ALL ,Any,Contains

All Operator

- ✓ All operator is used to check whether all the elements of a data source satisfy a specified condition.
- ✓ It returns a Boolean value.

```
0 references
class Program
{
    0 references
    static void Main(string[] args)
    {
        Student[] students = { new Student { Name="Kim", Marks=90 },
                new Student { Name="John", Marks=80 },
                new Student { Name="Lee", Marks=75 } };


        Console.ReadLine();
    }
}
```

```
0 references
static void Main(string[] args)
{
    Student[] students = { new Student { Name="Kim", Marks=90 },
            new Student { Name="John", Marks=80 },
            new Student { Name="Lee", Marks=75 } };

    var query = students.All(student => student.Marks > 80);

    var q = (from student in students
             select student).All(x => x.Marks > 70);


    Console.ReadLine();
}
```

Another example

```
{
    0 references
    static void Main(string[] args)
    {
        Student[] students = {
            new Student {
                Name ="Kim", Marks=90,
                        Subject = new List<Subject>(){
                            new Subject(){ SubjectName = "Math", SubjectMarks=75 },
                            new Subject(){ SubjectName = "English", SubjectMarks=80 },
                            new Subject(){ SubjectName = "Art", SubjectMarks=86 },
                            new Subject(){ SubjectName = "History", SubjectMarks=91 }
                        } },
                new Student { Name="John", Marks=80,
                    Subject = new List<Subject>(){
                            new Subject(){ SubjectName = "Math", SubjectMarks=89 },
                            new Subject(){ SubjectName = "English", SubjectMarks=91 },
                            new Subject(){ SubjectName = "Art", SubjectMarks=90 },
                            new Subject(){ SubjectName = "History", SubjectMarks=91 }
                        } },
                new Student { Name="Lee", Marks=75,
                    Subject = new List<Subject>(){
                            new Subject(){ SubjectName = "Math", SubjectMarks=75 },
                            new Subject(){ SubjectName = "English", SubjectMarks=80 },
                            new Subject(){ SubjectName = "Art", SubjectMarks=60 },
                            new Subject(){ SubjectName = "History", SubjectMarks=91 }
                        } } };
```

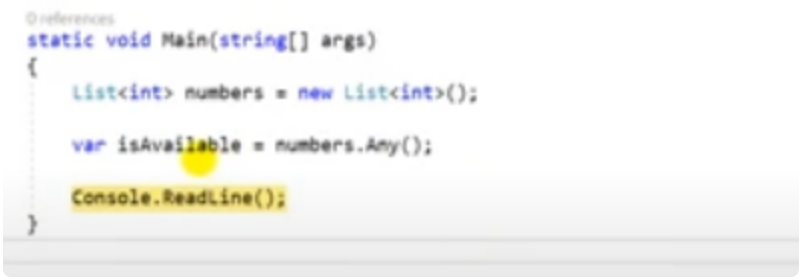
```
                            new Subject(){ SubjectName = "Art", SubjectMarks=90 },
                            new Subject(){ SubjectName = "History", SubjectMarks=91 }
                        } },
                new Student { Name="Lee", Marks=75,
                    Subject = new List<Subject>(){
                            new Subject(){ SubjectName = "Math", SubjectMarks=75 },
                            new Subject(){ SubjectName = "English", SubjectMarks=80 },
                            new Subject(){ SubjectName = "Art", SubjectMarks=60 },
                            new Subject(){ SubjectName = "History", SubjectMarks=91 }
                        } } };

        var studentss = students.Where(std => std.Subject.All(x => x.SubjectMarks > 70)).Select(std => std).ToList();

        var qs = (from std in students
                  where std.Subject.All(x => x.SubjectMarks > 70)
                  select std).ToList();

        Console.ReadLine();
    }
```

Any

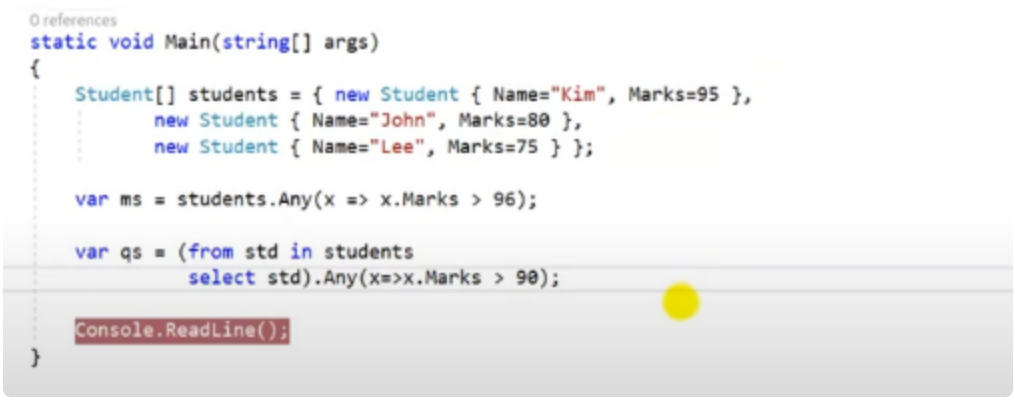
Any Operator

- ✓ Any operator is used to check whether at least one element of a data source satisfy a specified condition.
- ✓ Any is also used to check if a collection contains some data or not.
- ✓ It returns a Boolean value.

Checking list contains some elements

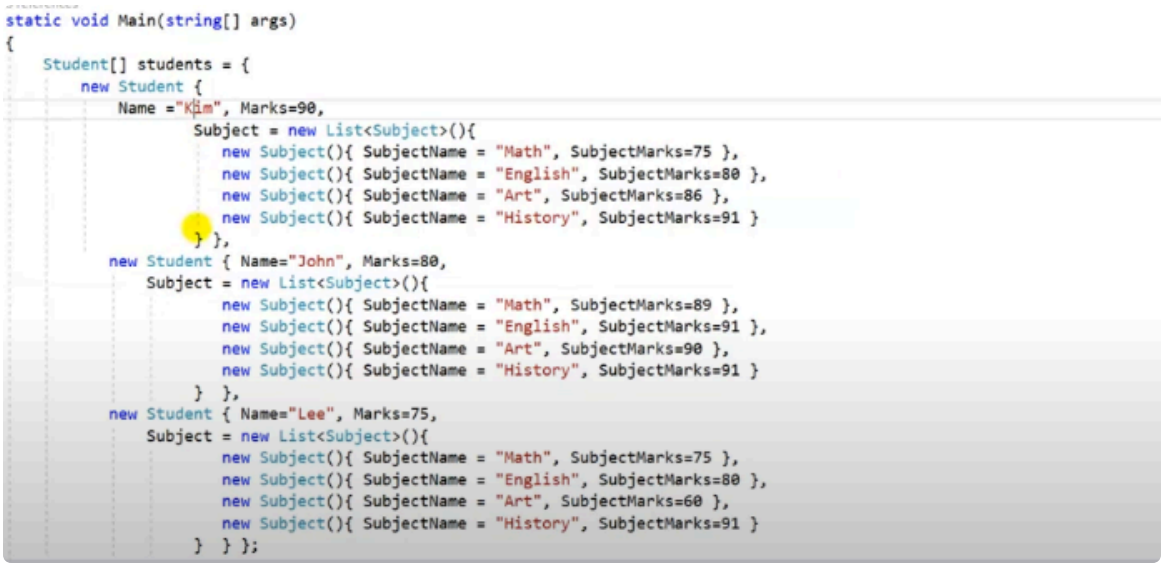


```
0 references
static void Main(string[] args)
{
    List<int> numbers = new List<int>();

    var isAvailable = numbers.Any();

    Console.ReadLine();
}
```

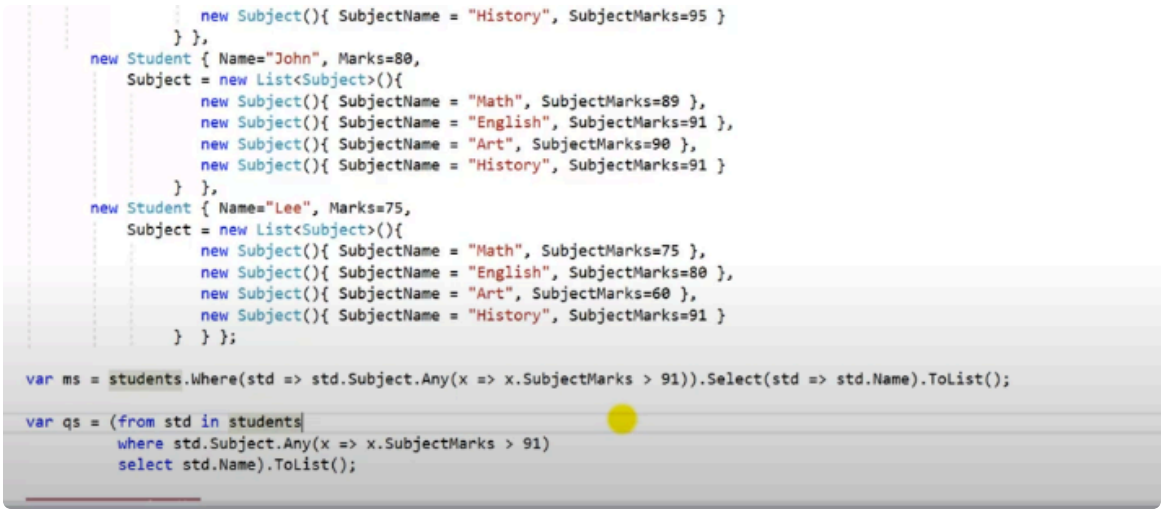


```
0 references
static void Main(string[] args)
{
    Student[] students = { new Student { Name="Kim", Marks=95 },
           new Student { Name="John", Marks=80 },
           new Student { Name="Lee", Marks=75 } };

    var ms = students.Any(x => x.Marks > 96);

    var qs = (from std in students
              select std).Any(x=>x.Marks > 90);

    Console.ReadLine();
}
```

Another example



```
static void Main(string[] args)
{
    Student[] students = {
        new Student {
            Name ="Kim", Marks=90,
                    Subject = new List<Subject>(){
                        new Subject(){ SubjectName = "Math", SubjectMarks=75 },
                        new Subject(){ SubjectName = "English", SubjectMarks=80 },
                        new Subject(){ SubjectName = "Art", SubjectMarks=86 },
                        new Subject(){ SubjectName = "History", SubjectMarks=91 }
                    } },
        new Student { Name="John", Marks=80,
                Subject = new List<Subject>(){
                        new Subject(){ SubjectName = "Math", SubjectMarks=89 },
                        new Subject(){ SubjectName = "English", SubjectMarks=91 },
                        new Subject(){ SubjectName = "Art", SubjectMarks=90 },
                        new Subject(){ SubjectName = "History", SubjectMarks=91 }
                    } },
        new Student { Name="Lee", Marks=75,
                Subject = new List<Subject>(){
                        new Subject(){ SubjectName = "Math", SubjectMarks=75 },
                        new Subject(){ SubjectName = "English", SubjectMarks=80 },
                        new Subject(){ SubjectName = "Art", SubjectMarks=60 },
                        new Subject(){ SubjectName = "History", SubjectMarks=91 }
                    } } };
```

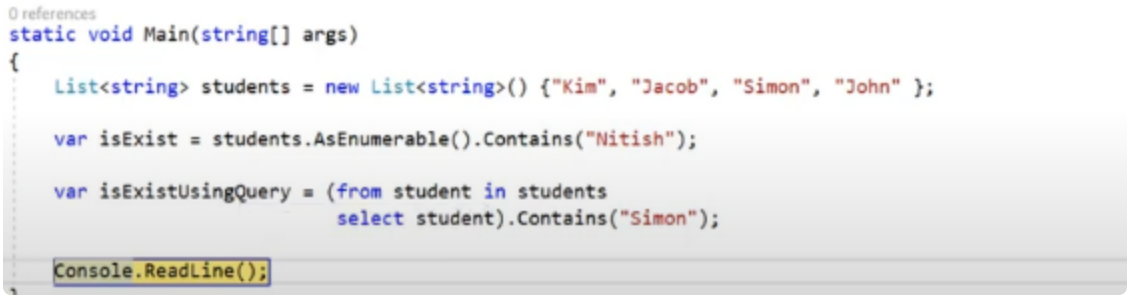


```
                new Subject(){ SubjectName = "History", SubjectMarks=95 }
            } },
        new Student { Name="John", Marks=80,
            Subject = new List<Subject>(){
                new Subject(){ SubjectName = "Math", SubjectMarks=89 },
                new Subject(){ SubjectName = "English", SubjectMarks=91 },
                new Subject(){ SubjectName = "Art", SubjectMarks=90 },
                new Subject(){ SubjectName = "History", SubjectMarks=91 }
            } },
        new Student { Name="Lee", Marks=75,
            Subject = new List<Subject>(){
                new Subject(){ SubjectName = "Math", SubjectMarks=75 },
                new Subject(){ SubjectName = "English", SubjectMarks=80 },
                new Subject(){ SubjectName = "Art", SubjectMarks=60 },
                new Subject(){ SubjectName = "History", SubjectMarks=91 }
            } };

var ms = students.Where(std => std.Subject.Any(x => x.SubjectMarks > 91)).Select(std => std.Name).ToList();

var qs = (from std in students
          where std.Subject.Any(x => x.SubjectMarks > 91)
          select std.Name).ToList();
```

Contains method

Contains Operator

- ✓ Contains operator is used to check whether a sequence (data source) contains a specified element.
- ✓ For a source of objects, Contains only check reference. To work with value we need to do some extra things.
- ✓ It returns a Boolean value.



```
0 references
static void Main(string[] args)
{
    List<string> students = new List<string>() {"Kim", "Jacob", "Simon", "John" };

    var isExist = students.AsEnumerable().Contains("Nitish");

    var isExistUsingQuery = (from student in students
                             select student).Contains("Simon");

    Console.ReadLine();
}
```

If you are working even you give the same value it will give false for that element has to be in the list

```
0 references
static void Main(string[] args)
{
    List<Student> students = new List<Student>()
   {
       new Student(){ Id = 1, Name = "Kim"},
       new Student(){ Id = 2, Name = "John"},
   };
    var std = new Student() { Id = 1, Name = "Kim" };

    students.Add(std);

    var isExist = students.Contains(std);

    Console.ReadLine();
}
```

To overcome this we have to override the comparer

create the comparer class

```
0 references
class StudentComparer : IEqualityComparer<Student>
{
    0 references
    public bool Equals(Student x, Student y)
    {
        if (object.ReferenceEquals(x, y))
        {
            return true;
        }

        if (object.ReferenceEquals(x, null) || object.ReferenceEquals(y, null))
        {
            return false;
        }

        return x.Id == y.Id && x.Name == y.Name;
    }

    0 references
    public int GetHashCode(Student obj)
    {
        throw new NotImplementedException();
    }
}
```

```
        return true;
    }

    if (object.ReferenceEquals(x, null) || object.ReferenceEquals(y, null))
    {
        return false;
    }

    return x.Id == y.Id && x.Name == y.Name;
}

0 references
public int GetHashCode(Student obj)
{
    if (Object.ReferenceEquals(obj, null))
    {
        return 0;
    }

    int idHashCode = obj.Id.GetHashCode();
    int nameHashCode = obj.Name == null ? 0 : obj.Name.GetHashCode();

    return idHashCode ^ nameHashCode;
}
```

Create this two methods in the class and create object in main and pass it

```
static void Main(string[] args)
{
    List<Student> students = new List<Student>()
    {
        new Student(){ Id = 1, Name = "Kim"},
        new Student(){ Id = 2, Name = "John"},
    };

    var comparer = new StudentComparer();

    var isExist = students.Contains(new Student() { Id = 1, Name = "Kim" }, comparer);

    var qs = (from std in students
              select std).Contains(new Student() { Id = 1, Name = "Kim" }, comparer);

    Console.ReadLine();
}
```