*For loop can be used for any type of collection, where as foreach loop can be used for array only.

- Sealed keyword is applied to the method & classes to prevent overriding and inheritance.

- Constructure is a method that is automatically executed when an object is created.

- String Builder class more memory than is actually needed. It is used for substitutions & appending or removing text from string.

- when want to modify the string without creating anew object use System.Text.StringBuilder class

Example: [88] ⟶ (C#) Heap

StringBuilder sb = new StringBuilder ("C#");

we an specify (length & capacity) as properties.

Char, Length

inbuild methods

Compare(—,—) return integer value of same in sort order

CompareTo(—)

Concat(—,—) concatinate two strings.

Contains(—)

EndsWith(—)

Equals(—,—)

GetEnumerator()

GetHashCode()

IndexOf(string)

Join(string, string[])

Replace(—,—)

Split(char[])

StartWith(String)

Substring(Int32) retrieve substring

ToCharArray() copy char in instance

Example ⇒ input.Replace("ay", "xa");

Code example,

① Spliting when a ',' occur.

```
string res = "a,b,c,d";
string [] res = res.split(',');
for ... print
    i < res.Length;
```

② check upper, lower, space in text.

```
int up=0; int lw=0; int sp=0;
for(int i=0; i< input.Lenght; i++)
{ if(char.IsUpper(input[i]))
    {     IsLower(      ))
          IsWhiteSpace(      ))
{ upper++;
  low++;
  sp++; }
```

Q your task is to separate the name contains only alphabets and may contain white space.

use: Split(), Replace(), IsLetter()

```
string name;
int i;
Console.WriteLine("enter the name:");
name = Console.ReadLine();
string[] namecoll = name.Split(",");
for(i=0;i<namecoll.Length,i++)
 { name = namecoll[i].Replace(" ","");
   if(char.IsLetter(namecoll[i],7))
    { console.writeline(name+"=>ok");
    }
    else
    {Console.WriteLine(name+"=>Notok");
    }
 }
```

=> Liam Zok, NOk Roca, Emma Boo3n

o/p => LiamZok => ok

NokRoca => ok

EmmaBoo3n => Not ok

Q wap to replace char of odd place by next alphabet. print invalid when not string present.

input	output
Ayusha2	Byvsia

```
string input;
string tempchar="";
bool flag=false;
Console.WriteLine("enter a string");
input=Console.ReadLine();
for(int i=0;i<input.Length;i++)
{ char temp=input[i];
 if(!char.IsLetter(temp))
 {console.writeline("invalid");
  break;}
 else if((i+1)%2!=0)
 { if(temp=='z')
    {tempchar+='a';}
   else if(temp=='Z')
    {tempchar+='A';}
   else
    {temp++;
     tempchar+=temp;}
 }
 else
 {tempchar+=temp;}
} // Loop ends here.
if(flag==false)
 {console.WriteLine(input+tempchar);}
```

Dictionary

- A dictionary is key value pairs
- It is present in System.Collections.Generic namespace.

Example

we will use customer class in this.

```
namespace Demo{
Class Program
{ public static void Main()
  { // creating Customer object.
    Customer cust1 = new Customer()
    { ID=101; Name='Hello'; };
    Customer cust2 = new Customer()
     { ID= 102, Name='Kali',};
    // creating dictionary<key,value>
    Dictionary< int , Customer> dictCust = new.....();
    // adding item to Dictionary
    dictCust.Add (cust1.ID , cust1);
    // printing value
Customer a= dictCust[101];
Console.WriteLine("ID={0}, Name={1}", a.ID, a.Name);
  }
}
pubic class Customer
 { public int ID{get;set;}
   public String Name{get;set;}
}
}
```

* we can print the values by the the help of foreach loop.

```
foreach(KeyValuePair<int, Customer>  custkvalue in dictCust)
                    ↑ variable (var)
{ custkvalue
  Customer cust = custkvalue.value
  Console.WriteLine(.....,cust.ID,cust.Name);
}
// it will print the values.
```

- for only keys → Customer (or) cust
  → values

```
foreach(int key in dictCust.Key)
  { Console.WriteLine(key); }
```

- ContainsKey()

File System

To read a filepath use string from user

```
path = @"c:\capgemini";
```

Reading all the dir present

```
string[] Dir = Directory.GetDirectories(path);
```

Listing subdirectories

```
foreach (string dir in Dir)
{ if (Directory.Exist(dir))
  { Console.WriteLine(dir);
    // for info relate to it
    var dirInfo = new DirectoryInfo(dir);
    Console.WriteLine(dirInfo.Name);
  }
}
```

For getting file use ".GetFiles"

① `var Files = Dictionary.GetFiles(path);`

// all files use `foreach(string fil in Files)`

② To check specific datatype file.

```
var f = Directory.Getfile(path, "*.txt", SearchOption.
        TopDirectoryOnly);
// for each to print.
```

Binary Serialize

① To do Serialization we will create a Class with methods

②

```
Class DataSerializer
{  // methods                                  lenahai        save to
   public void BinarySerializer(object data, string filepath)
   {  // Check file exist then delete.
      FileStream fileStream;
      BinaryFormatter bf = new BinaryFormatter();
      if (File.Exists(filepath)) File.Delete(filepath);
      // create new file.
      fileStream = File.Create(filepath);
      bf.Serialize(fileStream, data);
      fileStream.Close();
   }
   public Object BinaryDeserial(string filepath)
   {  Object obj = null;
      Filestream filestream;
      BinaryFormatter bf = new BinaryFormatter();
      if (File.Exists(filepath))
      {  fileStream = File.OpenRead(filepath);
         obj = bf.Deserializer(filestream);
         filestream.close(); }
```

② public class Person
{
Public string FirstName
{get; set;}
Public string LastName
{get; set;}

main method

① // taking obj for data of person
Person person = new Person() { FirstName = "J", LastName = "K" };
string filepath = "data.save";
DataSerializer datas = new DataSerializer();
Person P = null; //obj for person
datas.BinarySerializer(person, filepath);
P = datas.BinaryDeserial(filepath) as Person;

Console.WriteLine(P.FirstName);
}
//public class person

this help to initilize the attributes

use when parameter & instence var. has same name.

constructor

```
Class Employee
{ int empId;
  String name;
  public Employee(int empId, String name)   //in const
   { this.empId = empId;
     this.name = name;
   }
   void display()
     { console.writeline("ID:"+id+"Name"+name); }
  static void main(String args[])
   { Employee e = new Employee(101, "Tom");
     e.display(); }
}
```

Structures

- Struct is user defined data type, It contains related data & methods.
- It is of value type, a struct var cannot inherited from another struct variable.
- It is similar to class. Difference is it is class → Reference type.

Example

```
public struct co
 { public int x,y;
 public co(int p1,int p2)
   {  x= p1;
      y= p2;
   }
}
```

```
class Pgm
{ static void Main()
  { co pt= new co();
    co pt2= new co(10,10);
   Console.Write("co 1 : ");
   Console.WriteLine("x={0}, y={1}", pt.x, pt.y);
    "      "      "      ", pt2.x, pt2.y);
  }
}
```

Delegates are similar to a function pointer in C/C++.
It is an object which stores the reference/address of function.
function signature must match.

Example

```
namespace unicast
    public delegate int fn(int x, int y);
    class Pgm
    { static void main()
①     { fn obj = new fn(Add);
②       int r = obj.Invoke(10,20);
③       Console.Writeline(r); // sum
④     }
⑤     Public static int Add(int x, int y)
⑥       { return (x+y); }
⑦   }
⑧ }
```

multicast Note change int to void

add after ① ⇒ obj = obj + new fn(subt);

after 6th ⇒ public static void subt(int x, int y)
{ Console.Writeline(x-y); }

⑨ Remove 'int x'

Methods (function)

```
using System;
class Program
{ public static void Main()
  { // creating instance of class
    Program P = new Program();
    // to access the method
    P.EvenNumber();
    // if we will do static below then
    // we will use class directly program.
       // EvenNumber()
  }
public void EvenNumber()
  { int s = 0;
    while (s <= 20)
      { Console.WriteLine(s);
        s = s + 2;
      }
  }
}
```

① passing value Normaly by value to function

```
int i = 10;
EvenNum(i);
}
public void EvenNum(int j)
  { j = 100; }
// op of i => 10 only.
EvenNum(ref i); }
public void EvenNum(ref int j)
  { j = 100; }
// op of i = 100 → change value by reference
```

Static & Instance class members.

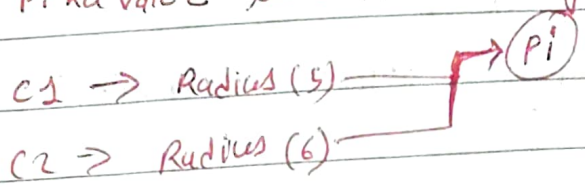
Hum instance use karte hai to har ek object k liye alag alag copy ban jayegi.

For example,

```
Class Circle
  { float -pi = 3.14; // static -pi = 3.14
    int -Radius;
  Public Circle (int Radius)
    { this.-Radius = Radius;
    }
  Public float Calculation()
    { return this.-Pi * this.-Radius * this.-Radius; }
}                    // Circle.-Pi
Class pgrm
  { P S V M ()
    { Circle c1 = new Circle(5);
      float area = c1.calculation();

      Circle c2 = new circle(6);
      float area2 = c2.Calculation();
    }
  }
// yaha pr C1 => one location
//        C2 => another location.
```

// Jab static use karenge tab Pi ka value sare share karenge.



Static method.

Static method no overloading.

Create a Product with properties → PId, PName, Pcost
in main method take input.
create DispProduct & Pass obj of Product.

Product.cs

```
namespace op1
{ Public class Product
  { //creating property
    public int PId{get;set;}
    public string PName{get;set;}
    public float pcost{get;set;}
  }
}
```

Program.cs

```
namespace op1
{ public class pgm
  { psvm()
    int ProductId;
    string Name;
    float Pcost;
    // Take input
    pgm pobj = new pgm();
    Product pro = new Product();
    pro.PId = productId;
    pro.PName = Name;
    pro.Pcost = Pcost;
    Console.WriteLine(pobj.Display(pro));
  }
  public string Display(Product obj)
  { int/string pid = obj.PId;
    string PName = obj.Pname;
    Console.WriteLine(obj.PId + obj.
      PName + P.Cost);
  }
}
```

Example2

To give value to property

```
Class Emp
  { int eId;
    public int Eid
      { set { eId = value; }
        get { return eId; }
      }
  }
```

call from main method

```
Class Pgm
{ s u m
  { Emp e1 = new Emp();
    e1.EId = 101;
    Console.WriteLine(e1.EId);
  }
}
```

Class, Property, Constructor,

__/__/____

Automated teller Machine,

create class Program => properties { AccountNum, Name, balance }
method in Program => pub str RegisterAccount (take input)
method deposit =>(pgm obj, cash)
method withDrawCash(pgm obj, cash)

```
Public class Program
{ double balance;
  string AcNum {get; set;}
  string AcName {get; set;}
  Public double Balance
  { set {balance = value;}
    get { return balance;}
  }

Public string RegisterAccount(string aNum, aName, double bal)
 { this.AcNum = aNum; this.AcName = aName; this.balance = bal;
   return "Registered";}
public double DepositCash (Program obj, double bal)
 { obj.Balance += bal; return obj.Balance; }
public double WithDrawCash(Program obj, double cash)
{ if(cash <= this.balance)
  { this.balance = this.balance - cash; return this.balance; }
}

psvm( )
 { // acept using while (choice != 4)
   Program pObj = new Program();
   // pObj. RegisterAccount( ---, ---, ---);
   //      . DepositCash( pObj, depositAmt);
   //console--. WithDrawCash( pObj, withAmt);
}}
```

class obj

Create a class to perform task like take input account type numbers, balance.

① create a public class with private members id, accountType, balance
② create public property for the same.
③ create a "withDraw" public method that return double & take amount as parameters.
④ To print details create GetDetails.

```
⇒ public class Account
  { int id;
    string accountType;
    double balance;
    // public property.
    public int Id
    { set { id = value; }
      get { return id; }
    }
    public string AccountType
    { set { accountType = value; }
      get { return accountType; }
    }
    public double Balance
    { set { balance = value; }
      get { return balance; }
    }
  // constructor to pass value.
  public Account (int id, string accountType, double balance)
    { this.id = id;
      this.accountType = accountType;
      this.balance = balance; }
  public Account()
    { }
//
Public bool WithDraw (double amount)
 { if (amount <= this.balance)
   { return true; }
   else { return false; }
 }
public string GetDetails()
{ Account acc = new Account();
 if (acc.WithDraw (this.balance) == true)
  {// this.id, this.accountType, this.wid... }
else { return "balance error"; } }
public class Pgm
 { psvm()
  {// Take input; declare var
   Account obj = new Account();
  Console.WriteLine (obj.GetDetails());
 }
}
```