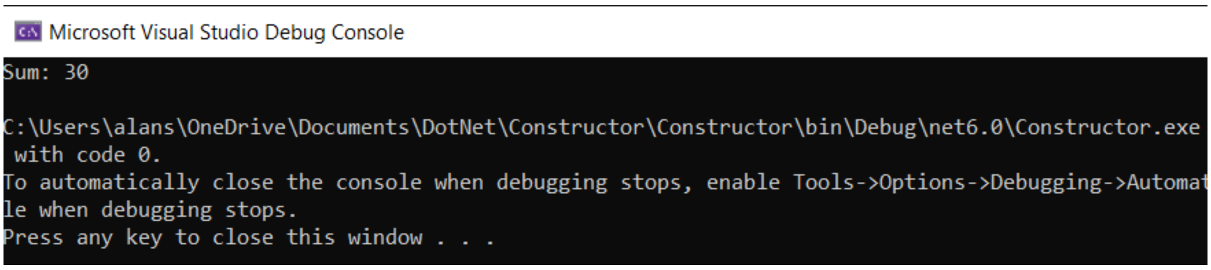
To Show Constructor, Method Overloading and Indexers

Constructor

Code:

```
using System;
namespace ParameterizedConstructor
{
    class Sum
    {
        private int x;
        private int y;
        public Sum(int a, int b)
        {
            x = a;
            y = b;
        }
        public int getSum()
        {
            return x + y;
        }
    }
    class Test
    {
        static void Main(string[] args)
        {
            Sum s = new Sum(20, 10);
            Console.WriteLine("Sum: {0}", s.getSum());
        }
    }
}
```

Output:



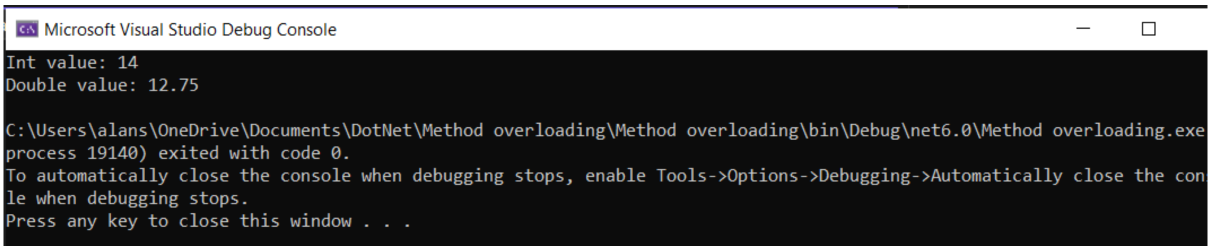
```
Sum: 30

C:\Users\alans\OneDrive\Documents\DotNet\Constructor\Constructor\bin\Debug\net6.0\Constructor.exe
 with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automat
le when debugging stops.
Press any key to close this window . . .
```

Method overloading

Code:

```
namespace MyApplication
{
    class Program
    {
        static int Add(int x, int y)
        {
            return x + y;
        }

        static double Add(double x, double y)
        {
            return x + y;
        }

        static void Main(string[] args)
        {
            int myNum1 = Add(5, 9);
            double myNum2 = Add(5.5, 7.25);
            Console.WriteLine("Int value: " + myNum1);
            Console.WriteLine("Double value: " + myNum2);
        }
    }
}
```

Output:



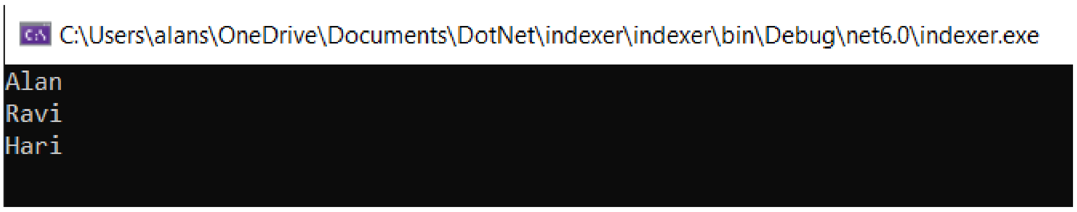
```
Int value: 14
Double value: 12.75

C:\Users\alans\OneDrive\Documents\DotNet\Method overloading\Method overloading\bin\Debug\net6.0\Method overloading.exe
process 19140) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the con
le when debugging stops.
Press any key to close this window . . .
```

Indexer:

Code:

```
using System;
namespace Indexer_example
{
    class Program
    {
        class IndexerClass
        {
            private string[] names = new string[10];
            public string this[int i]
            {
                get
                {
                    return names[i];
                }
                set
                {
                    names[i] = value;
                }
            }
        }
        static void Main(string[] args)
        {
            IndexerClass Team = new IndexerClass();
            Team[0] = "Alan";
            Team[1] = "Ravi";
            Team[2] = "Hari";
            for (int i = 0; i < 10; i++)
            {
                Console.WriteLine(Team[i]);
            }
            Console.ReadKey();
        }
    }
}
```

Output:


C:\Users\alans\OneDrive\Documents\DotNet\indexer\indexer\bin\Debug\net6.0\indexer.exe

```
Alan
Ravi
Hari
```

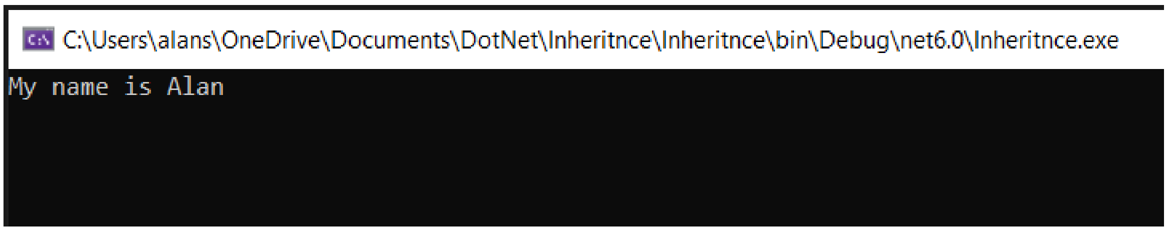
To Show Inheritance, Sealed Class and use of BASE keyword

Inheritance:

Code:

```
using System;

namespace Inheritance
{
    class A
    {

        public string name;

    }
    class B : A
    {

        public void getName()
        {
            Console.WriteLine("My name is " + name);
        }
    }

    class Program
    {

        static void Main(string[] args)
        {
            B obj = new B();
            obj.name = "Alan";
            obj.getName();
            Console.ReadLine();
        }

    }
}
```

Output:



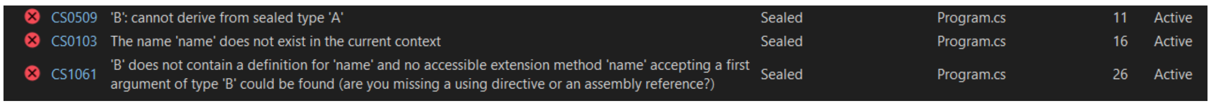
```
C:\Users\alans\OneDrive\Documents\DotNet\Inheritnce\Inheritnce\bin\Debug\net6.0\Inheritnce.exe
My name is Alan
```

Sealed Class

Code:

```
using System;

namespace Inheritance
{
   sealed class A
    {

        public string name;

    }
    class B : A
    {

        public void getName()
        {
            Console.WriteLine("My name is " + name);
        }
    }

    class Program
    {

        static void Main(string[] args)
        {
            B obj = new B();
            obj.name = "Alan";
            obj.getName();
            Console.ReadLine();
        }

    }
}
```

Output:



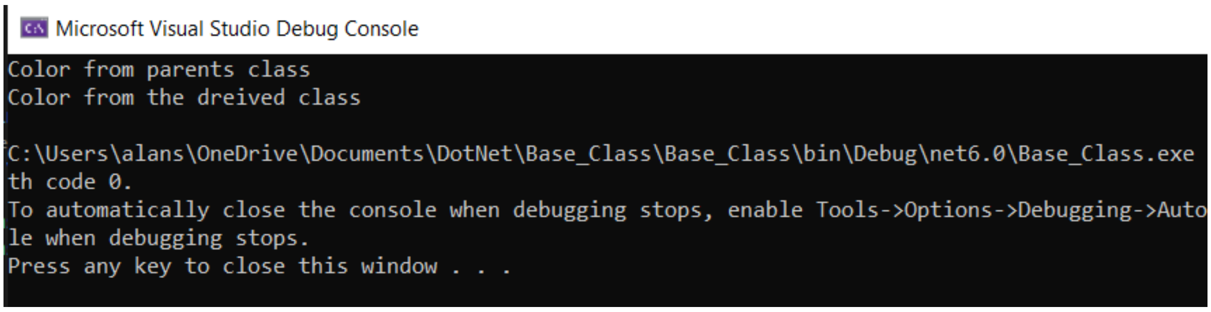
```
CS0509  'B': cannot derive from sealed type 'A'                                              Sealed   Program.cs   11   Active
CS0103  The name 'name' does not exist in the current context                                Sealed   Program.cs   16   Active
        'B' does not contain a definition for 'name' and no accessible extension method 'name' accepting a first
CS1061  argument of type 'B' could be found (are you missing a using directive or an assembly reference?)   Sealed   Program.cs   26   Active
```

base Keyword

Code:

```
using System;

public class A
{
    public string color = "Color from parents class";
}
public class B : A
{
    public string color = "Color from the dreived class";
        public void Show()
        {
            Console.WriteLine(base.color);
            Console.WriteLine(color);
        }
}
public class MainClass
{
    public static void Main()
    {
        B obj1 = new B();
        obj1.Show();
    }

}
```

Output:



Microsoft Visual Studio Debug Console

```
Color from parents class
Color from the dreived class

C:\Users\alans\OneDrive\Documents\DotNet\Base_Class\Base_Class\bin\Debug\net6.0\Base_Class.exe
th code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Auto
le when debugging stops.
Press any key to close this window . . .
```

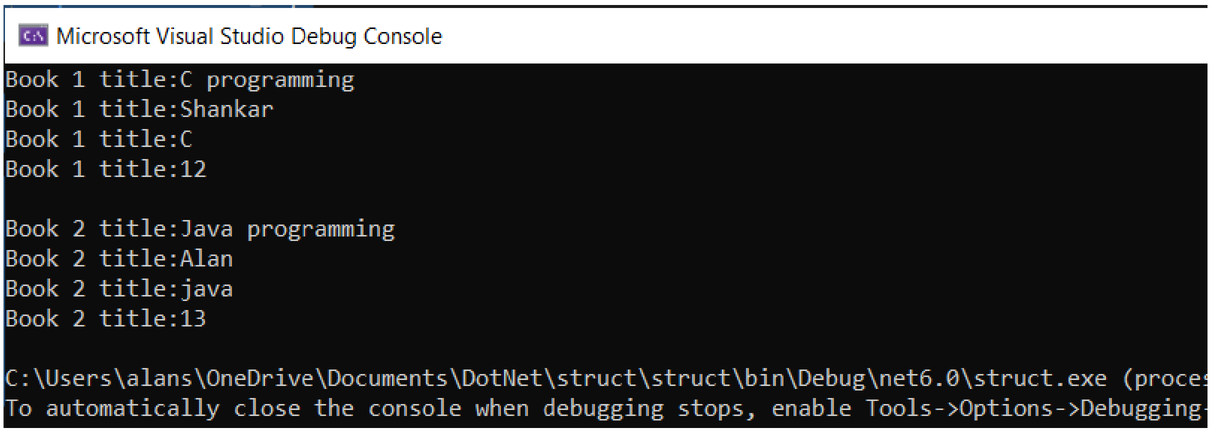
To Show Struct, Enum and Delegates

Struct

Code:

```
using System;
namespace Struct
{
    struct Books
    {
        public string title;
        public string author;
        public string subject;
        public int book_id;
    }

    public class testStructure
    {
        public static void Main(string[] args)
        {
            Books Book1;
            Books Book2;
            Book1.title = "C programming";
            Book1.author = "Shankar";
            Book1.subject = "C";
            Book1.book_id = 12;

            Book2.title = "Java programming";
            Book2.author = "Alan";
            Book2.subject = "java";
            Book2.book_id = 13;

            Console.WriteLine("Book 1 title:{0}", Book1.title);
            Console.WriteLine("Book 1 title:{0}", Book1.author);
            Console.WriteLine("Book 1 title:{0}", Book1.subject);
            Console.WriteLine("Book 1 title:{0}", Book1.book_id);

            Console.WriteLine("\nBook 2 title:{0}", Book2.title);
            Console.WriteLine("Book 2 title:{0}", Book2.author);
            Console.WriteLine("Book 2 title:{0}", Book2.subject);
            Console.WriteLine("Book 2 title:{0}", Book2.book_id);
        }
    }
}
```

Output:



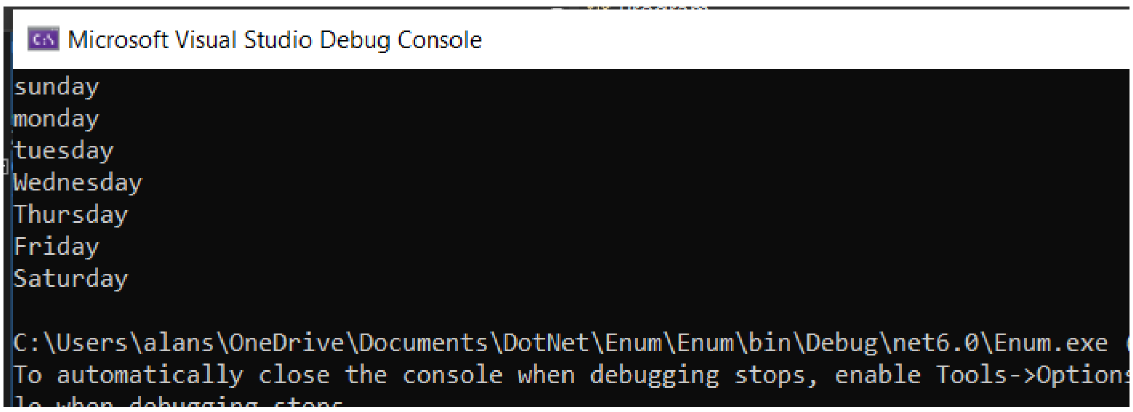
```
Book 1 title:C programming
Book 1 title:Shankar
Book 1 title:C
Book 1 title:12

Book 2 title:Java programming
Book 2 title:Alan
Book 2 title:java
Book 2 title:13

C:\Users\alans\OneDrive\Documents\DotNet\struct\struct\bin\Debug\net6.0\struct.exe (proce
To automatically close the console when debugging stops, enable Tools->Options->Debugging
```

Enum

Code:

```
using System;

// define an enum
enum Weekdays
{
    sunday,
    monday,
    tuesday,
    Wednesday,
    Thursday,
    Friday,
    Saturday
}
class Program
{
    public static void Main()
    {
        foreach (Weekdays d in Enum.GetValues(typeof(Weekdays)))
        {
            Console.WriteLine(d);
        }
    }
}
```

Output:



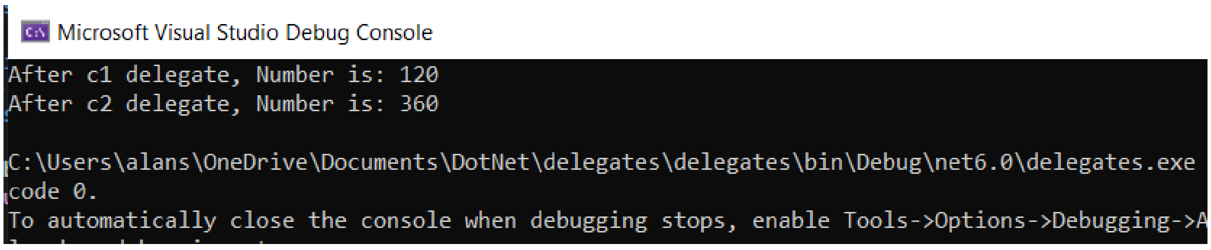
```
sunday
monday
tuesday
Wednesday
Thursday
Friday
Saturday

C:\Users\alans\OneDrive\Documents\DotNet\Enum\Enum\bin\Debug\net6.0\Enum.exe
To automatically close the console when debugging stops, enable Tools->Option
le when debugging stops.
```

Delegates

Code:

```
using System;
delegate int Calculator(int n);

public class DelegateExample
{
    static int number = 100;
    public static int add(int n)
    {
        number = number + n;
        return number;
    }
    public static int mul(int n)
    {
        number = number * n;
        return number;
    }
    public static int getNumber()
    {
        return number;
    }
    public static void Main(string[] args)
    {
        Calculator c1 = new Calculator(add);
        Calculator c2 = new Calculator(mul);
        c1(20);
        Console.WriteLine("After c1 delegate, Number is: " + getNumber());
        c2(3);
        Console.WriteLine("After c2 delegate, Number is: " + getNumber());

    }
}
```

Output:



```
After c1 delegate, Number is: 120
After c2 delegate, Number is: 360

C:\Users\alans\OneDrive\Documents\DotNet\delegates\delegates\bin\Debug\net6.0\delegates.exe
code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->A
```

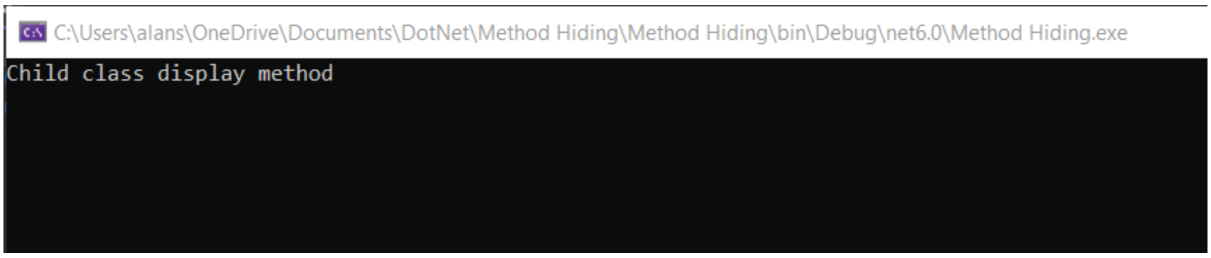
To Show Method Hiding and Method Override

Method Hiding

Code:

```
using System;
namespace MethodHiding
{
    class Class1
    {
        public void display()
        {
            Console.WriteLine("Parent class display method");
        }
    }
    class Class2 : Class1
    {
        public new void display()
        {
            Console.WriteLine("Child class display method");
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            Class2 obj = new Class2();
            obj.display();
            Console.ReadKey();
        }
    }
}
```

Output:



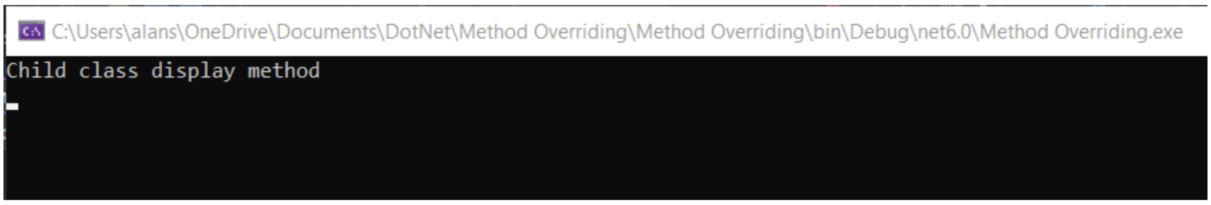
```
C:\Users\alans\OneDrive\Documents\DotNet\Method Hiding\Method Hiding\bin\Debug\net6.0\Method Hiding.exe
Child class display method
```

Method Overriding

Code:

```
using System;
namespace MethodHiding
{
    class Class1
    {
        public virtual void display()
        {
            Console.WriteLine("Parent class display method");
        }
    }
    class Class2 : Class1
    {
        public override void display()
        {
            Console.WriteLine("Child class display method");
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            Class2 obj = new Class2();
            obj.display();
            Console.ReadKey();
        }
    }
}
```

Output:



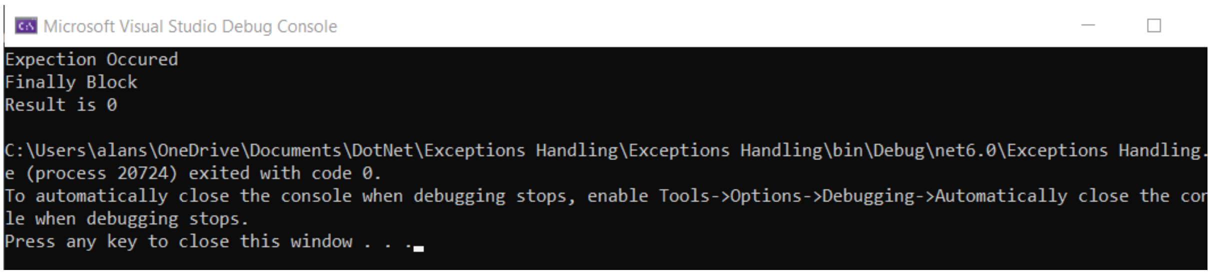
```
C:\Users\alans\OneDrive\Documents\DotNet\Method Overriding\Method Overriding\bin\Debug\net6.0\Method Overriding.exe
Child class display method
```

To Handle Exceptions in C#

Code:

```csharp
using System;

public class ExExample
{
    public static void Main()
    {
        int x = 0;
        int div = 0;

        try
        {
            div = 100 / x;
            Console.WriteLine("This is not executed");
        }
        catch (DivideByZeroException)
        {
            Console.WriteLine("Expection Occured");
        }
        finally
        {
            Console.WriteLine("Finally Block");
        }
        Console.WriteLine($"Result is {div}");
    }
}
```

Output:



```
Expection Occured
Finally Block
Result is 0

C:\Users\alans\OneDrive\Documents\DotNet\Exceptions Handling\Exceptions Handling\bin\Debug\net6.0\Exceptions Handling.
e (process 20724) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the co
le when debugging stops.
Press any key to close this window . . .
```

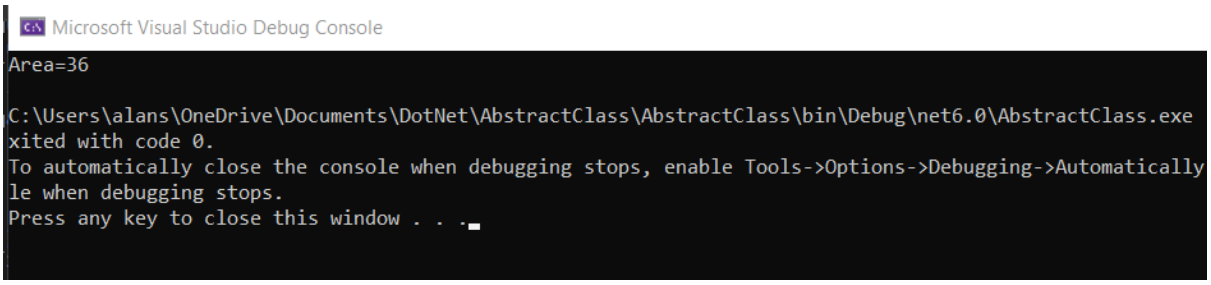
To Show Abstract Classes and Interfaces in C#

Abstract Class

Code:

```csharp
using System;
using System.Security.Principal;

abstract class AreaClass
{
    abstract public int Area();

}
class Square : AreaClass
{
    int side = 0;
    public Square(int n)
    {
        side = n;
    }
    public override int Area()
    {
        return side * side;     }
}
class Driver
{
    public static void Main()
    {
        Square s = new Square(6);
        Console.WriteLine("Area=" + s.Area());
    }
}
```

Output:



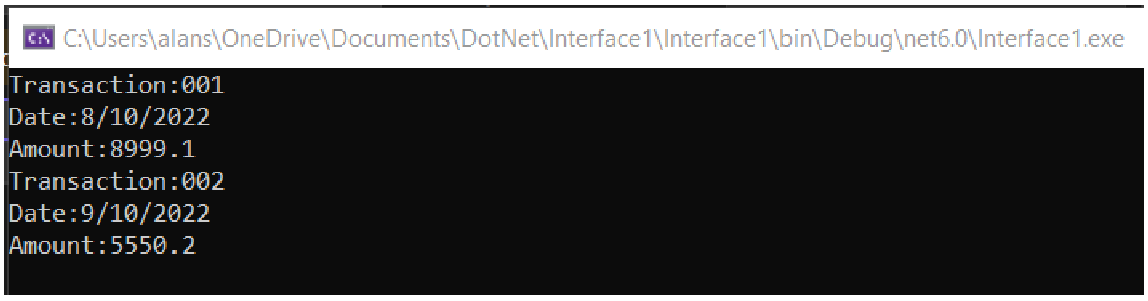
```
Area=36

C:\Users\alans\OneDrive\Documents\DotNet\AbstractClass\AbstractClass\bin\Debug\net6.0\AbstractClass.exe
xited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically
le when debugging stops.
Press any key to close this window . . .
```

Interface

Code:

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System;
namespace InterfaceApplication
{
    public interface ITransactions
    {
        void showTransaction();
        double getAmount();

    }
    public class Transaction : ITransactions
    {
        private string tCode;
        private string date;
        private double amount;
        public Transaction()
        {
            tCode = " ";
            date = " ";
            amount = 0.0;
        }
        public Transaction(string c, string d, double a)
        {
            tCode = c;
            date = d;
            amount = a;
        }
        public double getAmount()
        {
            return amount;
        }
        public void showTransaction()
        {
            Console.WriteLine("Transaction:{0}", tCode);
            Console.WriteLine("Date:{0}", date);
            Console.WriteLine("Amount:{0}", getAmount());
        }
    }
    class Tester
    {
        static void Main(string[] args)
        {
            Transaction t1 = new Transaction("001", "8/10/2022", 8999.10);
            Transaction t2 = new Transaction("002", "9/10/2022", 5550.20);
            t1.showTransaction();
            t2.showTransaction();
            Console.ReadKey();

        }
    }
}
```

Output:



C:\Users\alans\OneDrive\Documents\DotNet\Interface1\Interface1\bin\Debug\net6.0\Interface1.exe

```
Transaction:001
Date:8/10/2022
Amount:8999.1
Transaction:002
Date:9/10/2022
Amount:5550.2
```

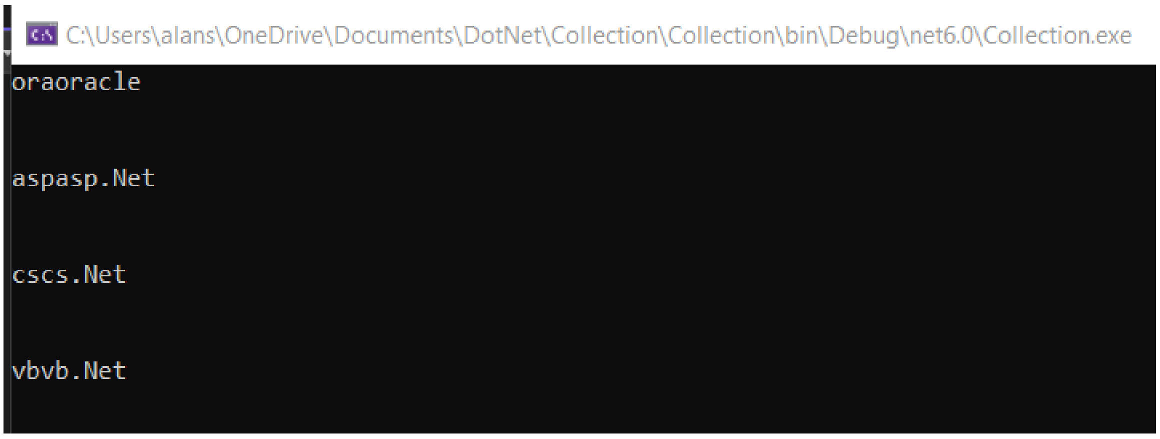
To Show Collections/ Generics and LINQ in C#

Collections

Code:

```
using System;
using System.Collections;
namespace System.Reflection.Metadata;

class Program
{
    static void Main(string[] args)
    {
        Hashtable ht = new Hashtable();
        ht.Add("ora", "oracle");
        ht.Add("vb", "vb.Net");
        ht.Add("cs", "cs.Net");
        ht.Add("asp", "asp.Net");
        foreach (DictionaryEntry d in ht)
        {
            Console.WriteLine(d.Key+""+d.Value);
            Console.WriteLine("\n");
        }
        Console.ReadKey();

    }
}
```

Output:



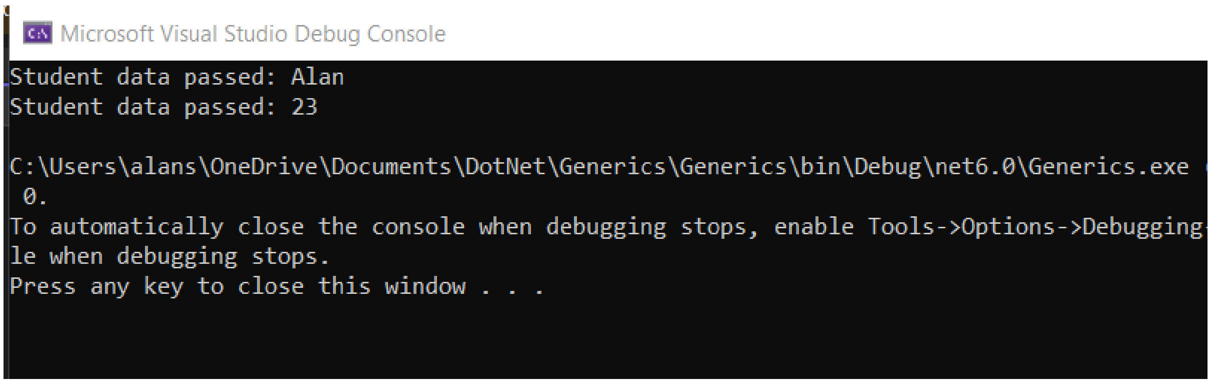
```
C:\Users\alans\OneDrive\Documents\DotNet\Collection\Collection\bin\Debug\net6.0\Collection.exe
oraoracle


aspasp.Net


cscs.Net


vbvb.Net
```

Generics

Code:

```
using System;

public class Student<T>
{
    public T data;
    public Student(T data)
    {
        this.data = data;
        Console.WriteLine("Student data passed: " + this.data);
    }
}

class Program
{
    static void Main()
    {
        Student<string> studentName = new Student<string>("Alan");

        Student<int> studentId = new Student<int>(23);
    }
}
```

Output:



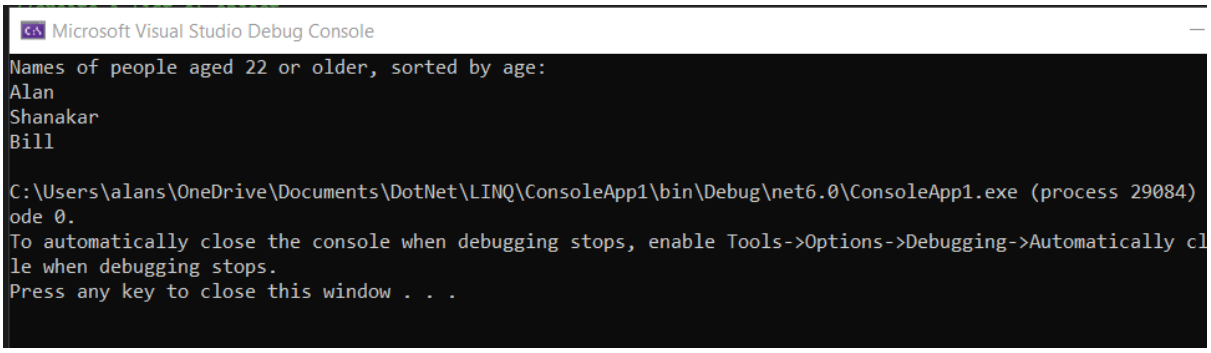
```
Student data passed: Alan
Student data passed: 23

C:\Users\alans\OneDrive\Documents\DotNet\Generics\Generics\bin\Debug\net6.0\Generics.exe
 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging
le when debugging stops.
Press any key to close this window . . .
```

LINQ

Code:

```
using System;
using System.Collections.Generic;
using System.Linq; //import the linq namespace
using System.Net.Cache;

class Program
{
    static void Main()
    {
        //create a list of object
        List<Person> people = new List<Person>
        {
            new Person{Name = "Alan",Age= 22},
            new Person{Name = "Shanakar",Age= 24},
            new Person{Name = "Bill",Age= 25},
            new Person{Name = "Ram",Age= 20},
            new Person{Name = "Rohan",Age= 19}
        };
        //query the list using LINQ
        var result = from p in people
                     where p.Age >= 22
                     orderby p.Age
                     select p.Name;
        //display the query result
        Console.WriteLine("Names of people aged 22 or older, sorted by age:");
        foreach (var name in result)
        {
            Console.WriteLine(name);
        }
    }
}
//define a custom class for the object in the list
class Person
{
    public string Name { get; set; }
    public int Age { get; set; }
}
```

Output:



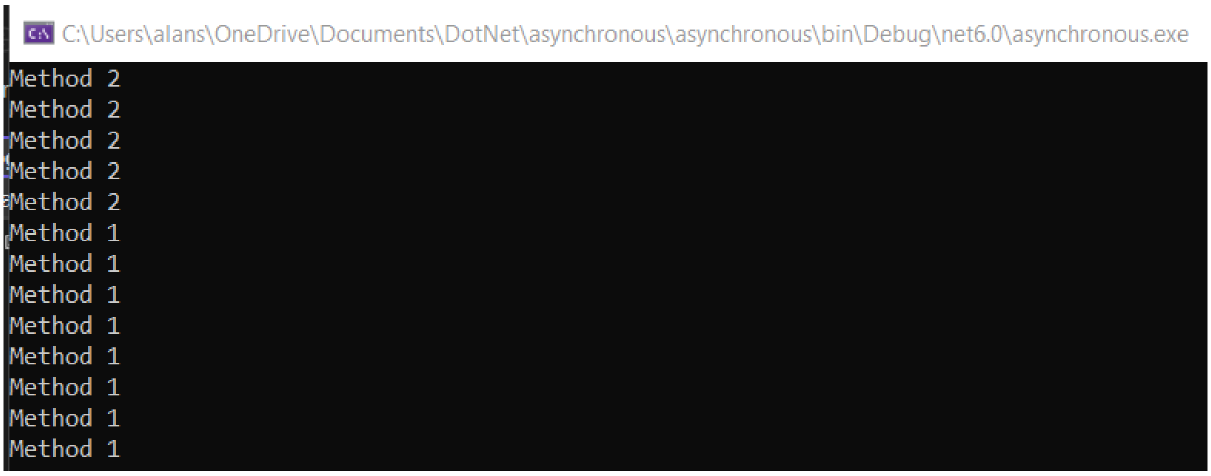
```
Names of people aged 22 or older, sorted by age:
Alan
Shanakar
Bill

C:\Users\alans\OneDrive\Documents\DotNet\LINQ\ConsoleApp1\bin\Debug\net6.0\ConsoleApp1.exe (process 29084)
ode 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically cl
le when debugging stops.
Press any key to close this window . . .
```

To Show Async and Awaits in C#.

Code:

```
using System;
using System.Threading.Tasks;
namespace MyconsoleApp
{
    class AsyncAwaitTest
    {
        static void Main(string[] args)
        {
            Method1();
            Method2();
            Console.ReadKey();
        }
        public static async Task Method1()
        {
            await Task.Run(() =>
            {
                for (int i = 0; i < 100; i++)
                {
                    Console.WriteLine("Method 1");
                }
            });
        }
        public static void Method2()
        {
            for (int i = 0; i < 50; i++)
            {
                Console.WriteLine("Method 2");
            }
        }
    }
}
```

Output:



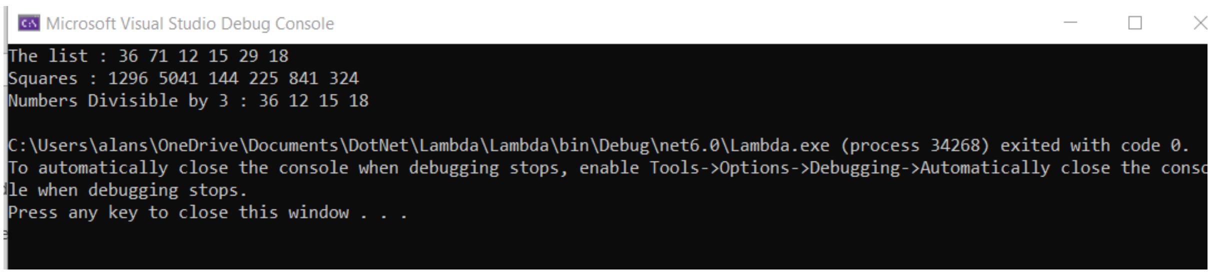
```
C:\Users\alans\OneDrive\Documents\DotNet\asynchronous\asynchronous\bin\Debug\net6.0\asynchronous.exe
Method 2
Method 2
Method 2
Method 2
Method 2
Method 1
Method 1
Method 1
Method 1
Method 1
Method 1
Method 1
Method 1
```

To Show Lambda Expression in C#

Code:

```
using System;
using System.Collections.Generic;
using System.Linq;

namespace Lambda_Expressions
{
    class Program
    {
        static void Main(string[] args)
        {
            List<int> numbers = new List<int>() {36, 71, 12,
                             15, 29, 18};

            Console.Write("The list : ");
            foreach (var value in numbers)
            {
                Console.Write("{0} ", value);
            }
            Console.WriteLine();

            var square = numbers.Select(x => x * x);

            Console.Write("Squares : ");

            foreach (var value in square)
            {
                Console.Write("{0} ", value);
            }
            Console.WriteLine();

            List<int> divBy3 = numbers.FindAll(x => (x % 3) == 0);

            Console.Write("Numbers Divisible by 3 : ");

            foreach (var value in divBy3)
            {
                Console.Write("{0} ", value);
            }
            Console.WriteLine();
        }
    }
}
```

Output:



```
The list : 36 71 12 15 29 18
Squares : 1296 5041 144 225 841 324
Numbers Divisible by 3 : 36 12 15 18

C:\Users\alans\OneDrive\Documents\DotNet\Lambda\Lambda\bin\Debug\net6.0\Lambda.exe (process 34268) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .
```

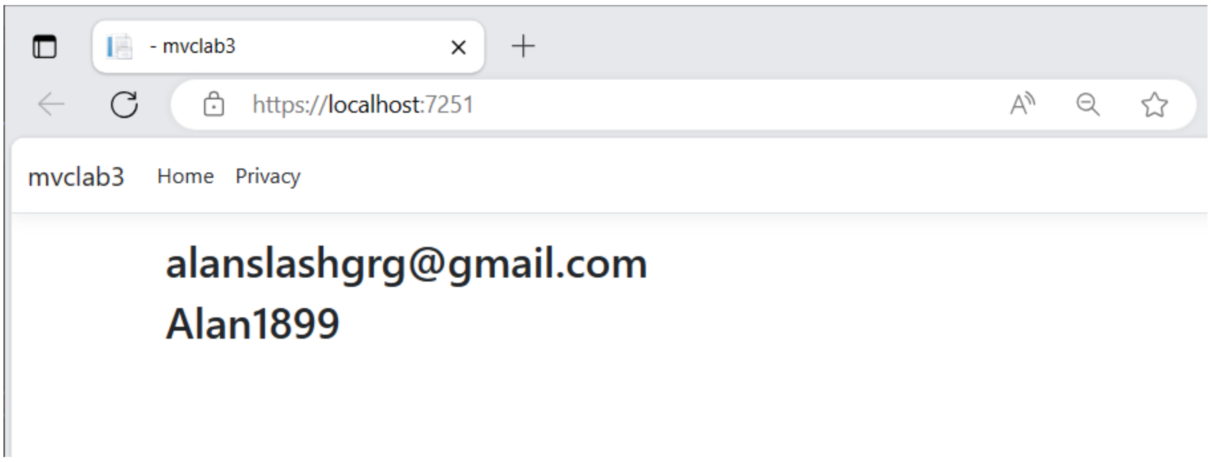
To Showcase ASP.NET MVC Architecture

Code:

```
//HomeController.cs
using Microsoft.AspNetCore.Mvc;
using mvclab3.Models;
using System.Diagnostics;

namespace mvclab3.Controllers
{
    public class HomeController : Controller
    {
        private readonly ILogger<HomeController> _logger;

        public HomeController(ILogger<HomeController> logger)
        {
            _logger = logger;
        }

        public IActionResult Index()
        {
            {
                MyForm myForm = new MyForm
                {

                    Email = " alanslashgrg@gmail.com",
                    Password = "Alan1899"
                };
                ViewBag.MyForm = myForm;

                return View(myForm);
            }
        }

        public IActionResult Privacy()
        {
            return View();
        }

        [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None,
NoStore = true)]
        public IActionResult Error()
        {
            return View(new ErrorViewModel { RequestId = Activity.Current?.Id
?? HttpContext.TraceIdentifier });
        }
    }
}

//Models-MyForm.cs
namespace mvclab3.Models
{
    public class MyForm
    {
        public string Email { get; set; }
        public string Password { get; set; }

    }
}
```

```
//Views-Home-Index.cshtml
@model MyForm
<h2>@ViewBag.MyForm.Email</h2>
<h2>@ViewBag.MyForm.Password</h2>
```

Output:



alanslashgrg@gmail.com

Alan1899