(T14)討論 GenerationOperators 的 Range、Repeat、Empty。討論 Quantifiers 的 All、Any、Contains
CourseGUID: 5ba9a6fe-7475-4b0c-8b99-bbcf7f5e2e1c
==
(T14)討論 GenerationOperators 的 Range、Repeat、Empty。討論 Quantifiers 的 All、Any、Contains
==

==

0. Summary

0.

0.1.
Three popular ways to solve the problems of Contains() and Equals() and SequenceEqual() for Reference Type, ClassA

0.1.1.
Override Equals() and GetHashCode() methods in ClassA

0.1.2.
If you can not access ClassA, then
Use another overloaded version of SequenceEqual(),Contains() method which can take a sub-class of IEqualityComparer as parameter.

0.1.3.
If you can not access ClassA, then
use Select() or SelectMany() to project into a new anonymous type,
which overrides Equals() and GetHashCode() methods.

0.2.
Three popular ways to solve the problems of Compare() and Sort() for Reference Type, ClassA

0.2.1.
ClassA implement IComparable<ClassA>
and then implement
//public int CompareTo(ClassA other)

0.2.2.

If you can not access ClassA, then
use other class to implement IComparer<ClassA>
E.g.
//public class ClassACompareName: IComparer<ClassA >
and then implement
public int Compare(ClassA current, ClassA other)

0.2.3.
If you can not access ClassA, then
use anonymous type to provide the method to compare.

1.
Range, Repeat, and Empty are Generation Operators.

1.1.
Enumerable.Range(Int32 start, Int32 count)
Reference:
https://msdn.microsoft.com/en-us/library/system.linq.enumerable.range(v=vs.110).aspx
Generates a sequence of integral numbers within a specified range.
Throws ArgumentOutOfRangeException
if count is less than 0
or if start + count -1 is larger than MaxValue.
E.g.

```
IEnumerable<int> intEnumerable = Enumerable.Range(11, 10);
```

//Create IEnumerable<int>{11, 12, ..., 20}

1.2.
Enumerable.Repeat<TResult>(TResult element, Int32 count)
Reference:
https://msdn.microsoft.com/en-us/library/bb348899(v=vs.110).aspx
Generates a sequence that contains one repeated value.
E.g.

```
IEnumerable<string> strEnumerable = Enumerable.Repeat("ITHandyguy", 3);
```

// [ITHandyguy] [ITHandyguy] [ITHandyguy]

1.3.
Enumerable.Empty<TResult>()
Reference:
https://msdn.microsoft.com/en-us/library/bb341042(v=vs.110).aspx
Returns an empty IEnumerable<T> that has the specified type argument.
E.g.
Enumerable.Empty<int>() - Returns an empty IEnumerable<int>
Enumerable.Empty<string>() - Returns an empty IEnumerable<string>

2.
All, Any, and Contains are Quantifiers.

2.1.
Enumerable.All<TSource>
(this IEnumerable<TSource> source, Func<TSource, Boolean> predicate)

Reference:
https://msdn.microsoft.com/en-us/library/bb548541(v=vs.110).aspx
Determines whether all elements of a sequence satisfy a condition.
Throws ArgumentNullException
if source or predicate is null.

E.g.1.
```
int[] intArr1 = { 1, 2, 3, 4, 11, 12, 13, 14 };
bool intArr1All = intArr1.All(x => x < 5);  //False
```

E.g.2.
```
int[] intArr2 = { 1, 2, 3, 4};
bool intArr2All = intArr2.All(x => x < 5);   //True
```

2.2.
Enumerable.Any<TSource>
(this IEnumerable<TSource> source, Func<TSource, Boolean> predicate)
Reference:
https://msdn.microsoft.com/en-us/library/bb534972(v=vs.110).aspx
Determines whether any element of a sequence satisfies a condition.
E.g.1.
```
int[] intArr1 = { 1, 2, 3, 4 };
bool intArr1Any = intArr1.Any();  //True
```

E.g.2.
```
int[] intArr2 = { 1, 2, 3, 4 };
bool intArr2Any = intArr2.Any(x => x > 5);   //False
```

2.3.
Enumerable.Contains<TSource>
(this IEnumerable<TSource> source, TSource value, IEqualityComparer<TSource> comparer)
Reference:
https://msdn.microsoft.com/en-us/library/bb339118(v=vs.110).aspx
Determines whether a sequence contains a specified element by using a specified IEqualityComparer<T>.

3.
//Enumerable.SequenceEqual<TSource>
//(this IEnumerable<TSource> first, IEnumerable<TSource> second)
or
//Enumerable.SequenceEqual<TSource>
//(this IEnumerable<TSource> first, IEnumerable<TSource> second, IEqualityComparer<TSource> comparer)
Determines whether two sequences are equal
by comparing their elements
by using a specified IEqualityComparer<T>
If 2 sequences are equal,
it means both sequences has the same length,
and same values is present in the same order in both the sequences.
==

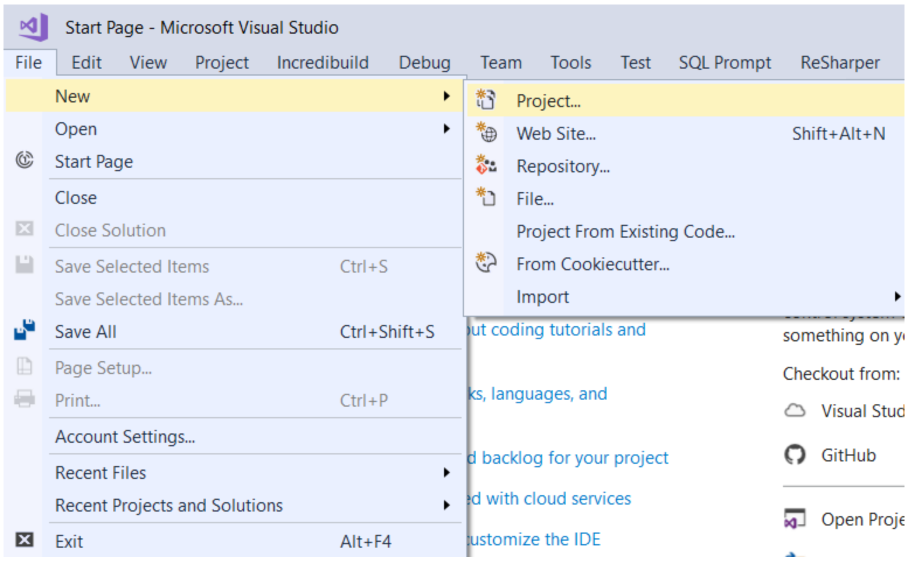
1. New Project

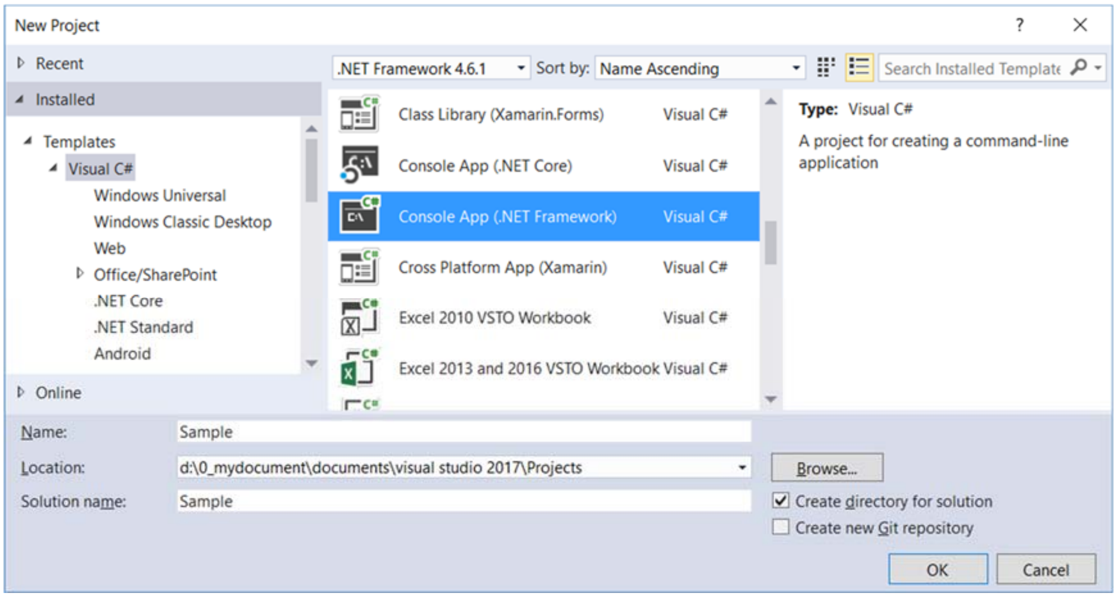
1.1. Create New Project : Sample

File --> New --> Project... -->

Visual C# --> **Console App (.Net Framework)** -->

Name: **Sample**




New Project
Recent
Installed
Templates
Visual C#
Windows Universal
Windows Classic Desktop
Web
Office/SharePoint
.NET Core
.NET Standard
Android
Online
Class Library (Xamarin.Forms) Visual C#
Console App (.NET Core) Visual C#
Console App (.NET Framework) Visual C#
Cross Platform App (Xamarin) Visual C#
Excel 2010 VSTO Workbook Visual C#
Excel 2013 and 2016 VSTO Workbook Visual C#
Search Installed Template
Browse...
OK
Cancel

.NET Framework 4.6.1

Sort by: Name Ascending

Type: Visual C#

A project for creating a command-line application

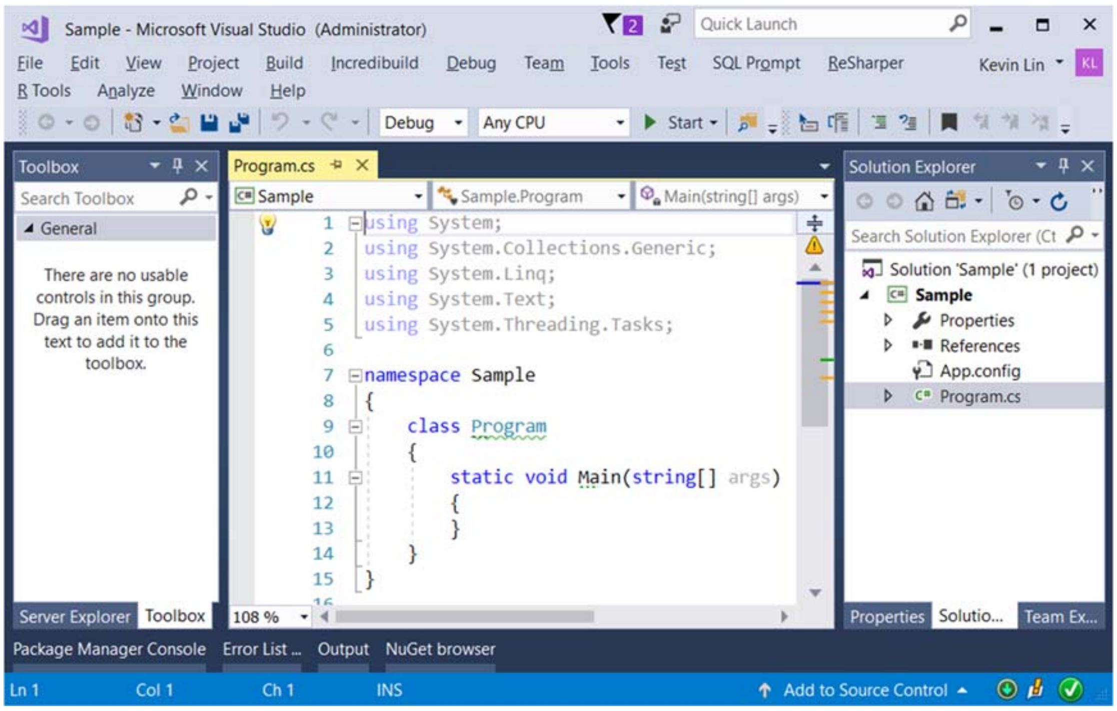
Name: Sample

Location: d:\0_mydocument\documents\visual studio 2017\Projects

Solution name: Sample

☑ Create directory for solution

☐ Create new Git repository



There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.

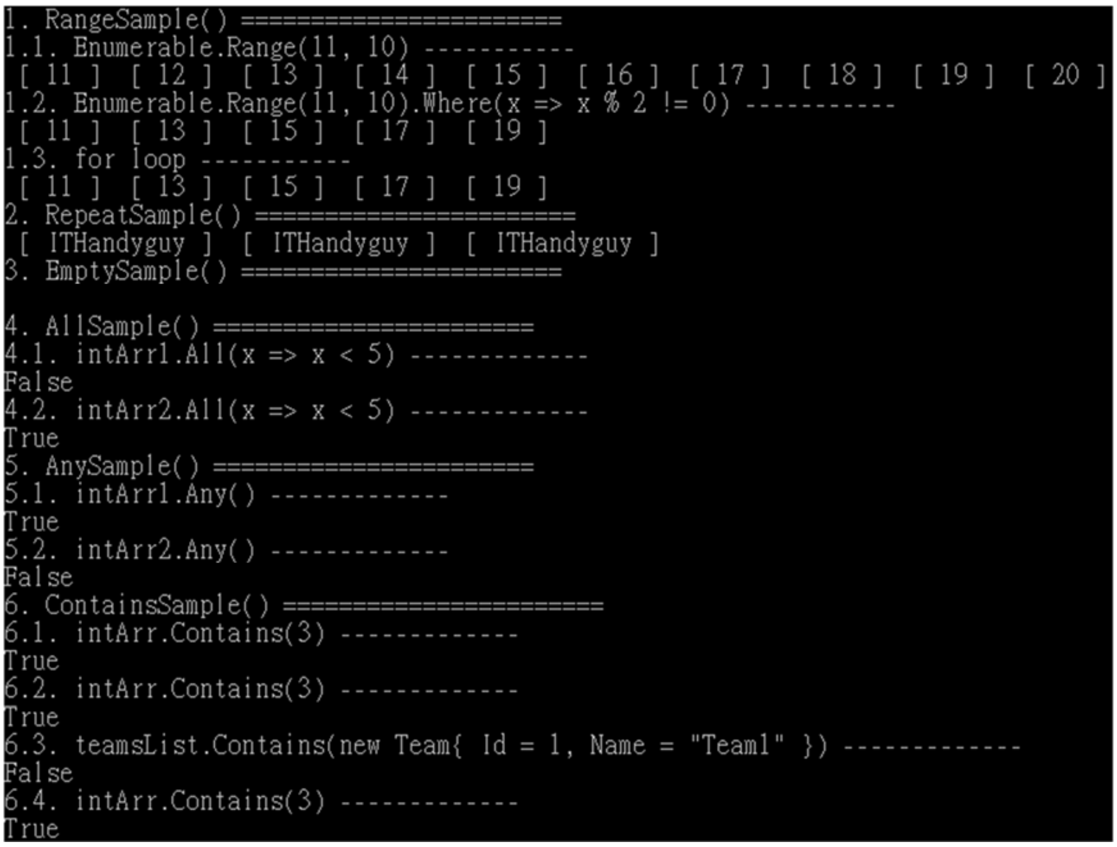
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Sample
{
    class Program
    {
        static void Main(string[] args)
        {
        }
    }
}
```

==

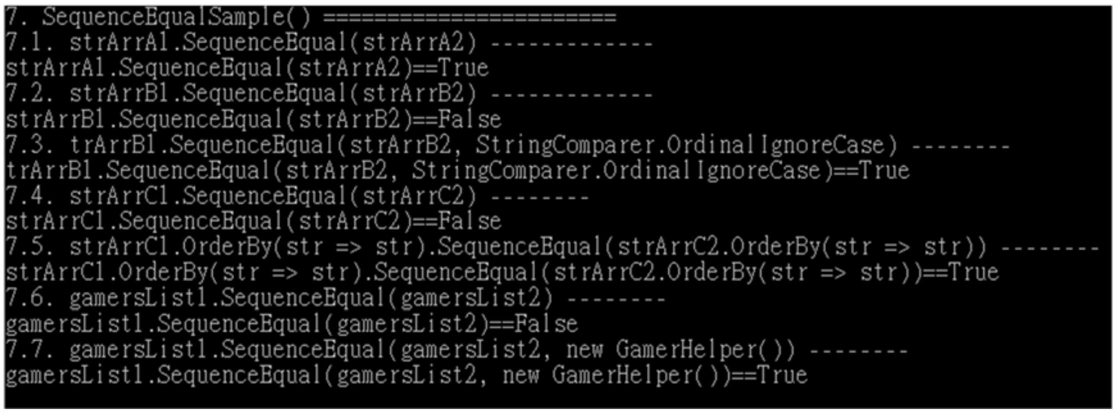
2. Sample : Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using OnLieGame;
namespace Sample
{
    class Program
     {
        static void Main(string[] args)
         {
            // 1. ===================================
            //RangeSample()
            Console.WriteLine("1. RangeSample() ====================== ");
             RangeSample();
            // 2. ===================================
            //RepeatSample()
            Console.WriteLine("2. RepeatSample() ====================== ");
             RepeatSample();
            // 3. ===================================
            //EmptySample()
            Console.WriteLine("3. EmptySample() ====================== ");
             EmptySample();
            // 4. ===================================
            //AllSample()
            Console.WriteLine("4. AllSample() ====================== ");
             AllSample();
            // 5. ===================================
            //AnySample
            Console.WriteLine("5. AnySample() ====================== ");
             AnySample();
            // 6. ===================================
            //ContainsSample
            Console.WriteLine("6. ContainsSample() ======================= ");
             ContainsSample();
            // 7. ===================================
            //SequenceEqualSample
            Console.WriteLine("7. SequenceEqualSample() ======================= ");
             SequenceEqualSample();
            Console.ReadLine();
         }


         // 1. ===================================
        //RangeSample()
        static void RangeSample()
         {
            //1.1. Enumerable.Range(11, 10) -------------------------------
            Console.WriteLine("1.1. Enumerable.Range(11, 10) ----------- ");
            IEnumerable<int> intEnumerable = Enumerable.Range(11, 10);
            foreach (int intEnumerableItem in intEnumerable)
             {
                 Console.Write($" [ {intEnumerableItem} ] ");
             }
            Console.WriteLine();
            // [ 11 ]  [ 12 ]  [ 13 ]  [ 14 ]  [ 15 ]  [ 16 ]  [ 17 ]  [ 18 ]  [ 19 ]  [ 20 ]
            //1.2. Enumerable.Range(11, 10).Where(x => x % 2 != 0) -------------------------------
            Console.WriteLine("1.2. Enumerable.Range(11, 10).Where(x => x % 2 != 0) ----------- ");
            IEnumerable<int> intEnumerableOddNumber = Enumerable.Range(11, 10).Where(x => x % 2 != 0);
            foreach (var intEnumerableOddNumberItem in intEnumerableOddNumber)
             {
                 Console.Write($" [ {intEnumerableOddNumberItem} ] ");
             }
            Console.WriteLine();
            // [ 11 ]  [ 13 ]  [ 15 ]  [ 17 ]  [ 19 ]
            //1.3. Enumerable.Range(11, 10).Where(x => x % 2 != 0) -------------------------------
            Console.WriteLine("1.3. for loop ----------- ");
            for (int i = 11; i <= 20; i++)
             {
                 if (i % 2 != 0)
                 {
                     Console.Write($" [ {i} ] ");
                 }
             }
            Console.WriteLine();
            // [ 11 ]  [ 13 ]  [ 15 ]  [ 17 ]  [ 19 ]
         }
        // 2. ===================================
        //RepeatSample()
        static void RepeatSample()
         {
            IEnumerable<string> strEnumerable = Enumerable.Repeat("ITHandyguy", 3);
            foreach (string strEnumerableItem in strEnumerable)
             {
                 Console.Write($" [ {strEnumerableItem} ] ");
             }
```

```csharp
        Console.WriteLine();
        // [ ITHandyguy ]  [ ITHandyguy ]  [ ITHandyguy ]
    }


// 3. ===================================
    //EmptySample()
    static void EmptySample()
    {
        //IEnumerable<int> intEnumerable2 =
        //    GetNull() == null ?
        //    Enumerable.Empty<int>() :
        //    GetNull();
        IEnumerable<int> intEnumerable =
            GetNull() ?? Enumerable.Empty<int>();
        foreach (int intEnumerableItem in intEnumerable)
        {
            Console.Write($" [ {intEnumerableItem} ] ");
        }
        Console.WriteLine();
    }
    static IEnumerable<int> GetNull()
    {
        return null;
    }
    //return an empty Enumerable<int>
    // 4. ===================================
    //AllSample()
    static void AllSample()
    {
        Console.WriteLine("4.1. intArr1.All(x => x < 5) ------------- ");
        int[] intArr1 = { 1, 2, 3, 4, 11, 12, 13, 14 };
        bool intArr1All = intArr1.All(x => x < 5);
        Console.WriteLine(intArr1All);
        //False
        Console.WriteLine("4.2. intArr2.All(x => x < 5) ------------- ");
        int[] intArr2 = { 1, 2, 3, 4};
        bool intArr2All = intArr2.All(x => x < 5);
        Console.WriteLine(intArr2All);
        //True
    }
    // 5. ===================================
    //AnySample
    static void AnySample()
    {
        Console.WriteLine("5.1. intArr1.Any() ------------- ");
        int[] intArr1 = { 1, 2, 3, 4 };
        bool intArr1Any = intArr1.Any();
        Console.WriteLine(intArr1Any);
        //True
        Console.WriteLine("5.2. intArr2.Any() ------------- ");
        int[] intArr2 = { 1, 2, 3, 4 };
        bool intArr2Any = intArr2.Any(x => x > 5);
        Console.WriteLine(intArr2Any);
        //False
        }
        // 6. ===================================
        //ContainsSample
        static void ContainsSample()
        {
            Console.WriteLine("6.1. intArr.Contains(3) ------------- ");
            int[] intArr = { 1, 2, 3, 4 };
            bool intArrContains3 = intArr.Contains(3);
            Console.WriteLine(intArrContains3);
            //True
            Console.WriteLine("6.2. intArr.Contains(3) ------------- ");
            string[] strArrTeamName = { "Team1", "Team2", "Team3" };
            bool strArrTeamNameContailsTeam2 =
strArrTeamName.Contains("team2", StringComparer.OrdinalIgnoreCase);
            Console.WriteLine(strArrTeamNameContailsTeam2);
            //True
            Console.WriteLine("6.3. teamsList.Contains(new Team{ Id = 1, Name = \"Team1\" }) -------------
");
            List<Team> teamsList = TeamHelper.GetSampleTeam();
            bool teamsListContainsTeam1 = teamsList.Contains(new Team{ Id = 1, Name = "Team1" });
            Console.WriteLine(teamsListContainsTeam1);
            //False
            Console.WriteLine("6.4. intArr.Contains(3) ------------- ");
            List<Gamer> gamersList = GamerHelper.GetSampleGamer();
            bool gamersListContainsName1 = gamersList.Contains(new Gamer { Id = 1, Name = "Name1", TeamId =
1 }, new GamerHelper());
            Console.WriteLine(gamersListContainsName1);
            //True
        }


       // 7. ===================================
        //SequenceEqualSample
        private static void SequenceEqualSample()
        {
            //7.1. strArrA1.SequenceEqual(strArrA2) -------------------------------
            Console.WriteLine("7.1. strArrA1.SequenceEqual(strArrA2) ------------- ");
            string[] strArrA1 = { "Name1", "Name2", "Name3" };
            string[] strArrA2 = { "Name1", "Name2", "Name3" };
            bool strArrA1SequenceEqualStrArrA2 =
                strArrA1.SequenceEqual(strArrA2);
            Console.WriteLine($"strArrA1.SequenceEqual(strArrA2)==" +
                              $"{strArrA1SequenceEqualStrArrA2}");
            //strArrA1.SequenceEqual(strArrA2)==True
            //7.2. strArrB1.SequenceEqual(strArrB2) -------------------------------
            Console.WriteLine("7.2. strArrB1.SequenceEqual(strArrB2) ------------- ");
            string[] strArrB1 = { "Name1", "Name2", "Name3" };
            string[] strArrB2 = { "name1", "name2", "Name3" };
            bool strArrB1SequenceEqualStrArrB2 =
                strArrB1.SequenceEqual(strArrB2);
            Console.WriteLine($"strArrB1.SequenceEqual(strArrB2)==" +
                              $"{strArrB1SequenceEqualStrArrB2}");
            //strArrB1.SequenceEqual(strArrB2)==False
            //7.3. trArrB1.SequenceEqual(strArrB2, StringComparer.OrdinalIgnoreCase) --------------------
            Console.WriteLine("7.3. trArrB1.SequenceEqual(strArrB2, StringComparer.OrdinalIgnoreCase) ----
---- ");
            bool strArrB1SequenceEqualStrArrB2IgnoreCase =
                strArrB1.SequenceEqual(strArrB2, StringComparer.OrdinalIgnoreCase);
            Console.WriteLine($"trArrB1.SequenceEqual(strArrB2,
StringComparer.OrdinalIgnoreCase)=={strArrB1SequenceEqualStrArrB2IgnoreCase}");
            //trArrB1.SequenceEqual(strArrB2, StringComparer.OrdinalIgnoreCase)==True
            //7.4. strArrC1.SequenceEqual(strArrC2) --------------------
            Console.WriteLine("7.4. strArrC1.SequenceEqual(strArrC2) -------- ");
            string[] strArrC1 = { "Name1", "Name3", "Name2" };
            string[] strArrC2 = { "Name2", "Name1", "Name3" };
            bool strArrC1SequenceEqualStrArrC2 =
                strArrC1.SequenceEqual(strArrC2);
            Console.WriteLine($"strArrC1.SequenceEqual(strArrC2)==" +
                              $"{strArrC1SequenceEqualStrArrC2}");
            //strArrC1.SequenceEqual(strArrC2)==False
            //7.5. strArrC1.OrderBy(str => str).SequenceEqual(strArrC2.OrderBy(str => str)) --------------
------
            Console.WriteLine("7.5. strArrC1.OrderBy(str => str).SequenceEqual(strArrC2.OrderBy(str =>
str)) -------- ");
            bool strArrC1OrderBySequenceEqualstrArrC2 = strArrC1.OrderBy(str => str)
                .SequenceEqual(strArrC2.OrderBy(str => str));
            Console.WriteLine($"strArrC1.OrderBy(str => str).SequenceEqual(strArrC2.OrderBy(str =>
str))=={strArrC1OrderBySequenceEqualstrArrC2}");
            //strArrC1.OrderBy(str => str).SequenceEqual(strArrC2.OrderBy(str => str))==True
            //7.6. gamersList1.SequenceEqual(gamersList2) --------------------
            Console.WriteLine("7.6. gamersList1.SequenceEqual(gamersList2) -------- ");
            List<Gamer> gamersList1 = GamerHelper.GetSampleGamer();
            List<Gamer> gamersList2 = GamerHelper.GetSampleGamer();
            bool gamersList1SequenceEqualGamersList2 = gamersList1.SequenceEqual(gamersList2);
            Console.WriteLine($"gamersList1.SequenceEqual(gamersList2)=={gamersList1SequenceEqualGamersLis
t2}");
            //gamersList1.SequenceEqual(gamersList2)==False
            //7.7. gamersList1.SequenceEqual(gamersList2, new GamerHelper()) --------------------
            Console.WriteLine("7.7. gamersList1.SequenceEqual(gamersList2, new GamerHelper()) -------- ");
            bool gamersList1SequenceEqualGamersList2V2 =
gamersList1.SequenceEqual(gamersList2, new GamerHelper());
            Console.WriteLine($"gamersList1.SequenceEqual(gamersList2, new
GamerHelper())=={gamersList1SequenceEqualGamersList2V2}");
            //gamersList1.SequenceEqual(gamersList2, new GamerHelper())==True
        }
    }
}


namespace OnLieGame
{
    public class Team
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public override string ToString()
        {
            return $"TeamId=={Id},TeamName={Name}";
        }
    }
    public class TeamHelper
    {
        public static List<Team> GetSampleTeam()
        {
            return new List<Team>
            {
                new Team { Id = 1, Name = "Team1"},
                new Team { Id = 2, Name = "Team2"},
                new Team { Id = 3, Name = "Team3"},
            };
        }
    }
    public class Gamer
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public int TeamId { get; set; }
        public override string ToString()
        {
            return $"GamerId=={Id},GamerName={Name},TeamId={TeamId}";
        }
    }
    public class GamerHelper : IEqualityComparer<Gamer>
    {
        public static List<Gamer> GetSampleGamer()
        {
            return new List<Gamer>
            {
                new Gamer { Id = 1, Name = "Name1", TeamId = 1 },
                new Gamer { Id = 2, Name = "Name2", TeamId = 2 },
                new Gamer { Id = 3, Name = "Name3", TeamId = 1 },
                new Gamer { Id = 4, Name = "Name4", TeamId = 1 },
                new Gamer { Id = 5, Name = "Name9", TeamId = 2 },
                new Gamer { Id = 6, Name = "Name10"}
            };
        }
        public bool Equals(Gamer x, Gamer y)
        {
            return y != null && x != null &&
                 x.Id == y.Id &&
                 x.Name == y.Name &&
                 x.TeamId == y.TeamId;
        }
        public int GetHashCode(Gamer obj)
        {
            return obj.Id.GetHashCode() ^
                 obj.TeamId.GetHashCode() ^
                 obj.Name.GetHashCode();
        }
    }
}
```



```
1. RangeSample() =======================
1.1. Enumerable.Range(11, 10) -----------
 [ 11 ]  [ 12 ]  [ 13 ]  [ 14 ]  [ 15 ]  [ 16 ]  [ 17 ]  [ 18 ]  [ 19 ]  [ 20 ]
1.2. Enumerable.Range(11, 10).Where(x => x % 2 != 0) -----------
 [ 11 ]  [ 13 ]  [ 15 ]  [ 17 ]  [ 19 ]
1.3. for loop -----------
 [ 11 ]  [ 13 ]  [ 15 ]  [ 17 ]  [ 19 ]
2. RepeatSample() =======================
 [ ITHandyguy ]  [ ITHandyguy ]  [ ITHandyguy ]
3. EmptySample() =======================

4. AllSample() =======================
4.1. intArr1.All(x => x < 5) -------------
False
4.2. intArr2.All(x => x < 5) -------------
True
5. AnySample() =======================
5.1. intArr1.Any() -------------
True
5.2. intArr2.Any() -------------
False
6. ContainsSample() =======================
6.1. intArr.Contains(3) -------------
True
6.2. intArr.Contains(3) -------------
True
6.3. teamsList.Contains(new Team{ Id = 1, Name = "Team1" }) -------------
False
6.4. intArr.Contains(3) -------------
True
```



```
7. SequenceEqualSample() =======================
7.1. strArrA1.SequenceEqual(strArrA2) -------------
strArrA1.SequenceEqual(strArrA2)==True
7.2. strArrB1.SequenceEqual(strArrB2) -------------
strArrB1.SequenceEqual(strArrB2)==False
7.3. trArrB1.SequenceEqual(strArrB2, StringComparer.OrdinalIgnoreCase) --------
trArrB1.SequenceEqual(strArrB2, StringComparer.OrdinalIgnoreCase)==True
7.4. strArrC1.SequenceEqual(strArrC2) --------
strArrC1.SequenceEqual(strArrC2)==False
7.5. strArrC1.OrderBy(str => str).SequenceEqual(strArrC2.OrderBy(str => str)) --------
strArrC1.OrderBy(str => str).SequenceEqual(strArrC2.OrderBy(str => str))==True
7.6. gamersList1.SequenceEqual(gamersList2) --------
gamersList1.SequenceEqual(gamersList2)==False
7.7. gamersList1.SequenceEqual(gamersList2, new GamerHelper()) --------
gamersList1.SequenceEqual(gamersList2, new GamerHelper())==True
```