```
using System;

public class A
{
    public void method()
    {
       Console.WriteLine("A");
    }
}

public class B : A
{
    public void method1()
    {
      Console.WriteLine("B");
    }
}

public class C : B
{
   public void method2()
   {
    Console.WriteLine("C");
   }
}

public class Program
{
    public static void Main()
    {
    A ObjA = new A();
    ObjA.method();

    A ObjB = new B();
    ObjB.method();

    B ObjB1 = new B();
    ObjB1.method1();

    A ObjC = new C();
    ObjC.method();

    C ObjC1 = new C();
    ObjC1.method2();

    }

}
```

```
//

Int32 a = 6;
object b = a;
long c = (long)b;


class A

{

    public A() => Console.WriteLine("A");

}

class B : A

{

    public B() => Console.WriteLine("B");

}

class Program

{

    static void Main()

    {

        B b = new B();

    }
}

class A
{
    public virtual void Show() => Console.WriteLine("A");
}

class B : A
{
    public new void Show() => Console.WriteLine("B");
}

class Program
{
    static void Main()
```

```
    {
        A a = new B();
        a.Show();
    }
}

<div class="col-sm-3"></div>
.forEach
for
emp=10
save = 5, update = 5
create type table
using merge
target table
when matched
up
when unmatched
in

var myVal  = {"12, 3, 5", "4, 7, 9", "6, 9, 11"}
js=5,nm,ag
<div id="md"></div>
let rows="";
$.each(js,function(ind,it){
rows+=`<tr data-id="${ind}"><td>${it.nm}</td><td>${it.ag}</td><td><button
data-id="${ind}" class="nameBtn">Get Name</button></td></tr>`;

});
let result =`<table></table>`;
$(document).on("click",".nameBtn",function(){
        let getId=$(this).data('id');
$(`tr[data-id="${getId}"]`).find('td:eq(0)').text();
});


public sealed class AppUtilities{

        private AppUtilities(){}

        public string EncryptData( string plainData){

//some logic
return encrypedData;
}
private object LockForAppUtil;
private static appUtil;
public static AppUtil {
lock(LockForAppUtil){
        appUtil ??= new AppUtil();
return appUtil;
```

```
}

}

}


Test

public static class ExtendTest{

        public static string ToLocalPath(this Test path){
                //some logic
                return $"/local/{pathValue}";
        }
}
Stu
var result = Stu.Select(x=> new {x.Name,Doj = x.Doj.ToString("MM/dd/yyyy")});

void SetValues(ref int a, out int b)
{
    a += 10;
    b = 20;
}

int x = 5;
int y;

SetValues(ref x, out y);
Console.WriteLine($"{x}, {y}");




using System;
public abstract class Shape
{
    public abstract double CalculateArea();

    public void PrintDetails()
    {
        Console.WriteLine("This is a shape.");
    }
}

public class Rectangle : Shape
{
    public double Width { get; set; }
    public double Height { get; set; }
    public override double CalculateArea()
```

```
    {
        return Width * Height;
    }
}

public interface IAnimal
{
    void MakeSound();
}
public class Dog : IAnimal
{
    public void MakeSound()
    {
        Console.WriteLine("Woof!");
    }
}

public class Cat : IAnimal
{
    public void MakeSound()
    {
        Console.WriteLine("Meow!");
    }
}
public class Program
{
    static void Main(string[] args)
    {
        Shape rectangle = new Rectangle();
        rectangle.PrintDetails();
        ((Rectangle)rectangle).Width = 5;
        ((Rectangle)rectangle).Height = 3;
        Console.WriteLine("Area of rectangle: " + rectangle.CalculateArea());

        IAnimal dog = new Dog();
        dog.MakeSound();
        IAnimal cat = new Cat();
        cat.MakeSound();
    }
}



int[] arr = {1,1,2,2,3,3,4,5,5};
output - 12345

int added = arr[0];
string r = added.ToString();
List<int> result = new List<int>(added);
for(int i = 1; i < arr.Length; i++){
```

```
        if(added != arr[i] ){
         r+=arr[i].toString();
        }
added = arr[i];

}


public class CustomExceptionHandler(RequestDelegate reqDel, IRestApiService iRest)
: IMiddleWare
{

        private readonly  RequestDelegate _reqDel = reqDel;


        public override Task async InvokeAsync(HttpContext httpContext){
                try{
        //use any logic here
//tran

}
        catch(Exception ex){}

        }
}


var employees = new List<(string Name, int Salary)>

{

    ("Akash", 5000),

    ("Bader", 7000),

    ("Manoj", 7000),

    ("Akansha", 4000),

    ("Pradeep", 6000),

    ("Garav", 5000)

};

get 2nd highest salary but among distinct salary.

var secondHDSalary =
employees.GroupBy(x=>x.Salary).Where(x=>x.Count()==1).Select(x=>x.Salary).OrderByDe
scending(x=> x).Skip(1).Take(1).First();
```

ICustRepo

uow
{
	ICustRepo a = _custRepo ?? new CustRepo();
}

public class LoadConfigData:IMiddleWare
{
private readonly IConfiguraion _config;
private readonly RequestDelegate _del;
	public LoadConfigData(RequestDelegate del, IConfiguration config){
_del=del;
_config=config;
}

}

UserRes

<Resource>
<MergeDictionaries src="UserRes" x:Name="res" />
</Resource>

<Button Style="{DynamicResource res.}"

Repo
:

public class BaseRepository<T>{

}

internal class UserRepository : BaseRepository<UserModel>{
	private st
}

service-

Task.WhenAll

med1
med2

```
int a =99;
object aBox = a;
int unboxed = (int)aBox;

(object EventData){
        Emp e = (Emp)EventData;
}
```