

Program-1 :Stack Implementation using OOP

```
using System;

public class Stack
{
    int[] stack;
    int size;
    int top = -1;

    public Stack()
    {
        Console.Write("Enter the size of Your Stack :");
        size = Convert.ToInt32(Console.ReadLine());
        stack = new int[size];
    }

    public bool OverFlow()
    {
        if (top == size)
        {
            return true;
        }
        else
        {
            return false;
        }
    }

    public bool UnderFlow()
    {
        if (top == -1)
        {
            return true;
        }
        else
        {
            return false;
        }
    }

    public void Push()
    {
        if (OverFlow())
        {
            Console.WriteLine("Stack Overflow");
        }
        else
        {
            Console.Write("\nEnter the Element to be Inserted to Stack :");
            int num = Convert.ToInt32(Console.ReadLine());
        }
    }
}
```

```

        stack[top + 1] = num;
        top = top + 1;
    }
}

public void Pop()
{
    if (UnderFlow())
    {
        Console.WriteLine("The Stack is Empty Right Now");
    }
    else
    {
        Console.WriteLine("\nElement Removed Successfully!\n");
        stack[top] = 0;
        top = top - 1;
    }
}

public void Peep()
{
    if (UnderFlow())
    {
        Console.WriteLine("\nThe Stack is Empty Right Now\n");
    }
    else
    {
        Console.WriteLine("\nThe Top of the Stack is :" + stack[top]+"\n");
    }
}

public void Display()
{
    //for (int i = 0; i <= top; i++)
    for (int i = top; i >= 0; i--)
    {
        Console.WriteLine("\n" + stack[i]);
    }
}

}

class ClientTest
{
    public static void Main()
    {
        Stack st = new Stack();
        try
        {

            for (int p = 0; ; p++)
            {
                //Console.Clear();
                Console.WriteLine("Enter Your Choice :\n1.Push\n2.Pop\n3.Peep\n4.display\n5.Exit\n");

                Console.WriteLine("-----");
                int choice = Convert.ToInt32(Console.ReadLine());

```

```

        Console.WriteLine("-----");

        if (choice == 5)
        {
            break;
        }
    else
    {

        switch (choice)
        {

            case 1:
                st.Push();
                break;

            case 2:
                st.Pop();
                break;

            case 3:
                st.Peep();
                break;

            case 4:
                st.Display();
                break;

            case 5: break;

        }

    }

}
catch (IndexOutOfRangeException e)
{
    Console.WriteLine("Stack is Empty");
}
catch (FormatException e)
{
    Console.WriteLine("Stack is Empty");
}

}
}

```

Program-2 :Substring User Defined

```

using System;

class ServerTest
{
    public string str = "";

    public ServerTest()
    {
        Console.Write("Enter Your String: ");
        str = Console.ReadLine();

        //Default Substring() Functions
        Console.WriteLine(str.Substring(3));
        Console.WriteLine(str.Substring(3,8));
    }

    public string SubString1(int startIndex)
    {
        string substr = "";
        for (int i = startIndex; i < str.Length; i++)
        {
            substr = substr + str[startIndex];
            startIndex++;
        }
        return substr;
    }

    public string SubString1(int startIndex, int length)
    {
        string substr = "";

        if (length > str.Length)
        {
            Console.WriteLine("Invalid Input IndexOutOfRange!");
        }
        else
        {
            for (int i = startIndex; i <= length + 2; i++)    //DOUBT: Why It need length+2
            {
                substr = substr + str[startIndex];
                startIndex++;
            }
        }
        return substr;
    }
}

class ClientTest
{
    public static void Main()
    {
        ServerTest s = new ServerTest();
    }
}

```

```
Console.WriteLine("-----");
Console.WriteLine(s.SubString(3));
Console.WriteLine(s.SubString(3, 8));
Console.WriteLine("-----");
//Pramod Vishnu Naik
}
}
```

Program-3: Sorting the N Strings Using User-Defined Function

```

class Pramod
{
    public void Sort(string[] str)
    {
        string temp;

        for(int i=0;i<str.Length;i++)
        {
            for(int j=0;j<str.Length;j++)
            {
                if(str[i].CompareTo(str[j])<0)
                {
                    temp = str[i];
                    str[i] = str[j];
                    str[j]=temp;
                }
            }
        }

        Console.WriteLine("-----");
        Console.WriteLine("The Sorted Array Elements are :");
        for(int i=0;i<str.Length;i++)
        {
            Console.WriteLine(str[i]);
        }
    }
}

class MainTest
{
    public static void Main()
    {
        //string[] str = { "PRAMOD", "Dangu", "Prajwal", "SUNIL", "Zabyulla" };

        Console.Write("Enter the Number of Strings:");
        int size = Convert.ToInt32(Console.ReadLine());

        string[] str = new string[size];

        Console.WriteLine("Enter {0} Strins :", size);

        for(int i=0;i<size;i++)
        {
            str[i] = Console.ReadLine();
        }

        Pramod p = new Pramod();

        p.Sort(str);
    }
}

```

Program-4 :Regex Split Function

```
using System.Text.RegularExpressions;
using System.Text;

class PramodRegex
{
    public static void Main()
    {
        //string reg = "( |,)" ; //RegEx Patter
        string str = "Amar,Akbar,Antony are Friends!";

        Regex reg = new Regex(" |,");
        StringBuilder sb = new StringBuilder();

        int count = 1;

        foreach(string sub in reg.Split(str))
        {
            sb.AppendFormat("{0}: {1}\n", count++, sub);
        }
        Console.WriteLine(sb);
    }
}
```

Program-5 : ArrayList Implementation using OOP

```
using System.Collections;
/*class Pramod
{
    ArrayList list = new ArrayList();
}*/

class MainTest
{
    public static void Main()
    {
        ArrayList list = new ArrayList();

        for (int p = 0; ; p++)
        {
            Console.WriteLine("\nEnter Your
Choice:\n1.Add\n2.Display\n3.Remove\n4.Insert\n5.Index\n6.Contains\n7.Clear\n8.Exit");
```

```

int choice = Convert.ToInt32(Console.ReadLine());

switch (choice)
{
    case 1:
        Console.WriteLine("\nEnter the Element to Be added to the List :");
        object obj = Console.ReadLine();
        list.Add(obj);
        break;

    case 2:

        if (list.Count <= 0)
        {
            Console.WriteLine("\nThe List is Currently Empty!\n");
        }
        else
        {
            Console.WriteLine("\nElements of the List Are :");
            foreach (object i in list)
            {
                Console.WriteLine(i);
            }
            Console.WriteLine("\n");
        }
        break;

    case 3: //-----Nested Switch to implement the Remove() and RemoveAt() Functions of List

        if (list.Count <= 0)
        {
            Console.WriteLine("\nThe List is Currently Empty!\n");
        }
        else
        {
            Console.WriteLine("\nEnter The way to Remove the Element from the List : \n1. Using The Object
Name \n2. Using the Index \n");

            int ch = Convert.ToInt32(Console.ReadLine());
            switch (ch)
            {
                case 1:
                    Console.WriteLine("\nEnter the Object to Remove From the List :");
                    object o = Console.ReadLine();
                    list.Remove(o);
                    Console.WriteLine("Object {0} Removed Successfully!", o);
                    Console.WriteLine();
                    break;

                case 2:
                    Console.WriteLine("\nEnter the Index of the Object to be removed :");
                    int q = Convert.ToInt32(Console.ReadLine());
                    list.RemoveAt(q);
                    Console.WriteLine("Object {0} Removed Successfully!", q);
                    Console.WriteLine();
                    break;
            }
        }
    }
}

```



```

    }
    break;

case 4:
    Console.Write("\nEnter the ele to be inserterd :");
    object ele = Console.ReadLine();
    Console.Write("\nEnter the index where you want to insert :");
    int indx = Convert.ToInt32(Console.ReadLine());
    list.Insert(indx, ele);
    break;

case 5:
    Console.Write("\nEnter the Object Name to get the Index :");
    object op = Console.ReadLine();

    Console.WriteLine("Index is :" + list.IndexOf(op) + "\n");
    break;

case 6:
    Console.Write("\nEnter the Object Name to Check The Availability :");
    object l = Console.ReadLine();

    Console.WriteLine(list.Contains(l));
    break;

case 7:
    list.Clear();
    Console.WriteLine("List is Empty Now!");
    break;

case 8:
    break;

    }
}

}
}

```

Program-6 : User Defined Exception Handling

//Program-1: Implementing the Exception by inheriting the Exception class

```

class NoMatchException : Exception
{
    public NoMatchException(string s) : base(s)
    {

```

```

    }
}

class Pramod
{
    public static void Main()
    {
        for(int i=0; ;i++)
        {
            string str;
            Console.Write("Enter the String :");
            str = Console.ReadLine();

            try
            {
                if(str!="India" && str!="india" && str!="INDIA")
                {
                    throw new NoMatchException("Input is Not INDIA");
                }
            }
            catch(NoMatchException e)
            {
                Console.WriteLine(e.Message);
            }
        }
    }
}

```

//Program-1: Exception Handling while Taking the Input from the User

```

/*class Pramod
{
    public static void Main()
    {
        for (int i = 0; ; i++)
        {
            int ii;
            try
            {
                Console.Write("Enter the Number :");
                ii = int.Parse(Console.ReadLine());
            }
            catch (Exception e)
            {
                Console.WriteLine("The Input is not a number!");
            }
        }
    }
}*/

```

Program-7 : Current and Saving Account :

```
class Account
{
    private double balance;

    public Account(double balance)
    {
        Balance = balance;
    }

    public double Balance
    {
        set
        {
            if (value >= 0)        //value is a keyword
            {
                this.balance = value;
            }
            else
            {
                Console.WriteLine("Balance Cannot be Negative!");
            }
        }

        get
        {
            return this.balance;
        }
    }

    public virtual void Credit(double amount)
    {
        if (amount > 0)
            Balance += amount;
        else
            Console.WriteLine("Negative Amount Cannot be Credited");
    }

    public virtual bool Debit(double amount)
    {
        bool OK = false;
        double minBal = 2000;
        if (Balance - amount >= minBal)
        {
            Balance = Balance - amount;
            OK = true; //Withdraw successfull-----this is used to apply tax or fee
        }
        else
    }
```

```

    {
        Console.WriteLine("Debit Amount Exceeds Account Balance!");
        OK = false; //no Withdrawal
    }
    return OK;
}

public void DisplayBalance()
{
    Console.Write("Your Remaining Balance is :" + Balance);
    Console.WriteLine("\n");
    //return this.Balance;

}

}

```

```

class SavingsAccount : Account
{
    private double intrest;

    public SavingsAccount(double balance, double intrest) : base(balance)
    {
        this.Intrest = intrest;
    }

    public double Intrest
    {
        set
        {
            if (value > 0) //value is a keyword
                this.intrest = value;
            else
                Console.WriteLine(" Intrest Amount Cannot be Negative");
        }

        get
        {
            return this.intrest;
        }
    }

    public double CalculateIntrest()
    {
        return Balance * Intrest; //Intrest is in % other wise / by 100
    }

}

```

```

class CurrentAccount : Account
{
    //Use either decimal or double DATA TYPE

    private double fee;

    public CurrentAccount(double balance, double fee) : base(balance)

```

```

{
    this.Fee = fee;
}

public double Fee
{
    set
    {
        if (value > 0)    //value is a keyword
            this.fee = value;
        else
            Console.WriteLine("Fee Amount Cannot be Negative!");
    }

    get
    {
        return this.fee;
    }
}

public override void Credit(double amount)
{
    base.Credit(amount);
    // base.Balance -= fee; // Balance is a Property
    Balance -= fee;
}

public override bool Debit(double amount)
{
    //return base.Debit(amount);
    if (base.Debit(amount))
    {
        Balance = Balance - fee;
        return true;
    }
    else
    {
        return false;
    }
}
}

class MainClass
{
    public static void Main()
    {
        double totalIntrest;

        SavingsAccount saving = new SavingsAccount(10000d, 0.1);

        //This is because We have to give the Intrest to the Saving account
        totalIntrest = saving.CalculateIntrest();
        saving.Credit(totalIntrest);
    }
}

```

```

CurrentAccount current = new CurrentAccount(30000, 10);

for (int i = 0; ; i++)
{

    Console.WriteLine("Enter your Option :\n1.Savings Account\n2.Current Account\n");

    int choice = Convert.ToInt32(Console.ReadLine());

    switch (choice)
    {
        case 1:

            Console.WriteLine("\nChoose your Option :\n1.Credit\n2.Debit\n3.Balance\n");
            int option = Convert.ToInt32(Console.ReadLine());

            switch (option)
            {
                case 1:
                    Console.Write("Enter your Credit Amount :");
                    int credit = Convert.ToInt32(Console.ReadLine());

                    saving.Credit(credit);

                    Console.WriteLine("\nAmount {0} Credited Successfully!\n", credit);
                    break;

                case 2:
                    Console.Write("Enter your Debit Amount :");
                    int debit = Convert.ToInt32(Console.ReadLine());

                    saving.Debit(debit);

                    Console.WriteLine("\nAmount {0} Debited Successfully!\n", debit);
                    break;

                case 3:
                    saving.DisplayBalance();
                    break;

            }
            break;

        case 2:

            Console.WriteLine("\nChoose your Option :\n1.Credit\n2.Debit\n3.Balance\n");
            int opt = Convert.ToInt32(Console.ReadLine());
            switch (opt)
            {
                case 1:
                    Console.Write("Enter your Credit Amount :");
                    int credit = Convert.ToInt32(Console.ReadLine());

                    current.Credit(credit);

                    Console.WriteLine("\nAmount {0} Credited Successfully!\n", credit);
                    break;
            }
        }
    }
}

```

```

        case 2:
            Console.Write("Enter your Debit Amount :");
            int debit = Convert.ToInt32(Console.ReadLine());

            current.Debit(debit);

            Console.WriteLine("\nAmount {0} Debited Successfully!\n", debit);
            break;
        case 3:
            current.DisplayBalance();
            break;
    }

    break;

    }
}

/* SavingsAccount saving = new SavingsAccount(10000d, 0.1);

totalIntrest = saving.CalculateIntrest();

Console.WriteLine("Balance Before Intrest{0}: ", totalIntrest);

saving.Credit(totalIntrest);
Console.WriteLine("Balance After Intrest{0}: ", saving.Balance);
Console.WriteLine("-----");

CurrentAccount current = new CurrentAccount(10000, 10);

Console.WriteLine("Current balance is {0}", current.Balance);
current.Credit(100000);
current.Debit(200);

Console.WriteLine("Total balance is {0}", current.Balance);
*/

}
}

```

Program-8: Photo and Album Using Properties and Indexers

```
using System;
class Photo
{
    private string title;

    public Photo(string title)
    {
        this.title = title;
    }
    public string Title
    {
        get
        {
            return this.title;
        }
    }
}

class Album
```



```

{

    private string[] alumb;
    int count = 0;
    //Photo p = new Photo();
    public Alumb(int size)
    {
        alumb = new string[size];
    }

    public string this[int i]
    {
        get
        {
            return alumb[i];
        }
    }

    public int this[string str]
    {
        get
        {
            //foreach(string i in alumb)
            //{
            //    alumb[count] = i;
            //}

            return Array.IndexOf(alumb, str);
        }
    }

    public void AddPhoto(String p)
    {

        alumb[count] = p;
        count++;
    }

}

class MainTest
{
    public static void Main()
    {
        Photo p1 = new Photo("Pramod");
        Photo p2 = new Photo("Dangu");

        Console.WriteLine(p1.Tittle);
        Console.WriteLine(p2.Tittle);

        Alumb alumb = new Alumb(10);

        //-----
        string str1 = p1.Tittle;
        alumb.AddPhoto(str1);

        string str2 = p2.Tittle;
        alumb.AddPhoto(str2);
    }
}

```

```

//-----

Console.WriteLine(albumb[0]);
Console.WriteLine(albumb["Pramod"]);

Console.WriteLine(albumb[1]);
Console.WriteLine(albumb["Dangu1"]);
}
}

```

Program-9 :Sum Of Two Complex Numbers

```

using System;
public class Complex //use either class or struct keyword
{
    public int real;
    public int imaginary;

    public Complex(int real, int imaginary)
    {
        this.real = real;
        this.imaginary = imaginary;
    }

    public static Complex operator +(Complex one, Complex two) //this method should be public and static
    {
        return new Complex(one.real + two.real, one.imaginary + two.imaginary);
    }

    public override string ToString()
    {
        return (String.Format("{0} + {1}i", real, imaginary));
    }
}

```

```
}
```

```
class Pramod
```

```
{
```

```
    public static void Main()
```

```
    {
```

```
        Complex comp1 = new Complex(5, 1);
```

```
        Complex comp2 = new Complex(6, 2);
```

```
        // Add both of them
```

```
        Complex res = comp1 + comp2;
```

```
        Console.WriteLine("First: {0}", comp1);
```

```
        Console.WriteLine("Second: {0}", comp2);
```

```
        // display the result
```

```
        Console.WriteLine("Result (Sum): {0}", res);
```

```
    }
```

```
}
```

```
//First: 7 + 1i
```

```
//Second: 2 + 6i
```

```
//Result (Sum): 9 + 7i
```