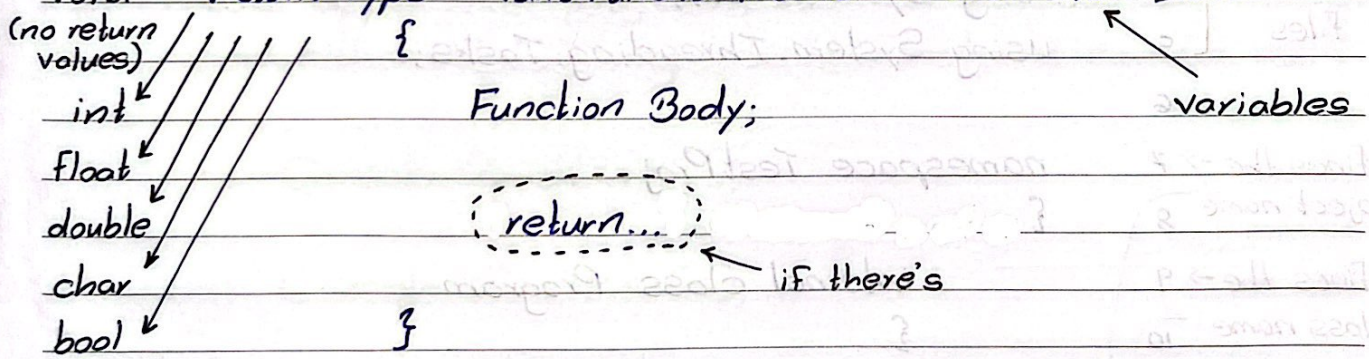
Functions in Programming.

*Apart From inbuilt Functions in programming, users are capable of defining functions based on the requirement. These Functions are known as user-defined functions.

*There're 4 user-defined function types available in programming.

01) No return type - No parameters
02) No return type - With parameters
03) With return type - No parameters
04) With return type - With parameters

*A user-defined Function will have the following structure. (Syntax of a user-defined function)



*Based on the return type that you have used, the return value should properly used.

Ex: If the return type is integer, you should return an integer value.

NSBM GREEN UNIVERSITY TOWN

C# Programming

* C# language is developed by Microsoft Coperations which capable of doing desktop application development.

* For C# coding, Microsoft Visual Studio IDE is used.

* C# programming is allowing you to create applications with Frontend and console based applications.

* Console based applications are running on console window and its source code structure can define as followed.

* C# is case sensitive.

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   namespace TestProj
8   {
9       internal class Program
10      {
11          static void Main(string[] args)
12          {
13
14          }
15      }
16  }
17
```

in built library files

defines the project name → 7

defines the class name → 9

main method of C# application → 11

* this name will change based on the project you create.

* class is a file which contains in the project.

* In the above source code, class name is program and that is the main class of the project.

* A project can contain only one main method and one program class.

Creating "Hello World" Program.

In C#, there're 2 inbuilt functions which can use to display content on the window.

01.) `Console.WriteLine("Hello World");`

02.) `Console.Write("Hello World");`

* In console application projects, to hold the console window on screen either of following inbuilt functions can be used.

01.) `Console.ReadLine();`

02.) `Console.ReadKey();`

* Console.WriteLine will display the content and move the curser to the next immediate line.

* Console.Write will display the content and the curser will remains on last printed point.

Data Types.

* In programming data types are being used to handle data at application execution level. All software applications are working with some set of data by using programming languages we should be able to access all of these data.

* Data types are being used to handle data in programming and C# is supporting Following data types in data handling.

01.) **int** - used to handle integer numerical values.

02.) **double** - used to handle numerical values with decimal points.

03.) **char** - used to handle single character data.

04.) **bool** - used to handle boolean data.

05.) **string** - used to handle text data with multiple characters.

Variable Declaration.

* In programming, depending on the requirement you can declare variables by using data types.

* When declaring a variable a specific data type should mention on a valid variable name should be given.

Syntax

DataType ValidVariableName = Initial Value;

Ex:

```
int sum = 70;
double batch = 18.1;
```

* When assigning values to string variables, the value should be given within two double quotes.

```
string name = "Alex";
```

* For character variable, the value should be given within a single quote.

```
char grade = 'A';
```

Question

01.) Declare 3 variables to assign your name, age and batch as values.

```
string name = "Afsaan";
int age     = 22;
double batch = 22.1;
```

① Displaying Variables on Console Application.

* To display variables Console.WriteLine inbuilt function will be used.

Ex:

```
int age = 70;
```

① `Console.WriteLine(age);`

or

② `Console.WriteLine("My age is " + age);`

* In ② example, a text value is merged with a variable value. In such situations, we call it as concatanation.

Question

01.) Using any loop, display first 10 natural numbers using C# console app.

```
{
    int count;
    For (count = 1; count <= 10; count++)
    {
        Console.WriteLine(count);
    }
    Console.ReadKey();
}
```

02.) Display Following pattern in C# console app using a For loop.

20
40
60
80
100

```
{
    int count;
    For (count = 20; count <= 100; count = count + 20)
    {                                  count += 20
        Console.WriteLine(count);
    }
    Console.ReadKey();
}
```

Global Variable Declaration

* In programming variables can be declared Globally, which means those variables can be used by the entire code.

→ Next Page

```
static void Main (string [] args)
{
    int count;  //Global Variable For main method.

    For (count = 20; count <= 100; count = count + 20)
    {
        Console.WriteLine (count);
    }
    count = 20;
    Console.ReadKey ();
}
```

Local Variable Declaration

* When declaring variables it can be declared For a specific score of project and those variables are known as local variables

```
static void Main (string [] args)
{
    For (int count = 20; count <= 100; count = count + 20)
                                        //local variable
    {
        Console.WriteLine (count);
    }
count = 20;
Console.ReadKey ();
}
```

* In above code, count variable is only known by For loop itself. Because of that count variable cannot be reused outside the for loop.

Getting String User Inputs.

* When taking the user inputs an inbuilt Function called Console.ReadLine(); is being used.

* The Following inbuilt function will scan the console window and get whatever the user input to the console application.

* The Following taken value need to be assigned to a variable where it can be used for further processing.

* In C# console applications, the default data type is string. Because of that all user inputs will consider as string user inputs initially.

→ Next Page

```
{
    class MainClass
    {
        static void Main (string [] args)
        {
        string name; //Declaring a string variable to store the value.
        Console.WriteLine ("Enter your name : ");
        name = Console.ReadLine (); //taking user input to the variable.
        Console.WriteLine ("Welcome" + name); //Displaying the variable.
        Console.ReadLine (); //avoid closing the console.
        }
    }
}
```

Getting Integer User Inputs

* When expecting an integer value, you should do type casting because the default data type of console application is string.

* When expecting integer inputs int.Parse() inbuilt function can be used.

* Using above method, the converted value now can be assigned to an integer variable.

* For integer inputs, you should always use type casting.

```
{
    class MainClass
    {
    static void Main (string [] args)
     {
       int age; //Declaring an integer variable to store the value.
       Console.WriteLine("Enter your age: ");
       age = int.Parse(Console.ReadLine()); //Taking user input
                                            //to the variable.
       Console.WriteLine("Your age is" + age); //Displaying the
                                               //variable.
       Console.ReadLine(); //avoid closing the console.
     }
    }
}
```

Method 02

* When doing the type casting apart From int.Parse, you can use Convert.ToInt32, 16, 64 inbuilt functions.

* This inbuilt functions will also convert string to integer and this will specifically decide how many bits are there in the converted integer.

```
age = Convert.ToInt32(Console.ReadLine());
```

Question

01.) Create a C# console application to check whether the user inserted integer value is divisible from the user inserted second integer value.

```
{
    static void Main (string [] args)
    {
        int num1, num2;

        Console.WriteLine ("Enter Number 01 : ");
        num1 = int.Parse (Console.ReadLine());

        Console.WriteLine ("Enter Number 02: ");
        num2 = int.Parse (Console.ReadLine());

        if (num1 % num2 == 0)
            Console.WriteLine ("Numbers are divisible");
        else
            Console.WriteLine ("Numbers are not divisible");

        Console.ReadKey();
    }
}
```

02.) Create a C# console application to check whether the user is eligible for voting if he or she is above 18 years. (With the user's name)

```
{
    static void Main(string[] args)
    {
        int age;
        string name;

        Console.WriteLine("Enter your name: ");
        name = Console.ReadLine();

        Console.WriteLine("Enter your age: ");
        age = int.Parse(Console.ReadLine());

        if(age>=18)
        {
            Console.WriteLine(name + " You're eligible");
        }
        else
        {
            Console.WriteLine(name + "You're not eligible");
        }

        Console.ReadKey();

    }
}
```

(01.) Create a C# console application project and display answers for 4 basic arithmetic operations based on 2 user input values.

```
int num1, num2;

Console.WriteLine("Enter First number: ");
num1 = int.Parse(Console.ReadLine());

Console.WriteLine("Enter second number: ");
num2 = int.Parse(Console.ReadLine());

Console.WriteLine("Summation is " + (num1 + num2));
Console.WriteLine("Substraction is " + (num1 - num2));
Console.WriteLine("Multiplication is " + (num1 * num2));
Console.WriteLine("Division is " + (num1 / num2));

Console.ReadKey();
```

(02.) Create a C# application to calculate area & circumferance of a circle, when user insert the radius value.

```
double rad;

Console.WriteLine("Enter the Radius Value: ");
rad = double.Parse(Console.ReadLine());

Console.WriteLine("Area of the circle is: " + (3.14 * rad * rad));
Console.WriteLine("Circumferance is: " + (2 * 3.14 * rad));
Console.ReadKey();
```

(03.) Create a C# application to get 10 integer inputs From the user & find out how many even & odd inputs are there.

```
int num;
int odd = 0, even = 0;

for (int i=1; i<=10; i++)
{
    Console.WriteLine("Enter Number : " + (i));
    num = int.Parse(Console.ReadLine());

    if (num%2 == 0)
        even++;
    else
        odd++;
}

Console.WriteLine("There are " + even + "numbers");
Console.WriteLine("There are" + odd + "numbers");

Console.ReadKey();
```

Arrays.

*Array is a basic data structure which used to store similar type of data (similar data type) inside computer application.

*The value inside the array known as an element and each element will be handled by unique address.

* In programming, these addresses are known as indexes and array indexes are always starting from 0.

* When declaring arrays in C#, it can be declared as an empty array or defined initialized array.

Syntax

Defined Initialized Array (Example 01)

```
static void Main (string [] args)
{
    int [] numbers = {1, 2, 3, 4};

    Console.ReadKey();
}
```

Empty Array (Example 02)

```
static void Main (string [] args)
{
    int [] numbers = new int [10];

    Console.ReadKey();
}
```

* When declaring arrays, we could use any data type and valid array name.

* In, Example 01, array size will automatically adjust based provided elements.

* In, Example 02, array size is defined as 10 which cannot be changed.

Question

01.) Create a C# console application project to assign Following integer values to an array and find the max & min values. 25, 41, 10, 78, 68, 36, 89, 37.

```
int[] arr = {25, 41, 10, 78, 68, 36, 89, 37};

int max = 0;

for (int i=0; i<arr.Length; i++)
{
    if (max < arr[i])
        max = arr[i];
}
Console.WriteLine(max);

int min = max;

For (int i=0; i<arr.Length; i++)
```

```
{
    if (min > arr[i])
        min = arr[i];
}
Console.WriteLine(min);

Console.ReadKey();
```

02.) Create a C# console application project to declare an integer array, size of 10 and get user inputs. Find the max & min values.

```
int[] arr = new int[10];

for(int i=0; i< arr.Length; i++)
{
    Console.WriteLine("Enter element " + (i+1));
    arr[i] = int.Parse(Console.ReadLine());
}
for(int i=0; i< arr.Length; i++)
{
    Console.WriteLine(arr[i]);
}

int max=0;

for(int i=0; i< arr.Length; i++)
{
    if(max < arr[i])
```

```
            max = arr[i];
        }
        Console.WriteLine(max);

        int min = max;

        for (int i=0; i< arr.Length; i++)
        {
            if (min > arr[i])
                min = arr[i];
        }
        Console.WriteLine(min);

        Console.ReadKey();
```

03.) ① Display the user inserted values to the array in ascending order.

```
    Console.WriteLine("Enter the numbers:");
    int count = int.Parse(Console.ReadLine());

    int[] numbers = new int[count];

    for (int i=0; i<count; i++)
    {
        Console.WriteLine("Enter a number: ");
        numbers[i] = int.Parse(Console.ReadLine());
    }
```

```
Array.Sort(numbers);

For (int i=0; i<count; i++)
{
    Console.WriteLine(numbers[i]);
}

Console.ReadLine();
```

Multi-dimensional Arrays

* C# programming language allows to create multi-dimensional arrays, which capable of storing values in more dimensions.

* The common form of multi-dimensional array is 2D array, which contains row and column values.

* When creating a 2D array in C#, row count should mention first and then the column count need to be mentioned.

* When accessing 2D arrays, a nested for loop need to be used to go through the array element.

Syntax

* When declaring a 2D array, a 'comma' (,) mark should mention inside the square bracket which defines/seperate row count and the column count of the declared array.

```
DataType[,] ValidArrayName = new DataType[Row count, Column count];
```

Example

```
static void Main(string[] args)
{
    int[,] arr = new int[3,4];
    Console.ReadLine();
}
```

NSBM GREEN UNIVERSITY TOWN

Question

01.) Create a C# console application project to declare a 2D array, which contains 2 rows and 4 columns. Get user inputs to the array from user and display all the values which contains in the array. Find the max & min values. Display the user inserted values in ascending order.

NSBM GREEN UNIVERSITY TOWN

C# Classes

* In OOP Programming Languages, project / application code can break down into different class files.

* Classes can be divided based on the Functionality of your code and the operation of your code.

* In a C# application, it could contain n number of classes depends on project score.

* For every project there should be a main class, which known as program.cs. This class file contains the main method of your application.

* Breaking down the project into different classes will allows you to reuse the code in the same project when required.

* As an example, when creating a calculator, the project code can be divided based on the operations such as summation, substraction, division, multiplication.

* class file always ends with .cs extension.

Ex: program.cs / summation.cs

* When declaring a class file, you can assign any valid class name.

Ex: employee.cs / student.cs

Structure of a class.

* When a class has been declared, it could contain n number of variables and n number of functions.

* These variables and functions are known as member variables & member functions.

* class will always start with a keyword called class which followed by a given class name. It contains 2 curly braces which will identify the boundry of the class.

* Inside the class boundry you can declare any number of user-defined functions & variables.

Access Specifiers / Access Modifiers

* In OOP Programming. this will decide the accessebility level of member variables & member functions in a specific class file.

* There're 3 main access specifiers contains in OOP Programming known as,
  01. public
  02. private
  03. protected

public.

*if a variable or method, declared under public access specifier, that variable or method can be used from anywhere in the project.

private.

*if a variable or method, declared under private access specifier, that variable or method can be used from only declared class.

protected.

*if a variable or method, declared under protected access specifier, that variable or method can be used from only declared class and its child class only.

*This concept is used in inheritence OOP concept.

Example.

```
class Student
{
    private int Sum;

    public void MyFunction();
    {
    }
}
```

* In above example, Student is the class file and inside the Student class, there's a private variable and public method.

* The declared private variable can be used only within the student class and the public function can be used in the entire project.

* When it come to classes, all operations & functionalities needed to be done inside a method or Function.

* After declaring methods, to execute them, it needed to be called from the main method, since compiler will only look into main method at execution level.

* By creating class objects, you can call these functions in the main method.

Creating a class object

* In OOP Programming, if your project contains a seperate class with some user-defined methods, to execute those methods, those needed to be called from the main method.

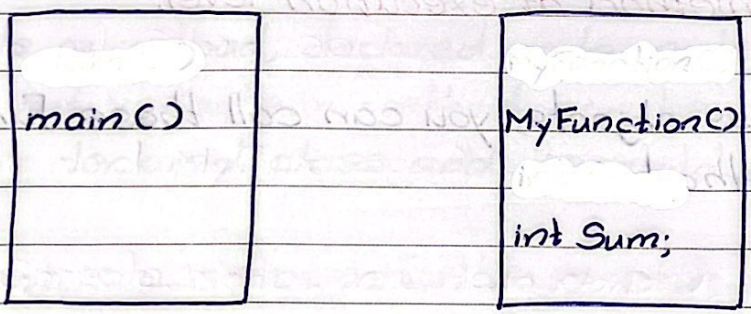
* Since the Function is located in a seperate class File, that Function can be only called through a class object.

* Using the created class object, all public methods and variables can be accessed by the other class.

Syntax of a Class Object.

* Assume that a project contains an added class called Student.cs and that class contains public void method called MyFunction.

* In the same class, assume that there's a public integer variable called Sum.




```
AddedClassName ValidObjectName = new AddedClassName();
```

Ex:-

```
① Student ObjStudent = new Student ();
② ObjStudent.MyFunction();
③ ObjStudent.Sum = 10;
```

* When executing the program, the compiler will look into the instructions which given inside the main method.

* When creating the class object in above scenario, that needed to be done inside the main method.

* When giving the object name, you can use any valid name without using any reserved keywords.

* In line ①, we're creating a class object by using the Student class and the object name is objStudent.

* In line ②, using the created object, the public method called MyFunction is executing.

* In line ③, using the object, the Sum variable is accessed and assigned the value 10.

Ex01:-

```
class Student
{
    public void MyFunction()
    {
        Console.WriteLine("Hello World");
    }
}
```

Student.cs

→ Next Page

Program.cs

```
class Program
{
    static void Main(string[] args)
    {
        Student objStudent = new Student();
        objStudent.MyFunction();
        Console.ReadKey();
    }
}
```

Ex 02:-

Student.cs

```
class Student
{
    public int Sum = 100;

    public void MyFunction()
    {
        Console.WriteLine("Hello World");
    }
}
```

Program.cs

```
class Program
{
    static void Main(string[] args)
    {
        Student objStudent = new Student();
        objStudent.MyFunction();
        Console.WriteLine(objStudent.Sum);
        Console.ReadKey();
    }
}
```

NSBM GREEN UNIVERSITY TOWN

Question

01.) Create a C# console application which contains an added class called employee and inside the class create a no return type no parameter method called EmpInfo. Inside the method, get the user's name and age as an input value and display them inside the method itself. Create class objects accordinly and call the above method.

```
public void EmpInfo()
{
    string Name;
    int Age;

    Console.WriteLine("Enter Your Name: ");
    Name = Console.ReadLine();

    Console.WriteLine("Enter Your Age: ");
    Age = int.Parse(Console.ReadLine());

    Console.WriteLine("Your name is" + Name + "and
    you are" + Age + "years old");
}
```

```
{
    Employee objEmp = new Employee();
    objEmp.EmpInfo();
    Console.ReadKey();
}
```

02.) Create a C# console application project which contains an added class called calculations and inside the class, create a method called converter, which is a no return type no parameter method. Inside the method get KM value from the user and convert it into CM and display the output.

```
public void Converter()
{
    double km;

    Console.WriteLine("Enter KM Value ");
    km = double.Parse(Console.ReadLine());

    Console.WriteLine("CM value is " + (km * 100000));
}
```

```
{
    Calculations objCal = new Calculations();
    objCal.Converter();
    Console.ReadKey();
}
```

03.) Create a C# console application project with an added class called calculator. Inside the calculator class, create a method called Sum, which will take 2 integer user inputs and then find the summation and return the answer out of the function. Create necessary class objects and call the above method.

```
public int Sum()
{
    int num1, num2;

    Console.WriteLine("Enter first number: ");
    num1 = int.Parse(Console.ReadLine());

    Console.WriteLine("Enter second number: ");
    num2 = int.Parse(Console.ReadLine());

    return (num1 + num2);
}
```

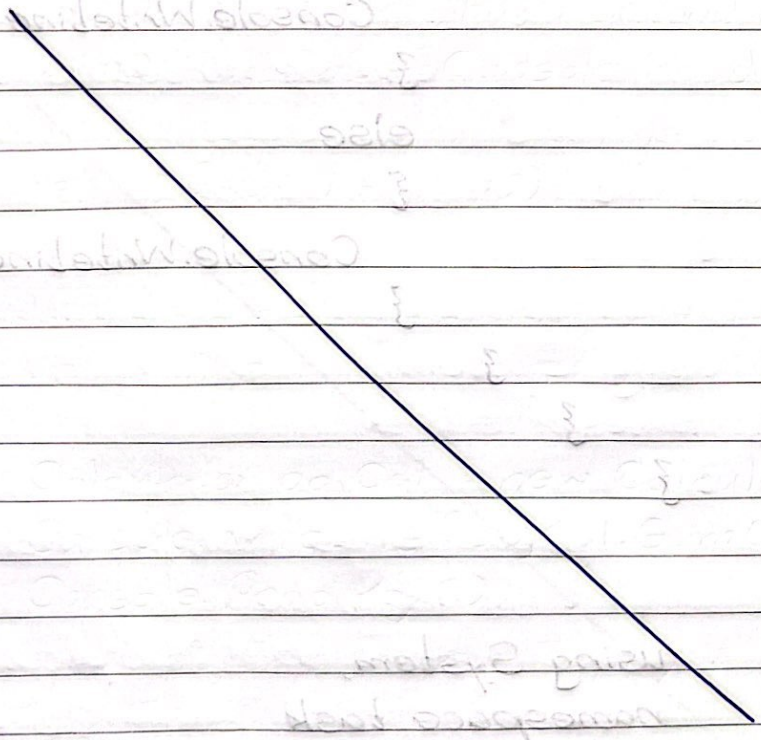
```
{
    Calculator objCal = new Calculator();
    Console.WriteLine(objCal.Sum());
    Console.ReadKey();
}
```

01.) Create a C# Console Application project to get 2 integer numerical inputs from the main method and pass them to a class called Calculations. Inside the class create a method called Summation and get user inserted values as parameters. Inside the method, Find the summation of above parameters and display the answer.

```
using System;
namespace task
{
    public class Calculations
    {
        public void Summation(int x, int y)
        {
            Console.WriteLine(x+y);
        }
    }
}
```

```
using System;
namespace task
{
    class Program
    {
        static void Main(string[] args)
        {
            int num1, num2;
```

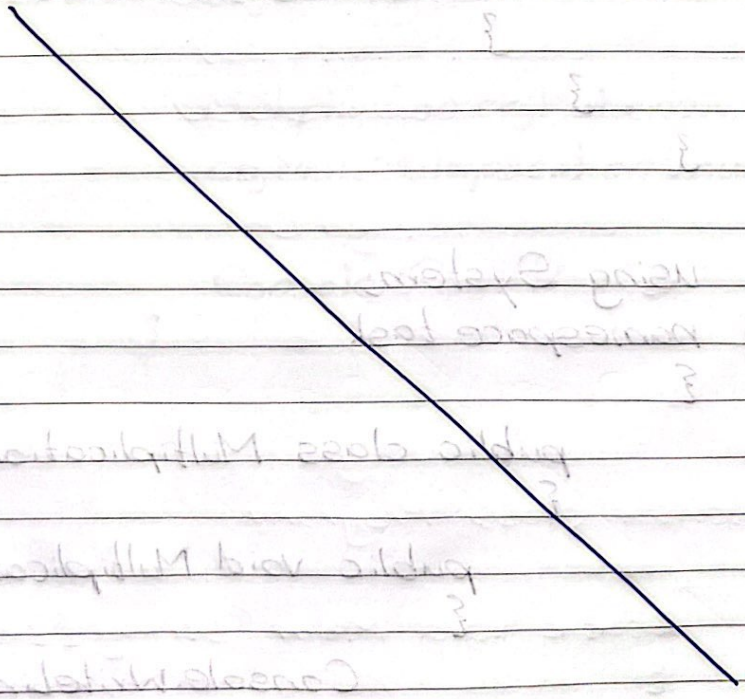
```
Console.WriteLine("Enter First number:");
num1 = int.Parse(Console.ReadLine());

Console.WriteLine("Enter second number:");
num2 = int.Parse(Console.ReadLine());

Calculations objCal = new Calculations();
objCal.Summation(num1, num2);

Console.ReadKey();
}
}
}
```



02.) Create a C# Console application project which contain a class called Validation and a method called Eligibility which take user's age as a parameter. Inside the method, display eligible if user is above 21 years old. Unless display not eligible.

```
using System;
namespace task
{
    public class Validation
    {
        public void Eligibility(int x)
        {
            if(x>=21)
            {
              Console.WriteLine("Eligible");
            }
            else
            {
              Console.WriteLine("Not Eligible");
            }
        }
    }
}


using System;
namespace task
{
    class Program
```

```
{
    static void Main (string [] args)
    {
        int age;

        Console.WriteLine ("Enter your age:");
        age = int.Parse (Console.ReadLine());

        Validation objVal = new Validation ();
        objVal.Eligibility(age);

        Console.ReadKey();
    }
}
}
```



03.) Create a C# console application project which contains 2 classes called Summation and Multiplication. Both of these classes are containing a method called operation which take 2 integer parameters. The above method will do the respective operation and display the answer in the method itself. From the main method, get 2 user inputs and pass them to classes to get the final answer.

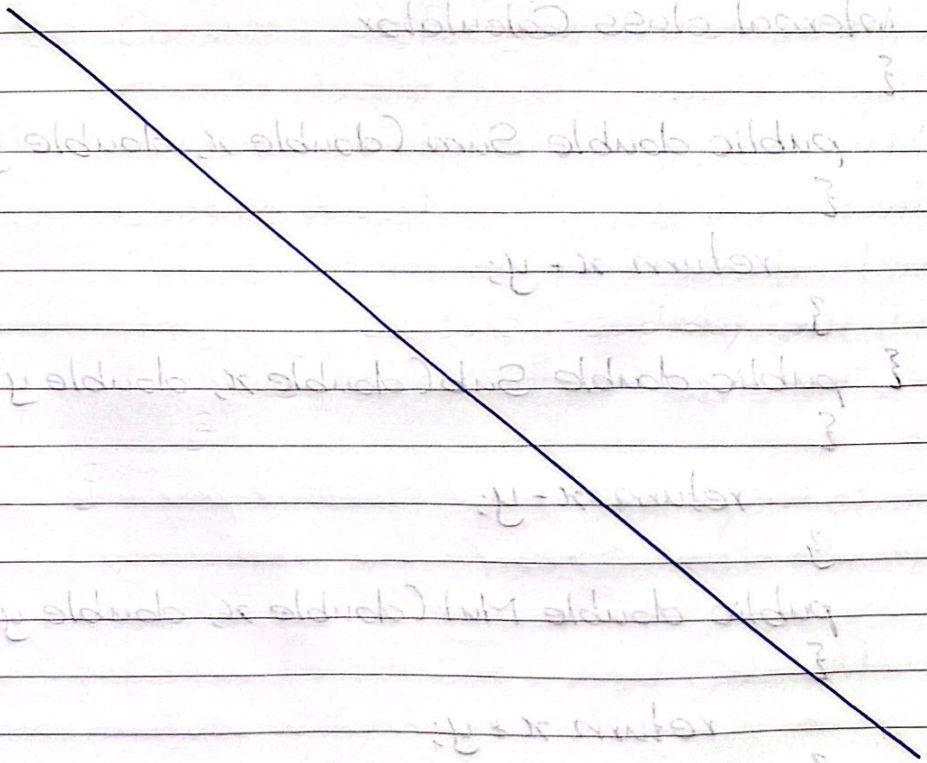
```
using System;
namespace task
{
    public class Summation
    {
        public void Summation(int x, int y)
        {
            Console.WriteLine(x+y);
        }
    }
}

using System;
namespace task
{
    public class Multiplication
    {
        public void Multiplication(int x, int y)
        {
            Console.WriteLine(x*y);
        }
    }
}
```

```
using System;
namespace task
{
    class Program
    {
        static void Main(string[] args)
        {
            int num1, num2;

            Console.WriteLine("Enter first number: ");
            num1 = int.Parse(Console.ReadLine());

            Console.WriteLine("Enter second number: ");
            num2 = int.Parse(Console.ReadLine());

            Summation objSum = new Summation();
            objSum.Summation(num1, num2);

            Multiplication objMul = new Multiplication();
            objMul.Multiplication(num1, num2);

            Console.ReadKey();
        }
    }
}
```

01.) Create a C# console application project with an extra class called Employee, which contains a method called SalaryCalculation. Method is taking working hours as parameters. If employee work more than 10 hours, return 25,000. If employee work more than 20 hours, return 45,000. If the employee's working hours less than 10 hours return 5,000. Display the final salary from the main method.

```
internal class Employee
{
    public int SalaryCalculation (int WH)
    {
        if (WH > 10)
            return 25000;
        else if (WH > 20)
            return 45000;
        else if (WH < 10)
            return 5000;
        else
            return 0;
    }
}

internal class Program
{
    static void Main (string [] args)
    {
        int WH;
```

```
Console.WriteLine("Enter your working hours: ");
WH = int.Parse(Console.ReadLine());
Employee objEmp = new Employee();
int val = objEmp.SalaryCalculation(WH);
Console.WriteLine("Your salary is: "+val);
Console.ReadKey();
}
}
```



02.) Create a C# console application project with a seperate class called Calculator, which contains 4 methods called; Summation, Subtraction, Multiplication, Division. Each method will take 2 double parameters and perform the operation and return the answer. Answer should be return to the main method and display in the main method. From the main method, you should display following key values to the user.

1 → Summation
2 → Subtraction
3 → Multiplication
4 → Division.

Get the user choice by using the above key values and perform the calculations.

```
internal class Calculator
{
    public double Sum (double x, double y)
    {
        return x + y;
    }
    public double Sub(double x, double y)
    {
        return x - y;
    }
    public double Mul (double x, double y)
    {
        return x * y;
    }
```

```
        public double Div (double x, double y)
        {
            return x/y;
        }
    }

class Program
{
    static void Main (string [] args)
    {
        double num1, num2;

        Console.WriteLine("Enter First number: ");
        num1 = double.Parse (Console.ReadLine());

        Console.WriteLine("Enter second number: ");
        num2 = double.Parse (Console.ReadLine());

        Console.WriteLine ("1 - Summation")
        Console.WriteLine ("2 - Substraction")
        Console.WriteLine ("3 - Multiplication")
        Console.WriteLine ("4 - Division")

        Console.WriteLine("Enter your choice ");
        int cho = int.Parse (Console.ReadLine());

        Calculator objCal = new Calculator();
```

```
if (cho == 1)
    Console.WriteLine ("Your answer is " +
                        objCal.Sum (num1, num2));
else if (cho == 2)
    Console.WriteLine ("Your answer is " +
                        objCal.Sub (num1, num2));
else if (cho == 3)
    Console.WriteLine ("Your answer is " +
                        objCal.Mul (num1, num2));
else if (cho == 4)
    Console.WriteLine ("Your answer is " +
                        objCal.Div (num1, num2));
else
    Console.WriteLine ("Invalid Input");

Console.ReadKey ();
}
}
```

Object Oriented Programming with C#.

* In OOP there're 4 main programming concepts are available which known as,
  01. Encapsulation.
  02. Inheritence.
  03. Polymorphism.
  04. Abstraction.

* Apart from above 4 main programming concepts, there're some OOP practices which available in OOP.

* Constructors, Structures, Interfaces are few examples for those practices.

What is a Constructor?

* Constructor is a special type of method which can use in OOP and a constructor will get called whenever you create a class object by using that class name.

* As an example, assume that there's a class called Student.cs which contains a constructor and that constructor will get called when you create a class object using Student class.

* Constructors are usually used when initializing variables, declaring variables simultaneously, when dealing with another class.

* When declaring a constructor, it should contain following characteristics.
    01. Constructor doesn't have a return type. Because of that nothing will get return from constructor.
    02. Constructor should have an access specifier and most of the time, this is a public access specifier.
    03. When writing a constructor the constructor name should be the class name.
    04. One class could contain n number of constructors by changing the parameter list of the constructor.

* Inside a constructor, you can do almost anything such as declaring variables, initializing variables, getting user inputs, doing operations, display ing content etc.

* In programming there're 2 main constructors are available known as,
    01. Default constructor.
    02. Parameterized constructor.

* Default constructor will never contain parameters and parameterized constructor will always contain number of parameters using different data types.

* A class could always contain a 1 default constructor.

Default Constructor.

```
class Person
{
    int a, b;

    public Person() //default constructor.
    {
        a = 200;
        b = 400;
    }
}
```

Parameterized Constructor.

```
class Person
{
    int a, b;

    public Person(int x, int y) //parameterized constructors.
    {
        a = x;
        b = y;
    }
}
```

01.)

I. Create a C# console application project, which contains a class called Employee and inside the class create a default constructor. Then display Hello World.

```
internal class Employee
{
    public Employee()
    {
        Console.WriteLine("Hello World");
    }
}

internal class Program
{
    static void Main(string[] args)
    {
        Employee objEmp = new Employee();

        Console.ReadKey();
    }
}
```

II. Add a no return type, no parameter method to the employee class to display your name.

```
internal class Employee
{
    public Employee()
    {
        Console.WriteLine("Hello World");
    }

    public void MyInfo()
    {
        Console.WriteLine("Afsaan");
    }
}


internal class Program
{
    static void Main(string[] args)
    {
        Employee objEmp = new Employee();

        objEmp.MyInfo();

        Console.ReadKey();
    }
}
```

02.) Create a C# console application project with a class called person where the person class is having parameterized constructor with 2 integer parameters. Inside the constructor, display the summation of 2 integer parameters.

```
internal class Person
{
    public Person(int x, int y)
    {
        Console.WriteLine(x+y);
    }
}


internal class Program
{
    static void Main(string[] args)
    {
        Person objPer = new Person(10,20);

        Console.ReadKey();
    }
}
```

*For a parameterized constructor, values needed to be passed at the point of creating the class object.

*Based on the parameter list values should be passed accordingly.

Question

Explain the differences between public and private access specifiers.

*Access specifiers will decide the accessability level of member variables and member functions.
*If a member declared under public access specifier, that member will be available to all other classes in project namespace.
*If a member declared under private access specifier, that member will be available only to the class it declared.

Object Oriented Programming Concepts

*In OOP there're 4 OOP concepts are available to minimize the complexity of the development and improve the efficiency of application development.
*There're number of advantages using OOP concepts in programming and these concepts can use accordinly under the given requirement.

01.) Encapsulation - Package the variables and functions into a single unit.
02.) Abstraction - Show only the essentials to the outside use.
03.) Inheritance - Inherit common variables and functions.
04.) Polymorphism - Function the object depending on the data.

Encapsulation.

* In encapsulation, we're trying to access a private variable from another class by using public methods.

* If a variable declared under private access specifier that cannot be accessed directly from another class. But using a public method and parameters that variable can be accessed by any other class in the same namespace. (project)

* Since we're restricting the private variable to the class itself, encapsulation also known as a method of hiding data.

* In encapsulation a special method name is being used when declaring public methods. Those methods are known as get and set methods. (Getters and Setters)

* A set method always used to set a value to the private variable and set method is always a public method.

* A get method can be used to return the value out of the class when required. This is also a public method.

NOTE

Use of get and set keywords are not mandatory in programming but as a good programming practice these keywords are being used, to identify that encapsulation has been used in the given project.

*Set method is always a void method and will always take a parameter to the method.

*Get method always return a value out of the function.

Example

```
namespace ConsoleAppl
{                                          Encap Class
    class EncapClass
    {
        private int age;

        public void setAge (int ageFromUser)
        {
            age = ageFromUser;
        }
        public int getAge()
        {
            return age;
        }
    }
}
```

```
namespace ConsoleApp1
{                                                    Program Class
    class Program
    {
        static void Main (string [] args)
        {
            EncapClass ec = new EncapClass();
            ec.setAge
            Console.WriteLine ("Your Age is"+ ec.getAge());
            Console.ReadKey();
        }
    }
}
```

Add another private variable to the same code and create relevant getters and setters using encapsulation. Private variable should contain user's name.

```
namespace ConsoleApp1
{
    class EncapClass
    {
        private int age;
        private stringname;

        public void setAge (int ageFromUser)
        {
            age = ageFromUser;
        }
```

```
        public void setName (string nameFromUser)
        {
            name = nameFromUser;
        }

        public int getAge()
        {
            return age;
        }

        public string getName()
        {
            return name;
        }
    }
}
```

```
namespace ConsoleApp1
{
    class program
    {
        static void Main (string [] args)
        {
            EncapClass ec = new EncapClass ();
            ec.setAge(23);
            ec.setName("Afsaan");
            Console.Writeline("Your age is" + ec.getAge());
            Console.Writeline("Your name is" + ec.getName());
            Console.ReadKey();
        }
    }
}
```

* Encapsulation Worksheet.

(01.) Your age is: 100

(02.) The code in the Main method of the MainClass will no longer be able to directly access the 'a' variable in the HelloClass, if it has been converted to private access modifier. (hello.a)

In reality, a compiler warning stating that 'a' is unavailable owing to its protection level would appear if we attempt to build the code with 'a' set to private.

(03.) EncapData Class

```
public class EncapData
{
    private double radius;
    private double pi = Math.PI;

    public double radius
    {
        get
```

```
        {
            return radius;
        }
        set
        {
            radius = value;
        }
    }
    public double rea
    {
        get
        {
            return pi * radius * radius;
        }
    }
    public double circumference
    {
        get
        {
            return 2 * pi * radius;
        }
    }
}
```

Program Class

```
class Program
{
    static void Main(string[] args)
    {
```

```
Console.WriteLine("Enter the radius of the circle: ");
double radius = double.Parse(Console.ReadLine());

EncapData edata = new EncapData();
edata.radius = radius;

double area = edata.area

double circumference = edata.circumference;

Console.WriteLine("Area of the circle: " + area);
Console.WriteLine("Circumference of the circle: " +
                                        circumference);

Console.ReadKey();
}
}
```

04.) Arithmetic Class

```
public class arithmetic
{
    private double num1;
    private double num2;

    public double Num1
    {
        get
        {
```

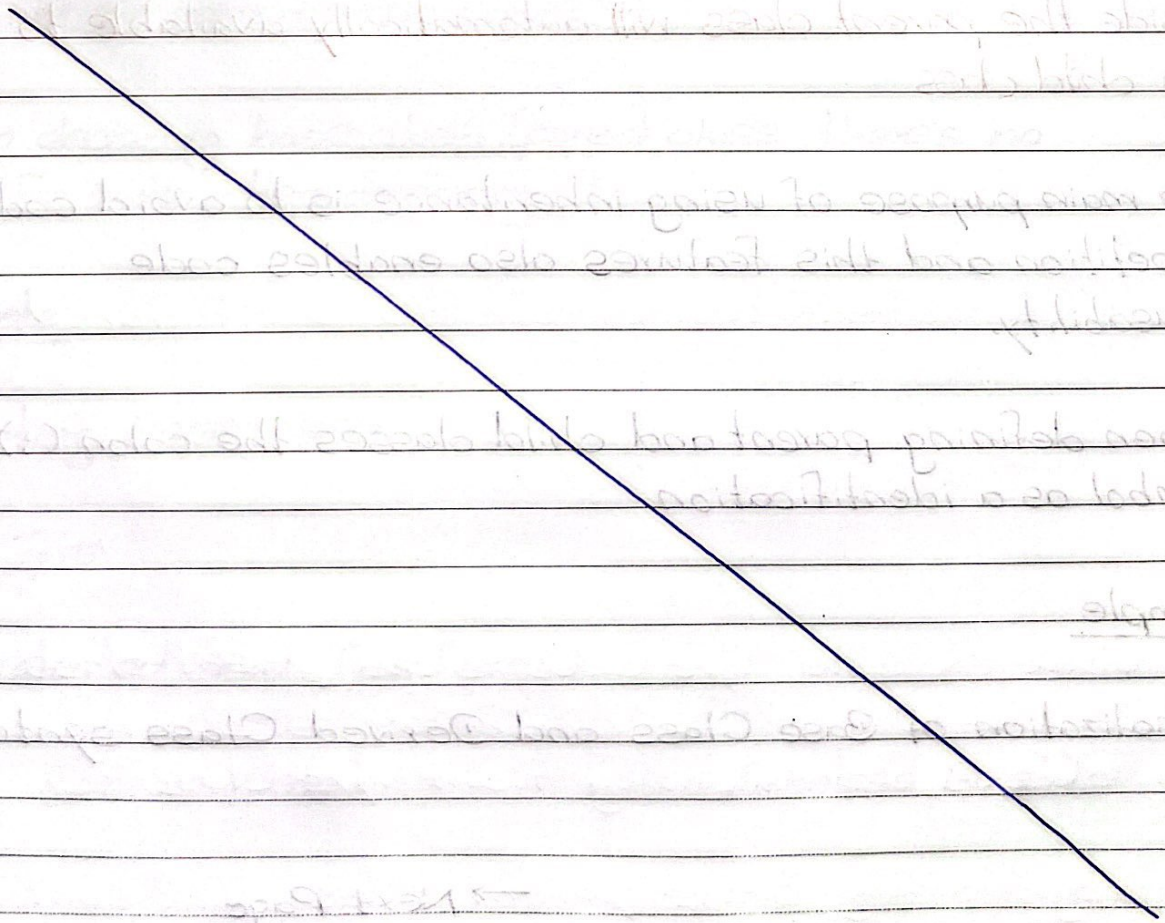
```
        return num1;
    }
    set
    {
        num1 = value;
    }
}

public double Num2
{
    get
    {
        return num2;
    }
    set
    {
        num2 = value;
    }
}

public double Add()
{
    return num1 + num2;
}

public double Sub()
{
    return num1 - num2;
}
```

```
public double Mul()
{
    return num1 * num2;
}

public double Div()
{
    if (num2 == 0)
    {
        Console.WriteLine("Error: Can't divide by Zero!");
        return -1;
    }
    return num1 / num2;
}
}
```

Program Class

```
class Program
{
    static void Main (string [] args)
    {
        Console.WriteLine ("Enter the first number: ");
        double num1 = double.Parse (Console.ReadLine());

        Console.WriteLine ("Enter the second number: ");
        double num2 = double.Parse (Console.ReadLine());
```

```
Arithmetic arith = new Arithmetic();
arith.Num1 = num1;
arith.Num2 = num2;

Console.WriteLine (arith.Add());
Console.WriteLine(arith.Sub());
Console.WriteLine(arith.Mul());
Console.WriteLine(arith.Div());

Console.ReadKey();
}
}
```



Inheritance.

*In this OOP concept there should be minimum of 2 classes available in a project.

*These 2 classes should declare under parent and child declaration.

*In programming, parent class is also known as based class and child class is known as derived class.

*In inheritence, member variables and functions which declared under public and protected access specifier inside the parent class will automatically available to its child class.

*The main purpose of using inheritance is to avoid code repetition and this feature also enables code reusability.

*When defining parent and child classes the colon (:) symbol as a identification.

Example

Initialization of Base Class and Derived Class syntax.

→ Next Page.

```
<access_specifier> class <base_class>
    {
        ...  <-- parent class / base class
    }

class <derived_class> : <base_class>
    {
        ...  <-- child class / derived class
    }
```

*In above syntax, when declaring the child class, colon (:) symbol has been used right after the child class which followed by parent class name.

*When declaring base class / parent class there's no specific rule when declaring it.

Example

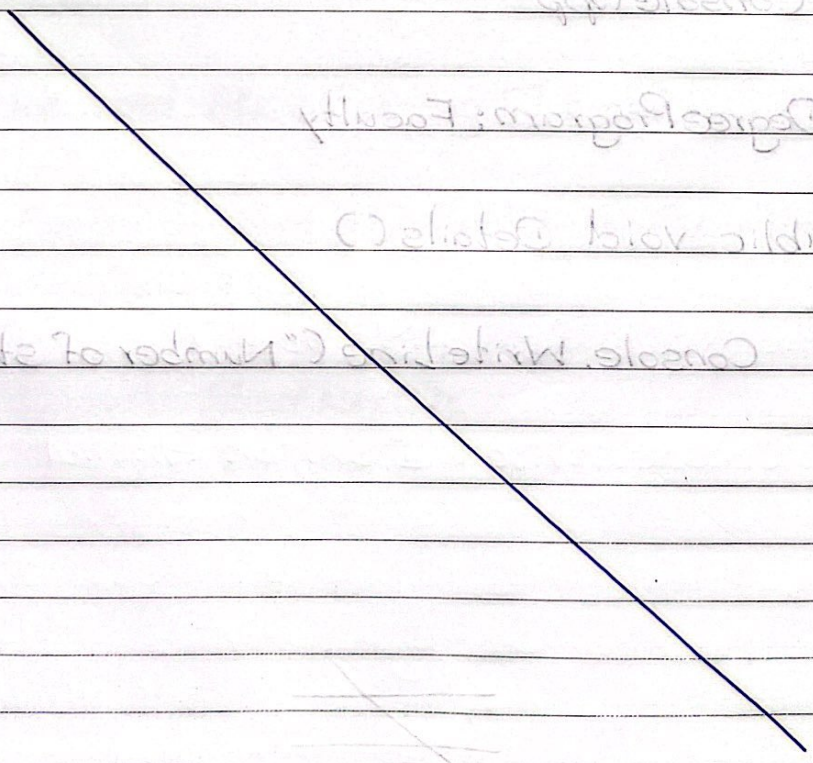
```
//creating base class

class Tyre
{
    protected void TyreType()
    {
        Console.WriteLine("This is a Tubeless Tyre");
    }
}
```

```
//creating child class

class Scooter : Tyre
{
    public void ScooterType()
    {
        Console.WriteLine("Scooter Color is Red");
        Tyre Type();
    }
}

//creating child class

class Car: Tyre
{
    public void CarType()
    {
        Console.WriteLine("   This is a Ferrari");
        TyreType();
    }
}
```

*In above example, Tyre class is the parent class of Scooter and Car classes. Which means Scooter and Car classes are inheriting from Tyre class.

*In programming single parent class could contain multiple child classes.

```
class Program
{
    static void Main (string [] args)
    {
        Scooter sc = new Scooter ();
        sc.ScooterType();

        Car c = new Car ();
        c.CarType();

        Console.ReadKey();
    }
}
```



NSBM GREEN UNIVERSITY TOWN

Inheritance Task.

- Create a C# console application.
- Add 2 classes called "Faculty" and "Degree Program".
- Make "Degree Program" as the derived class. (Child class)
- Inside "Faculty" class create an integer variable called "student" and assign value 100.
- Inside the "Degree Program" class create a void method called "Details" and print the "student" variable.
- Call the method "Details" inside main method via a class project.

DegreeProgram.cs

```
namespace ConsoleApp
{
    class DegreeProgram: Faculty
    {
        public void Details()
        {
            Console.WriteLine("Number of students: " +
                                                students);
        }
    }
}
```

Faculty.cs

```
namespace ConsoleApp
{
    class Faculty
    {
        public int student = 100;
    }
}
```

Program.cs

```
namespace ConsoleApp
{
    class program
    {
        static void Main (string [] args)
        {
            DegreeProgram = new DegreeProgram ();
            dp.Details ();

            Console.ReadKey ();
        }
    }
}
```

Inheritance Exercise

- Create a console application with 2 added classes called "Animal" and "Dog".
- "Dog" is derived class (child class) of "Animal class" (base class)
- Inside the "Animal class", create a method.
- Inside the method, print "I am an Animal"
- Inside the "Dog Class" create a method and display "I have four legs."
- Inside the main method, create relavent class object and display as follows.
- "I am an Animal. I have four legs."

Animal.cs

```
namespace ConsoleApp
{
    internal class Animal
    {
        public void AnimalMethod()
        {
            Console.Write("I am an Animal ");
        }
    }
}
```

Dog.cs

```
namespace ConsoleApp
{
    internal class Dog: Animal
    {
        public void DogMethod()
        {
            Animal.Method();

            Console.WriteLine("I have four legs");
        }
    }
}
```

Program.cs

```
namespace ConsoleApp
{
    internal class program
    {
        static void Main(string[] args)
        {
            Dog objDog = new Dog();
            objDog.DogMethod();

            Console.ReadKey();
        }
    }
}
```

NSBM GREEN UNIVERSITY TOWN

Windows Form Applications.

* C# language is supporting graphical user interface (GUI) application developments.

* These applications are capable of handling multiple user interactions using different types of controlles such as buttons, text boxes, combo boxes etc.

* As a developer, you can develop GUI applications by adding above controllers based on the requirement.

* To develop GUI applications, Windows Form Application (.net Framework) is being used.

* In a single form application project there could be n number of forms (user interfaces).

01.) Create a windows form application which contains a lable and a button controller. Once user clicks on the button, the text of the lable should convert into "Hello World".

02.) Create a windows form application which contains 3 buttons called Blue, Yellow and Red. Once user clicks on above buttons, the background color of the application should change accordingly. Add a Default button to change the background color of the application to default color.

03.) Create a windows form application which contains a lable and a button. Once you click on the button, the button click count should disply on the lable itself.

Getting User Inputs.

* In windows Form application text box controller has been used to get user inputs to the application.

* From the text box controller, these input values can be assigned to variables from backend C# class File.

* Once the value has been taken into the variable, that value can be used For modifications, such as adding content, doing calculations etc.

* When taking user inputs through text boxes, every user is considered as a string input. When expecting a numerical value, necessary type casting needed to be done based on expected data type.

01.) Create a windows form application to get user's name, batch and degree using 3 seperate text boxes and once the user clicks on the OK button all 3 values should display on the bottom of the screen. Note that all input values are string inputs. Add a cancel button to clear the content on all text boxes and labels.

02.) Create a windows form application to get 2 integer inputs to the application and display the summation once user clicks on the Sum button.

Add 3 more buttons for Subtraction, Multiplication, Division and display the answers for respective operation.

Form Navigation

* In windows Form application projects, there could be number of user interfaces which contains in one project

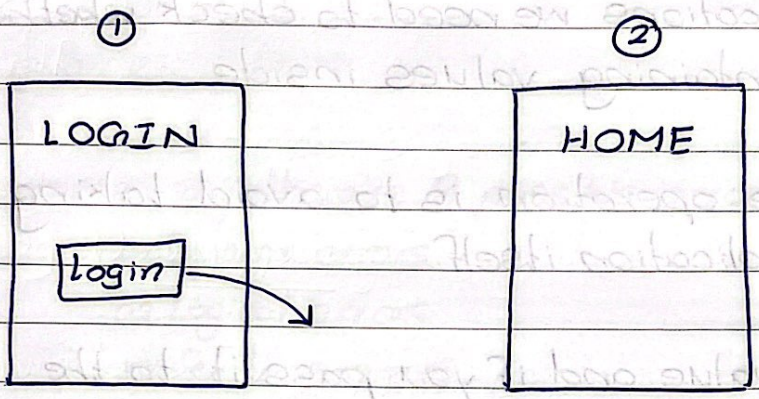
* In reality we need to combine all of these user interfaces together to work as one unit.

* In developing, we're connecting these interfaces by using a button click event.

* In form applications, each UI has a dedicated class and when navigating through the forms, we're using those classes to create class objects.

* Through the created class object we can navigate users to the next user interface.

Example



*In above diagram it contains 2 obsses called Login.cs and Home.cs.

*When user clicks on the login button user should navigate to home window and disable the login form.

*Inside the login button click event, we should create a class object and through the object Show function needed to be called.

```
Home objhm = new Home();
this.Hide();
objhm.Show();
```

Modify the same application by adding 2 text boxes for username and passwords. If user name is equal to Admin and password is equal to admin123 only user will direct to the home.

Form Validations

* When developing applications we need to check whether all text boxes are containing values inside.

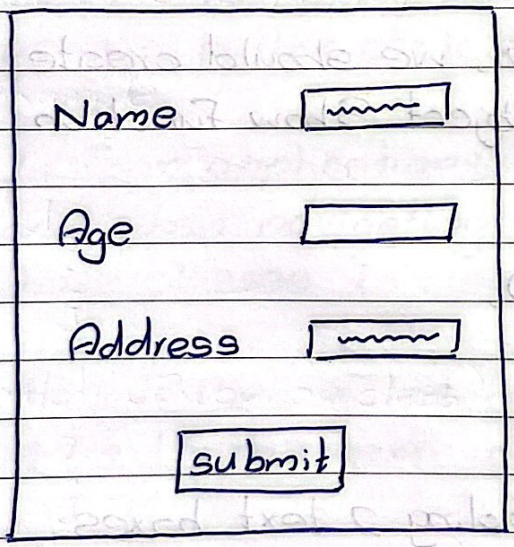
* The importancy of the operation is to avoid taking null values to the application itself.

* Once you have a null value and if you pass it to the database, database will populate with null values.

* In application development above process is known as form validation.

* Validation can be done by using a if condition.

Example



Name	Age	Address
	Null	

```
if (txtName.Text != "" && txtAge.Text = "" && txtAdd.Text = "")
```

Exception Handling.

What is an exception?

*In programming there're 3 possible error types which known as,

01. Syntax errors.
02. Logical errors.
03. Runtime errors.

*Exception is known as a run time error which can occur at the application execution level.

*Exception handling is handling runtime errors and providing better user experience to the users.

*Assume that there's an application to find out the division value of two given numbers. For the second number if user input 0 mistakenly the application will get crashed due to an exception.

*When developing applications all possible exceptions should handle by the developer.

Handling an Exception.

*To handle exceptions in OOP try and catch block will be used.

*In the try block we should include the code that you assume which might throw an exception at the execution level.

* If there's an exception, From the try block compiler will move into the catch statement / block and execute the given instructions.

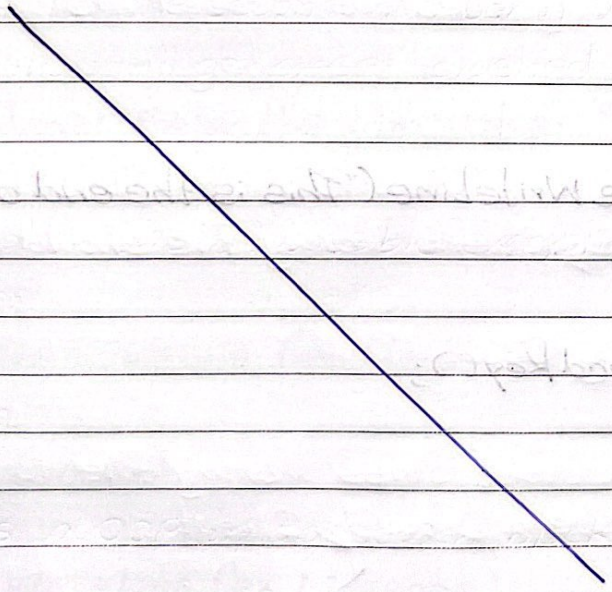
```
try
  {

  }
catch (Exception class)
  {

  }
```

* With a single try statement there could be multiple catch statements in programming.

```
try
  {
  }
catch (Exception class)
  {
  }
catch (Exception class)
  {
  }
```

* After using try and catch blocks if only required a finally block can be used at the end of the code. if or if not there's an exception finally block will execute continue the program.

```
try
  {
  }
catch (Exception class)
  {
  }
Finally
  {
  }
```



Example

```
static void Main(string[] args)
{
    Console.WriteLine("Enter Number One: ");
    int Num1 = int.Parse(Console.ReadLine());

    Console.WriteLine("Enter Number Two: ");
    int Num2 = int.Parse(Console.ReadLine());

    try
    {
        Console.WriteLine("Division is: " + (Num1 / Num2));
    }
    catch (Exception xyz)
    {
        Console.WriteLine(xyz.Message);
    }
    Finally
    {
        Console.WriteLine("This is the end of the application");
    }

    Console.ReadKey();

}
```

* In above example, exception is a general exception class which is capable of handling all possible runtime errors at the execution level.

* xyz is known as the created instance of the exception class.

01.) Create a console application project to convert given l values to ml and display the answers. use exception handling to avoid inserting character based values for l by users.

```
class Program
{
  static void Main(string[] args)
  {
    int l = 0;

    Console.WriteLine("Enter the litre value: ");

    try
    {
      l = int.Parse(Console.ReadLine());
    }
    catch (Exception ex)
    {
      Console.WriteLine(ex.Message);
    }

    Console.WriteLine("Mililitre value is: " + (l * 1000));
    Console.ReadKey();
  }
}
```

Database Connections

ADO.net

*When connecting applications with the database in .net development ADO.net library files will be used.

*After connecting an application with a database from the application users are capable of creating, reading, updating, deleting data. This process is known as CRUD Operation.

*By using ADO.net we're capable of connecting SQL, Excel, Access, NOSQL databases with the application.

*Depending on the database that you're connecting the library file that you have to use will be changed.

*All ADO.net library files are located inside System.Data namespace.

*When connecting applications with SQL Databases System.Data.SqlClient library file should import.

Steps of Inserting Data.

01.) Import the relevant class library file to the C# class file.
- Using System.Data.SqlClient

02.) Create a class object by using SqlConnection class by passing the connection string as a parameter.

Connection String will define the location of the database.

03.) Write an appropiate SQL Query to insert data by using relavent table name.

04.) Create a class object by using SqlCommand cmd by passing query and connection object.