

## C# - 12

### Collections

#### Dynamic Array

When I create a integer array So but default what data it will store?

```
class NosArray
{
    int[] nos = new int[5];
9 references
```

-----it will store 0, because it is a integer type and we have seen integer will be 0 type, it will always be initialized to 0

----if it is a reference type

if i write like this

```
int[] nos;
```

then it will store null in it

but we are initializing it, as we are initializing it and its of integer type, so by default it will store integer value i.e 0

Now How do you get the index number ?

Imagine you are in mall and you are maintaining the stall where you take peoples bag and give them token



you will always know weather it is blank or it is filled

So in our case we always fill this and when someone comes and say empty this we'll empty this

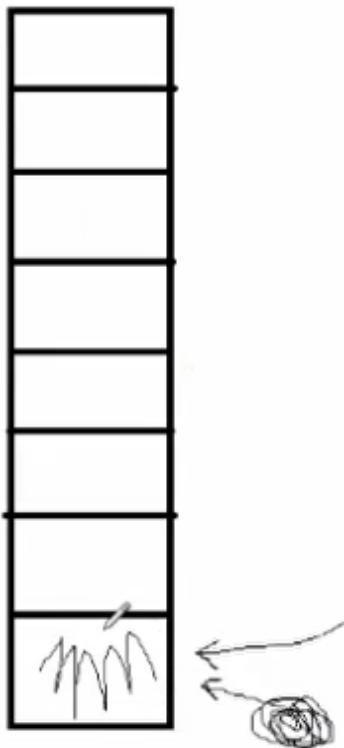
Now how do you know weather it is empty or filled, From where did you start

So you will hold a particular pointer (Pointer is just a location ), and search weather it is empty or not

So i need a pointer which say weather it is filled or not ,

we'll take a pointer cnt and initialize it to 0

if you are initializing it to 0 means you already have something in it



**So you should initialize it to -1**

```
int[] nos = new int[5];
int cnt = -1;
```

a reference

now as it is initialize to -1  
we cannot directly say

```
int[] nos = new int[5];
int cnt = -1;
9 references
public void Add(int no)
{
    nos[cnt] = no;
}
```

before putting it to my pointer i need to increment the pointer by +1

```
2 references
class NosArray
{
    int[] nos = new int[5];
    int cnt = -1;
9 references
    public void Add(int no)
    {
        cnt++;
        nos[cnt] = no;
    }
}
```

Now when i do this there are chances that when i read, this will surely put all the numbers

now when i debug look here

cnt is -1

A screenshot of the Microsoft Visual Studio IDE interface. The top menu bar includes File, Edit, View, Git, Project, Build, Debug, Test, Analyze, Tools, Extensions, and Window. The toolbar below has icons for file operations like Open, Save, and Build. The status bar at the bottom shows "121 %", "No issues found", and several tabs: Call Stack, Breakpoints, Exception Settings, Command Window, Immediate Window, Output, Error List ..., and Autos.

The main window displays the code editor with two files open: "DynArray.cs" and "ConsoleApp". The "ConsoleApp" tab is active. The code in "ConsoleApp" is:

```
2 references
class NosArray
{
    int[] nos = new int[5];
    int cnt = -1;
    public void Add(int no)
    {
        cnt++;
        nos[cnt] = no;
    }
    //public ? GetNos()
    //{
    //    return ?;
    //}
}

0 references
class DynArray
{}
```

The line "cnt++;" is highlighted in yellow, and the variable "cnt" is underlined with a red squiggly line, indicating a potential error or warning. A red circular breakpoint icon is visible on the left margin next to line 15. The status bar at the bottom indicates "No issues found".

Now when i say cnt++ it becomes 0  
now nos is having 5 value

The screenshot shows the Microsoft Visual Studio IDE interface. The title bar displays "DynArray.cs" and "ConsoleApp". The code editor window contains the following C# code:

```
2 references
class NosArray
{
    int[] nos = new int[5];
    int cnt = -1;
9 references
public void Add(int no)
{
    cnt++;
    nos[cnt] = no; ≤ 1ms elapsed
}
//public
//{
//    return
//}
}
0 references
class DynArray
{
```

A tooltip is displayed over the line of code "nos[cnt] = no;". The tooltip shows the variable "nos" with its type "int[5]" and a list of its elements:

	nos	[int[5]]
[0]	0	(
[1]	0	)
[2]	0	
[3]	0	
[4]	0	

The status bar at the bottom of the IDE shows "121 %", "No issues found", and a dropdown menu.

so in nos of 0, i'll store the number i.e 10

DynArray.cs X ConsoleApp

ConsoleApp

ConsoleApp.CollectionsDem

```
2 references
9 class NosArray
10 {
11     int[] nos = new int[5];
12     int cnt = -1;
13     9 references
14     ►| public void Add(int no)
15     {
16         cnt++;
17         nos[cnt] = no; ≤ 1ms elapsed
18     }
19     //public ? GetNos()
20     //{
21     //    return ?;
22     //}
23 }
24
0 references
25 class DynArray
26 {
```

121 % No issues found

Call Stack Breakpoints Exception Settings Command Window Immediate Window Output Error List

The screenshot shows the Microsoft Visual Studio IDE interface. The code editor displays two classes: `NosArray` and `DynArray`. The `NosArray` class has a constructor that initializes a `nos` array of size 5 with all elements set to 0. The `Add` method increments a counter `cnt` and assigns the value of `no` to the `cnt`-th index of the `nos` array. The `DynArray` class is currently empty. A tooltip is visible over the line of code `nos[cnt] = no;`, showing the variable `nos` with its type `[int[5]]`. The tooltip content is a table:

[0]	10
[1]	0
[2]	0
[3]	0
[4]	0

The status bar at the bottom indicates "No issues found".

```
DynArray.cs  ✘ × ConsoleApp
ConsoleApp
ConsoleApp.Collectors
2 references
class NosArray
{
    int[] nos = new int[5];
    int cnt = -1;
    9 references
    public void Add(int no)
    {
        cnt++;
        nos[cnt] = no;
    }
    //public
    //{
    //    return
    //}
}
0 references
class DynArray
{}
```

Similarly I'm calling add method to add 11, 12, 13, 14 .....

DynArray.cs X ConsoleApp

ConsoleApp

0 references

ConsoleApp.CollectionsDe

```
25 class DynArray
26 {
27     static void Main()
28     {
29         int[] nos = new int[10];
30         NosArray nosArray = new NosArray();
31         nosArray.Add(10);
32         nosArray.Add(11);
33         nosArray.Add(12);
34         nosArray.Add(13);
35         nosArray.Add(14);
36         nosArray.Add(15);
37         nosArray.Add(16);
38         nosArray.Add(17);
39         nosArray.Add(18);
40     }
41 }
42 }
43 }
44 }
```

121 % No issues found

Call Stack Breakpoints Exception Settings Command Window Immediate Window Output Error List

now again I'm calling nosArray.Add(11)

DynArray.cs X ConsoleApp

C# ConsoleApp

```
7  namespace ConsoleApp.CollectionsDemo
8  {
9      class NosArray
10     {
11         int[] nos = new int[5];
12         int cnt = -1;
13         public void Add(int no)
14         {
15             cnt++;
16             nos[cnt] = no;
17         }
18
19         //public ? GetNos()
20         //{
21         //    return ?;
22         //}
23     }
24 }
```

2 references  
ConsoleApp.CollectionsDemo

9 references  
ConsoleApp.CollectionsDemo

0 references

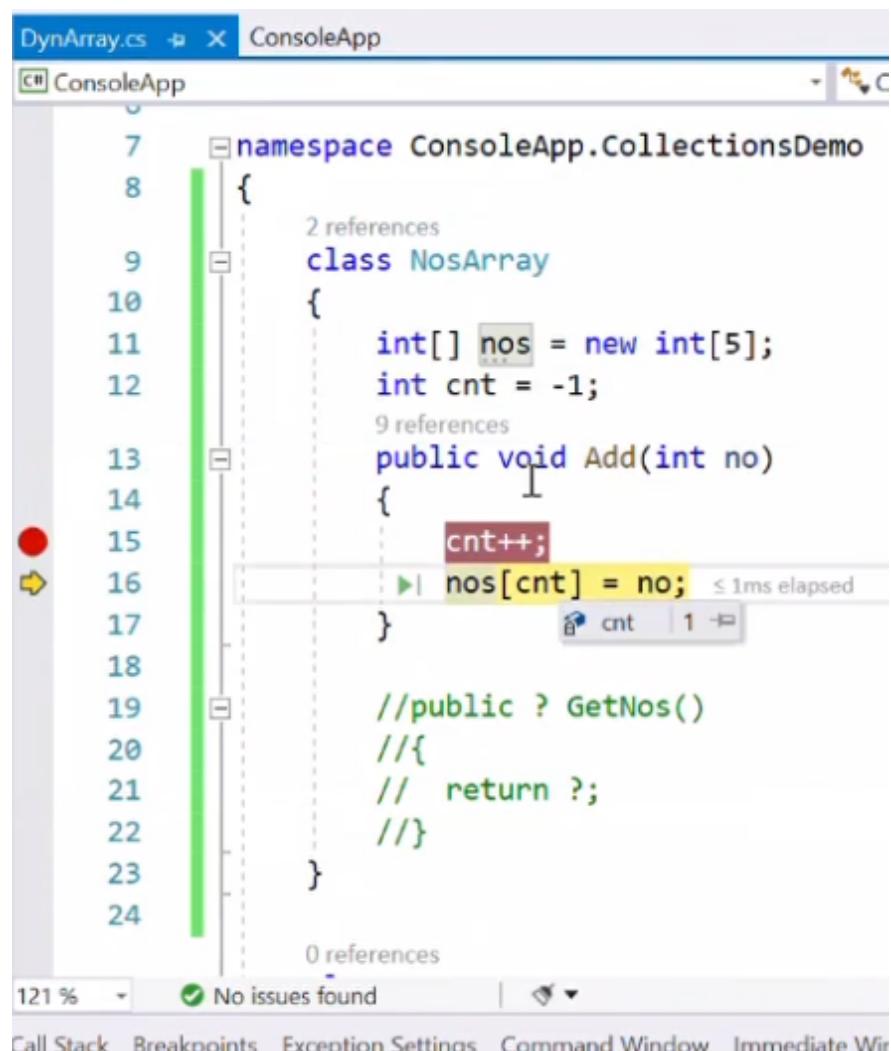
DynArray.cs X ConsoleApp

ConsoleApp

```
7     namespace ConsoleApp.CollectionsDemo
8     {
9         class NosArray
10        {
11            int[] nos = new int[5];
12            int cnt = -1;
13            public void Add(int no)
14            {
15                cnt++;
16                nos[cnt] = no;    ⏴ 1ms elapsed
17            }
18
19            //public ? GetNos()
20            //{
21            //    return ?;
22            //}
23        }
24
0 references
```

121 % No issues found

Call Stack Breakpoints Exception Settings Command Window Immediate Win



The screenshot shows a code editor window for a C# file named 'DynArray.cs' within a project 'ConsoleApp'. The code defines a class 'NosArray' with an array 'nos' of size 5. A tooltip is displayed over the line 'nos[cnt] = no;', showing the current state of the array:

[0]	10
[1]	11
[2]	0
[3]	0
[4]	0

```
7     namespace ConsoleApp.CollectionsDemo
8     {
9         class NosArray
10        {
11            int[] nos = new int[5];
12            int cnt = -1;
13            public void Add(int no)
14            {
15                cnt++;
16                nos[cnt] = no;
17            }
18        }
19    }
20
21
22
23
24 }
```

look 11 stored here

similarly 12, 13, 14 will be stored here

now when we are trying to insert 15 look at this array is already filled

The screenshot shows a Microsoft Visual Studio code editor window for a C# project named "ConsoleApp". The current file is "DynArray.cs". A tooltip is displayed over the line of code "nos[cnt] = no;". The tooltip shows a table with the following data:

	nos	[int[5]]
[0]	10	
[1]	11	
[2]	12	
[3]	13	
[4]	14	

The code in "DynArray.cs" is as follows:

```
7  namespace ConsoleApp.CollectionsDemo
8  {
9      class NosArray
10     {
11         int[] nos = new int[5];
12         int cnt = -1;
13         public void Add(int no)
14         {
15             cnt++;
16             nos[cnt] = no; < 1ms elapsed
17         }
18         //public
19         //{
20         //    return
21         //}
22     }
23 }
24
```

The status bar at the bottom indicates "121 %", "No issues found", and tabs for "Call Stack", "Breakpoints", "Exception Settings", "Command Window", and "Immediate Window".

So when array is filled, where will i store the data

## So when i run this I'll get **Array Index out of bond Exception**

So how will I control that

How will you extend the length

When will you extend the length or Recreate new array

----- when `cnt == nos.length - 1`

S if this condition is true then

1. I create a new temp array
2. Copy data from nos to temp array
3. Re-create the temp array
4. Copy data from temp array to nos
5. Clear temp array

DynArray.cs\* X ConsoleApp

ConsoleApp

ConsoleApp.CollectionsDemo.NosArray

```
7  namespace ConsoleApp.CollectionsDemo
8  {
9      class NosArray
10     {
11         int[] nos = new int[5];
12         int cnt = -1;
13         public void Add(int no)
14         {
15             if(cnt == nos.Length-1)
16             {
17                 //1. create new temp array
18                 //2. copy data from nos to temp array
19
20                 //3. re-create temp array
21                 //3. copy data from temp array to nos
22                 //4. clear temp array
23             }
24
25             cnt++;
26             nos[cnt] = no;
27         }
28     }
29 }
```

The screenshot shows the Microsoft Visual Studio IDE with the file `DynArray.cs` open. The code implements a dynamic array class with an `Add` method. The code uses a temporary array `temp` to copy data from the original array `nos` before increasing its size and copying back. Handwritten annotations explain the steps:

- `int[] temp = new int[nos.Length];` ← creating new array equal to length of nos array
- `CopyTo()` ← we use this to copy data
- `nos.CopyTo(temp, 0);` ← I'm adding just 10 to whatever I have I have 5 so as I add 10 the size of my array will become 15

```
12     int cnt = -1;
13     public void Add(int no)
14     {
15         if(cnt == nos.Length-1)
16         {
17             //1. create new temp array
18             int[] temp = new int[nos.Length];
19             //2. copy data from nos to temp array
20             nos.CopyTo(temp, 0);
21             //3. re-create temp array
22             nos = new int[nos.Length + 10];
23             //3. copy data from temp array to nos
24             temp.CopyTo(nos, 0);
25             //4. clear temp array
26             temp = null;
27         }
28
29         cnt++;
30         nos[cnt] = no;
31     }
32
33     //public ? GetNos()
34     //{
35 }
```

my cnt value is 4

The screenshot shows the Microsoft Visual Studio IDE interface. The title bar displays "DynArray.cs" and "ConsoleApp". The code editor window contains the following C# code:

```
12     int cnt = -1;
13     9 references
14     public void Add(int no)
15     {
16         if(cnt == nos.Length-1)
17         {
18             //1. create new temp array
19             int[] temp = new int[nos.Length];
20             //2. copy data from nos to temp array
21             nos.CopyTo(temp, 0);
22             //3. re-create temp array
23             nos = new int[nos.Length + 10];
24             //3. copy data from temp array to nos
25             temp.CopyTo(nos, 0);
26             //4. clear temp array
27             temp = null;
28         }
29         cnt++;
30         nos[cnt] = no;
31     }
```

The code editor has a green vertical ruler on the left and a status bar at the bottom indicating "121 %". A red circular icon with a yellow question mark is visible on the far left of the editor area. The status bar also shows "No issues found".

nos.length is 5 that's a reason we are doing -1 to it , so that it match 4 == 4

The screenshot shows the Microsoft Visual Studio IDE interface. The title bar displays "DynArray.cs" and "ConsoleApp". The solution explorer on the left shows "ConsoleApp" and "ConsoleApp.CollectionsDemo.N". The code editor window contains the following C# code:

```
12     int cnt = -1;
13     9 references
14     public void Add(int no)
15     {
16         if(cnt == nos.Length-1)
17         {
18             //1. create new temp array
19             int[] temp = new int[nos.Length];
20             //2. copy data from nos to temp array
21             nos.CopyTo(temp, 0);
22             //3. re-create temp array
23             nos = new int[nos.Length + 10];
24             //3. copy data from temp array to nos
25             temp.CopyTo(nos, 0);
26             //4. clear temp array
27             temp = null;
28         }
29         cnt++;
30         nos[cnt] = no;
31     }
```

A tooltip for the variable "nos.Length" is displayed at line 18, position 45, showing the value "5". A red circular icon with a yellow arrow is visible on the far left of the code editor.

so condition matches

and temp array is created of size 5

DynArray.cs X ConsoleApp

ConsoleApp

ConsoleApp.CollectionsDe

```
12     int cnt = -1;
13     9 references
14     public void Add(int no)
15     {
16         if(cnt == nos.Length-1)
17         {
18             //1. create new temp array
19             int[] temp = new int[nos.Length];
20             //2. copy > temp | {int[5]} => temp array
21             nos.CopyTo(temp, 0); 1ms elapsed
22             //3. re-create temp array
23             nos = new int[nos.Length + 10];
24             //3. copy data from temp array to nos
25             temp.CopyTo(nos, 0);
26             //4. clear temp array
27             temp = null;
28         }
29         cnt++;
30         nos[cnt] = no;
31     }
```

121 % No issues found

Call Stack Breakpoints Exception Settings Command Window Immediate Window Output Error List

then i'm copying all the data to temp

DynArray.cs   X   ConsoleApp

ConsoleApp   +   ConsoleApp.CollectionsDemo.

```
12     int cnt = -1;
9 references
13     public void Add(int no)
14     {
15         if(cnt == nos.Length-1)
16         {
17             //1. create new temp array
18             int[] temp = new int[nos.Length];
19             //2. copy temp [int[5]] → temp array
20             nos.CopyTo(temp, 0);
21             //3. re-create nos [2]
22             nos = new int[nos.Length + 10]; ← 1ms elapse
23             //3. copy temp [4] → mp array to nos
24             temp.CopyTo(nos, 0);
25             //4. clear temp array
26             temp = null;
27         }
28         cnt++;
29         nos[cnt] = no;
30     }
31 
```

121 %   No issues found

Call Stack   Breakpoints   Exception Settings   Command Window   Immediate Window   Output   Error List   A

after this I'll resize nos array to +10, so it becomes of size 15 and everything is been initialize to 0

Remember: When ever you are creating a new integer array, it will be initialize to 0, not null

The screenshot shows the Microsoft Visual Studio IDE interface. The title bar displays "DynArray.cs" and "ConsoleApp". The code editor window contains C# code for a dynamic array class. A tooltip is displayed over the line of code "int[] temp = new int[nos.Length];". The tooltip details the creation of a temporary array "temp" with 5 elements, labeled as a "temp array". Below this, another tooltip shows the assignment of a new array "nos" with 10 elements, labeled as a "array". The main code in the editor is as follows:

```
12     int cnt = -1;
13     public void Add(int no)
14     {
15         if(cnt == nos.Length-1)
16         {
17             //1. create new temp array
18             int[] temp = new int[nos.Length];
19             //2. copy temp | [int[5]] → temp array
20             nos.CopyTo(temp, 0);
21             //3. re-create nos
22             nos = new int[nos.Length + 10];
23             //3. copy data from temp array to nos
24             temp.CopyTo(nos, 0);
25             //4. clear temp array
26             temp = null;
27         }
28         cnt++;
29         nos[cnt] = no;
30     }
31 }
```

The status bar at the bottom indicates "121 %", "No issues found", and a zoom level. The menu bar includes "Call Stack", "Breakpoints", "Exception Settings", "Command Window", "Immediate Window", "Output", "Error List", and "A".

after that we copy temp data to nos

DynArray.cs ✘ × ConsoleApp

ConsoleApp

ConsoleApp.CollectionsDemo.NosArray

```
12     int cnt = -1;
13
14     public void Add(int no)
15     {
16         if(cnt == nos.Length-1)
17         {
18             //1. create new temp array
19             int[] temp = new int[nos.Length];
20             //2. copy data from nos to temp array
21             nos.CopyTo(temp, 0);
22             //3. re-create temporary array
23             nos = new int[nos.Length];
24             //3. copy data from temp array to nos
25             temp.CopyTo(nos, 0);
26             //4. clear temp array
27             temp = null;    ↳ nos [int[15]] +→
28         }
29         cnt++;
30         nos[cnt] = no;
31     }
```

121 % No issues found

Call Stack Breakpoints Exception Settings Command Window Immediate Window Output Error List Autos Loc

Ready

after that I initialize temp to null

The screenshot shows the Microsoft Visual Studio IDE interface. The title bar displays "DynArray.cs" and "ConsoleApp". The code editor window contains the following C# code:

```
int cnt = -1;
9 references
public void Add(int no)
{
    if(cnt == nos.Length-1)
    {
        //1. create new temp array
        int[] temp = new int[nos.Length];
        //2. copy data from nos to temp array
        nos.CopyTo(temp, 0);
        //3. re-create temp array
        nos = new int[nos.Length + 10];
        //3. copy data from temp array to nos
        temp.CopyTo(nos, 0);
        //4. clear temp array
        temp = null;
    }
    cnt++;
    nos[cnt] = no;
}
```

A red dot on the left margin indicates a break point at line 18. A tooltip at the bottom of the editor window shows the variable "temp" with the value "null".

now it will store me the next number

so 15 will be stored

DynArray.cs X ConsoleApp

ConsoleApp

ConsoleApp.CollectionsDemo.No

```
int cnt = -1;
9 references
public void Add(int no)
{
    if(cnt == nos.Length-1)
    {
        [0] 10 e new temp array
        [1] 11 = new int[nos.Length];
        [2] 12 data from nos to temp array
        [3] 13 temp, 0);
        [4] 14 create temp array
        [5] 15 int[nos.Length + 10];
        [6] 16 data from temp array to nos
        [7] 17 0(nos, 0);
        [8] 18 0 temp array
        [9] 19 0 ;
        [10] 20 0 ;
        [11] 21 0 ;
        [12] 22 0 ;
        [13] 23 0 ;
        [14] 24 0 ;
    }
    cnt++;
    nos[cnt] = no;
}
< 1ms e nos {int[15]} ↗
```

121 % No issues found

then it go for 16

DynArray.cs

ConsoleApp

C# ConsoleApp

ConsoleApp.Collections

```
0 references
40     static void Main()
41     {
42         int[] nos = new int[10];
43         NosArray nosArray = new NosArray();
44         nosArray.Add(10);
45         nosArray.Add(11);
46         nosArray.Add(12);
47         nosArray.Add(13);
48         nosArray.Add(14);
49         nosArray.Add(15);
50         nosArray.Add(16); // Line 50 highlighted in yellow
51         nosArray.Add(17);
52         nosArray.Add(18);
53     }
54 }
55 }
56 }
57 }
```

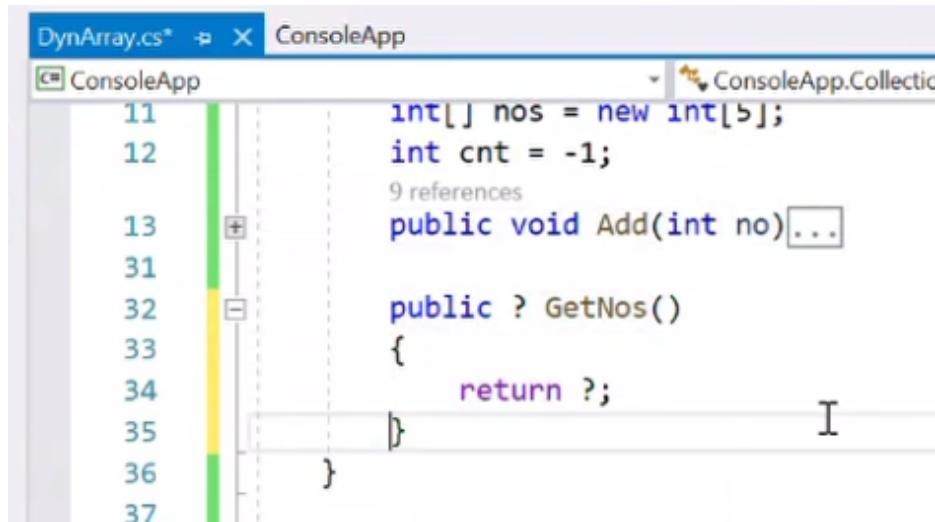
121 %

No issues found

Call Stack Breakpoints Exception Settings Command Window Immediate Window Output Error L

similarly 17, 18 all this numbers will be stored

Now How will you write get method?



```
DynArray.cs*  X  ConsoleApp
C# ConsoleApp
11 int[] nos = new int[5];
12 int cnt = -1;
13 public void Add(int no){...}
31
32 public ? GetNos()
33 {
34     return ?;
35 }
36
37 }
```

so for 1st ? obviously as it is a array it will return the numeric array only so int

```
11 int[] nos = new int[5];
12 int cnt = -1;
13 9 references
14 public void Add(int no)...
15
16 0 references
17 public int[] GetNos()
18 {
19     return nos;
20 }
21
22 }
```

Now we need last modified Index

what if we return nos

```
11 public void Add(int no)...
12
13 31
14
15 32 public int[] GetNos()
16 {
17     return nos;
18 }
19
20 }
```

DynArray.cs X ConsoleApp

ConsoleApp

ConsoleApp.CollectionsDemo.DynArray

```
38 class DynArray
39 {
40     static void Main()
41     {
42         int[] nos = new int[10];
43         NosArray nosArray = new NosArray();
44         nosArray.Add(10);
45         nosArray.Add(11);
46         nosArray.Add(12);
47         nosArray.Add(13);
48         nosArray.Add(14);
49         nosArray.Add(15);
50         nosArray.Add(16);
51         nosArray.Add(17);
52         nosArray.Add(18);

53         foreach (var no in nosArray.GetNos())
54         {
55             Console.WriteLine(no);
56         }
57     }
58 }
59 }
60 }
```

121 %

No issues found

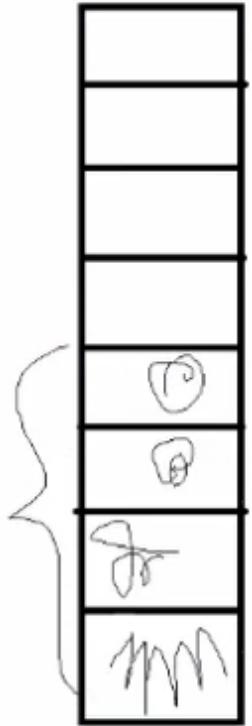
```
DynArray.cs  ConsoleApp
ConsoleApp.cs Select C:\WINDOWS\system32\cmd.exe
11
10
11
12
13
13
31
14
15
32
16
33
17
18
34
0
35
0
36
0
0
0
Output 0
Show output
Build started: Project: ConsoleApp, Configuration: Debug Any CPU ...
Press any key to continue . . .
1>----- R
```

problem is we get the 0 also here

but when i say get

I want to get all the numbers meaning from 10 to 18, i don't want that 0's

how to avoid this 0's



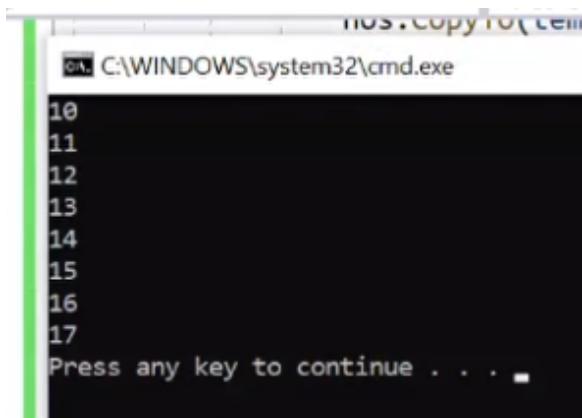
we have to return this 4 item from here

But how do you know that you have 4 item here, who will know that i have 4 items

-----cnt

you need to use some technique to read till cnt

```
1 reference
public int[] GetNos()
{
    int[] temp = new int[cnt];
    for (int i = 0; i < cnt; i++)
    {
        temp[i] = nos[i];
    }
    return temp;
}
```



The screenshot shows a Windows Command Prompt window with the title "nos.Copy10\temp". The window displays the output of a program, which is a list of integers from 10 to 17. The text in the window is as follows:

```
C:\WINDOWS\system32\cmd.exe
10
11
12
13
14
15
16
17
Press any key to continue . . .
```

i'm getting till 17 but i should get till 18

<= aega

```
1 reference
public int[] GetNos()
{
    int[] temp = new int[cnt];
    for (int i = 0; i <= cnt; i++)
    {
        temp[i] = nos[i];
    }
    return temp;
}
```

T

for this at line 34 we have to take cnt + 1

```
1 reference
32
33
34  int[] temp = new int[cnt + 1];
35  for (int i = 0; i <= cnt; i++)
36  {
37      temp[i] = nos[i];
38  }
39  return temp;
40
```

```
aAr  C:\WINDOWS\system32\cmd.exe
5
6 10
7 11
8 12
9 13
10 14
11 15
12 16
13 17
14 18
15 Press any key to continue . . . ■
16
17
```

by doing dry run we found the error

nos.lengtht = 4	
nos	cnt
10	0
11	1
12	2
13	3
14	4


so this is how people started thinking like array has its own disadvantage what shall we do  
we'll write some code which will automatically increment the array allow the user to store particular data  
so they created a package called collection and in the collection they started adding all this thing

### **In collection we have 2 different type**

1. **Generic**
2. **Non-Generic**

Earlier we used to use Non-Generic

### **What is Non-Generic collection ?**

----Same array

But here it store me Object

The screenshot shows a code editor window in Visual Studio with the file NonGenCollection.cs open. The code defines a class NonGenCollection with a Main() method. A tooltip is displayed over the word 'ArrayList' at line 13, which is underlined with a red squiggle. The tooltip contains the error message 'CS0103 The name 'ArrayList' does not exist in the current context' and a list of suggestions:

- using System.Collections;
- System.Collections.ArrayList
- Generate variable 'ArrayList'
- Use expression body for methods

The suggestion 'using System.Collections;' is highlighted with a green background.

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace ConsoleApp.CollectionsDemo
8  {
9      class NonGenCollection
10     {
11         static void Main()
12         {
13             ArrayList
14             using System.Collections;
15             System.Collections.ArrayList
16             Generate variable 'ArrayList'
17             Use expression body for methods
18         }
19     }
20 }
```

Look at this ArrayList comes from using System.Collections  
not from Collection.Generic

The screenshot shows a Microsoft Visual Studio code editor window. The title bar indicates the file is "NonGenCollection.cs\*" and the project is "ConsoleApp\*". The code editor displays the following C# code:

```
1  using System;
2  using System.Collections;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace ConsoleApp.CollectionsDemo
9  {
10     class NonGenCollection
11     {
12         static void Main()
13         {
14             ArrayList
15         }
16     }
17 }
18
```

A tooltip is visible over the word "ArrayList", showing the full namespace path: "ConsoleApp.CollectionsDemo.NonGenCollection.ArrayList". The code editor uses color coding for syntax, and there are some yellow squiggly underlines and a yellow warning sign icon on line 3.

Look at this it allows me to store anything

NonGenCollection.cs X ConsoleApp\*

ConsoleApp

ConsoleApp.CollectionsDemo.NonGenC

```
1  using System;
2  using System.Collections;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace ConsoleApp.CollectionsDemo
9  {
10    class NonGenCollection
11    {
12      static void Main()
13      {
14        ArrayList list = new ArrayList();
15        list.Add(10);
16        list.Add("Shashi");
17        list.Add(20);
18        list.Add(40);
19        list.Add("Ravi");
20      }
21    }
22 }
```

121 %

No issues found

## **Now what is the disadvantage of using this kind of list**

Now as I'm allowing the user to add 10 as well as "Shashi"

So how will you search , How will you know what datatype user have entered we don't know

then i have to run a for loop till list.count

then take individual element

then check this particular element is of the integer type or it is string type

then Unbox it, then fetch the data

```
NonGenCollection.cs  X  ConsoleApp*
ConsoleApp          ConsoleApp.CollectionsDemo.NonG

0 references
class NonGenCollection
{
    0 references
    static void Main()
    {
        ArrayList list = new ArrayList();
        list.Add(10);
        list.Add("Shashi");
        list.Add(20);
        list.Add(40);
        list.Add("Ravi");

        for (int i = 0; i < list.Count; i++)
        {
            }
    }
}
```

So that's becomes a biggest disadvantage

That's a reason they introduce something called Generics

which is a part of collection

```
| using System.Collections.Generic;
```

Collection.Generic

## Now What is there in my Generic?

1. **List** : yesterday in trainer trainee app we used that , in the same way

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace ConsoleApp.CollectionsDemo
8  {
9      class ListDemo
10     {
11         static void Main()
12         {
13             List<int> nos = new List<int>();
```

121 % | No issues found | Ln: 13 Ch: 36 Col: 45 TABS CRLF

Solution Explorer

- Solution 'ConsoleApp' (2 of 2 pr)
- AccessTest
- ConsoleApp
  - Properties
  - References
  - CollectionsDemo
    - DynArray.cs
    - IndexersDemo.cs
    - ListDemo.cs
    - NonGenCollection.cs
  - AccessDemo.cs
  - App.config
  - ArrayDemo1.cs
  - ArrayDemo2.cs
  - BoxingDemo.cs
  - BreakContinueDemo.cs
  - Calc.cs
  - ClassDemo.cs
  - CommandLineDemo.cs
  - ConvertType.cs
  - EnumDemo.cs
  - GoToDemo.cs
  - IfDemo1.cs
  - IfDemo2.cs

nos has a add method, look at this it takes only number type

The screenshot shows a code editor window with the file `ListDemo.cs` open. The code is as follows:

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace ConsoleApp.CollectionsDemo
8  {
9      class ListDemo
10     {
11         static void Main()
12         {
13             List<int> nos = new List<int>();
14             nos.Add(10);
15         }
16     }
17 }
18
```

A tooltip is displayed over the `nos.Add(10)` line, providing documentation for the `Add` method:

`void List<int>.Add(int item)`  
Adds an object to the end of the `List<T>`.  
*item*: The object to be added to the end of the `List<T>`. The value can be null for reference types.

When i try to add some other data it gives me error

The screenshot shows a Microsoft Visual Studio interface with the following details:

- Title Bar:** Shows "ListDemo.cs" and "ConsoleApp".
- Solution Explorer:** Shows "ConsoleApp" and "ConsoleApp.CollectionsDemo.ListDemo".
- Code Editor:** Displays the following C# code:

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace ConsoleApp.CollectionsDemo
8  {
9      class ListDemo
10     {
11         static void Main()
12         {
13             List<int> nos = new List<int>();
14             nos.Add(10);
15             nos.Add("Shashi");
16         }
17     }
18 }
19
```
- Tooltips and Error:** A tooltip for the string "Shashi" is displayed, stating: "class System.String Represents text as a sequence of UTF-16 code units." Below it, an error message is shown: "CS1503: Argument 1: cannot convert from 'string' to 'int'".

So this becomes a advantage of generics  
it takes only the type which you pass here

ListDemo.cs\* ✘ X ConsoleApp\*

ConsoleApp

ConsoleApp.CollectionsDemo.ListDe

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace ConsoleApp.CollectionsDemo
8  {
9      class ListDemo
10     {
11         static void Main()
12         {
13             List<int> nos = new List<int>();
14             nos.Add(10);
15             nos.Add();
16         }
17     }
18 }
19
```

based on this it take only that data

A screenshot of the Microsoft Visual Studio IDE showing a C# file named `ListDemo.cs`. The code demonstrates the use of a `List<int>` collection. The code is as follows:

```
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace ConsoleApp.CollectionsDemo
8  {
9      class ListDemo
10     {
11         static void Main()
12         {
13             List<int> nos = new List<int>();
14             nos.Add(10);
15             nos.Add(20);
16             nos.Add(2);
17             nos.Add(6);
18             nos.Add(1);
19             nos.Add(3);
20
21         }
22     }
23 }
24
25
```

Explore the methods what list have

Just write `nos.` you will get all the methods what list have

ListDemo.cs\* ✘ X ConsoleApp\*

ConsoleApp

ConsoleApp.CollectionsDemo.ListDemo

```
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace ConsoleApp.CollectionsDemo
8  {
9      class ListDemo
10     {
11         static void Main()
12         {
13             List<int> nos =
14                 new List<int>();
15
16             nos.Add(1);
17             nos.Add(2);
18             nos.Add(3);
19
20             foreach (int n in nos)
21             {
22                 Console.WriteLine(n);
23             }
24         }
25     }
}
```

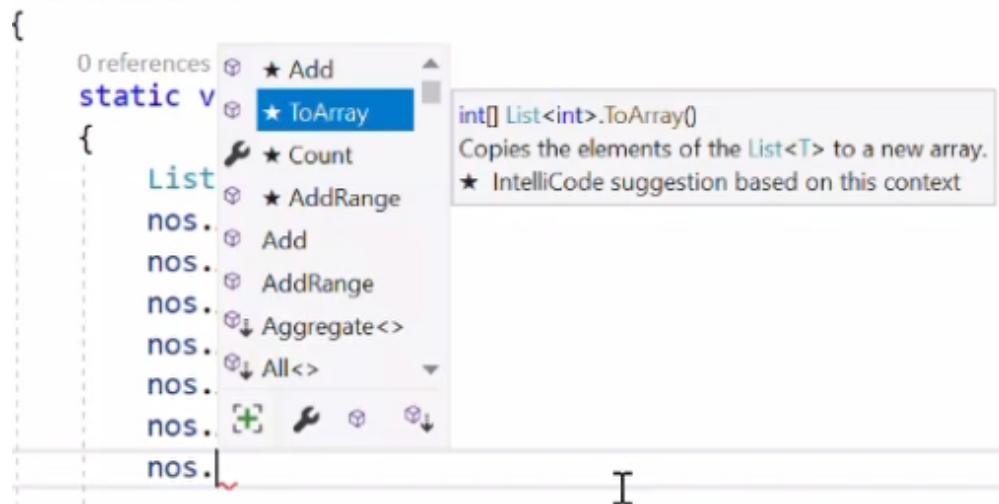
Add

void List<int>.Add(int item)  
Adds an object to the end of the List<T>.

★ IntelliCode suggestion based on this context

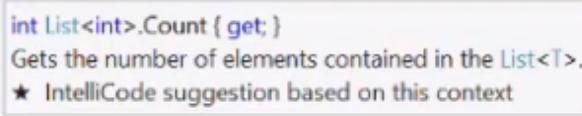
?w List<int>();

**ToArray()** --- it will gone to copy the elements of the list to a new array



**Count()** --- it will return me the length of the list, number of element in the list

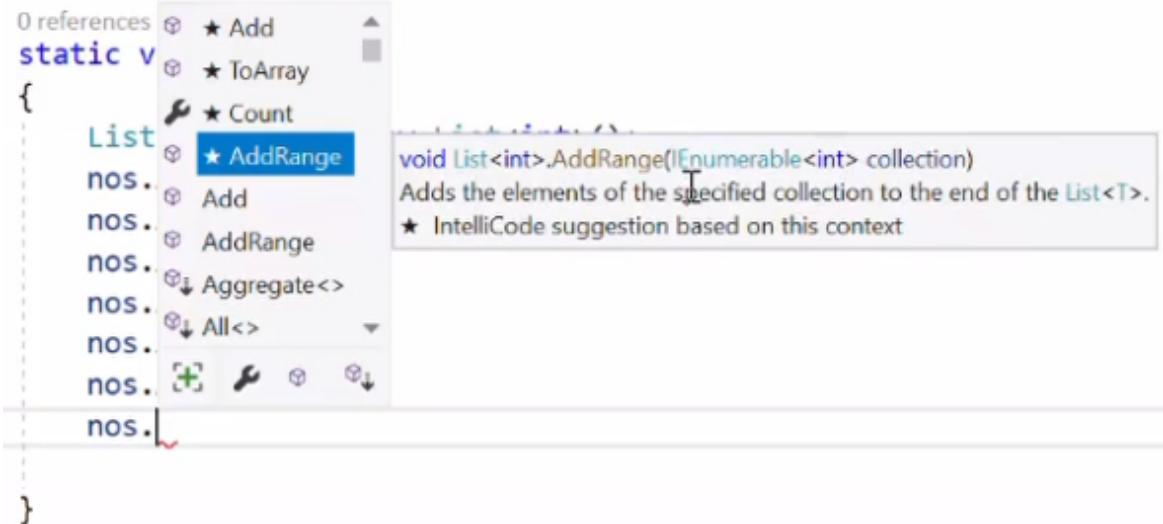
```
class ListDemo
{
    static void Main()
    {
        List<int> nos = new List<int>();
        nos.Add(1);
        nos.Add(2);
        nos.Add(3);
        nos.AddRange(new int[] { 4, 5, 6 });
        nos.Count;
    }
}
```



The screenshot shows a code editor with an open IntelliSense tooltip for the `Count` property of a `List<int>`. The tooltip contains the following information:

- Signature: `int List<int>.Count { get; }`
- Description: Gets the number of elements contained in the `List<T>`.
- Note: ★ IntelliCode suggestion based on this context

**AddRange()** --- it will take a existing array and it will add this existing array, look at the parameter what we are passing here, you need to pass collection as a parameter it adds the elements of the specified collection to the end of the list



and it returns me nothing its a void type

I have a list of integers, I call it as evenNos, i have initialize it as 2 4 6 8, ill say addrange and pass evenNos

ListDemo.cs   X   ConsoleApp\*

ConsoleApp

ConsoleApp.CollectionsDemo.ListC

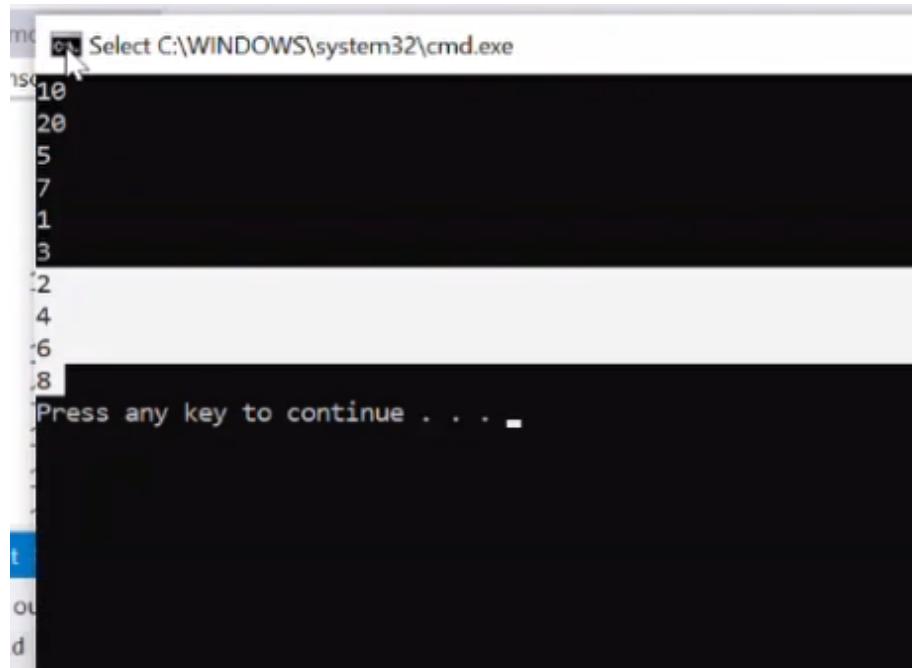
```
7     namespace ConsoleApp.CollectionsDemo
8     {
9         class ListDemo
10        {
11            static void Main()
12            {
13                List<int> evenNos = new List<int>()
14                {
15                    2,4,6,8
16                };
17                List<int> nos = new List<int>();
18                nos.Add(10);
19                nos.Add(20);
20                nos.Add(5);
21                nos.Add(7);
22                nos.Add(1);
23                nos.Add(3);
24                nos.AddRange(evenNos);
25
26                foreach (var item in nos)
27                {
28                    Console.WriteLine(item);

```

121 %   No issues found

Output   Package Manager Console   Error List ...   Immediate Window

and print it using foreach



A screenshot of a Windows Command Prompt window titled "Select C:\WINDOWS\system32\cmd.exe". The window displays a list of integers from 1 to 10, each on a new line. The text is white on a black background. At the bottom of the window, there is a message "Press any key to continue . . .". The command prompt's title bar and some of the window's interface are visible around the edges.

```
10
20
5
7
1
3
2
4
6
8
Press any key to continue . . .
```

**Clear()** --- it removes all the elements from the list

---

similarly look for other things like **average**, **Binarysearch** and all binary search you cannot run directly, you need I Comparer for it

ListDemo.cs\* X ConsoleApp

ConsoleApp

```
11  static void Main()
12  {
13      List<int> evenNos = new List<int>()
14      {
15          2,4,6,8
16      };
17      List<int> nos = new List<int>();
18      nos.Add(10);
19      nos.Add(20);
20      nos.Add(5);
21      nos.Add(7);
22      nos.Add(1);
23      nos.Add(3);
24      nos.AddRange(evenNos);
25      nos.
26      foreach (int item in nos)
27      {
28          Console.WriteLine(item);
29      }
30  }
31  }
32  }
33  }
34 }
```

121 % 2 0 ← →

ConsoleApp.CollectionsDemo.ListDemo

nos.

foreach (int item in nos)

Average

(extension) double IEnumerable<int>.Average()  
Computes the average of a sequence of int values.

This screenshot shows a Microsoft Visual Studio code editor window. The current file is 'ListDemo.cs' in a project named 'ConsoleApp'. The cursor is positioned at the end of the line 'nos.' on line 25. A tooltip is displayed, showing the 'Average' extension method from the 'IEnumerable<int>' interface. The tooltip includes the method signature '(extension) double IEnumerable<int>.Average()' and a brief description: 'Computes the average of a sequence of int values.'. Below the tooltip, the code editor shows the declaration of the 'Average' method and other members of the 'IEnumerable<int>' interface. The status bar at the bottom left indicates a zoom level of 121% and shows two errors (red X) and zero warnings (yellow triangle). The title bar shows the full path 'ConsoleApp.CollectionsDemo.ListDemo'.

## What is **Count** and **Capacity**

this 5 is my capacity here

```
namespace ConsoleApp.CollectionsDemo
{
    class NosArray
    {
        int[] nos = new int[5];
        int cnt = -1;
    }
}
```

and this -1 is count

```
using System.Threading.Tasks;

namespace ConsoleApp.CollectionsDemo
{
    class NosArray
    {
        int[] nos = new int[5];
        int cnt = -1;
    }
}
```

**Element at** ---- I want to know what is the element at a given position

DynArray.cs ListDemo.cs\* X ConsoleApp

ConsoleApp

```
11 static void Main()
12 {
13     List<int> evenNos = new List<int>()
14     {
15         2,4,6,8
16     };
17     List<int> nos = new List<int>();
18     nos.Add(10);
19     nos.Add(20);
20     nos.Add(5);
21     nos.Add(7);
22     nos.Add(1);
23     nos.Add(3);
24     nos.AddRange(evenNos);
25     nos.
26     fore Count<>
27     {
28         DefaultIfEmpty<>
29         Distinct<>
30         ElementAt<> tem;
31     }
32 }
```

Find, FindAll -----

A screenshot of the Visual Studio IDE showing a C# code editor. The file is named `ListDemo.cs`. The code is as follows:

```
11 static void Main()
12 {
13     List<int> evenNos = new List<int>()
14     {
15         2, 4, 6, 8
16     };
17     List<int> nos = new List<int>();
18     nos.Add(10);
19     nos.Add(20);
20     nos.Add(5);
21     nos.Add(7);
22     nos.Add(1);
23     nos.Add(3);
24     nos.AddRange(evenNos);
25     nos.|
26     fore<ElementAt<> tem;
27     {
28         ElementAtOrDefault<> tem;
29     }
30 }
31 }
32 }
33 }
```

The cursor is at line 25, character 11, after `nos.`. A code completion dropdown menu is open, listing various methods starting with `FindAll`, which is highlighted in blue. Other visible methods include `ElementAt`, `ElementAtOrDefault`, `Equals`, `Except`, `Exists`, `Find`, and `FindIndex`.

you can write predicate like this **x goes to x == i want to find 6**

A screenshot of a C# code editor showing a portion of a file named `ListDemo.cs`. The code is as follows:

```
11 static void Main()
12 {
13     List<int> evenNos = new List<int>()
14     {
15         2,4,6,8
16     };
17     List<int> nos = new List<int>();
18     nos.Add(10);
19     nos.Add(20);
20     nos.Add(5);
21     nos.Add(7);
22     nos.Add(1);
23     nos.Add(3);
24     nos.AddRange(evenNos);
25     nos.Find(x => x == 6);
26     foreach (var item in nos)
27     {
28         Console.WriteLine(item);
29     }
30 }
31 }
32 }
33 }
34 }
```

The code uses LINQ's `Find` method on a `List<int>` named `nos`. The `Find` method takes a predicate lambda expression `x => x == 6` as its argument. A yellow lightbulb icon is shown next to the `Find` call, indicating a potential issue or warning.

now what my **find** does is, it will return me a integer

```
nos.Find(x => x == 6);
```

**foreach** (int *list*.Find(*Predicate<int>* *match*)  
Searches for an element that matches the conditions defined by the specified predicate, and returns the first occurrence within the entire *List<T>*.

**Returns:**  
The first element that matches the conditions defined by the specified predicate, if found; otherwise, the default value for type *T*.

**Exceptions:**  
*ArgumentNullException*

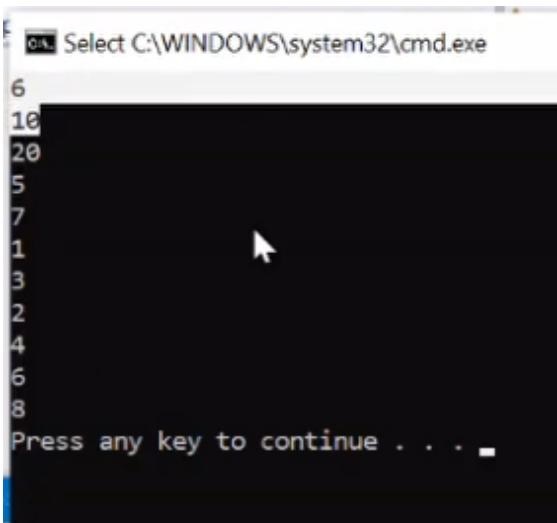
you can put that in *Console.WriteLine*

A screenshot of the Visual Studio IDE showing a code editor window. The window title is "ConsoleApp". The code being edited is in the file "ListDemo.cs" under the namespace "ConsoleApp.CollectionsDemo.ListDemo". The code demonstrates the use of the `Find` method on a `List<int>` to find a specific value (6) and then iterate over the list to print each item. The code is as follows:

```
11 static void Main()
12 {
13     List<int> evenNos = new List<int>()
14     {
15         2,4,6,8
16     };
17     List<int> nos = new List<int>();
18     nos.Add(10);
19     nos.Add(20);
20     nos.Add(5);
21     nos.Add(7);
22     nos.Add(1);
23     nos.Add(3);
24     nos.AddRange(evenNos);
25     Console.WriteLine(nos.Find(x => x == 6));
26     foreach (var item in nos)
27     {
28         Console.WriteLine(item);
29     }
30 }
31 }
32 }
33 }
34 }
```

The status bar at the bottom shows "121 %", "No issues found", and a dropdown menu.

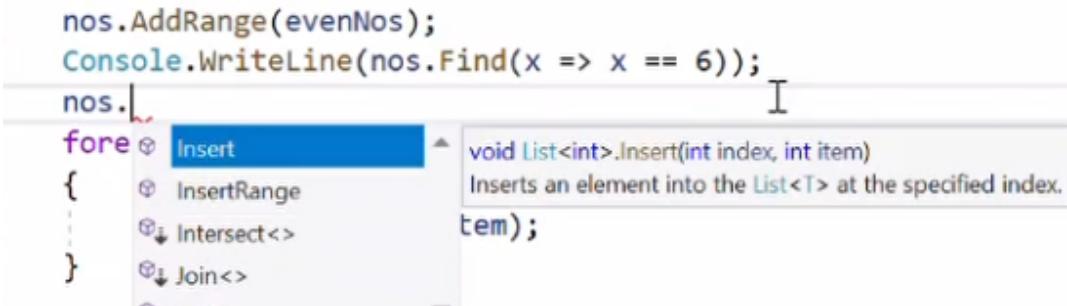
it will give me that particular number



```
6
10
20
5
7
1
3
2
4
6
8
Press any key to continue . . .
```

Where ever you have the predicate, you can always check for the predicate, this will return weather the particular number exists or not

you have **insert, insertrange** ...



in Array list this will be very difficult, going to index and adding the particular value

you have **max---** it will give me maximum number in the list

, **min** --- will give me the minimum number in the list

you have **Remove**, **Remove at** --- you can go to particular position and you can remove element from that position

you have **Reverse** -----it will reverse the whole list

**Sort** --- it will sort the list

**Sum** --- it will sum all the list

so this are some of the methods of list, you can explore more like this

## Stack

Create a new stack, checkout for the methods

The screenshot shows the Microsoft Visual Studio IDE interface. On the left, the Solution Explorer displays a solution named 'ConsoleApp' containing two projects: 'AccessTest' and 'ConsoleApp'. The 'ConsoleApp' project is expanded, showing its files: 'Properties', 'References', and a folder 'CollectionsDemo' which contains 'DynArray.cs', 'IndexersDemo.cs', 'ListDemo.cs', 'NonGenCollection.cs', and 'StackDemo.cs'. On the right, the main code editor window is open to 'StackDemo.cs' under the 'ConsoleApp.CollectionsDemo' namespace. The code is as follows:

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace ConsoleApp.CollectionsDemo
8  {
9      0 references
10     class StackDemo
11     {
12         0 references
13         static void Main()
14         {
15             Stack<int> stacks = new Stack<int>();
16         }
17     }
18 }
```

Similarly check for **Queue**

The screenshot shows a code editor window with the following code:

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace ConsoleApp.CollectionsDemo
8  {
9      class StackDemo
10     {
11         static void Main()
12         {
13             Stack<int> stacks = new Stack<int>();
14             Queue<int> queues = new Queue<int>();
15             //HashSet
16             //Dictionary<I
17         }
18     }
19 }
20
```

The code uses the `System`, `System.Collections.Generic`, `System.Linq`, `System.Text`, and `System.Threading.Tasks` namespaces. It defines a `StackDemo` class within the `ConsoleApp.CollectionsDemo` namespace. The `Main` method creates a `Stack<int>` and a `Queue<int>`. There are two commented-out lines at the end of the class definition.

## HashSet

Now see If I don't know about hashset

just like any other collection i'll create a hashset and try methods on it, like what methods it will have

A screenshot of the Microsoft Visual Studio IDE interface. The main window shows a code editor with the file `HashSetDemo.cs` open. The code demonstrates the use of the `HashSet<int>` collection. The code editor features color-coded syntax highlighting and a vertical ruler on the left side. A tooltip or code completion dropdown is visible at the bottom of the code editor area, showing the method signature `nos.Add`. The status bar at the bottom displays the zoom level (121%), error count (1), and line number (Ln: 1). The menu bar includes File, Edit, View, Git, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help, and a search bar. The toolbar contains various icons for file operations like Open, Save, and Build. The Solution Explorer on the left shows the project structure with files like `ConsoleApp`, `ConsoleApp.cs`, and `ConsoleApp.CollectionsDemo`. The Task List shows one item: `Main()`.

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace ConsoleApp.CollectionsDemo
8 {
9     class HashSetDemo
10    {
11        static void Main()
12        {
13            HashSet<int> nos = new HashSet<int>();
14            nos.Add(
15        }
16    }
17 }
18
```

my hashset will also support add method

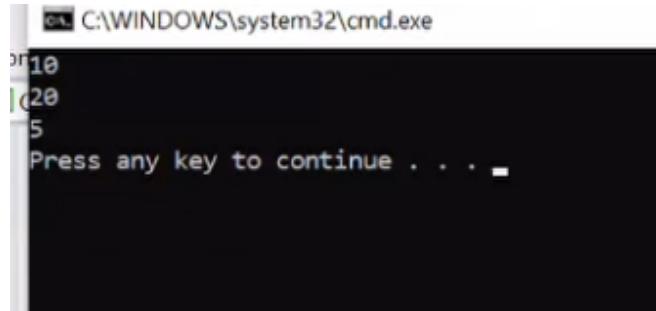
A screenshot of the Microsoft Visual Studio IDE interface. The title bar shows "ConsoleApp" and "HashSetDemo.cs". The menu bar includes File, Edit, View, Git, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, and Help. The toolbar has various icons for file operations like Open, Save, and Build. The code editor displays the following C# code:

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace ConsoleApp.CollectionsDemo
8  {
9      class HashSetDemo
10     {
11         static void Main()
12         {
13             HashSet<int> nos = new HashSet<int>();
14             nos.Add(10);
15             nos.Add(20);
16             nos.Add(10);
17             nos.Add(5);
18
19             foreach (var no in nos)
20             {
21                 Console.WriteLine(no);
22             }
23         }
24     }
25 }
```

The code editor features color-coded syntax highlighting: blue for keywords, green for comments, and purple for strings. A vertical green bar highlights the entire class definition. A yellow lightbulb icon is located at line 19, indicating a potential issue or warning. The status bar at the bottom shows "121 %", "No issues found", and a small icon.

the expected output is 10 20 10 5

but i got this



```
C:\WINDOWS\system32\cmd.exe
10
20
5
Press any key to continue . . .
```

So instead of me creating hashset lets check with list

ConsoleApp    HashSetDemo.cs\* X

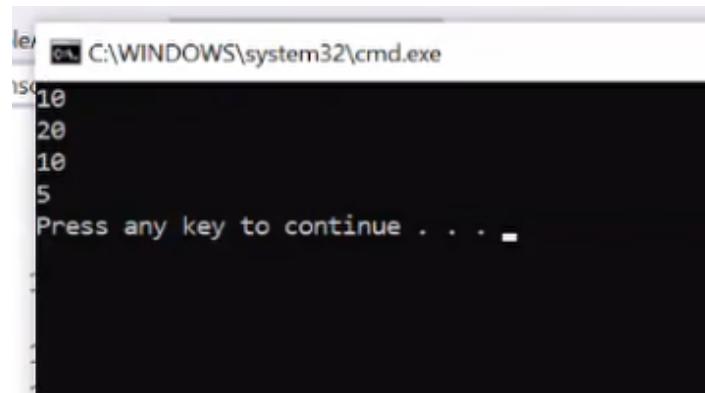
ConsoleApp

ConsoleApp.CollectionsDemo.HashSetDemo

```
7  namespace ConsoleApp.CollectionsDemo
8  {
9      class HashSetDemo
10     {
11         static void Main()
12         {
13             List<int> nos = new List<int>();
14             //HashSet<int> nos = new HashSet<int>();
15             nos.Add(10);
16             nos.Add(20);
17             nos.Add(10);
18             nos.Add(5);

19             foreach (var no in nos)
20             {
21                 Console.WriteLine(no);
22             }
23         }
24     }
25 }
26 }
27 }
```

now i'm getting expected output

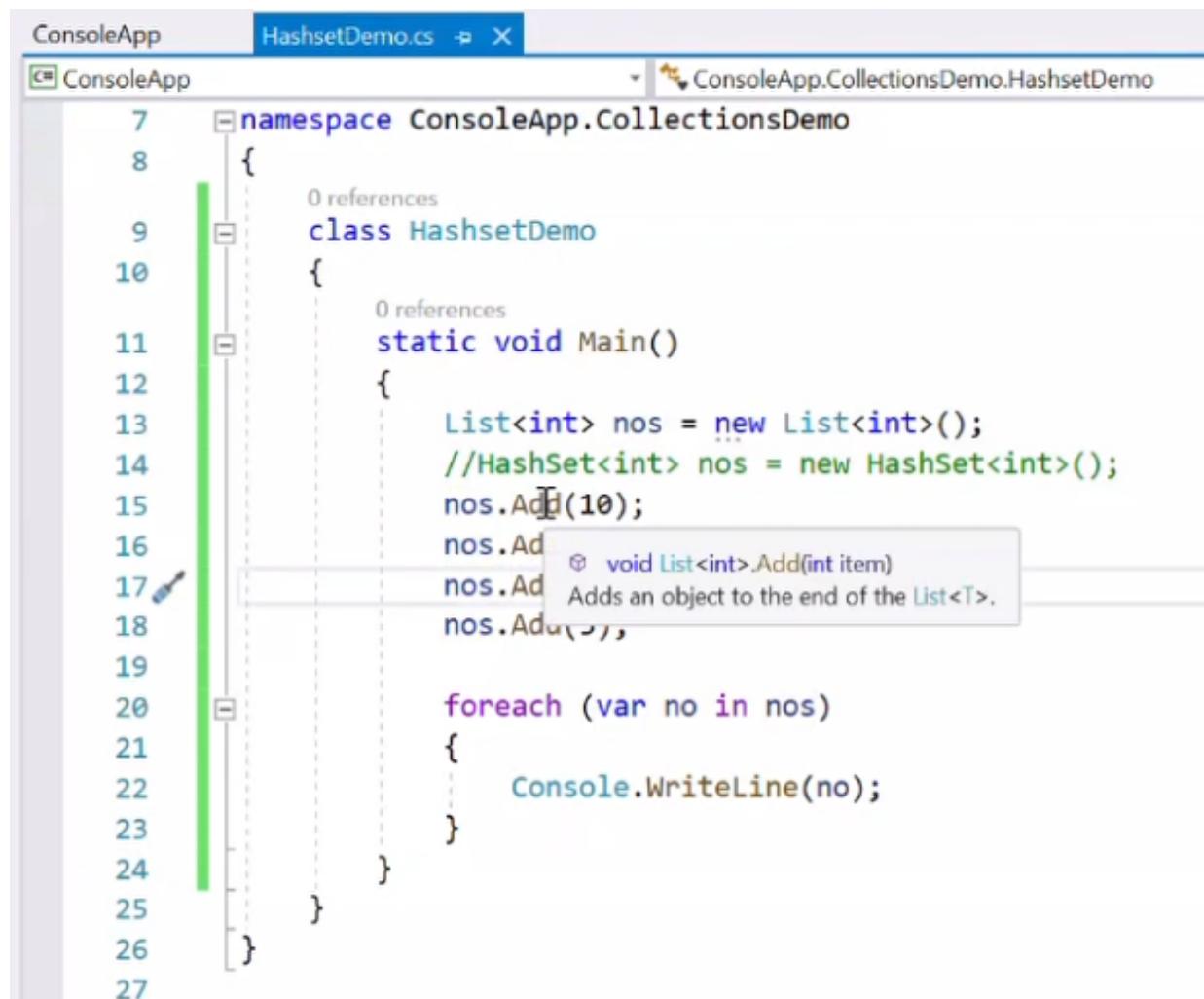


```
le C:\WINDOWS\system32\cmd.exe
150
10
20
10
5
Press any key to continue . . .
```

**But in HashSet why am i not getting 10 twice  
because HashSet is a set of unique element, So it will always  
capture the unique data**

So how do you know 10 is stored twice or not

So look at my add method



```
ConsoleApp          HashSetDemo.cs  X
ConsoleApp          ConsoleApp.CollectionsDemo.HashSetDemo

7  namespace ConsoleApp.CollectionsDemo
8  {
9      class HashSetDemo
10     {
11         static void Main()
12         {
13             List<int> nos = new List<int>();
14             //HashSet<int> nos = new HashSet<int>();
15             nos.Add(10);
16             nos.Add(20);
17             nos.Add(30);
18             nos.Add(40);
19
20             foreach (var no in nos)
21             {
22                 Console.WriteLine(no);
23             }
24         }
25     }
26 }
27
```

in list add method is of void type, so you will not know weather it has added the particular element or not, so it will be difficult for you to know weather the particular element is added or not

**so it's difficult to know with list**

**but look at this with hashset**

ConsoleApp

ConsoleApp.CollectionsDemo.HashSetDemo

Main()

```
7  namespace ConsoleApp.CollectionsDemo
8  {
9      class HashSetDemo
10     {
11         static void Main()
12         {
13             //List<int> nos = new List<int>();
14             HashSet<int> nos = new HashSet<int>();
15             nos.Add(10);
16             nos.    ⚡ bool HashSet<int>.Add(int item)
17             nos.    Adds the specified element to a set.
18             nos.    Returns:
19             nos.        true if the element is added to the HashSet<T> object; false if the element is already present.
20             foreach (var no in nos)
21             {
22                 Console.WriteLine(no);
23             }
24         }
25     }
26 }
27 }
28 }
```

121 %

No issues found

Ln: 14 Ch:

Output Package Manager Console Error List Immediate Window

**with hashset add is of boolean type**, that's how you will know weather the element is added or not in the hashset So you can always put a message,it is not added or it's a duplicate element

Apart from this you can see, we have a count method just like list, we have a Remove method , Contains , average, comparer, count .....  
all this methods are still there just like your list

## Dictionary

**Speciality of dictionary is it will store the data in a key, value pair**

and dictionary is something which we usually use alot in project

A screenshot of a C# code editor showing a tooltip for the `Dictionary<TKey, TValue>` class. The code editor window has tabs for "DictDemo.cs\*" and "ConsoleApp\*". The current file is "ConsoleApp.cs" under the namespace "ConsoleApp.CollectionsDemo". The code defines a `DictDemo` class with a `Main` method. A tooltip is displayed over the `Dictionary<TKey, TValue>` type, providing its definition: "Represents a collection of keys and values." and noting "`TKey`: The type of the keys in the dictionary."

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace ConsoleApp.CollectionsDemo
8  {
9      class DictDemo
10     {
11         static void Main()
12         {
13             Dictionary<TKey, TValue>
14         }
15     }
16 }
17
```

so dictionary of , i'll put my key as integer and my value as string

```
DictDemo.cs* ✘ X ConsoleApp*  
ConsoleApp  
ConsoleApp.CollectionsDemo.DictDemo  
Main()  
1  using System;  
2  using System.Collections.Generic;  
3  using System.Linq;  
4  using System.Text;  
5  using System.Threading.Tasks;  
6  
7  namespace ConsoleApp.CollectionsDemo  
8  {  
9      class DictDemo  
10     {  
11         static void Main()  
12         {  
13             Dictionary<int, string> contacts = new Dictionary<int, string>();  
14         }  
15     }  
16 }  
17
```

now how do you store the data, so we'll try  
try with add



so yes dictionary will support add method which is of void type



so this is how we add the data

now to fetch the data again we'll use the foreach loop

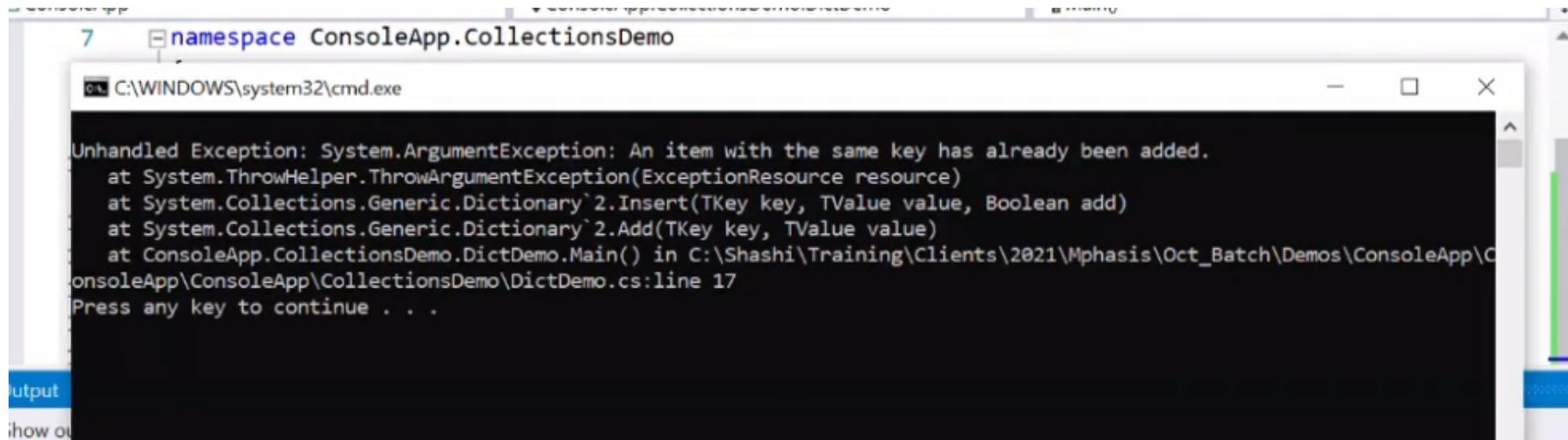
keep this in your mind when ever you are creating a collection it should always be a plural , it cannot be a book, it should be books

The screenshot shows a Microsoft Visual Studio interface with the following details:

- Title Bar:** Shows "DictDemo.cs" and "ConsoleApp\*" as tabs.
- Project Explorer:** Shows "ConsoleApp" as the selected project.
- Toolbox:** Shows "ConsoleApp.CollectionsDemo.DictDemo" and "Main()".
- Code Editor:** Displays the following C# code:

```
7  namespace ConsoleApp.CollectionsDemo
8  {
9      class DictDemo
10     {
11         static void Main()
12         {
13             Dictionary<int, string> contacts = new Dictionary<int, string>();
14             contacts.Add(1, "Shashi");
15             contacts.Add(2, "Sham");
16             contacts.Add(3, "Ravi");
17             contacts.Add(1, "Anju");
18             contacts.Add(5, "Soman");
19
20             foreach (var contact in contacts)
21             {
22                 Console.WriteLine(contact);
23             }
24         }
25     }
26 }
27 }
```
- Tooltips:** A tooltip for the variable "contact" is displayed, indicating it is a local variable of type `KeyValuePair<int, string>`.
- Status Bar:** Shows "121 %", "No issues found", "Ln: 22 Ch: 30 Col: 42", "TABS", and "CI".

when i move my mouse pointer on this, see contact is in key value pair  
so when i execute this i'll get error



The screenshot shows a Windows Command Prompt window titled "ConsoleApp.CollectionsDemo". The command entered is "C:\WINDOWS\system32\cmd.exe". The output window displays the following error message:

```
Unhandled Exception: System.ArgumentException: An item with the same key has already been added.
  at System.ThrowHelper.ThrowArgumentException(ExceptionResource resource)
  at System.Collections.Generic.Dictionary`2.Insert(TKey key, TValue value, Boolean add)
  at System.Collections.Generic.Dictionary`2.Add(TKey key, TValue value)
  at ConsoleApp.CollectionsDemo.DictDemo.Main() in C:\Shashi\Training\Clients\2021\Mphasis\Oct_Batch\Datas\ConsoleApp\Collections\ConsoleApp\CollectionsDemo\DictDemo.cs:line 17
Press any key to continue . . .
```

which says

An item with the same key has already been added.

we, already have data in the particular key

see by mistake we wrote 1 twice

**So Remember whenever the data in key repeated it throws**

## System.ArgumentException

it is a argument exception, not arguments exception

```
contacts.Add(1, "Shashi");
contacts.Add(2, "Sham");
contacts.Add(3, "Ravi");
contacts.Add(1, "Anju");
contacts.Add(5, "Soman");
```

so it should have been 4

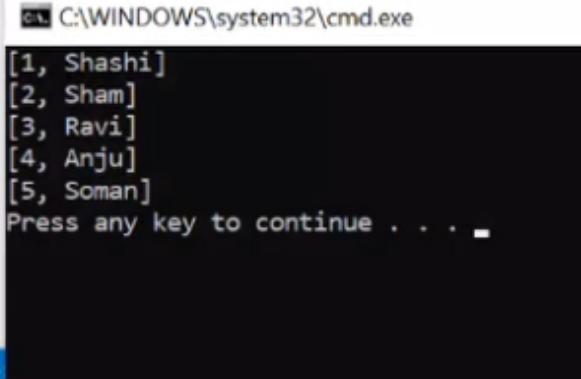
The screenshot shows a Microsoft Visual Studio code editor window. The title bar indicates the file is "DictDemo.cs\*" and the project is "ConsoleApp". The code editor displays the following C# code:

```
7  namespace ConsoleApp.CollectionsDemo
8  {
9      class DictDemo
10     {
11         static void Main()
12         {
13             Dictionary<int, string> contacts = new Dictionary<int, string>();
14             contacts.Add(1, "Shashi");
15             contacts.Add(2, "Sham");
16             contacts.Add(3, "Ravi");
17             contacts.Add(4, "Anju");
18             contacts.Add(5, "Soman");
19
20             foreach (var contact in contacts)
21             {
22                 Console.WriteLine(contact);
23             }
24         }
25     }
26 }
27
```

The code defines a class named "DictDemo" with a static "Main" method. Inside "Main", a dictionary "contacts" is created and populated with five entries. A "foreach" loop then iterates over the dictionary, printing each entry to the console.

now see we got the output

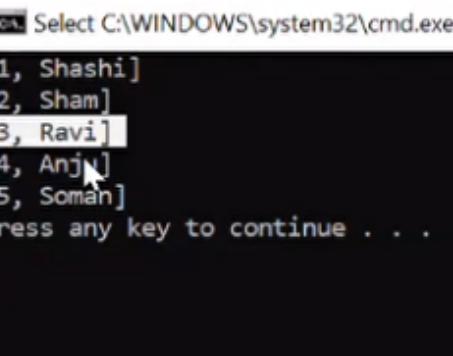
```
namespace ConsoleApp1
{
    class Program
    {
        static void Main(string[] args)
        {
            var dict = new Dictionary<string, string>()
            {
                {"Shashi", "1"}, {"Sham", "2"}, {"Ravi", "3"}, {"Anju", "4"}, {"Soman", "5"}
            };
            foreach (var item in dict)
            {
                Console.WriteLine($"[{item.Key}, {item.Value}]");
            }
            Console.WriteLine("Press any key to continue . . .");
        }
    }
}
```



when you look into this there is a square bracket ,

```
Console.WriteLine($"[{item.Key}, {item.Value}]");

```



so this is one object, this is one key, value pair what I'm getting

I want to differentiate between my key and value

so you can write like this

DictDemo.cs\* ✘ X ConsoleApp

ConsoleApp

ConsoleApp.CollectionsDemo.DictDemo

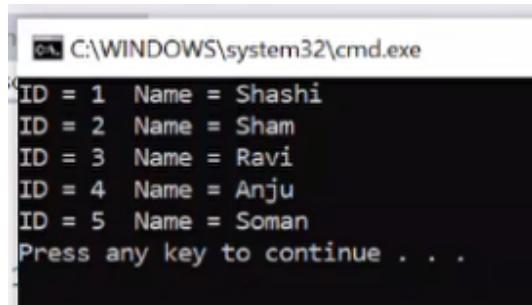
Main()

```
7  namespace ConsoleApp.CollectionsDemo
8  {
9      class DictDemo
10     {
11         static void Main()
12         {
13             Dictionary<int, string> contacts = new Dictionary<int, string>();
14             contacts.Add(1, "Shashi");
15             contacts.Add(2, "Sham");
16             contacts.Add(3, "Ravi");
17             contacts.Add(4, "Anju");
18             contacts.Add(5, "Soman");
19
20             foreach (var contact in contacts)
21             {
22                 Console.WriteLine("ID = {0} \tName = {1}", contact.Key, contact.Value);
23             }
24         }
25     }
26 }
27
```

121 %

No issues found

Ln: 22 Ch: 34 Col: 46 TAB



C:\WINDOWS\system32\cmd.exe

```
ID = 1 Name = Shashi
ID = 2 Name = Sham
ID = 3 Name = Ravi
ID = 4 Name = Anju
ID = 5 Name = Soman
Press any key to continue . . .
```

lets look into some complex thing in dictionary

Imagine I want to store **name with multiple contacts**

**How do you store this**

there were multiple solution to do this 1 employee can have many phone numbers

We already did this, in traier trainee app, 1 trainer can have many trainees

DictDemo1.cs X ConsoleApp\*

C# ConsoleApp

```
namespace ConsoleApp.CollectionsDemo
{
    class Employee
    {
        public string Name { set; get; }
        private List<string> contacts = new List<string>();

        public void AddContact(string Contact)
        {
            this.contacts.Add(Contact);
        }

        public List<string> GetContacts()
        {
            return this.contacts;
        }
    }

    class DictDemo1
    {
    }
}
```

121 % No issues found

Output Package Manager Console Error List ... Immediate Window

now tomorrow i want to create a contact i'll just write

The screenshot shows a code editor window with the following details:

- Title Bar:** DictDemo1.cs - ConsoleApp\*
- Project Bar:** ConsoleApp - ConsoleApp.CollectionsDemo.DictDemo1
- Code Area:** The code is written in C# and defines a class named DictDemo1. It contains a public method GetContacts() which returns a List<string>. It also contains a static void Main() method where an Employee object is created, its name is set to "Shashi", and two contacts are added.

```
18
19     public List<string> GetContacts()
20     {
21         return this.contacts;
22     }
23 }
24
25 class DictDemo1
26 {
27     static void Main()
28     {
29         Employee employee = new Employee();
30         employee.Name = "Shashi";
31         employee.AddContact("8971472005");
32         employee.AddContact("8839938388");
33     }
34 }
35
36
```

- Status Bar:** Shows 121 % zoom, a green checkmark icon, and the message "No issues found".

if it's a right thought - See its 2 min's job

wrong thought - it will be 2 years job and even after 2 years you cannot fix it because you are on wrong path, don't make the things complicated

we are taking about dictionary, then all of us have a common mind set , that shashi is talking about dictionary so solution is dictionary, yaa may be solution is dictionary but have you ever tried something else

Please keep thinking of something else there, then only you will get multiple solution, **don't stop at one solution**

So when i do this in list, so why should i go with dictionary

now here if you see, the employee is bounded with the name and the contact

```
2 references
class Employee
{
    1 reference
    public string Name { set; get; }
    I private List<string> contacts = new List<string>();
}

2 references
```

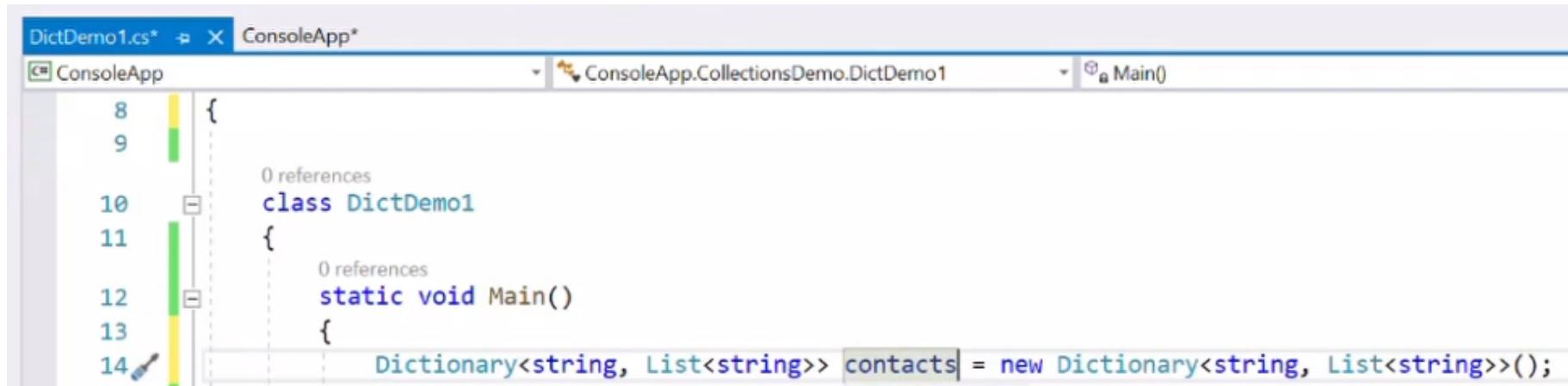
So contact is a part of employee, it is wrapped in the same box, called employee, we cannot segregate it if i want to access this contact , i can go only through employee like employee.contacts

So there are chances where this design may be wrong for some time

i want to loosely couple this 2 things, i want to keep employee name seprate and contact seprate , but they should be access via same pair , they should be club together

at that point of time you can think of a dictionary

So in the dictionary i'll select the name as a key which will be my string and contact no as list because we have to store many contacts



```
DictDemo1.cs*  ✘ X ConsoleApp*
ConsoleApp  ✘ ConsoleApp.CollectionsDemo.DictDemo1  ✘ Main()
8   {
9
10  class DictDemo1
11  {
12    static void Main()
13    {
14      Dictionary<string, List<string>> contacts = new Dictionary<string, List<string>>();
```

but what if you have 2 person with the same name, you cannot store that with the string, you should take employee id in that case as employee id will always be unique

The screenshot shows a Microsoft Visual Studio code editor window. The title bar indicates the project is named "ConsoleApp" and the file is "DictDemo1.cs". The code itself is as follows:

```
8  {
9
10 class DictDemo1
11 {
12     static void Main()
13     {
14         Dictionary<string, List<string>> contacts = new Dictionary<string, List<string>>();
15         List<string> contacts1 = new List<string>();
16         contacts1.Add("8971472005");
17         contacts1.Add("8383884838");
18         contacts.Add("Shashi", contacts1);
19
20         List<string> contacts2 = new List<string>();
21         contacts2.Add("7782722005");
22         contacts2.Add("9829924838");
23         contacts.Add("Ravi", contacts2);
24         //Employee employee = new Employee();
25         //employee.Name = "Shashi";
26         //employee.AddContact("8971472005");
27         //employee.AddContact("8839938388");
28     }
29 }
```

now to get data we use foreach loop  
and as our contact is also a list we use foreach loop there too

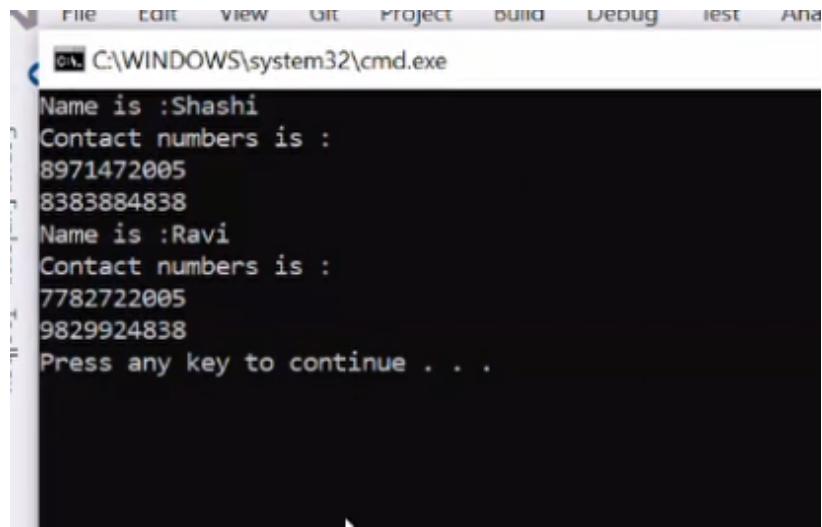
DictDemo1.cs X ConsoleApp\*

ConsoleApp

```
15     List<string> contacts1 = new List<string>();
16     contacts1.Add("8971472005");
17     contacts1.Add("8383884838");
18     contacts.Add("Shashi", contacts1);
19
20     List<string> contacts2 = new List<string>();
21     contacts2.Add("7782722005");
22     contacts2.Add("9829924838");
23     contacts.Add("Ravi", contacts2);
24
25     foreach (var contact in contacts)
26     {
27         var name = contact.Key;
28         Console.WriteLine("Name is :" + name);
29         Console.WriteLine("Contact numbers is : ");
30         var contactNos = contact.Value;
31         foreach (var no in contactNos)
32         {
33             Console.WriteLine(no);
34         }
35     }
36
37     //Employee employee = new Employee();
38     //employee.Name = "Shashi";
```

121 % No issues found

Ln: 1



```
Name is :Shashi
Contact numbers is :
8971472005
8383884838
Name is :Ravi
Contact numbers is :
7782722005
9829924838
Press any key to continue . . .
```

**Assignment :**  
**WAP to print mark report  
like this**

\*Untitled - Notepad

File Edit Format View Help

### Mark Details

---

Name: Shashikanth

#### Test Details

---

Test Code - T001

Subject Code - S001

Subject Mark - 30

Mark Scored - 23

Remark - P

---

Test Code - T002

Subject Code - S002

Subject Mark - 30

Mark Scored - 26

Remark - P

---

Test Code - T003

Subject Code - S003

Subject Mark - 30

Mark Scored - 29

Remark - P

---



## **.NET Collections and Generics**

## Objectives

- After the end of the Session the participants should be able to:
  - Problems with Array.
  - Understand what are Collections.
    - What?
    - Why?
    - How?
  - Understand Generics- GenX Collections.
  - What are the built-in Classes of Generics
  - What are the Interfaces of Generics
  - Create Custom Generics
  - Implementing Custom generics.

The slide has a blue header bar with a navigation menu containing items like Home, About, Services, Portfolio, Contact, and Log Out. Below the header is a dark grey sidebar with a vertical list of numbers from 1 to 12. The main content area has a white background with a black header bar labeled 'Current Scenario'. The text in the main area is as follows:

- Arrays.
  - The Most simplest way of storing data as a Single Unit.
  - Fixed size is not always the Real time scenario.
  - How Could you achieve dynamic features in an Array?
  - [Example](#)

Here in this Example, I have a Static Array of Employees initially set with the size 0. Currently No Employees are added to the Array. A function called `AddNewEmployee` is used to add a new Employee to the Array.

Limitation is Array is fixed in Size. So I have to think on a logic to achieve the mechanism of Adding Employees to the current set, without loosing my old Data.

## Collections

- What?
  - An object that stores a group of elements together as one unit.
  - A group of data of similar type semantically grouped for a purpose.
  - Arrays are the most primitive type of Collections available in all Languages.
- Why?
  - Traditional Arrays are fixed in Size.
  - Real time Applications needs an ability to append, Remove or modify the existing Data as per their needs.
  - An Example without an Idea of Dynamic Data.
- How?
  - .NET framework provides a huge sets of classes which fulfill the issues of Collections.
  - Grouped under the namespace [System.Collections](#).

## Collections under .NET Terminology

- Anything that allows to refer each element within it is called Collection.
- Simply put, It's a class that implements an interface called `IEnumerable`.

The screenshot shows a Microsoft Word slide with the following content:

**System.Collections**

- The .NET framework has a variety of built in Classes which provide a puzzling selection of collection objects, each with a somewhat specialized purpose.
- Supports 4 General types of Collections
  - Non Generic
  - Specialized
  - Bit based.
  - Generic

The slide has a dark blue header bar with the title "System.Collections". The main content area is white with black text. There are two solid black horizontal bars, one above the bullet points and one below them. The Word ribbon is visible at the top, showing tabs like Home, Insert, Design, Animations, Slide Show, Review, View, and Developer. The status bar at the bottom shows page numbers from 1 to 12.

The principal benefit of collections is that they standardize the way groups of objects are handled by our programs. All collections are designed around a set of clearly defined interfaces. Several built-in implementations of these interfaces, such as **ArrayList**, **Hashtable**, **Stack**, and **Queue**, are provided, which we can

## Generic vs. Non Generic

- Non Generic stores the data as Object
  - Not type safe
  - Available in v1.0 and v1.1
- Generic are type safe way of storing the objects as Data structures
  - Available since v2.0
  - Easy way of storing objects.
- Example of Non Generic Version.

## Using Non Generic Class

```
ArrayList list = new ArrayList();
list.Add(123); //Stores the data as object(Boxed Value)
list.Add("SomeName"); //Accepts this, but Wrong interpretation.
//Compiler does not throw Error.
```

## Using Generic Class

```
List<int> list = new List<int>();
list.Add(123); //Stores the data as integer.
list.Add("SomeName"); /*Does not Accepts this and
Compiler tells about this Error.*/
```

## Retrieval Issues

- Using Non Generics:

```
foreach(object element in list)
{
    if (element is int)//Check first for type safety.
    {
        //Integer Operations
    }
    else
    {
        //Non Integer Operations
    }
}
```

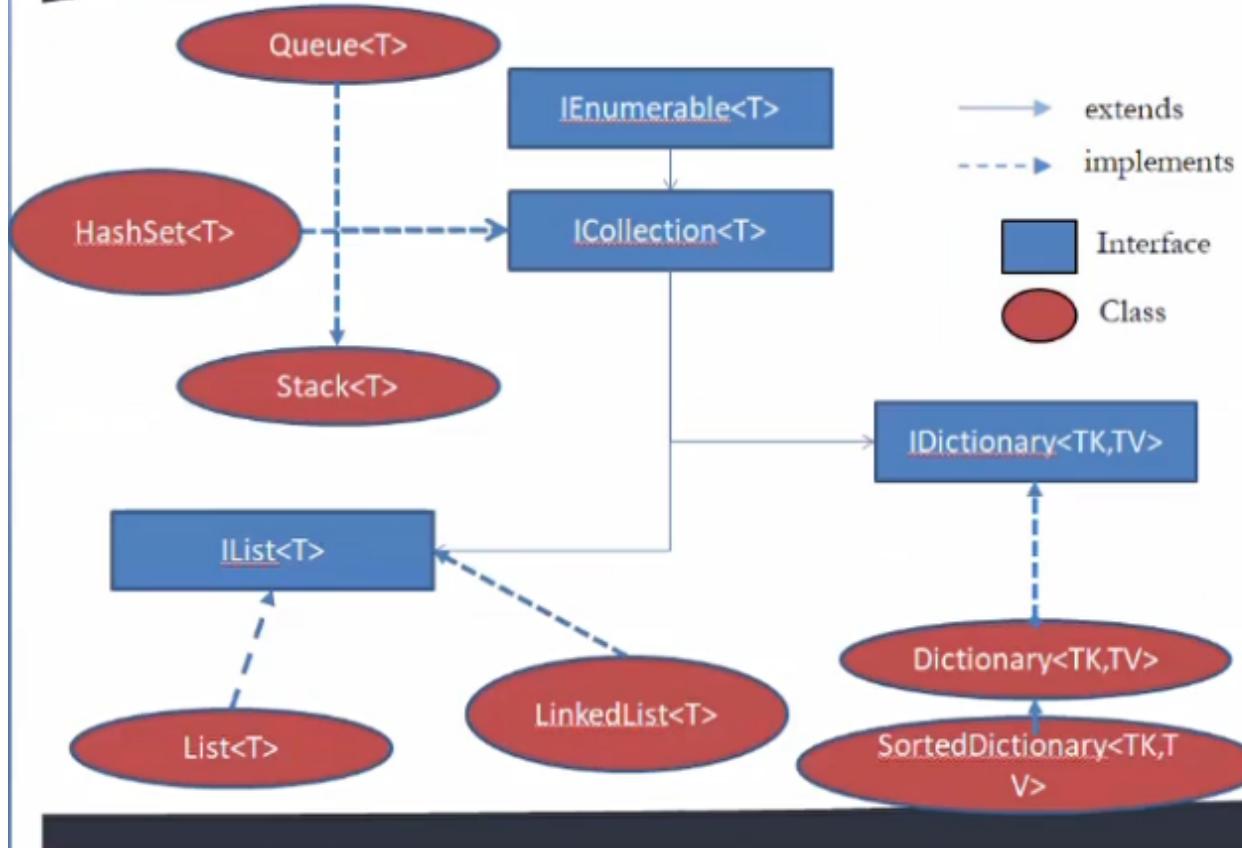
- Using Generics:

```
foreach(int element in list)//No Need to check for Integer
{
    //Integer Operations
}
```

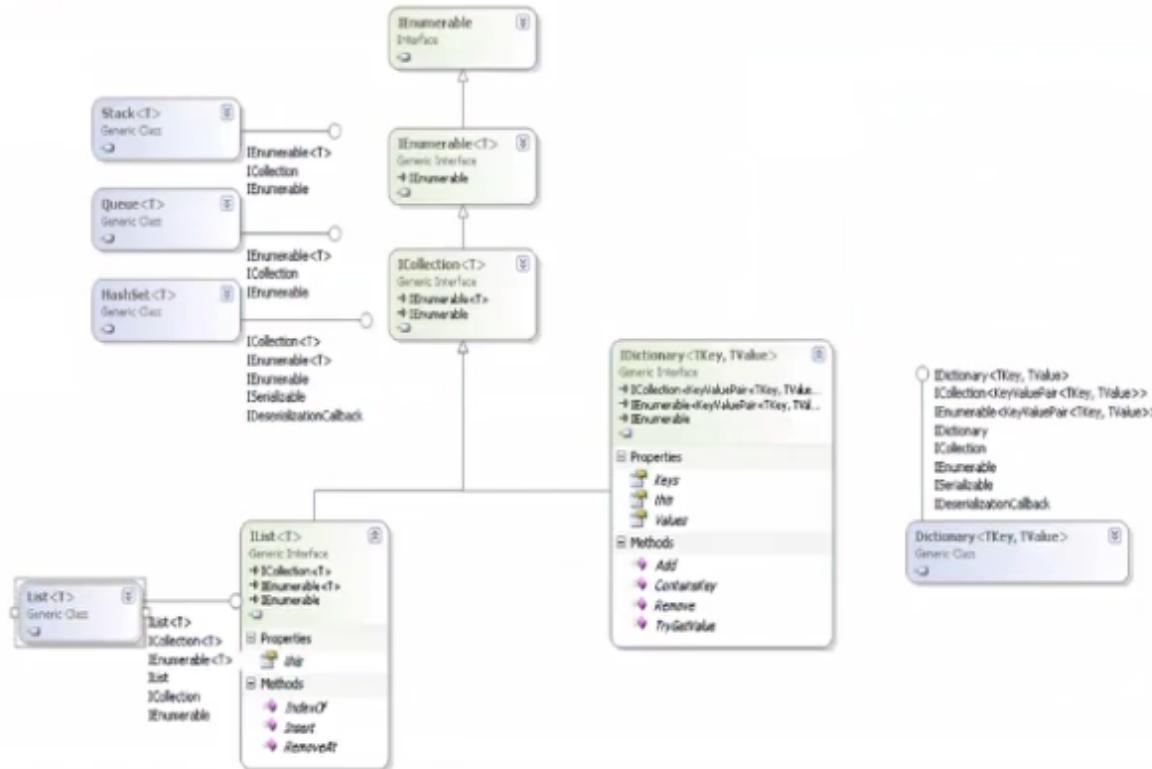
## Generics

- GenX Collections.
- The whole idea is to avoid accidental Type mismatches.
- Similar to Templates of C++.
- Have Classes to perform type safe way of storing the data.

## The Classes and Interfaces of Generics.



## Complete Hierarchy



## Classes under Generics

Class	Description
List<T>	A dynamic array. Provides functionality similar to that found in the non-generic <a href="#">ArrayList</a> class.
Dictionary<TK, TV>	Stores key/value pairs. Provides functionality similar to that found in the non-generic <a href="#">Hashtable</a> class.
HashSet<T>	Represents a set of values.
LinkedList<T>	Stores elements in a doubly linked list.
Queue<T>	A first-in, first-out list. Provides functionality similar to that found in the non-generic <a href="#">Queue</a> class.
SortedDictionary<TK, TV>	A sorted list of key/value pairs.
SortedList<TK, TV>	A sorted list of key/value pairs. Provides functionality similar to that found in the non-generic <a href="#">SortedList</a> class.
Stack<T>	A first-in, last-out list. Provides functionality similar to that found in the non-generic <a href="#">Stack</a> class.

## The List<T> Design

The diagram illustrates the design of the List<T> class. It is a generic class that implements the `IList<T>`, `ICollection<T>`, and `IEnumerable<T>` interfaces. The class has the following members:

- Properties:** Capacity, Count, this
- Methods:** Add, Clear, CopyTo (+ 2 overloads), Exists, Find, FindAll, List (+ 2 overloads), Remove, Sort (+ 3 overloads), ToArray
- Nested Types:** Enumerator (a struct that implements `IEnumerator<T>`, `IDisposable`, and `IEnumerator`)

- Provides a Generic Dynamic Array kind of Data storage in it.
- Stores the data as First In Last Out.
- Simplest form of storing the data where we can modify the structure at any point of time.

Some of the Important Methods:

Add->Adds the Element to the Last of the list.

Remove->Removes the Specified Element from the List.

## List<T> Usage

- Lists are Dynamic Array.
- Element is added to the Last of the list.
- The Count gives the number of elements within the List.



## Code Snippet

```
class Phone{...}

class Program
{
    static void Main(string[] args)
    {
        List<Phone> mobiles = new List<Phone>();//Initially no Phones will be available
        mobiles.Add(new Phone { PhoneID = 111, PhoneModel = "N73", PhoneCost = 13200 });
        mobiles.Add(new Phone { PhoneID = 112, PhoneModel = "xPhoria", PhoneCost = 18000 });
        mobiles.Add(new Phone { PhoneID = 113, PhoneModel = "5800Xpress", PhoneCost = 12200 });
//Phones are added using Add Method...

        mobiles.Remove(mobiles[2]);
    }
}
```