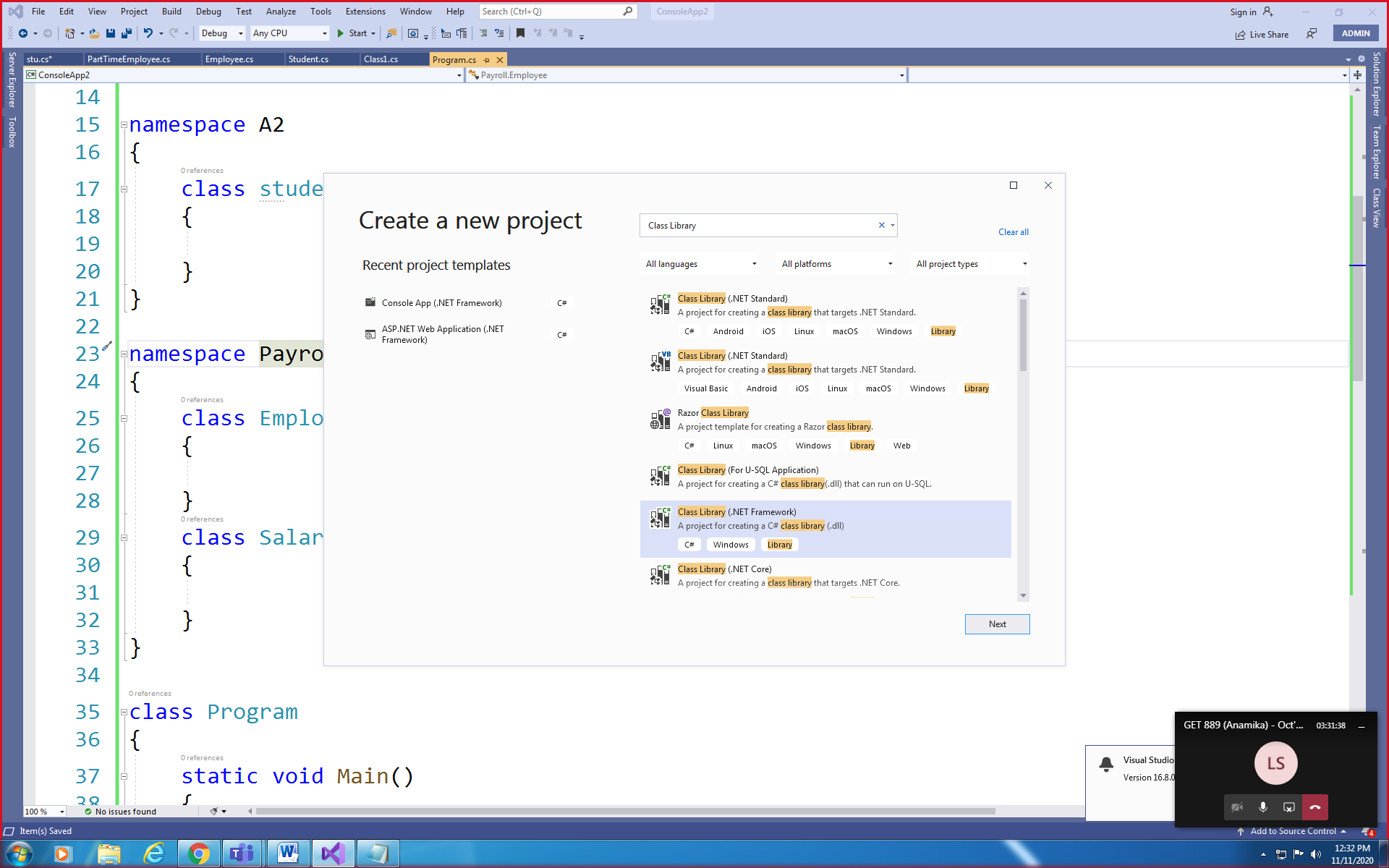
Function : Set of statements > Reusability

Project > assembly > Unit of deployment

.exe / .dll

.exe > which we can execute

.dll > can not execute , it does not Main method



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace MyFunctions

{

public class NumericFunctions

{

public int add(int x, int y)

{

return x + y;

}

public bool IsEven(int num)

{

if (num % 2 == 0)

return true;

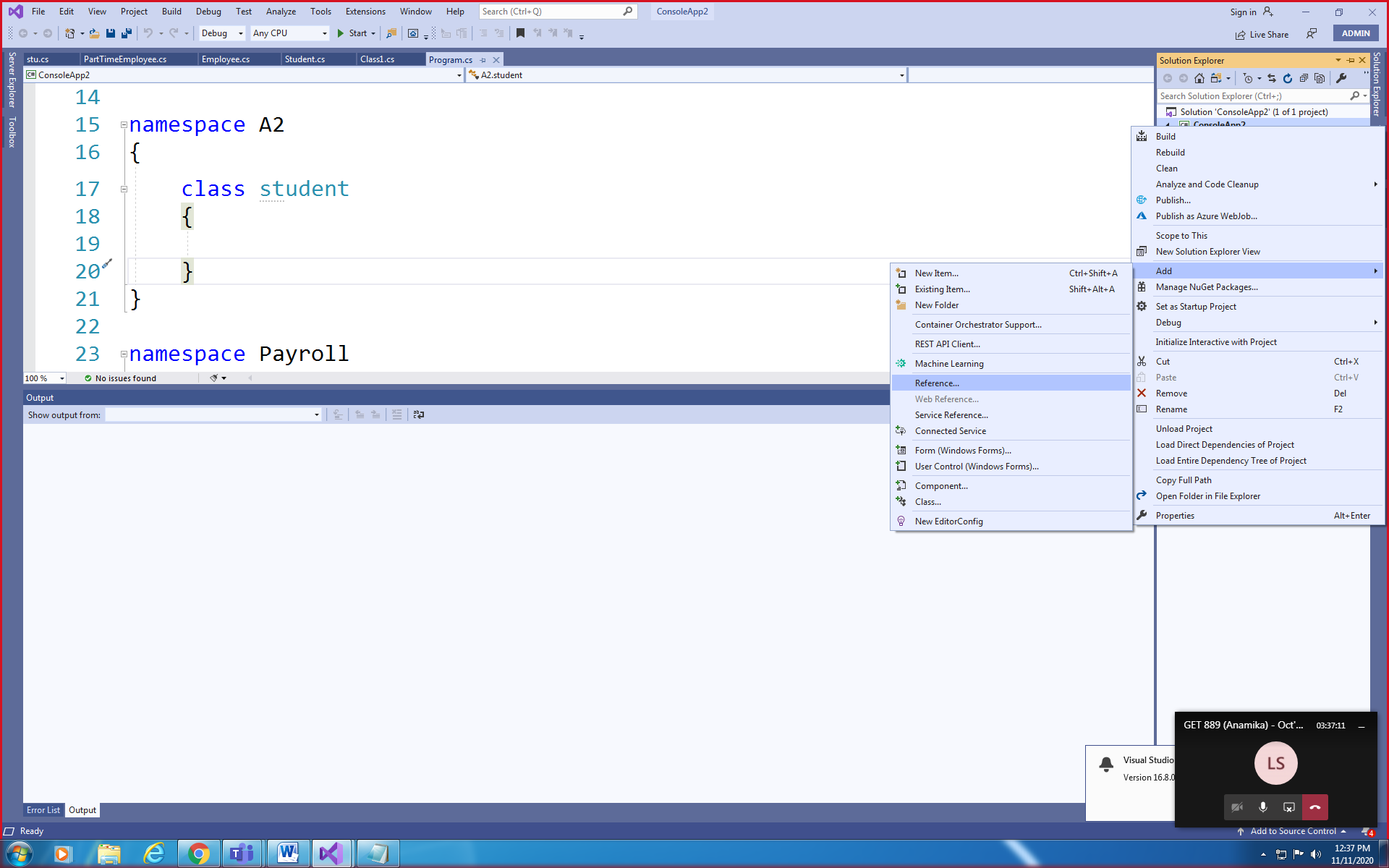
else

return false;

}

}

}



Console.WriteLine(x.add(10,2));

Console.WriteLine(x.IsEven(10));

C:\Users\Win7\source\repos\MyFunctions\MyFunctions\bin\Debug