using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System;

//using MyCustomConsole;

//using Fruits;

//using Outter.Middle.Inner;

//namespace MyCustomConsole

//{

// public class Console

// {

// public static void WriteLine(string text = "")

// {

// System.Console.WriteLine(text);

// }

// }

//}

//namespace Cs\_Lesson7

//{

// internal class Program

// {

// static void Main(string[] args)

// {

// System.Console.WriteLine("Salam {0}","Dunya");

// MyCustomConsole.Console.WriteLine("Salam");

// }

// }

//}

//namespace Fruits

//{

// class Apple

// {

// }

// class Mango

// {

// }

//}

//namespace Cs\_Lesson7

//{

// class Program

// {

// static void Main(string[] args)

// {

// var apple = new Fruits.Apple();

// }

// }

//}

#region Example 2

//namespace Outter

//{

// namespace Middle

// {

// class Samsung

// {

// public Samsung()

// {

// }

// }

// namespace Inner

// {

// class A

// {

// }

// class B

// {

// }

// }

// }

//}

//namespace Network

//{

// namespace HTTP

// {

// namespace HTTPClient

// {

// public class HTTPClientResponse

// {

// }

// }

// }

//}

//namespace Project

//{

// using HNC = Network.HTTP.HTTPClient;

// public class Program

// {

// static void Main(string[] args)

// {

// //var obj = new B();

// HNC.HTTPClientResponse asd = new HNC.HTTPClientResponse();

// }

// }

//}

#endregion

#region Oveerriding

//namespace Project

//{

// class Student

// {

// public string Name { get; set; }

// public string Surname { get; set; }

// public int Age { get; set; }

// public override string ToString()

// {

// return $"Name : {Name} Surname : {Surname} Age : {Age}";

// }

// public override bool Equals(object obj)

// {

// if (obj is Student other)

// {

// return this.Name == other.Name && this.Age == other.Age && this.Surname == other.Surname;

// }

// return false;

// }

// }

// class Program

// {

// static void Main(string[] args)

// {

// Student student = new Student()

// {

// Age = 23,

// Name = "John",

// Surname = "Johnlu"

// };

// Student student2 = new Student()

// {

// Age = 23,

// Name = "John",

// Surname = "Johnlu"

// };

// //Console.WriteLine(student);

// //Console.WriteLine(student);

// //Console.WriteLine(student.GetHashCode());

// //Student student2 = student;

// //Console.WriteLine(student2.GetHashCode());

// Console.WriteLine(student.Equals(student2));

// Console.WriteLine(Object.ReferenceEquals(student,student2));

// }

// }

//}

#endregion

#region OperatorOverloading

namespace Microsoft

{

public class Point

{

public int X { get; set; }

public int Y { get; set; }

public override string ToString()

{

return $"X : {X} Y : {Y}";

}

public override bool Equals(object obj)

{

if (obj is Point other)

{

return other.X == this.X && other.Y == this.Y;

}

return false;

}

public static Point operator ++(Point point)

{

point.X++;

point.Y++;

return point;

}

public static Point operator --(Point point)

{

point.X--;

point.Y--;

return point;

}

public static bool operator==(Point p1, Point p2)

{

return p1.Equals(p2);

}

public static bool operator!=(Point p1, Point p2)

{

return !p1.Equals(p2);

}

public static Point operator -(Point p1, Point p2)

{

return new Point { X = p1.X - p2.X, Y = p1.Y - p2.Y };

}

public static Point operator +(Point p1, Point p2)

{

return new Point { X = p1.X + p2.X, Y = p1.Y + p2.Y };

}

public static bool operator >=(Point p1, Point p2)

{

return p1.X >= p2.X && p1.Y >= p2.Y;

}

public static bool operator <=(Point p1, Point p2)

{

return p1.X <= p2.X && p1.Y <= p2.Y;

}

}

class Vector

{

public int X { get; set; }

public int Y { get; set; }

public int Z { get; set; }

public override string ToString()

{

return $"X : {X} Y : {Y} Z : {Z}";

}

public override bool Equals(object obj)

{

if (obj is Vector other)

{

return other.X == this.X && other.Y == this.Y && other.X == this.Z;

}

return false;

}

public static Vector operator ++(Vector vector)

{

vector.X++;

vector.Y++;

vector.Z++;

return vector;

}

public static Vector operator --(Vector vector)

{

vector.X--;

vector.Y--;

vector.Z--;

return vector;

}

public static Vector operator +(Vector v1, Vector v2)

{

return new Vector { X = v1.X + v2.X, Y = v1.Y + v2.Y, Z = v1.Z + v2.Z };

}

public static Vector operator -(Vector v1, Vector v2)

{

return new Vector { X = v1.X - v2.X, Y = v1.Y - v2.Y, Z = v1.Z - v2.Z };

}

public static Vector operator \*(Vector v1, Vector v2)

{

return new Vector { X = v1.X \* v2.X, Y = v1.Y \* v2.Y, Z = v1.Z \* v2.Z };

}

public static Vector operator /(Vector v1, Vector v2)

{

return new Vector { X = v1.X / v2.X, Y = v1.Y / v2.Y, Z = v1.Z / v2.Z };

}

public static bool operator ==(Vector v1, Vector v2)

{

return v1.Equals(v2);

}

public static bool operator !=(Vector v1, Vector v2)

{

return !v1.Equals(v2);

}

public static bool operator >=(Vector v1, Vector v2)

{

return v1.X >= v2.X && v1.Y >= v2.Y && v1.Z >= v2.Z;

}

public static bool operator <=(Vector v1, Vector v2)

{

return v1.X <= v2.X && v1.Y <= v2.Y && v1.Z <= v2.Z;

}

public static bool operator >(Vector v1, Vector v2)

{

return v1.X > v2.X && v1.Y > v2.Y && v1.Z > v2.Z;

}

public static bool operator <(Vector v1, Vector v2)

{

return v1.X < v2.X && v1.Y < v2.Y && v1.Z < v2.Z;

}

class Program

{

static void Main(string[] args)

{

#region MyRegion

Point p1 = new Point()

{

X = 10,

Y = 20

};

Point p2 = new Point()

{

X = 10,

Y = 20

};

Point p3 = new Point()

{

X = 10,

Y = 30

};

//p1++;

Console.WriteLine(p1 >= p2);

Console.WriteLine(p1 >= p3);

//if (p1.Equals(p2))

//{

// Console.WriteLine("They are same");

//}

//else

//{

// Console.WriteLine("They are not same");

//}

#endregion

}

}

}

}

#endregion