**Installation Tools:**

1. .Net Core SDK (translate/execute C# code into instruction for the machine)
2. Any Editor like (VS code)
3. Add C# extension in VS code

Any high level language needs additional software to execute the code

Create program

1. Create folder of project (contain 2 folders src, test)
2. Command line in src (dotnet new console)
3. Command line in test(dotnet new xunit)
4. Build command (dotnet test)
5. To add reference of project inside project (dotnet add reference ../src/src.csproj)

**Variables:**

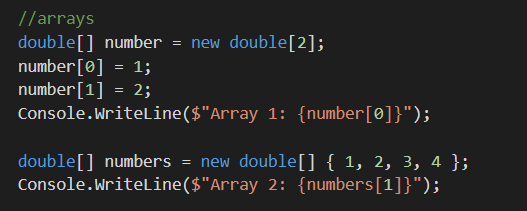
We can use var for all types

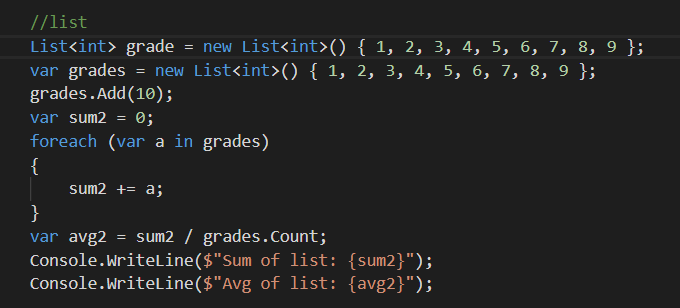
Double x =3.45 the same as var x =3.45

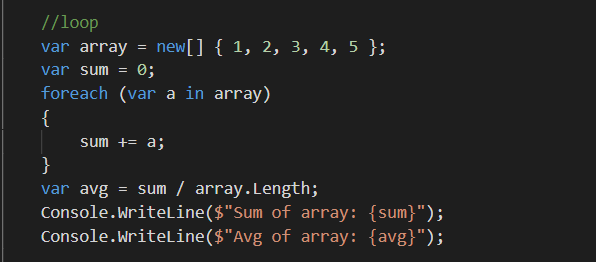
Need first initialization to use variable

Wrong   
int x;   
x=x+1;

Right   
int x=0;  
x=x+1;

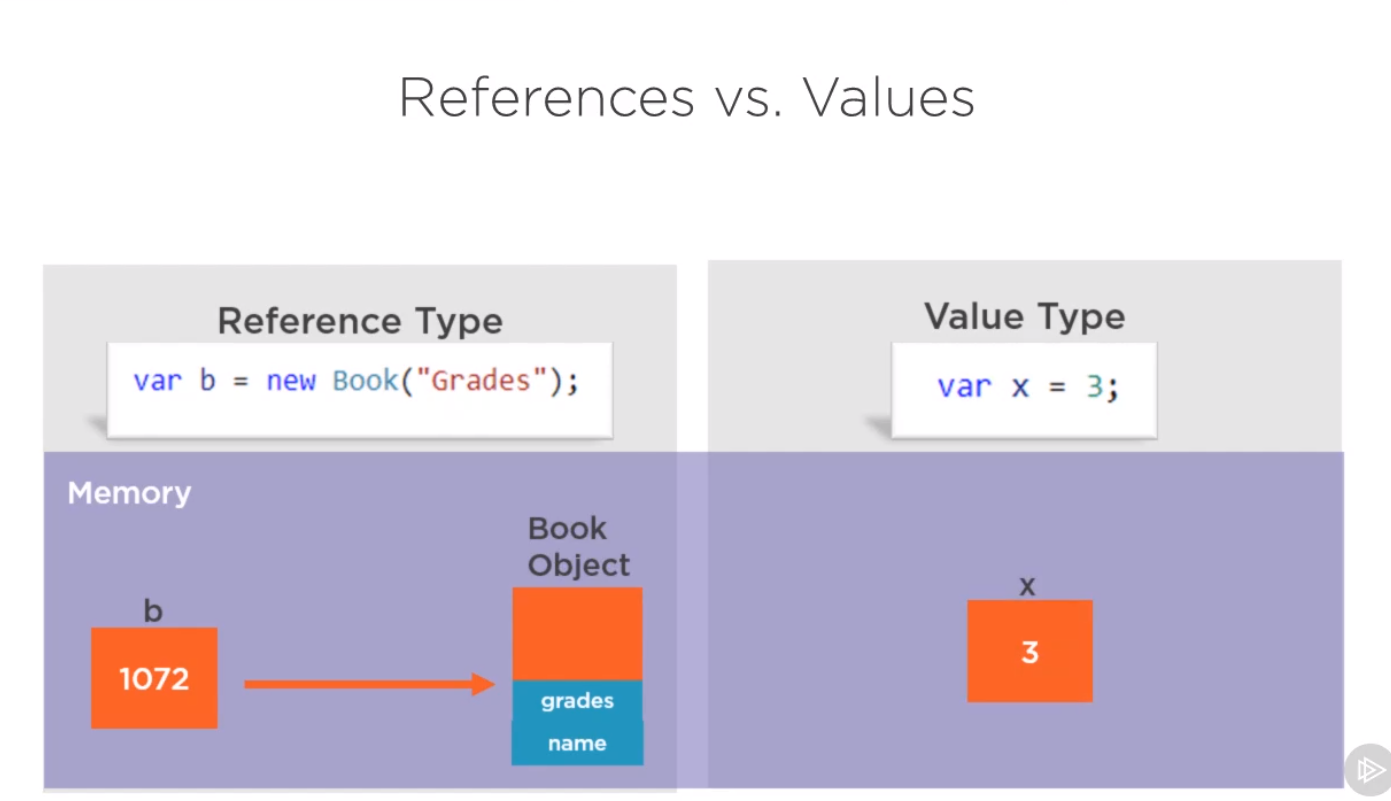
**Arrays:  
**

**List:**

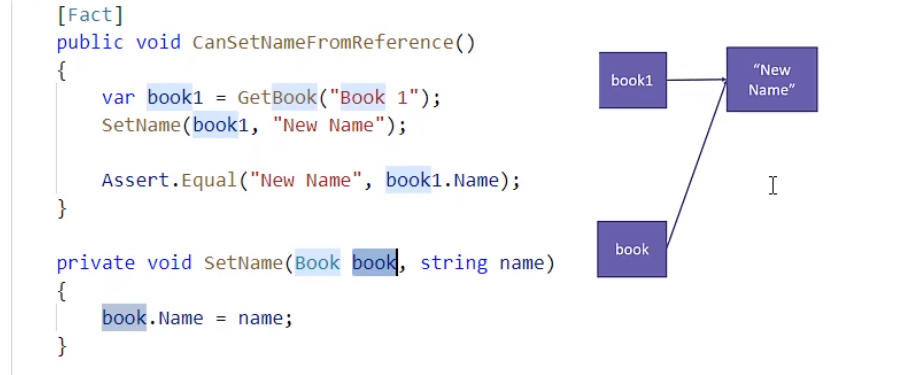
**Loop:**  


**Default access modifier of class : internal (can use it in the same project)**

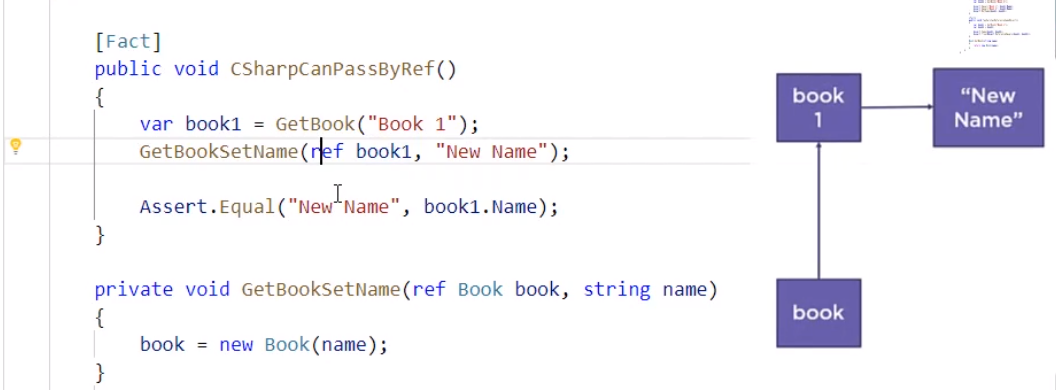
**Reference & value Types:**

****

****

****

****

****

**Solution file:**

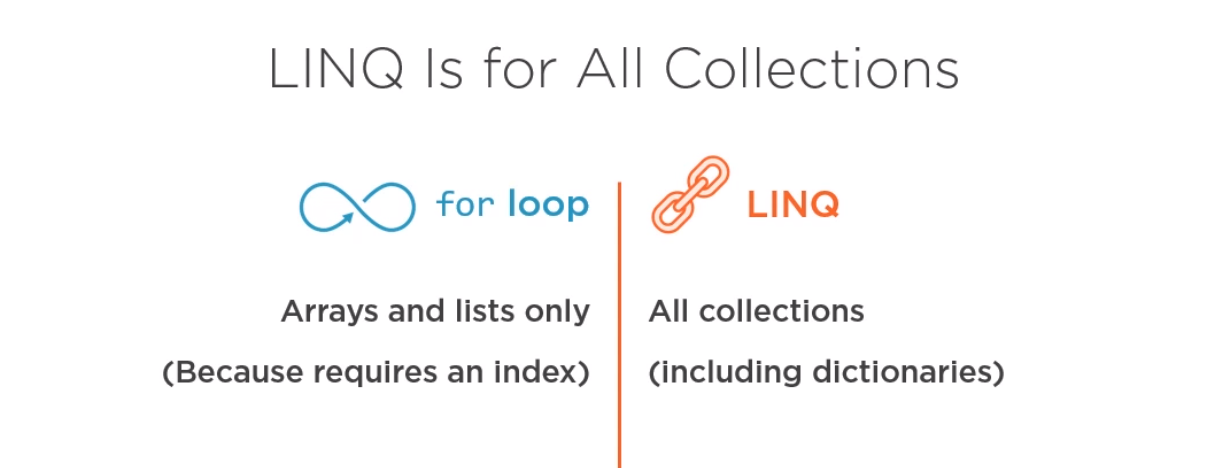
To build and run multi projects

1. Command line: Dotnet new sln (in folder that contains all project or another folder)
2. Add ur projects (Command line: dotnet sln add (path of csproj))
3. Command line: dotnet build (to build ur projects)

**Collections:** array – list – dictionary

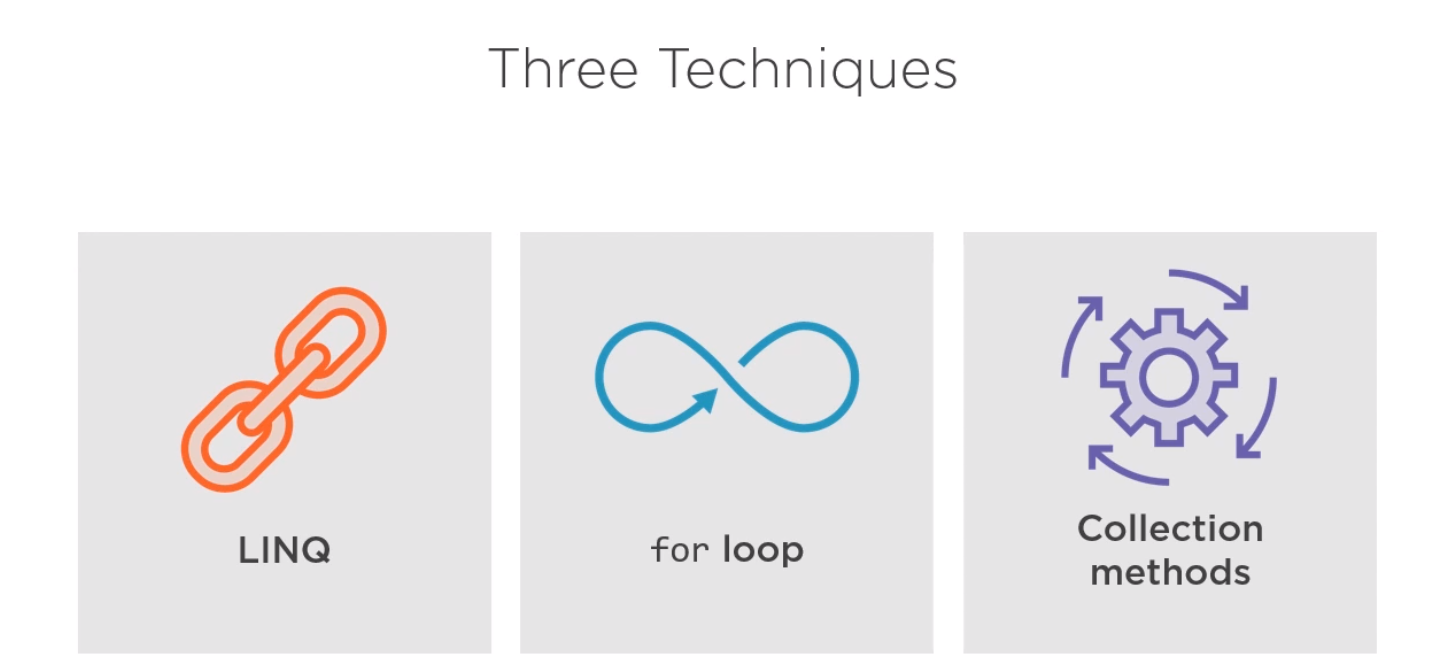
Immutable(array,…) collection -> can’t be modified once instantiated

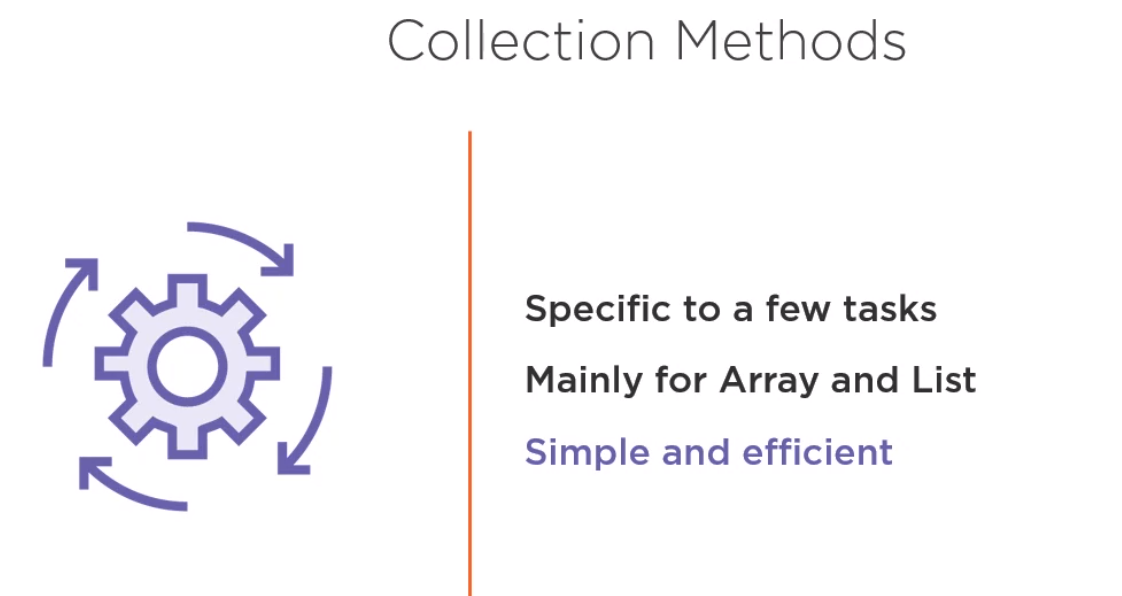
Concurrent(array,…) collection -> if you need to access collection from multiple threads



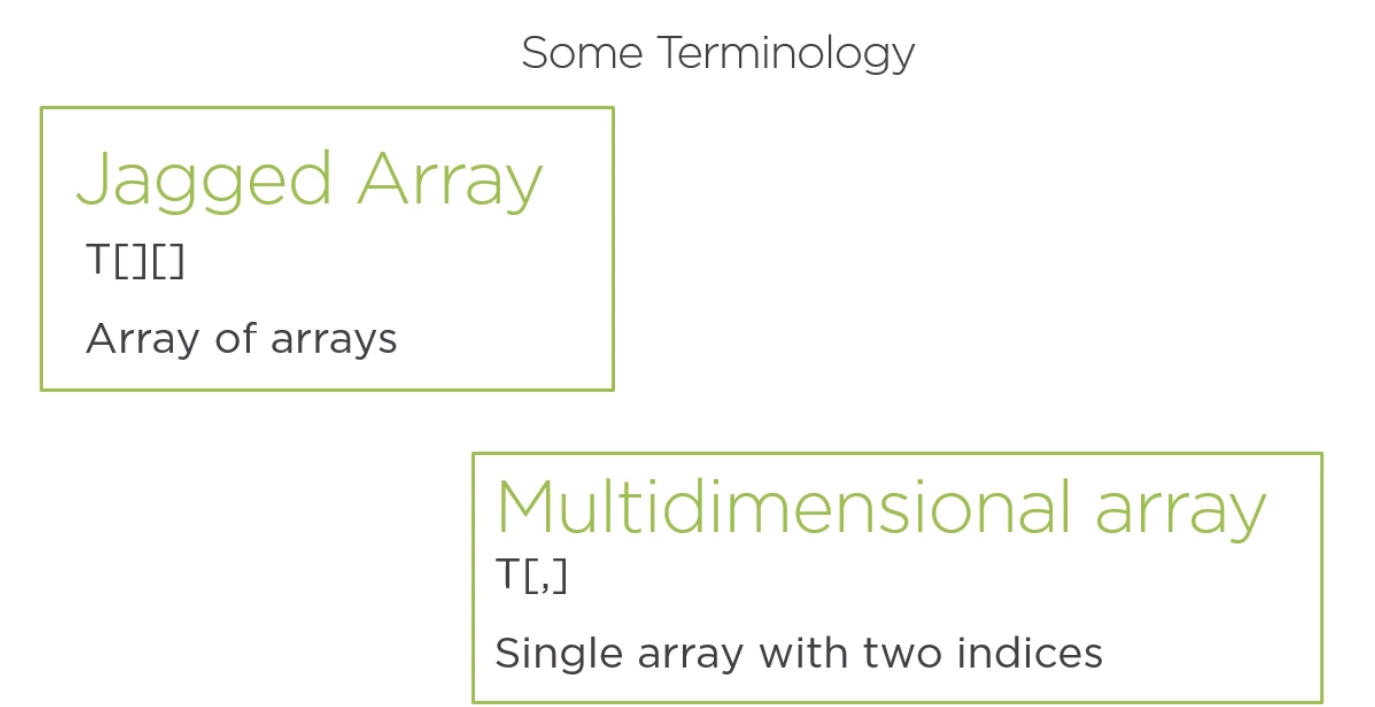
Linq : methods help the user to query data , linq is read only can’t modify the data , not only for collections , you can use linq to access data taken from external DB

3 techniques to treat with collections



Jagged Array vs multidimensional array



**Indexers:**

An indexer allows an object to be indexed such as an array. When you define an indexer for a class, this class behaves similar to a virtual array. You can then access the instance of this class using the array access operator ([ ]).

class IndexedNames {

private string[] namelist = new string[size];

static public int size = 10;

public IndexedNames() {

for (int i = 0; i < size; i++)

namelist[i] = "N. A.";

}

public string this[int index] {

get {

string tmp;

if( index >= 0 && index <= size-1 ) {

tmp = namelist[index];

} else {

tmp = "";

}

return ( tmp );

}

set {

if( index >= 0 && index <= size-1 ) {

namelist[index] = value;

}

}

}

static void Main(string[] args) {

IndexedNames names = new IndexedNames();

names[0] = "Zara";

names[1] = "Riz";

names[2] = "Nuha";

names[3] = "Asif";

names[4] = "Davinder";

names[5] = "Sunil";

names[6] = "Rubic";

for ( int i = 0; i < IndexedNames.size; i++ ) {

Console.WriteLine(names[i]);

}

Console.ReadKey();

}

}

**Delegates:**

**Events:**

Generics:

List and dictionary a type of generic