**Fizz.cs:**

using LinkGroup.FizzBuzzHandler;

namespace LinkGroup.Fizz

{

public class DisplayFizz : IFizzBuzz

{

private readonly IFizzBuzz \_fizzBuzzHandler;

public DisplayFizz(IFizzBuzz fizzBuzzHandlerService)

{

\_fizzBuzzHandler = fizzBuzzHandlerService;

}

/// <summary>

/// This Method will display Fizz if Input provided is Divisors by 3.

/// </summary>

/// <param name="num"></param>

/// <returns></returns>

public string methodCommon(int num)

{

if (num % 3 == 0)

return "Fizz";

return \_fizzBuzzHandler.methodCommon(num);

}

}

}

**Buzz.cs**

{

public class DisplayBuzz : IFizzBuzz

{

/// <summary>

/// This Method will display Buzz if Input provided is Divisors by 5.

/// </summary>

/// <param name="num"></param>

/// <returns></returns>

public string methodCommon(int num)

{

if (num % 5 == 0)

return "Buzz";

return num.ToString();

}

}

}

**FizzBuzz.cs**

using LinkGroup.FizzBuzzHandler;

namespace LinkGroup.FizzBuzz

{

public class DisaplyFizzBuzz : IFizzBuzz

{

private readonly IFizzBuzz \_fizzBuzzHandler;

public DisaplyFizzBuzz(IFizzBuzz fizzBuzzHandlerService)

{

\_fizzBuzzHandler = fizzBuzzHandlerService;

}

/// <summary>

/// This Method will display Fizz Buzz if Input provided is Divisors by 15.

/// </summary>

/// <param name="num"></param>

/// <returns></returns>

public string methodCommon(int num)

{

if (num % 15 == 0) // Here, Divisors of 3 and 5 will also be Divisors by 15. To avoid Multiple execution.

return "Fizz Buzz";

return \_fizzBuzzHandler.methodCommon(num);

}

}

}

**IValidation.cs**

namespace LinkGroup.InputValidationInterface

{

public interface IValidation

{

string validateInput(string input);

}

}

**IFizzBuzzHandler.cs**

namespace LinkGroup.FizzBuzzHandler

{

public interface IFizzBuzz

{

string methodCommon(int num);

}

}

**Validation.cs**

using System;

using LinkGroup.InputValidationInterface;

namespace LinkGroup.InputValidation

{

public class Validation : IValidation

{

public readonly IValidation \_validateInput;

public Validation() { }

#region Validation Logic

/// <summary>

/// This Method will validate the Input.

/// </summary>

/// <param name="input"></param>

/// <returns></returns>

public string validateInput(string input)

{

try

{

if (input != null && input.TrimEnd() == "")

{

throw new Exception("Input is empty. Please enter some Input.");

}

else if (Convert.ToInt32(input) <= 0)

{

throw new Exception("Input is less than or equal to zero. Please write Valid Input.");

}

}

catch (Exception)

{

Console.WriteLine("Validation Failed.");

throw;

}

return input;

}

#endregion

}

}

**Program.cs**

using System;

using LinkGroup.Fizz;

using LinkGroup.Buzz;

using LinkGroup.FizzBuzzHandler;

using LinkGroup.InputValidationInterface;

namespace LinkGroup

{

class Program

{

static void Main(String[] args)

{

IFizzBuzz \_fizzBuzzHandler = new FizzBuzz.DisaplyFizzBuzz(new DisplayFizz(new DisplayBuzz()));

IValidation \_validateInput = new InputValidation.Validation();

try

{

Console.Write("Enter number : ");

string input = Console.ReadLine();

int num = Convert.ToInt32(\_validateInput.validateInput(input));

for (int i = 1; i <= num; i++)

{

string res = \_fizzBuzzHandler.methodCommon(i);

Console.WriteLine(res);

}

}

catch (Exception e)

{

Console.WriteLine("Error Occured because " + e.Message);

}

}

}

}

**SQL Queries:**

-----------------------------------------------------------------------------------------------

--1. Show all investors

SELECT

\*

FROM

dbo.investors WITH(NOLOCK)

-----------------------------------------------------------------------------------------------

--2. Show all active investors ordered by surname

SELECT

\*

FROM

dbo.investors WITH(NOLOCK)

WHERE

active = 'Y'

ORDER BY

surname

-----------------------------------------------------------------------------------------------

--3. Show all the orders for investor 12 in January 2012

SELECT

\*

FROM

orders WITH(NOLOCK)

WHERE

investor\_id = 12

AND order\_date > '2011-12-31'

AND order\_date < '2012-02-01'

-----------------------------------------------------------------------------------------------

--4. Show the number of orders and quantities for investor 12 during January 2012 grouped by fund name

SELECT

o.investor\_id,

f.fund\_name,

COUNT(order\_id) as order\_count,

SUM(o.quantity) as order\_quantity

FROM

dbo.orders o WITH(NOLOCK)

inner join dbo.funds f WITH(NOLOCK) ON o.fund\_id = f.fund\_id

and o.investor\_id = 12

AND o.order\_date > '2011-12-31'

AND o.order\_date < '2012-02-01'

GROUP BY

o.investor\_id,

f.fund\_name

-----------------------------------------------------------------------------------------------

---5. Show all investors who represent a company

--This is needed for a mail merge and must have the following two columns:

--• MAIL\_NAME (combined forename and surname)

--• BUSINESS\_NAME

SELECT

CONCAT(forename, surname) as mail\_name,

business\_name

FROM

dbo.investors WITH(NOLOCK)

WHERE

business\_name is not null

ORDER BY

mail\_name

-----------------------------------------------------------------------------------------------

--6. Now add an extra mail merge column.

--• ORDER\_COUNT (the total number of orders for the investor)

SELECT

ConCat(forename,surname) as mail\_name,

business\_name,

COUNT(order\_id) as order\_count

FROM investors I WITH(NOLOCK)

LEFT JOIN orders O WITH(NOLOCK)

ON I.investor\_id=O.investor\_id

WHERE

business\_name IS NOT NULL

GROUP BY

I.forename,

I.surname,

business\_name

ORDER BY ConCat(forename,surname)

-----------------------------------------------------------------------------------------------

--7. Create a new fund with an id of 200

INSERT INTO

dbo.funds (fund\_id, fund\_name)

VALUES

(200, 'TEST FUND')

SELECT

\*

FROM

dbo.funds

-----------------------------------------------------------------------------------------------

--8. Show quantities bought and sold across all orders

SELECT

buy\_Sell,

sum(quantity) as quantity

FROM

dbo.orders WITH(NOLOCK)

GROUP BY

buy\_sell

-----------------------------------------------------------------------------------------------

--9. Now include the total value (£'s) of the orders

SELECT

buy\_Sell,

sum(quantity) as quantity,

sum(quantity \* price) as total\_value

FROM

dbo.orders WITH(NOLOCK)

GROUP BY

buy\_sell

-----------------------------------------------------------------------------------------------

---10. Which investors had no orders

SELECT

\*

FROM

dbo.investors WITH(NOLOCK)

WHERE

investor\_id not in (

SELECT

investor\_id

FROM

dbo.orders

)

-----------------------------------------------------------------------------------------------

---11. Which three orders had the greatest value (£'s)

SELECT

top 3 order\_id,

(quantity \* price) as total\_value

FROM

dbo.orders WITH(NOLOCK)

ORDER BY

total\_value DESC

-----------------------------------------------------------------------------------------------

--12. All the investors started with £1000. What are their current values after processing all the orders

--Buy orders will increase an investor's total, sell orders will reduce it.

SELECT

i.investor\_id,

i.forename,

i.surname,

sum(o.quantity \* o.price) as total\_value

FROM

dbo.investors i WITH(NOLOCK)

inner join dbo.orders o WITH(NOLOCK) on o.investor\_id = i.investor\_id

GROUP BY

i.investor\_id,

i.forename,

i.surname

ORDER BY

i.investor\_id

-----------------------------------------------------------------------------------------------