**C# ASSIGNMENT NO: 1**

1. Write a Simple console Application Calculator with the help of Visual Studio .NET IDE which will perform following operations on two numbers:
   1. Addition.
   2. Subtraction.
   3. Multiplication.
   4. Division

**CODE:**

using System;

namespace CSharpAssignment1

{

class Calculator

{

static void Main()

{

int num1, num2, choice, result;

Console.WriteLine("Calculator Program");

Loop:

Console.Write("Enter 1st number: ");

num1 = Convert.ToInt32(Console.ReadLine());

Console.Write("Enter 2nd number: ");

num2 = Convert.ToInt32(Console.ReadLine());

Console.WriteLine($"The two numbers are: {num1} and {num2}");

Console.WriteLine("What operation would you like to perform? \n 1. Addition \n 2. Subtraction \n 3. Multiplication \n 4. Division");

Console.Write("Enter your choice in number (1,2,3 or 4):");

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

switch (choice)

{

case 1:

result = num1 + num2;

Console.WriteLine($"Addition: {result}");

break;

case 2:

result = num1 - num2;

Console.WriteLine($"Subtraction: {result}");

break;

case 3:

result = num1 \* num2;

Console.WriteLine($"Multiplication: {result}");

break;

case 4:

if (num2 ==0)

{

Console.WriteLine($"Exception: Cannot divide {num1}/{num2}");

break;

}

else {

result = num1 / num2;

Console.WriteLine($"Division: {result}");

break;

}

default:

Console.WriteLine("Enter valid option!");

break;

}

Console.Write("Do you want to exit? (Y/N):");

char choose = Convert.ToChar(Console.ReadLine());

if ( choose == 'N' || choose == 'n')

goto Loop;

else

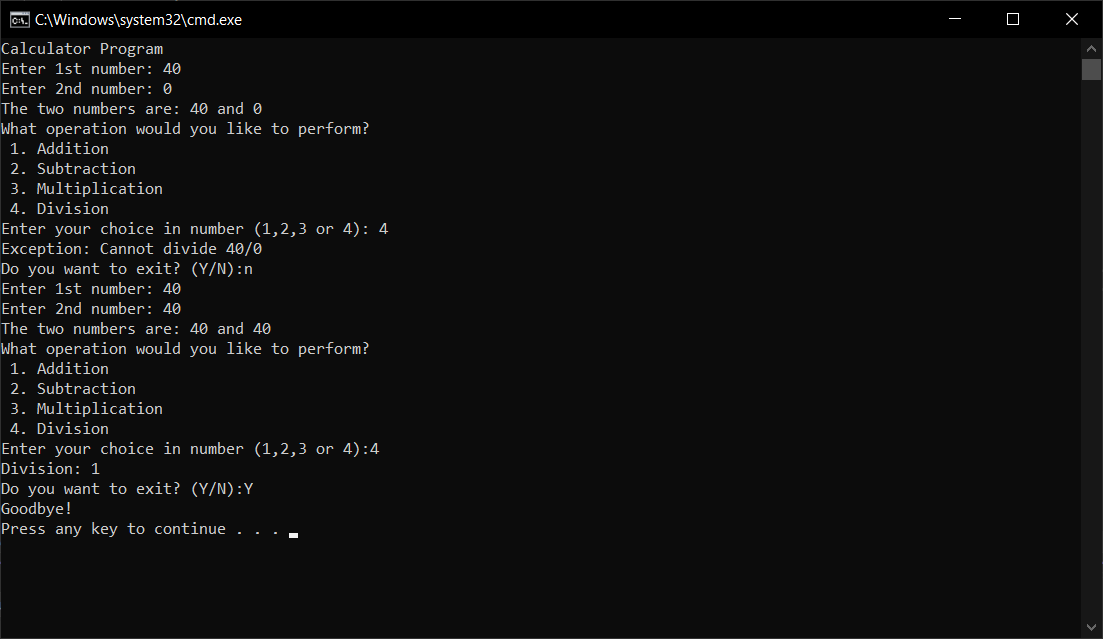
Console.WriteLine("Goodbye!");

}

}

}

**OUTPUT:**



Accept input from user and display results on console. Make use of loops, switch case wherever required.

1. Accept average marks of five students. Display the highest marks obtained.

**CODE:**

using System;

namespace CSharpAssignment1

{

class averageMarks

{

public static void Main()

{

int[] marks = new int[5];

int highestAverage = 0;

Console.WriteLine("Enter average marks for 5 students:");

for (int i = 0; i < marks.Length; i++)

{

Console.Write($"Enter avg marks for {i + 1} student:");

marks[i] = Convert.ToInt32(Console.ReadLine());

}

Console.WriteLine("Average Marks entered for 5 students are :");

foreach (int mark in marks)

{

Console.Write($"{mark} ");

}

for (int i = 0; i < marks.Length; i++)

{

if (marks[i] > highestAverage)

{

highestAverage = marks[i];

}

}

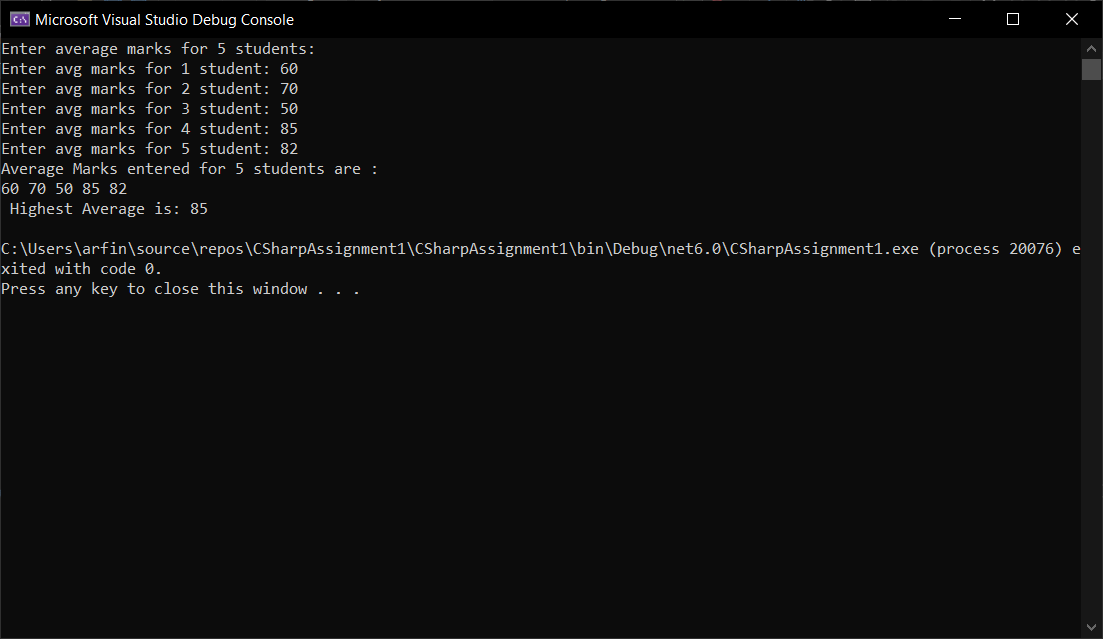
Console.WriteLine($"\n Highest Average is: {highestAverage}");

}

}

}

**OUTPUT:**



1. Write a static method to accept param array of integers. The method should find the sum of all the integers passed and display the result. Write a client program to call the method.

CODE:

using System;

namespace CSharpAssignment1

{

class staticParamSum

{

public static void Sum(params int[] ints) //Static method

{

int sum = 0;

foreach (int num in ints)

{

sum += num;

}

Console.WriteLine("Sum of " + ints.Length + $" numbers: {sum}");

}

public static void Main()

{

int length;

Loop:

Console.Write("Enter length of the array: ");

length = Convert.ToInt32(Console.ReadLine());

try

{

if (length == 0 || length < 0)

throw new IndexOutOfRangeException("Length of array lesser than or equal to zero.\n Enter appropriate length of array.");

}

catch

{

Console.WriteLine("Length of array lesser than or equal to zero.\nEnter appropriate length of array.");

goto Loop;

}

Console.WriteLine($"Enter {length} elements for array:");

int[] ints = new int[length];

for (int i = 0; i < ints.Length; i++)

{

ints[i] = Convert.ToInt32(Console.ReadLine());

}

Console.Write("Array elements: { ");

for (int i = 0; i < ints.Length; i++)

{

if (i == ints.Length-1)

Console.Write(ints[i] + " }\n\n");

else

Console.Write(ints[i] + ", ");

}

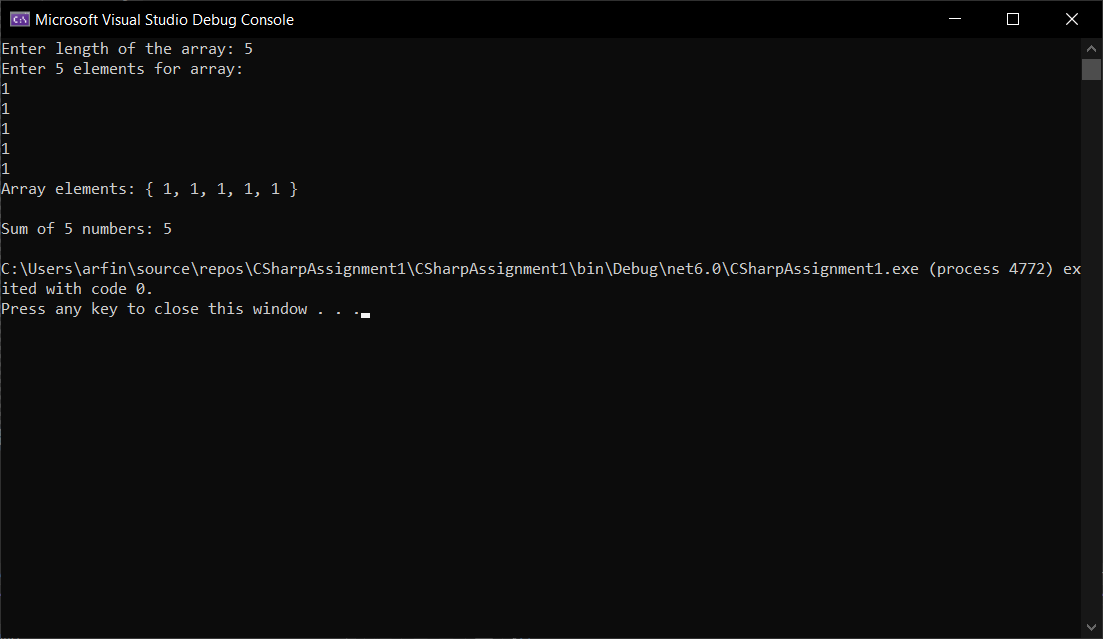
Sum(ints); // Calling static method in main.

}

}

}

OUTPUT:



1. Write a method to swap two integers. The client code should call the method and print the swapped value.

CODE:

using System;

namespace CSharpAssignment1

{

class Swapped

{

void swapped(int num1, int num2)

{

int c;

c = num2;

num2 = num1;

num1 = c;

Console.WriteLine($"\nNumbers after swapping: {num1} and {num2}");

}

public static void Main()

{

int a, b;

Console.WriteLine("Swapping Numbers Program");

Console.Write("Enter 1st number:");

a = Convert.ToInt32(Console.ReadLine());

Console.Write("Enter 2nd number:");

b = Convert.ToInt32(Console.ReadLine());

Console.WriteLine($"Numbers before swapping: {a} and {b}");

Swapped s = new Swapped();

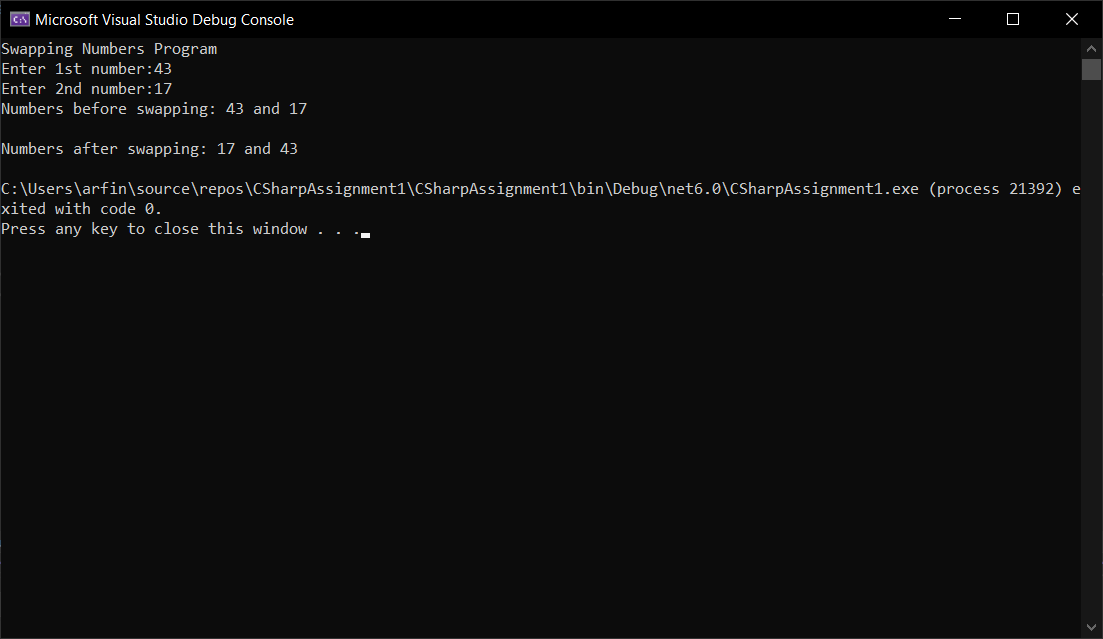
s.swapped(a, b);

}

}

}

OUTPUT:



1. Write a single method that calculates the area and circumference of the circle. The area and circumference should be displayed through the client code

CODE:

using System;

namespace CSharpAssignment1

{

class Circle

{

static void CircleDimensions(int r, out double area, out double circumference)

{

area = 3.142 \* r \* r;

circumference = 2 \* 3.142 \* r;

}

public static void Main()

{

int radius;

double area, circumference;

Console.Write("Enter radius for circle: ");

radius = Convert.ToInt32(Console.ReadLine());

CircleDimensions(radius,out area,out circumference);

Console.WriteLine($"Radius : {radius}");

Console.WriteLine($"Area : {area} sq. units");

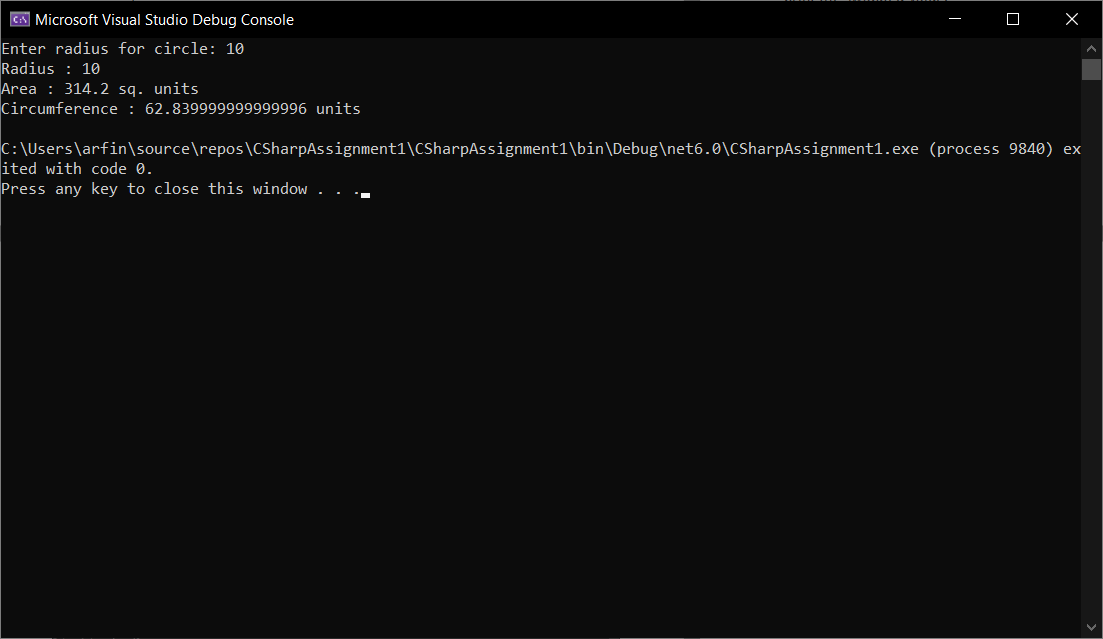
Console.WriteLine($"Circumference : {circumference} units");

}

}

}

OUTPUT:



1. Create a structure Book which contains the following members:

***bookId, title, price, bookType***

Type of the book should an enumerated data type with values as ***Magazine, Novel, ReferenceBook, Miscellaneous***. Write a console based application to do the following tasks.

* 1. Accept the details of the book
  2. Display the details of the book. The type of book should be displayed as a string e.g.:

Magazine

Note: Use methods for accepting and displaying details.

**CODE:**

using System;

namespace CSharpAssignment1

{

struct Book

{

public string bookId;

public string title;

public string price;

public enum bookType

{

Magazine = 1,

Novel = 2,

ReferenceBook = 3,

Miscellaneous = 4

}

public void getValue(string id, string t, string p, bookType type)

{

bookId = id;

title = t;

price = p;

string bookType = type.ToString();

Console.WriteLine(bookType);

Console.WriteLine($"Book Id: {bookId} \t Book Title: {title}\t Book Price: {price}\t Book Type: {bookType}");

}

}

class BookApplication

{

public static void Main()

{

Book book = new Book();

string bookId, title, price;

Console.WriteLine("Welcome to Book Application");

Console.Write("Enter Book Id: ");

bookId = Console.ReadLine();

Console.Write("Enter Book Title: ");

title = Console.ReadLine();

Console.Write("Enter Book price: ");

price = Console.ReadLine();

Loop:

Console.Write("Choose your book type:\n 1. Magazine\t 2. Novel\t 3. Reference Book\t 4. Miscellaneous\n Enter your choice in number (1,2,3 or 4):");

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

switch (choice)

{

case 1:

book.getValue(bookId, title, price, Book.bookType.Magazine);

break;

case 2:

book.getValue(bookId, title, price, Book.bookType.Novel);

break;

case 3:

book.getValue(bookId, title, price, Book.bookType.ReferenceBook);

break;

case 4:

book.getValue(bookId, title, price, Book.bookType.Miscellaneous);

break;

default:

Console.WriteLine("Invalid Book Type. PLease select from the given options");

goto Loop;

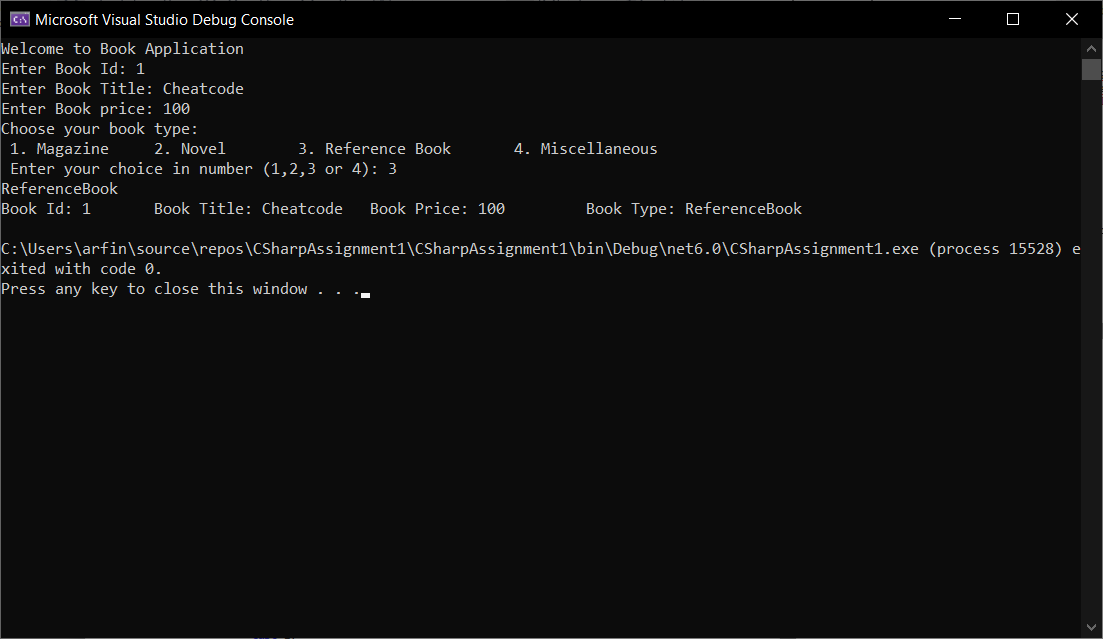
}

}

}

}

**OUTPUT:**



**C# ASSIGNMENT NO: 2**

**OBJECT ORIENTED PROGRAMMING**

1. Develop Employee Management System for Litware Organization. Write a Class Library project LitwareLib.
   1. Add class Employee with following private members:
      * EmpNo int
      * EmpName string
      * Salary double
      * HRA double
      * TA double
      * DA double
      * PF double
      * TDS double
      * NetSalary double
      * GrossSalary double.

Write methods for accepting EmpNo, EmpName and Salary. HRA, TA, DA, PPF, TDS, NET, GROSS should be calculated automatically. Follow the table for calculations.

|  |  |  |  |
| --- | --- | --- | --- |
| Salary | HRA % of Salary | TA % of Salary | DA % of Salary |
| <5000 | 10 | 5 | 15 |
| <10000 | 15 | 10 | 20 |
| <15000 | 20 | 15 | 25 |
| <20000 | 25 | 20 | 30 |
| >=20000 | 30 | 25 | 35 |

## GrossSalary = Salary + HRA + TA + DA.

Calculate PF, TDS and Net salary in a function named “CalculateSalary()”

## PF = 10 % of GrossSalary.

## TDS = 18 % of GrossSalary.

NetSalary = GrossSalary – (PF + TDS).

**CODE:**

**Main Console Application:**

using System;

using LitwareLib;

class Program

{

public static void Main()

{

int empid;

string empname;

double salary;

Console.WriteLine("Welcome to Litware Organization");

Console.Write("Enter your Employee ID: ");

empid = Convert.ToInt32(Console.ReadLine());

Console.Write("Enter your Name: ");

empname = Console.ReadLine();

Console.Write("Enter your Salary: ");

salary = Convert.ToDouble(Console.ReadLine());

Employee employee = new Employee();

employee.GetEmployeeData(empid, empname, salary);

employee.CalculateSalary();

}

}

**LitwareLib.dll**

using System;

namespace LitwareLib

{

public class Employee

{

// Defining data members

private int Empno;

private string EmpName;

private double Salary, HRA, TA, DA, PF, TDS, NetSalary, GrossSalary;

// Constructor initializes variables to zero.

public Employee()

{

Empno = 0; Salary = 0; HRA = 0; TA = 0; DA = 0;

EmpName = "Employee Name";

}

// Getting information about Employee

public void GetEmployeeData(int Empno, string EmpName, double Salary)

{

this.Empno = Empno;

this.EmpName = EmpName;

this.Salary = Salary;

Console.WriteLine($"Employee Details:\nID: {this.Empno}\tName: {this.EmpName}!\n");

CalculateGrossSalary();

}

// Calculating GrossSalary

public void CalculateGrossSalary()

{

// Calculating various HRA, TA, DA.

if (this.Salary < 5000)

{

HRA = (10 \* this.Salary) / 100;

TA = (5 \* this.Salary) / 100;

DA = (15 \* this.Salary) / 100;

GrossSalary = Salary + HRA + TA + DA;

Console.Write($"Gross Salary: {GrossSalary}");

}

else if (this.Salary < 10000)

{

HRA = (15 \* this.Salary) / 100;

TA = (10 \* this.Salary) / 100;

DA = (20 \* this.Salary) / 100;

GrossSalary = Salary + HRA + TA + DA;

Console.Write($"Gross Salary: {GrossSalary}");

}

else if (this.Salary < 15000)

{

HRA = (20 \* this.Salary) / 100;

TA = (15 \* this.Salary) / 100;

DA = (25 \* this.Salary) / 100;

GrossSalary = Salary + HRA + TA + DA;

Console.Write($"Gross Salary: {GrossSalary}");

}

else if (this.Salary < 20000)

{

HRA = (25 \* this.Salary) / 100;

TA = (20 \* this.Salary) / 100;

DA = (30 \* this.Salary) / 100;

GrossSalary = Salary + HRA + TA + DA;

Console.Write($"Gross Salary: {GrossSalary}");

}

else if (this.Salary >= 20000)

{

HRA = (30 \* this.Salary) / 100;

TA = (25 \* this.Salary) / 100;

DA = (35 \* this.Salary) / 100;

GrossSalary = Salary + HRA + TA + DA;

Console.Write($"Gross Salary: {GrossSalary}");

}

}

//Calculating NetSalary

public void CalculateSalary()

{

PF = (10 \* GrossSalary) / 100;

TDS = (18 \* GrossSalary) / 100;

NetSalary = GrossSalary - (PF + TDS);

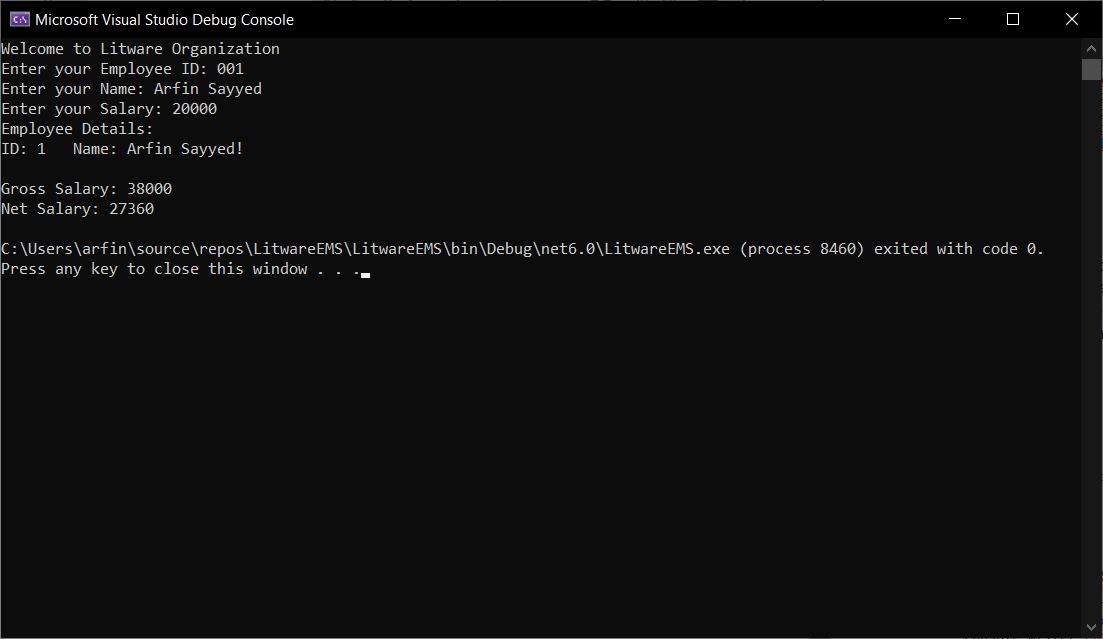
// Console.WriteLine($"\nPF: {PF} \t TDS: {TDS}");

Console.WriteLine($"\nNet Salary: {NetSalary}");

}

}

}

OUTPUT:

e) Write a console application Employee Management which allow HR staff member to register newly joined employee with EmpNo, EmpName and Salary. Display gross salary of employee on console. LitwareLib class Library will be used in Test console application for creating objects and invoking functionality of Employee class. Use Exception Handling mechanism wherever necessary.

**CODE:**

using System;

using LitwareLib;

class Test

{

public static void Main()

{

int EmpNo;

string EmpName;

double Salary;

Employee NewEmployee = new Employee();

Console.WriteLine("Welcome to Employee Managament Tool for HR");

Console.Write("Enter New Employee's ID: ");

EmpNo = Convert.ToInt32(Console.ReadLine());

Console.Write("Enter New Employee's Name: ");

EmpName = Console.ReadLine();

Console.Write("Enter Salary: ");

Salary = Convert.ToDouble(Console.ReadLine());

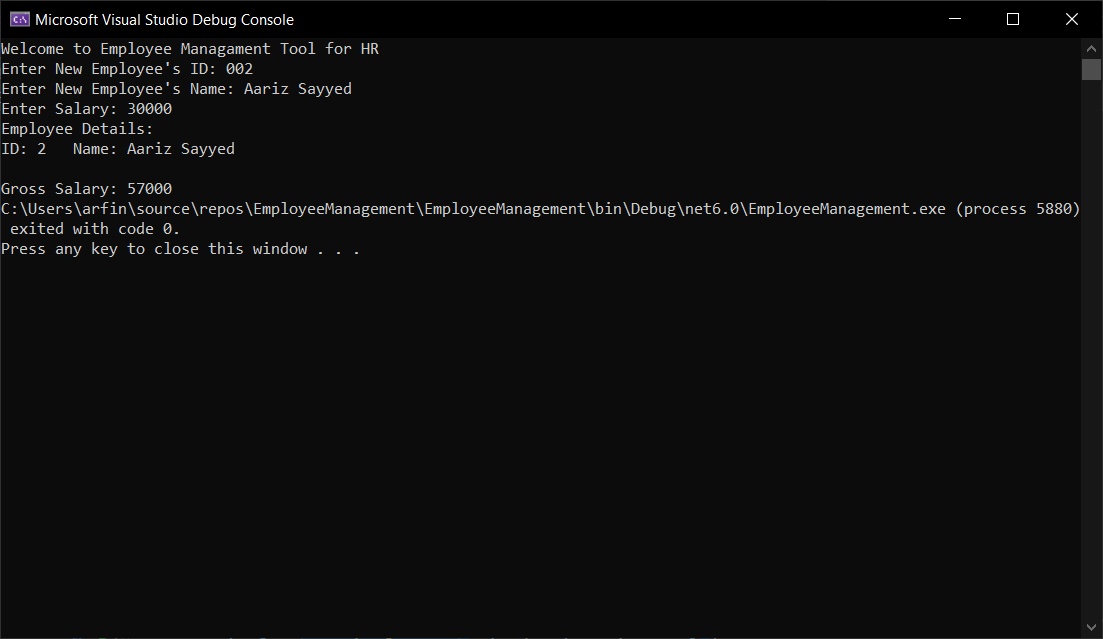
Employee employee = new Employee();

employee.GetEmployeeData(EmpNo, EmpName, Salary);

}

}

**OUTPUT:**



**C# ASSIGNMENT NO: 3 INHERITANCE AND POLYMORPHISM**

## Assignments to be done in this session

1. Create a hierarchy of Employee, Manager, MarketingExecutive in Employee Management System. They should have the following functionality.
   1. Manager with following private members.
      * Petrol Allowance: 8 % of Salary.
      * Food Allowance : 13 % of Salary.
      * Other Allowances : 3% of Salary.

Calculate GrossSalary by adding above allowances. Override CalculateSalary() method to calculate Net Salary. NetSalary. PF calculation should not consider above allowances.

* 1. MarketingExecutive with following private members.
     + Kilometer travel
     + Tour Allowances : Rs 5/- per Kilometer (Automatically generated).
     + Telephone Allowances : Rs.1000/-

Calculate GrossSalary by adding above allowances. Override CalculateSalary(). NetSalary,PF calculation should not consider above allowances.

**CODE:**

**Employee Management**

using System;

using LitwareLib;

class Test

{

public static void Main()

{

int EmpNo;

string EmpName;

double Salary;

Console.WriteLine("Welcome to Employee Management System!");

Console.Write("Enter Employee Number: ");

EmpNo = Convert.ToInt32(Console.ReadLine());

Console.Write("Enter Employee Name: ");

EmpName = Console.ReadLine();

Console.Write("Enter Salary: ");

Salary = Convert.ToDouble(Console.ReadLine());

Console.Write("Are you an/a 1. Employee\t 2. Manager\t 3. Marketing Executive \nEnter your choice in Number: ");

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

switch (choice)

{

case 1:

Employee employee = new Employee();

employee.GetEmployeeData(EmpNo,EmpName,Salary);

employee.CalculateGrossSalary();

employee.CalculateSalary();

break;

case 2:

Manager manager = new Manager();

manager.GetEmployeeData(EmpNo, EmpName, Salary);

manager.ManagerGrossSalary();

manager.CalculateSalary();

break;

case 3:

MarketingExecutive me = new MarketingExecutive();

me.GetEmployeeData(EmpNo, EmpName, Salary);

Console.WriteLine("Enter Kilometer Travel allowed");

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

double TA = 5 \* km;

me.MEGrossSalary(TA);

me.CalculateSalary();

break;

default:

Console.WriteLine("Invalid choice");

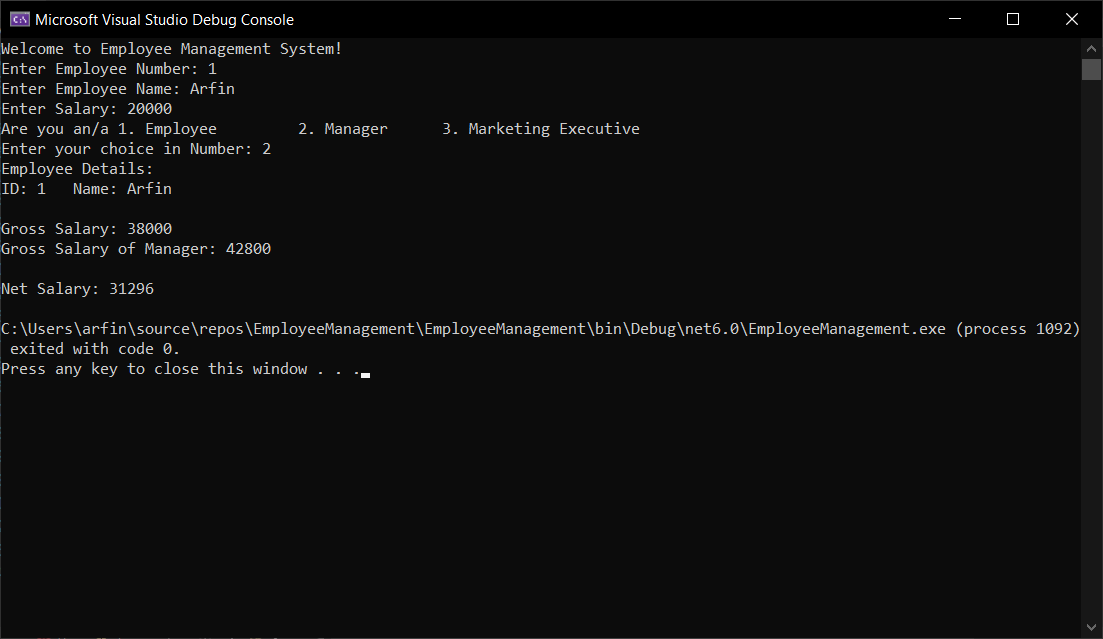
break;

}

}

}

**OUTPUT:**



Implement IPrintable interface for every Employee which will allow to print details of Employee on console.

Code:

LitwareLib.dll

using System;

namespace LitwareLib

{

interface IPrintable

{

void DisplayEmployee();

}

public class Employee : IPrintable

{

// Defining data members

private int Empno;

private string EmpName;

private double Salary,HRA, TA, DA, PF, TDS, NetSalary, GrossSalary;

public void DisplayEmployee()

{

Console.WriteLine("Using IPrintable to display Employee Details:");

Console.WriteLine($"ID: {Empno}\t Name: {EmpName} \tSalary: {Salary}");

Console.WriteLine($"Allowances: \nHouse Rent Alowance:{HRA}, Travel Allowance: {TA}, Dearness Allowance: {DA}");

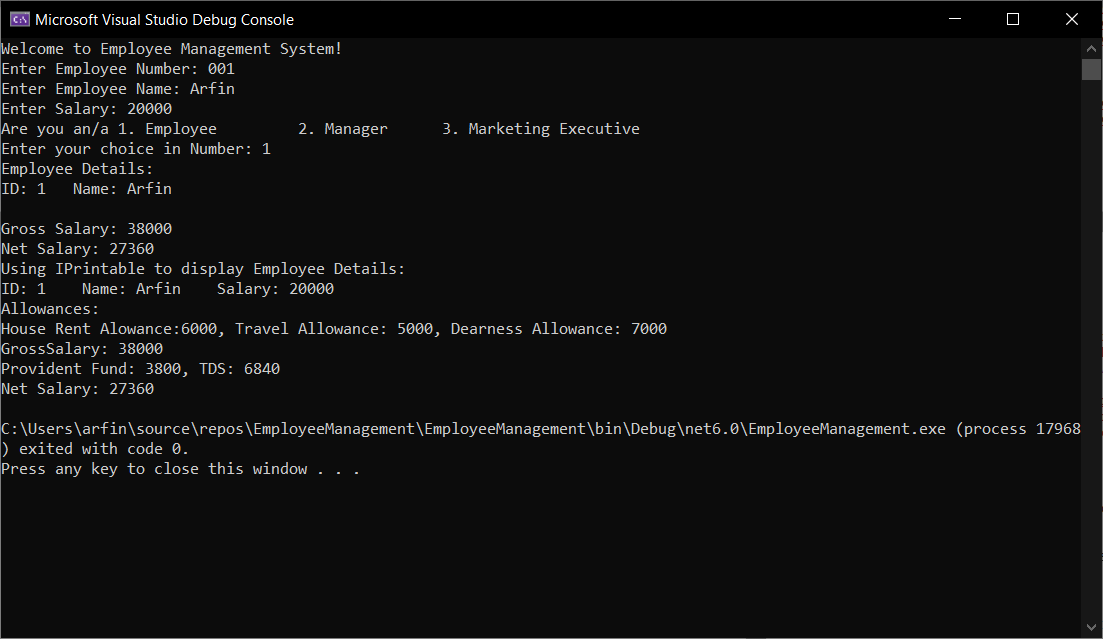
Console.WriteLine($"GrossSalary: {GrossSalary}");

Console.WriteLine($"Provident Fund: {PF}, TDS: {TDS}");

Console.WriteLine($"Net Salary: {NetSalary}");

}

OUTPUT:



1. Write a class called MyStack with following members.
   1. integer array
   2. integer variable to store top position
   3. size of the array.

Implement Push() and Pop() operation. Implement ICloneable interface to perform cloning. Write a client application to perform cloning.

**CODE:**

using System;

namespace CSharpAssigments

{

public class MyStack : ICloneable

{

public int[] array;

public int top, SizeOfArray;

public MyStack(int SizeOfArray)

{

this.SizeOfArray = SizeOfArray;

array = new int[this.SizeOfArray];

top = -1;

}

void Push(int x)

{

if (top < array.Length)

{

top++;

array[top] = x;

Console.WriteLine($"Pushed Element: {x}");

}

else

{

Console.WriteLine("Stack Overflow");

Environment.Exit(1);

}

}

void Pop()

{

if (top >= 0)

{

int x = array[top];

array[top] = 0;

--top;

Console.WriteLine($"Popped Element: {x}");

}

else

{

Console.WriteLine("Stack Underflow");

Environment.Exit(1);

}

}

void ViewStack()

{

foreach(int element in array)

{

Console.WriteLine(element);

}

}

public object Clone()

{

var stack = (MyStack)MemberwiseClone();

return stack;

}

public static void Main()

{

Console.Write("Enter size of the array: ");

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

MyStack stack = new MyStack(SizeofArray);

Loop:

Console.WriteLine("What action would you like to perform?\n1. Push \t2.Pop \t\t3. View Stack");

Console.Write("Enter your choice in number: ");

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

switch (choice)

{

case 1:

Console.Write("Enter element to be pushed: ");

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

stack.Push(num);

break;

case 2:

stack.Pop();

break;

case 3:

stack.ViewStack();

break;

default:

Console.WriteLine("Invalid input!");

goto Loop;

break;

}

Console.Write("Do you wish to continue? (Y/N): ");

char wish = Convert.ToChar(Console.ReadLine());

if(wish == 'y' || wish == 'Y')

{

goto Loop;

}

else if(wish == 'n' || wish == 'N')

{

Console.WriteLine("Stack: ");

stack.ViewStack();

Console.WriteLine("Thanks!");

}

else

{

Console.WriteLine("Invalid Input!");

}

Console.WriteLine("Cloned stack using Clone from ICloneable.");

var cloned = (MyStack)stack.Clone();

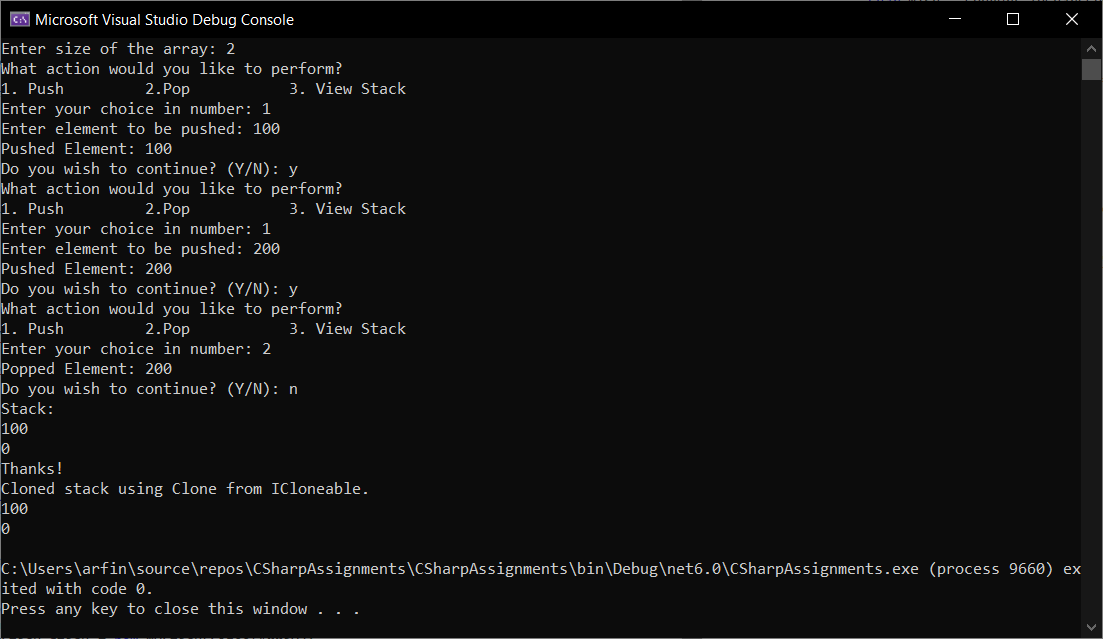
cloned.ViewStack();

}

}

}

**OUTPUT:**



1. Create a custom exception class named StackException. The Push()and Pop() method should throw object of StackException when the stack is full or empty respectively.

CODE: using System;

namespace CSharpAssigments

{

public class MyStack : ICloneable

{

public int[] array;

public int top, SizeOfArray;

public MyStack(int SizeOfArray)

{

this.SizeOfArray = SizeOfArray;

array = new int[this.SizeOfArray];

top = -1;

}

void Push()

{

if (top < array.Length-1)

{

Console.Write("Enter element to be pushed: ");

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

++top;

array[top] = num;

Console.WriteLine($"Pushed Element: {num}");

}

else

{

throw (new StackException ("Stack Overflow: Stack is full, Can't Push!"));

/\*Console.WriteLine("Stack Overflow");

Environment.Exit(1);\*/

}

}

void Pop()

{

try

{

if (top >= 0)

{

int x = array[top];

array[top] = 0;

--top;

Console.WriteLine($"Popped Element: {x}");

}

else

{

/\* Console.WriteLine("Stack Underflow");

Environment.Exit(1); \*/

throw (new StackException("Stack Underflow: Stack is Empty, Can't Pop!"));

}

}

catch (StackException exception)

{

Console.WriteLine(exception);

}

}

void ViewStack()

{

foreach(int element in array)

{

Console.WriteLine(element);

}

}

public object Clone()

{

var stack = (MyStack)MemberwiseClone();

return stack;

}

public static void Main()

{

Console.Write("Enter size of the array: ");

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

MyStack stack = new MyStack(SizeofArray);

Loop:

Console.WriteLine("What action would you like to perform?\n1. Push \t2.Pop \t\t3. View Stack");

Console.Write("Enter your choice in number: ");

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

switch (choice)

{

case 1:

stack.Push();

break;

case 2:

stack.Pop();

break;

case 3:

stack.ViewStack();

break;

default:

Console.WriteLine("Invalid input!");

goto Loop;

break;

}

Console.Write("Do you wish to continue? (Y/N): ");

char wish = Convert.ToChar(Console.ReadLine());

if(wish == 'y' || wish == 'Y')

{

goto Loop;

}

else if(wish == 'n' || wish == 'N')

{

Console.WriteLine("Stack: ");

stack.ViewStack();

Console.WriteLine("Thanks!");

}

else

{

Console.WriteLine("Invalid Input!");

}

Console.WriteLine("Cloned stack using Clone from ICloneable.");

var cloned = (MyStack)stack.Clone();

cloned.ViewStack();

}

}

public class StackException : Exception

{

public StackException(string message) : base(message)

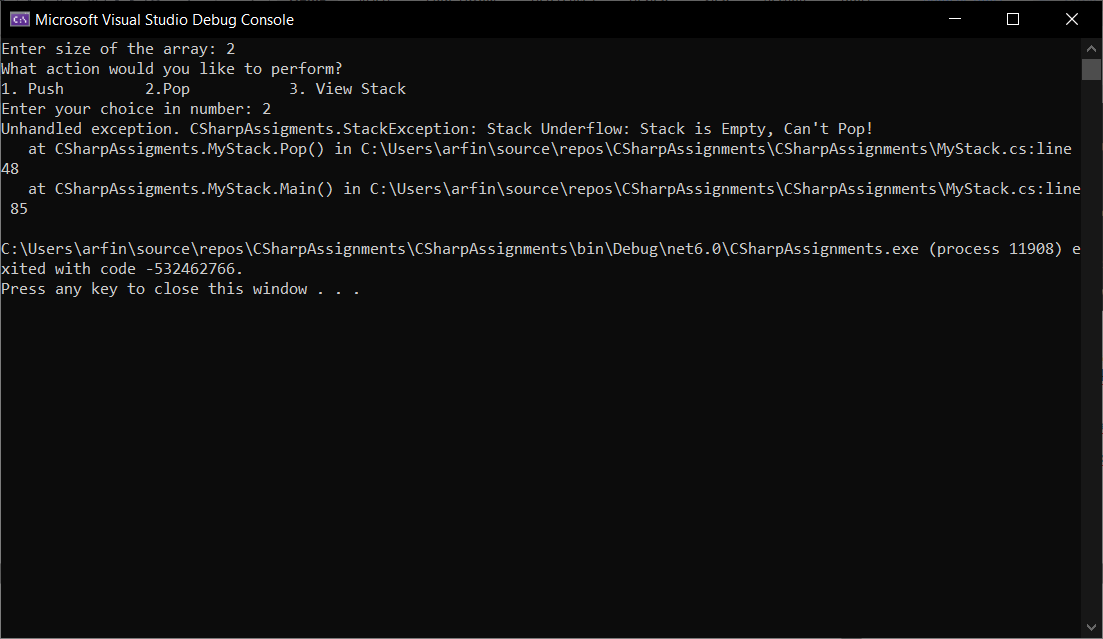
{

}

}

}

OUTPUT:



**C# ASSIGNMENT NO: 4**

1. Create a custom exception class named StackException. The Push()and Pop() method should throw object of StackException when the stack is full or empty respectively.

CODE:

using System;

namespace CSharpAssigments

{

public class MyStack : ICloneable

{

public int[] array;

public int top, SizeOfArray;

public MyStack(int SizeOfArray)

{

this.SizeOfArray = SizeOfArray;

array = new int[this.SizeOfArray];

top = -1;

}

void Push()

{

if (top < array.Length-1)

{

Console.Write("Enter element to be pushed: ");

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

++top;

array[top] = num;

Console.WriteLine($"Pushed Element: {num}");

}

else

{

throw (new StackException ("Stack Overflow: Stack is full, Can't Push!"));

/\*Console.WriteLine("Stack Overflow");

Environment.Exit(1);\*/

}

}

void Pop()

{

try

{

if (top >= 0)

{

int x = array[top];

array[top] = 0;

--top;

Console.WriteLine($"Popped Element: {x}");

}

else

{

/\* Console.WriteLine("Stack Underflow");

Environment.Exit(1); \*/

throw (new StackException("Stack Underflow: Stack is Empty, Can't Pop!"));

}

}

catch (StackException exception)

{

Console.WriteLine(exception);

}

}

void ViewStack()

{

foreach(int element in array)

{

Console.WriteLine(element);

}

}

public object Clone()

{

var stack = (MyStack)MemberwiseClone();

return stack;

}

public static void Main()

{

Console.Write("Enter size of the array: ");

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

MyStack stack = new MyStack(SizeofArray);

Loop:

Console.WriteLine("What action would you like to perform?\n1. Push \t2.Pop \t\t3. View Stack");

Console.Write("Enter your choice in number: ");

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

switch (choice)

{

case 1:

stack.Push();

break;

case 2:

stack.Pop();

break;

case 3:

stack.ViewStack();

break;

default:

Console.WriteLine("Invalid input!");

goto Loop;

break;

}

Console.Write("Do you wish to continue? (Y/N): ");

char wish = Convert.ToChar(Console.ReadLine());

if(wish == 'y' || wish == 'Y')

{

goto Loop;

}

else if(wish == 'n' || wish == 'N')

{

Console.WriteLine("Stack: ");

stack.ViewStack();

Console.WriteLine("Thanks!");

}

else

{

Console.WriteLine("Invalid Input!");

}

Console.WriteLine("Cloned stack using Clone from ICloneable.");

var cloned = (MyStack)stack.Clone();

cloned.ViewStack();

}

}

public class StackException : Exception

{

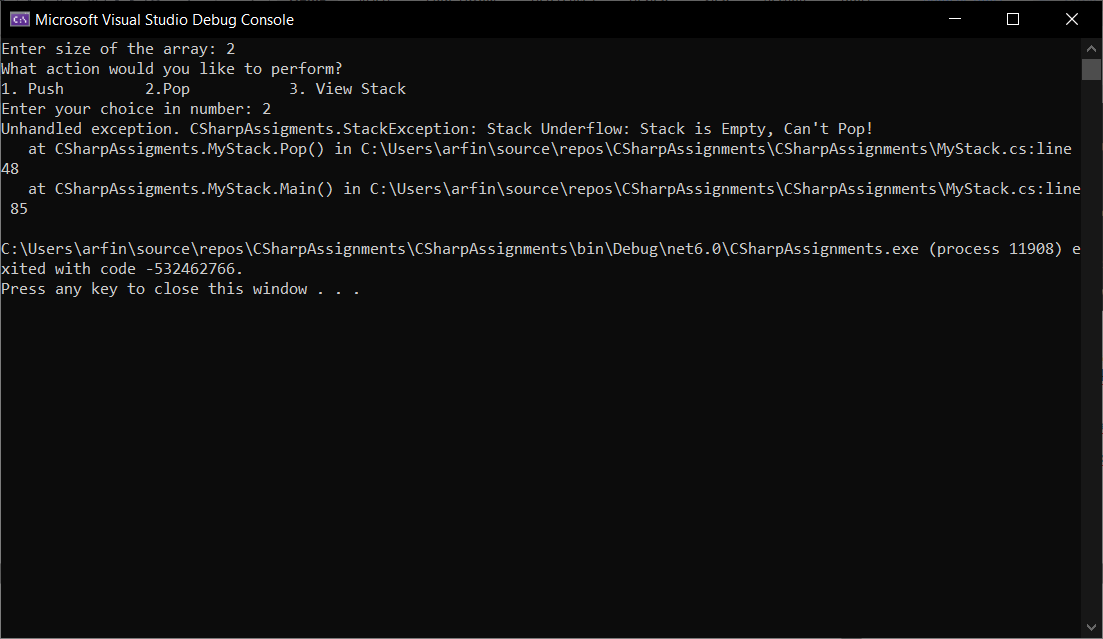
public StackException(string message) : base(message)

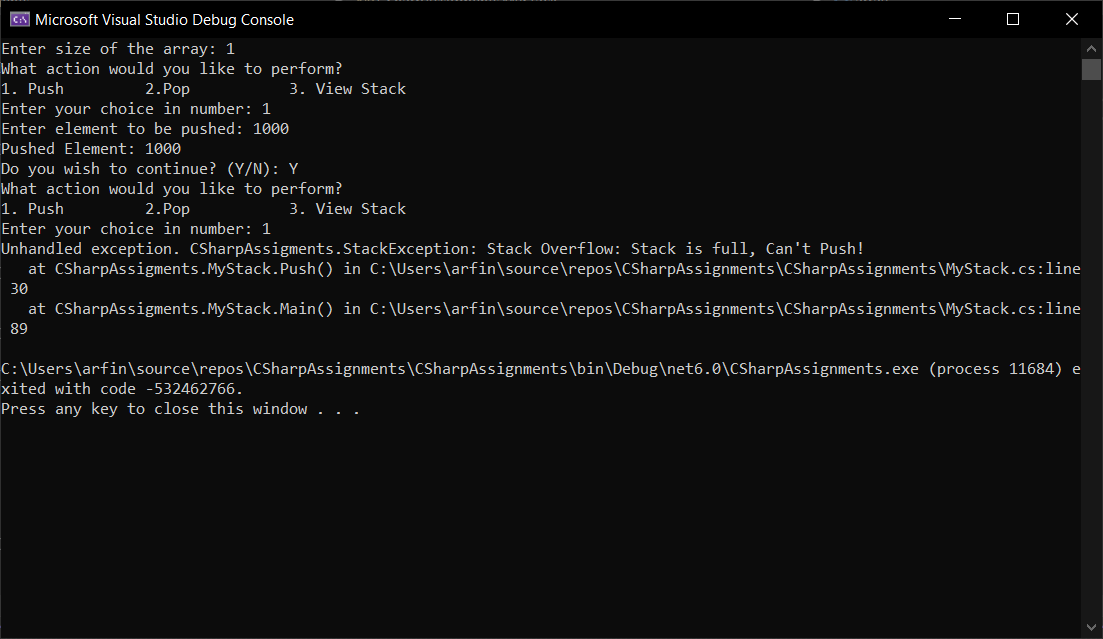
{

}

}

}





**C# ASSIGNMENT NO: 5**

1. **Create following types of arrays**
   1. **Integer**
   2. **String**

**Use System.Array class to perform following operations on them**

*Copy, Sort, Clear, Reverse*

**CODE:**

using System;

using System.Collections.Generic;

using System.Text;

namespace CSharpAssigments

{

class ArrayOperations

{

public void IntegerOperations()

{

Console.Write("Enter size of 1st array: ");

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

int[] arr = new int[size];

int[] arr2 = new int[size];

Console.WriteLine("Enter elements in the 1st array: ");

for (int i = 0; i < size; i++)

{

arr[i] = Convert.ToInt32(Console.ReadLine());

}

Console.WriteLine("1st Array Elements:");

foreach (int elements in arr)

{

Console.Write($"{elements} ");

}

// Copying array to another array.

Array.Copy(arr, arr2, size);

Console.WriteLine("\nElements copied from 1st to 2nd array are:");

foreach (int elements in arr2)

{

Console.Write($"{elements} ");

}

//Sorting elements in an Array.

Array.Sort(arr);

Console.WriteLine("\nSorted Elements: ");

foreach (int sortedElement in arr)

{

Console.Write($"{sortedElement} ");

}

//Clear Arrays

Array.Clear(arr, 2, 2);

Console.WriteLine("\nClear Elements: ");

foreach (int elements in arr)

{

Console.Write($"{elements} ");

}

//Reversing Array

Array.Reverse(arr);

Console.WriteLine("\nReversed Elements: ");

foreach (int b in arr)

{

Console.Write($"{b} ");

}

}

public void StringOperations()

{

Console.Write("\nEnter Size of an Array: ");

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

string[] str = new string[size];

string[] str2 = new string[size];

Console.WriteLine("Enter string elements in 1st array: ");

for (int i = 0; i < size; i++)

{

str[i] = Console.ReadLine();

}

Console.WriteLine("\nString Elements in 1st Array: ");

foreach (string elements in str)

{

Console.Write($"{elements} ");

}

Array.Copy(str, str2, size);

Console.WriteLine("\nCopying strings from 1st array to 2nd array: \nString Elements in 2nd Array ");

foreach (string elements in str2)

{

Console.Write($"{elements} ");

}

//Sorting String

Array.Sort(str);

Console.WriteLine("\nSort Strings:");

foreach (string a in str)

{

Console.Write($"{a} ");

}

Array.Clear(str, 1, 2);

Console.WriteLine("\nClear Strings:");

foreach (string n in str)

{

Console.Write($"{n} ");

}

Array.Reverse(str);

Console.WriteLine("\nReverse Strings:");

foreach (string b in str)

{

Console.WriteLine(b);

}

}

public static void Main(string[] args)

{

ArrayOperations MyArray = new ArrayOperations();

MyArray.IntegerOperations();

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

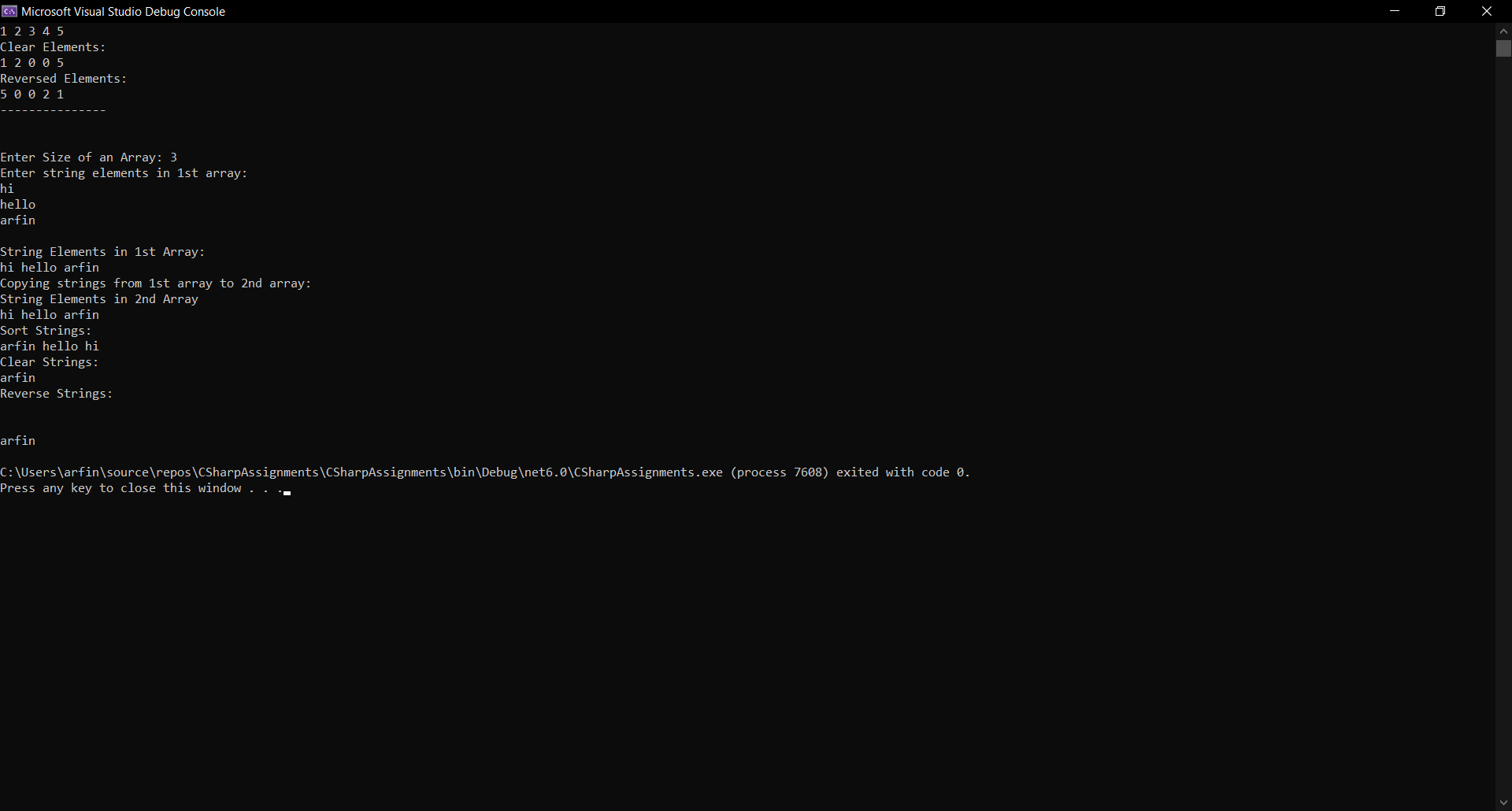
MyArray.StringOperations();

}

}

}

**OUTPUT:**



Accept input from user through Console.

1. Use collection class such as ArrayList to hold more than one employee objects in Employee Management application. Display all Employee details which are stored in collection.

CODE:

using System;

using System.Collections;

using LitwareLib;

namespace CSharpAssigments

{

public class EmployeeManagementApplication

{

public static void Main()

{

Employee emp1 = new Employee();

Employee emp2 = new Employee();

Employee emp3 = new Employee();

int id;

string name;

double salary;

ArrayList mylist = new ArrayList();

mylist.Add(emp1); mylist.Add(emp2); mylist.Add(emp3);

foreach(Employee employee in mylist)

{

Console.Write("Enter ID: ");

id = Convert.ToInt32(Console.ReadLine());

Console.Write("Enter Name: ");

name = Console.ReadLine();

Console.Write("Enter Salary: ");

salary = Convert.ToDouble(Console.ReadLine());

employee.GetEmployeeData(id,name,salary);

}

Console.WriteLine("Displaying employee details from Employee ArrayList - \n");

foreach(Employee employee in mylist)

{

Console.WriteLine($"ID: {employee.\_Empno} \t Name: {employee.\_EmpName} \t Salary: {employee.\_Salary}");

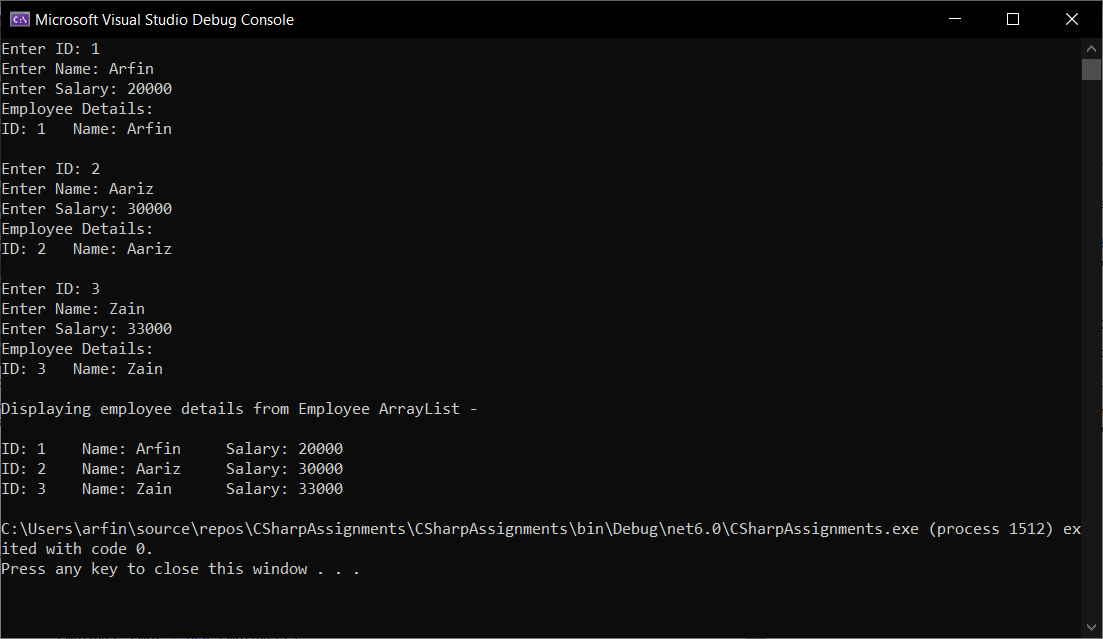
}

}

}

}

OUTPUT:



1. Write a console based program to create a linked list of Employee objects using the generic class List<>.Perform following operations on the list:
2. Add a new employee
3. Display the list of employees.
4. Total number of employees in the list

CODE:

using System;

using System.Collections.Generic;

using LitwareLib;

namespace CSharpAssignments

{

public class LinkedList

{

public static void Main() {

Employee emp1 = new Employee();

Employee emp2 = new Employee();

int id;

string name;

double salary;

List<Employee> EmployeeList = new List<Employee>();

Console.WriteLine("Performing Operation: Adding new employee to List.");

EmployeeList.Add(emp1); EmployeeList.Add(emp2);

foreach (Employee employee in EmployeeList)

{

Console.Write("Enter ID: ");

id = Convert.ToInt32(Console.ReadLine());

Console.Write("Enter Name: ");

name = Console.ReadLine();

Console.Write("Enter Salary: ");

salary = Convert.ToDouble(Console.ReadLine());

employee.GetEmployeeData(id, name, salary);

}

Console.WriteLine("\nPerforming Operation: Displaying the list of employees in List.");

foreach (Employee employee in EmployeeList)

{

Console.WriteLine($"ID: {employee.\_Empno} \t Name: {employee.\_EmpName} \t Salary: {employee.\_Salary}");

}

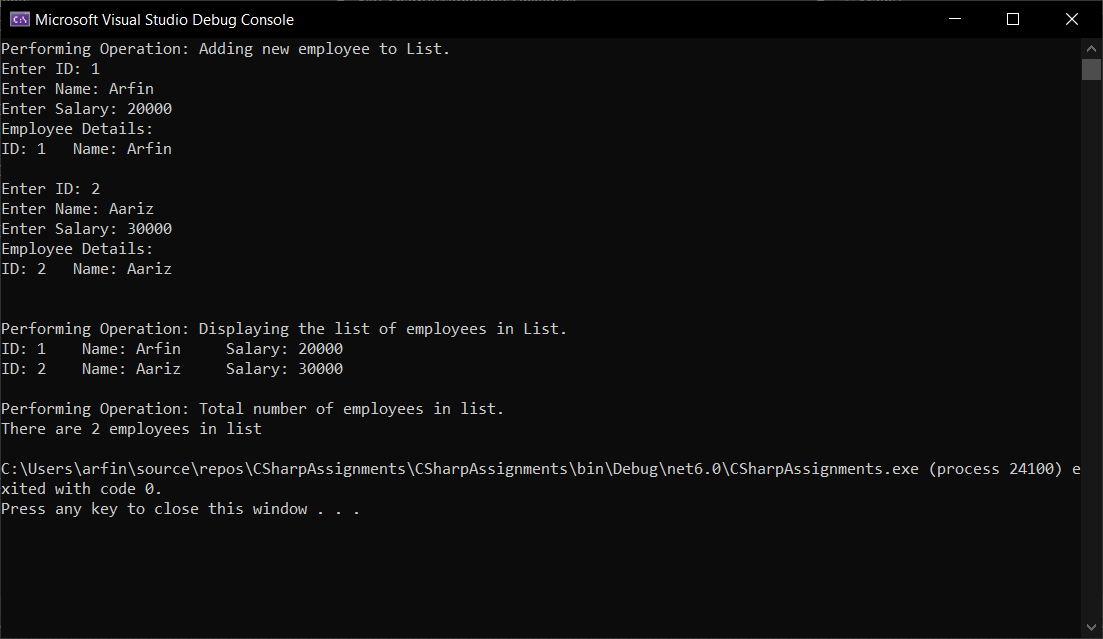
Console.WriteLine("\nPerforming Operation: Total number of employees in list.");

Console.WriteLine("There are " + EmployeeList.Count +" employees in list");

}

}

}

OUTPUT:

1. Write Custom Generic class MyStack based on assignment of previous session, with

Push() and Pop() methods to store any kind of .NET Type.

CODE:

using System;

using System.Collections.Generic;

using System.Linq;

namespace CSharpAssignments

{

using System;

namespace CSharpAssigments

{

public class MyStack<Type>

{

public Type[]? array;

public int top, SizeOfArray;

public MyStack(int SizeOfArray)

{

this.SizeOfArray = SizeOfArray;

array = new Type[this.SizeOfArray];

top = -1;

}

public void Push(Type num)

{

if (top < array.Length - 1)

{

++top;

array[top] = num;

Console.WriteLine($"Pushed Element: {num}");

}

else

{

throw (new StackException("Stack Overflow: Stack is full, Can't Push!"));

/\*Console.WriteLine("Stack Overflow");

Environment.Exit(1);\*/

}

}

public void Pop()

{

try

{

if (top >= 0)

{

Type x = array[top];

--top;

Console.WriteLine($"Popped Element: {x}");

}

else

{

throw (new StackException("Stack Underflow: Stack is Empty, Can't Pop!"));

}

}

catch (StackException exception)

{

Console.WriteLine(exception);

}

}

public void ViewStack()

{

foreach (Type element in array)

{

Console.WriteLine(element);

}

}

}

public class Program

{

public static void Main()

{

Console.Write("Enter size of the array: ");

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

MyStack<int> stack = new MyStack<int>(SizeofArray);

Loop:

Console.WriteLine("What action would you like to perform?\n1. Push \t2.Pop \t\t3. View Stack");

Console.Write("Enter your choice in number: ");

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

switch (choice)

{

case 1:

stack.Push(1);

break;

case 2:

stack.Pop();

break;

case 3:

stack.ViewStack();

break;

default:

Console.WriteLine("Invalid input!");

goto Loop;

break;

}

Console.Write("Do you wish to continue? (Y/N): ");

char wish = Convert.ToChar(Console.ReadLine());

if (wish == 'y' || wish == 'Y')

{

goto Loop;

}

else if (wish == 'n' || wish == 'N')

{

Console.WriteLine("Stack: ");

stack.ViewStack();

Console.WriteLine("Thanks!");

}

else

{

Console.WriteLine("Invalid Input!");

}

}

}

public class StackException : Exception

{

public StackException(string message) : base(message)

{

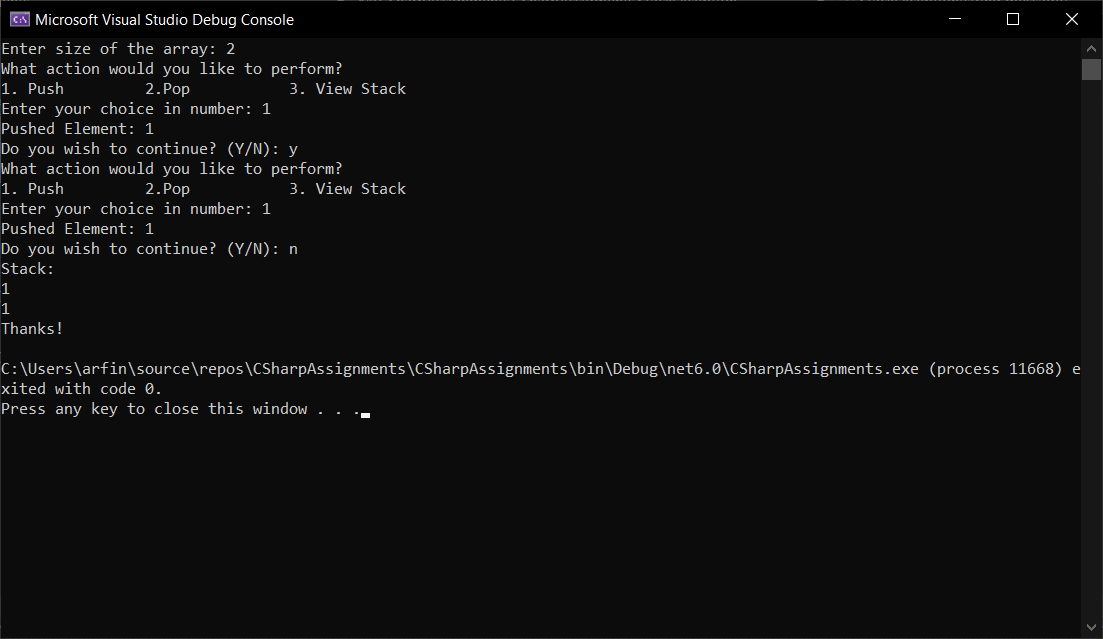
}

}

}

}

OUTPUT:



1. In the assignment 3 above, add a functionality to search an employee on name in the List<>.

Console.WriteLine("\n Searching name: Arfin in List");

var found = EmployeeList.FirstOrDefault(employee => employee.\_EmpName == "Arfin");

if (found != null)

{

Console.WriteLine("Element exists");

}

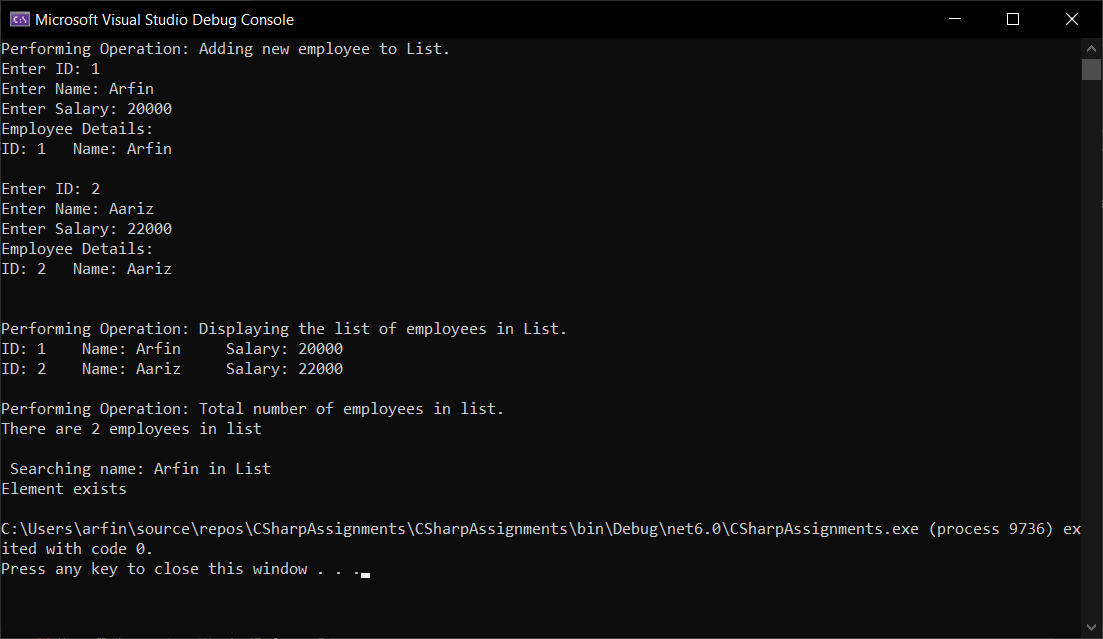
0.else

{

Console.WriteLine("Employee name doesn't exists");

}

OUTPUT:



1. Create a class named Player that contains Player name, runs scored as data members. Create a class named Team that contains an array of Player. Implement IEnumerable interface for class Team.

Write a console based application to create an object named India. Use foreach loop to iterate through the object India to display information about its players.

CODE:

using System;

using System.Collections;

namespace CSharpAssignments

{

class Player

{

public string Name { get; set; }

public int Run { get; set; }

public Player(string name, int run)

{

Name = name;

Run = run;

}

}

class Team : IEnumerable

{

private Player[] playerArray = new Player[4];

public Team()

{

playerArray[0] = new Player("Virat Kohli", 25);

playerArray[1] = new Player("MS Dhoni", 35);

playerArray[2] = new Player("KL Rahul", 29);

playerArray[3] = new Player("MD Shami", 34);

}

public IEnumerator GetEnumerator()

{

Console.WriteLine("Players in Team: ");

foreach (Player player in playerArray)

{

Console.WriteLine($"Name: {player.Name} Run: {player.Run}");

}

return playerArray.GetEnumerator();

}

}

public class MyProgram

{

public static void Main(string[] args)

{

Team India = new Team();

India.GetEnumerator();

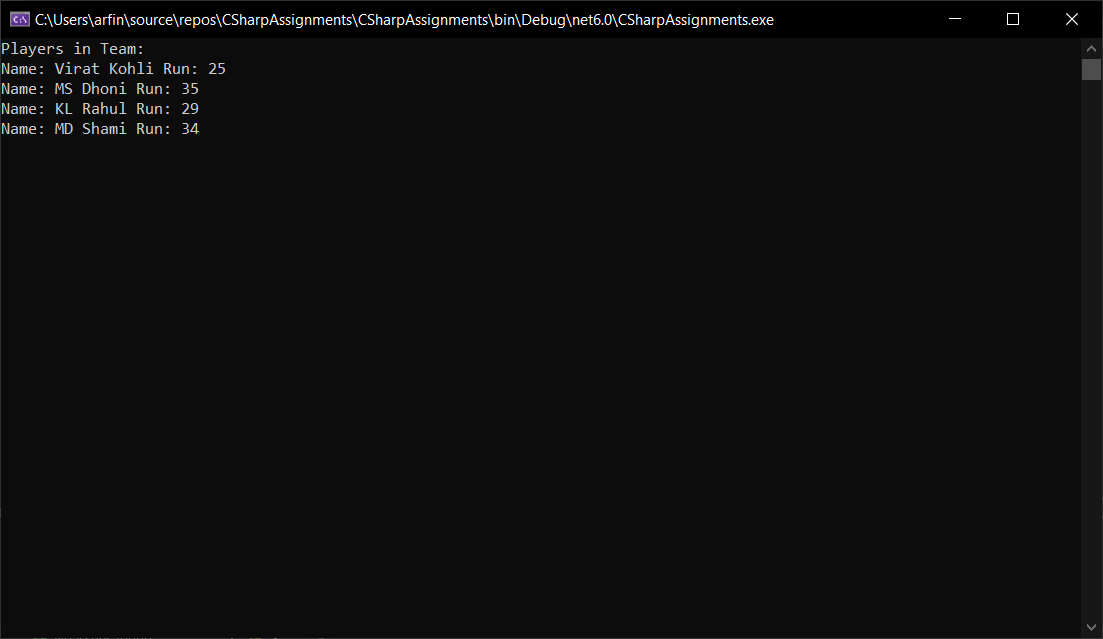
Console.ReadLine();

}

}

}

OUTPUT:



1. Use an iterator to iterate through the players in the above example.

CODE:

using System;

using System.Collections;

namespace CSharpAssignments

{

class Player

{

public string Name { get; set; }

public int Run { get; set; }

public Player(string name, int run)

{

Name = name;

Run = run;

}

}

class Team : IEnumerable

{

private Player[] playerArray = new Player[4];

public Team()

{

playerArray[0] = new Player("Virat Kohli", 25);

playerArray[1] = new Player("MS Dhoni", 35);

playerArray[2] = new Player("KL Rahul", 29);

playerArray[3] = new Player("MD Shami", 34);

}

public IEnumerator GetEnumerator()

{

Console.WriteLine("Players in Team: ");

foreach(Player player in playerArray)

{

yield return player;

}

}

}

public class MyProgram

{

public static void Main(string[] args)

{

Team India = new Team();

India.GetEnumerator();

foreach (Player player in India)

{

Console.WriteLine($"Name: {player.Name} Run: {player.Run}");

}

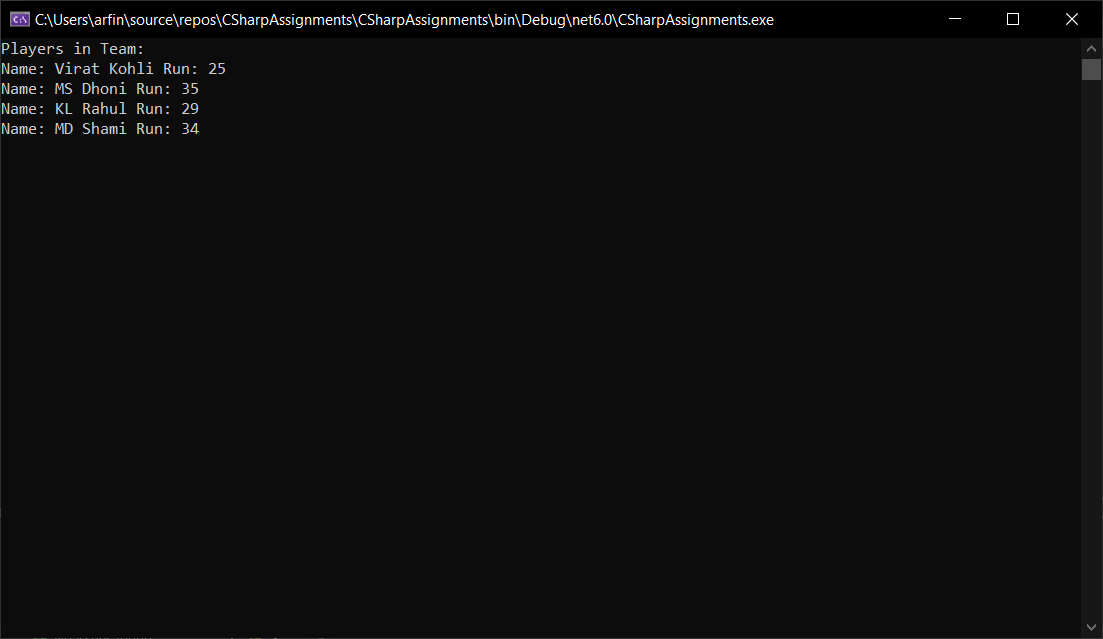
Console.ReadLine();

}

}

}

OUTPUT:



**C# ASSIGNEMNT NO: 6**

1. In Assignment 3 of previous session, print the details of Managers with the help of delegate

EmployeeDelegate. (UniCast Delegate)

CODE:

using System;

using LitwareLib;

public delegate void EmployeeDelegate();

class Test

{

public static void Main()

{

int EmpNo;

string EmpName;

double Salary;

Console.WriteLine("Welcome to Employee Management System!");

Console.Write("Enter Employee Number: ");

EmpNo = Convert.ToInt32(Console.ReadLine());

Console.Write("Enter Employee Name: ");

EmpName = Console.ReadLine();

Console.Write("Enter Salary: ");

Salary = Convert.ToDouble(Console.ReadLine());

Console.Write("Are you an/a 1. Employee\t 2. Manager\t 3. Marketing Executive \nEnter your choice in Number: ");

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

switch (choice)

{

case 1:

Employee employee = new Employee();

employee.GetEmployeeData(EmpNo,EmpName,Salary);

employee.CalculateGrossSalary();

employee.CalculateSalary();

employee.DisplayEmployee();

break;

case 2:

Manager manager = new Manager();

manager.GetEmployeeData(EmpNo, EmpName, Salary);

manager.ManagerGrossSalary();

manager.CalculateSalary();

manager.DisplayEmployee();

EmployeeDelegate mymanager = new EmployeeDelegate(manager.DisplayManager);

mymanager.Invoke();

break;

case 3:

MarketingExecutive me = new MarketingExecutive();

me.GetEmployeeData(EmpNo, EmpName, Salary);

Console.WriteLine("Enter Kilometer Travel allowed");

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

double TA = 5 \* km;

me.MEGrossSalary(TA);

me.CalculateSalary();

me.DisplayEmployee();

break;

default:

Console.WriteLine("Invalid choice");

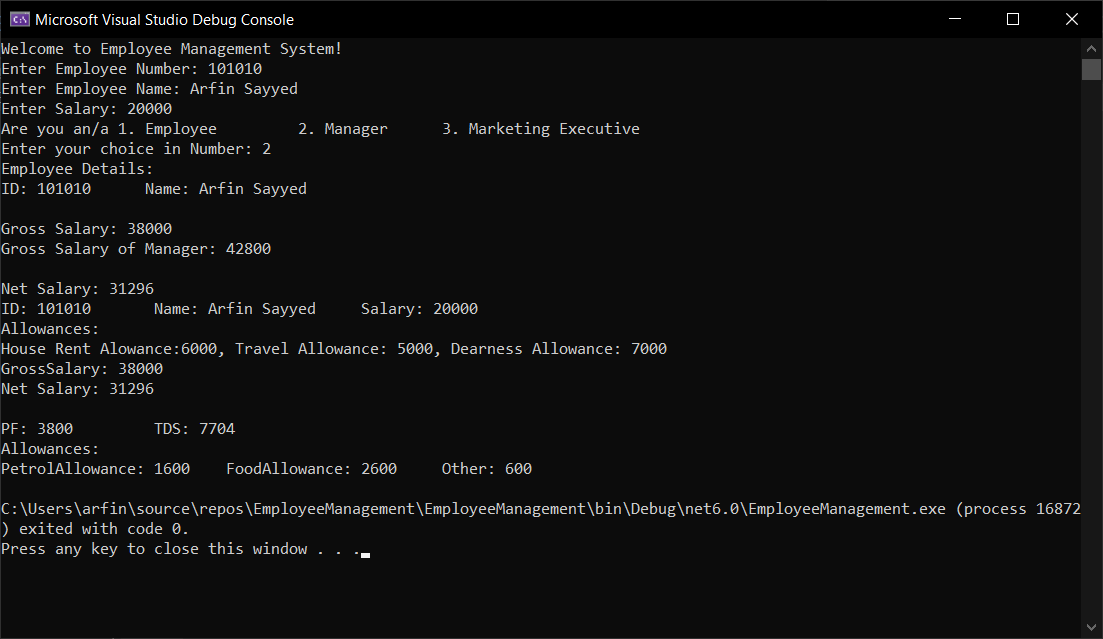
break;

}

}

}

OUTPUT:



1. Write a console Application for banking domain, having Account class with data members as account number, customer name, balance . It should have Withdraw and Deposit methods for performing banking transaction. It should also define UnderBalance, BalanceZero events. These events would be raised when balance of account is less than certain amount and equal to zero respectively.

CODE:using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace BankingDomainApplication

{

public class Account

{

private double \_AccountNumber;

private string \_CustomerName;

private double \_Balance;

public double AccountNumber

{

get { return \_AccountNumber; }

set { \_AccountNumber = value; }

}

public string CustomerName

{

get { return \_CustomerName; }

set { \_CustomerName = value; }

}

public double Balance

{

get { return \_Balance; }

set { \_Balance = value; }

}

public void getAccountHolderData(double AccountNumber, string CustomerName)

{

this.AccountNumber = AccountNumber;

this.CustomerName = CustomerName;

}

public void Withdraw(double amount)

{

double AmountToBeWithdrawn = amount;

if(amount <= Balance)

{

Balance = Balance - AmountToBeWithdrawn;

this.\_Balance = Balance;

Console.WriteLine($"Your balance has been debited with Rs. {AmountToBeWithdrawn}.");

}

else

{

Console.WriteLine("Transaction Failed! Amount to be withdrawn greater than Account Balance");

}

}

public void Deposit(double amount)

{

double AmountToBeDeposited = amount;

if(amount <= 0)

{

Console.WriteLine("Transaction Failed, Deposit amount greater than 0.");

}

else {

Balance = Balance + AmountToBeDeposited;

this.\_Balance = Balance;

Console.WriteLine($"Your balance has been credited with Rs. {AmountToBeDeposited}.");

}

}

public double DisplayBalance()

{

return Balance;

}

public static void Main()

{

Account AccountHolder = new Account();

Console.Write("Enter Account Number: ");

double AccountNumber = Convert.ToInt32(Console.ReadLine());

Console.Write("Enter Name: ");

string CustomerName = Console.ReadLine();

AccountHolder.getAccountHolderData(AccountNumber, CustomerName);

if (AccountHolder.Balance == 0)

{

Console.WriteLine($"Balance: {AccountHolder.Balance} \n");

Console.WriteLine("Kindly maintain 50 as minimum balance");

}

Loop:

Console.Write("What would you like to do? \n1. Deposit \t2. Withdraw \t3. View Balance \nEnter your choice: ");

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

switch (choice)

{

case 1:

Console.Write("Enter amount to be deposited: ");

double amount = Convert.ToDouble(Console.ReadLine());

AccountHolder.Deposit(amount);

break;

case 2:

Console.Write("Enter amount to be withdrawed: ");

double amountwithdrawn = Convert.ToDouble(Console.ReadLine());

AccountHolder.Withdraw(amountwithdrawn);

break;

case 3:

double balance = AccountHolder.DisplayBalance();

Console.Write($"Balance: {balance}");

break;

default:

Console.Write("Invalid Input! Try again later.");

break;

}

//Registering 2 events.

UnderBalanceClass info = new UnderBalanceClass();

info.UnderBalance += whenUnderBalance; //Registering an event.

ZeroBalanceClass info1 = new ZeroBalanceClass();

info1.ZeroBalance += whenZeroBalance; //Registering an event.

//Event Handlers

static void whenUnderBalance()

{

Console.WriteLine("Balance under the mentioned limit, Under Balance Event Raised!");

}

static void whenZeroBalance()

{

Console.WriteLine("Balance is Zero, Zero Balance Event Raised!");

}

Console.Write("\nDo you wish to perform more operations? (Y/N): ");

char wish = Convert.ToChar(Console.ReadLine());

if (wish == 'Y' || wish == 'y')

{

if (AccountHolder.Balance == 0)

{

info1.isZeroBalance();

}

else if (AccountHolder.Balance < 50)

{

info.isUnderBalance();

}

else { goto Loop; }

}

else if( wish == 'N' || wish == 'n')

{

Console.WriteLine("GoodBye");

}

}

}

//\*-- EVENT HANDLING STARTS HERE --\*

//Delegate Declaration.

public delegate void BalanceEvents();

public class UnderBalanceClass // Event Class Declaration

{

public event BalanceEvents UnderBalance; // Declaring Events

public void isUnderBalance()

{

OnUnderBalance();

}

protected virtual void OnUnderBalance()

{

UnderBalance?.Invoke();

}

}

public class ZeroBalanceClass // Event Class Declaration

{

public event BalanceEvents ZeroBalance; // Declaring Events

public void isZeroBalance()

{

OnZeroBalance();

}

protected virtual void OnZeroBalance()

{

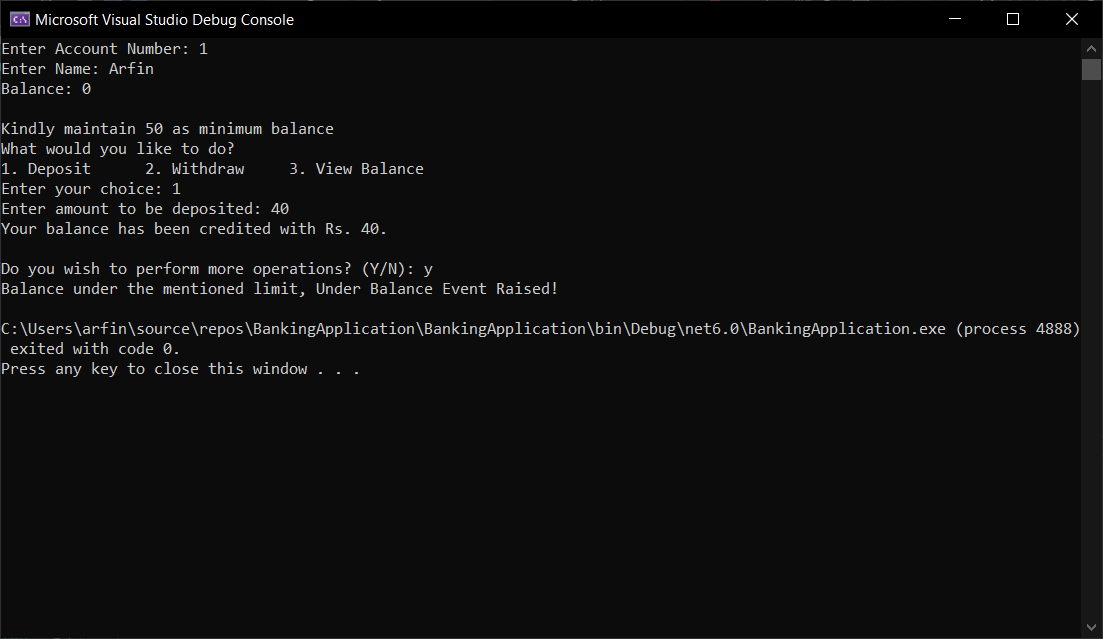
ZeroBalance?.Invoke();

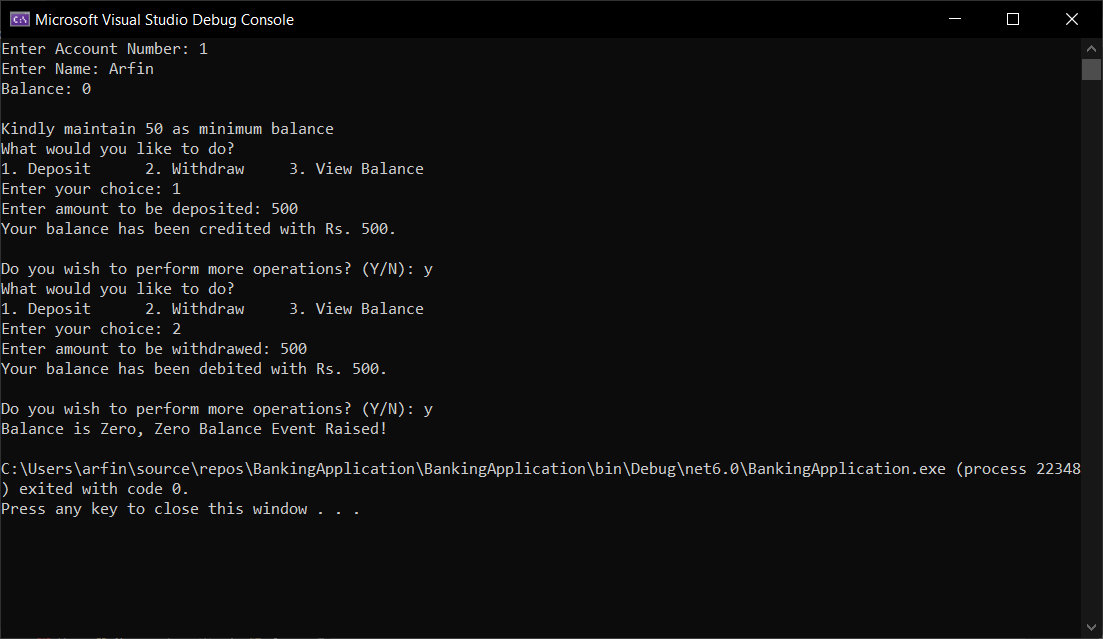
}

}

}

OUTPUT:





## Now try these to get a complete grip…

1. Write separate ICICIBank class with EventHandlers for UnderBalance and

BalanceZero Events.

* 1. Show message "Transaction cannot be continued as balance is insufficient/zero in Account".

CODE:

public class BalanceEventsClass // Event Class Declaration

{

public event BalanceEvents UnderBalance; // Declaring Events

public void isUnderBalance()

{

OnUnderBalance();

}

protected virtual void OnUnderBalance()

{

UnderBalance?.Invoke();

}

public event BalanceEvents ZeroBalance; // Declaring Events

public void isZeroBalance()

{

OnZeroBalance();

}

protected virtual void OnZeroBalance()

{

ZeroBalance?.Invoke();

}

}

public class ICICIBank : BalanceEventsClass

{

protected override void OnUnderBalance()

{

Console.WriteLine("Transaction cannot be continued as balance is insufficient in Account");

}

protected override void OnZeroBalance()

{

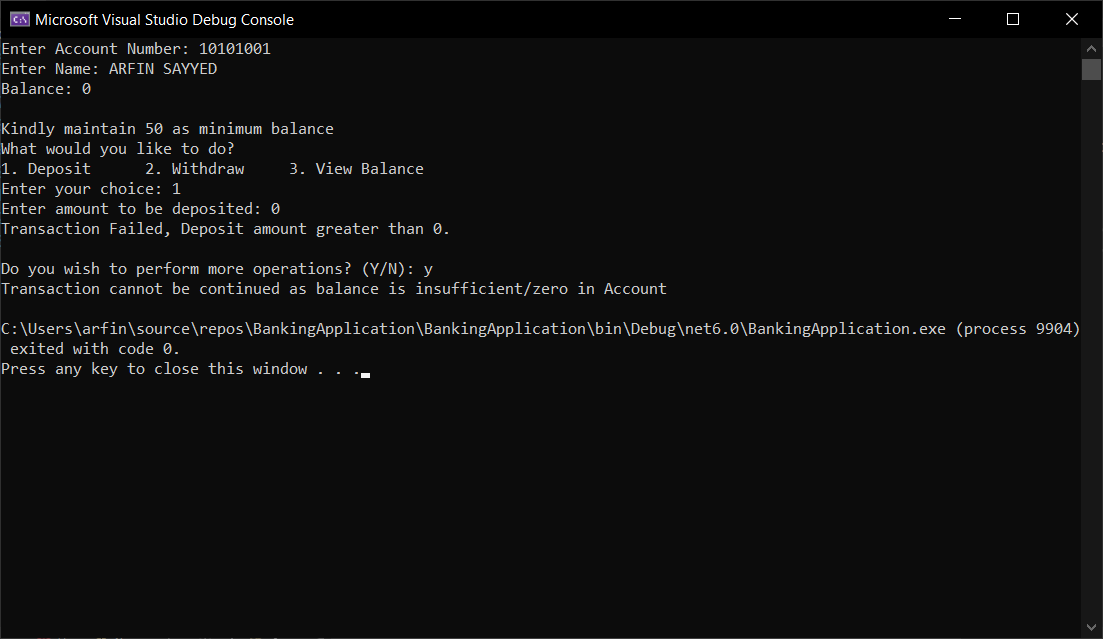
Console.WriteLine("Transaction cannot be continued as balance is insufficient/zero in Account");

}

}

}

OUTPUT:



1. Write separate HDFCBank class with EventHandlers for UnderBalance and

BalanceZero Events.

* 1. EventHandler should allow account holder to withdraw money till balance goes to Rs - 1000;below this show message "Transaction cannot be continued below specified limit of Rs.-1000".

CODE:

Public static void Main(){

BalanceEventsClass info2 = new HDFCBank(); //<---- 6.5

info2.UnderBalance += whenUnderBalance;

case 2:

Console.Write("Enter amount to be withdrawed: ");

double amountwithdrawn = Convert.ToDouble(Console.ReadLine());

if( AccountHolder.Balance - amountwithdrawn < 1000)

{

info2.isUnderBalance();

break;

}

AccountHolder.Withdraw(amountwithdrawn);

break;

}

public class HDFCBank : BalanceEventsClass

{

protected override void OnUnderBalance()

{

Console.WriteLine("Transaction cannot be continued below specified limit of Rs.-1000");

}

protected override void OnZeroBalance()

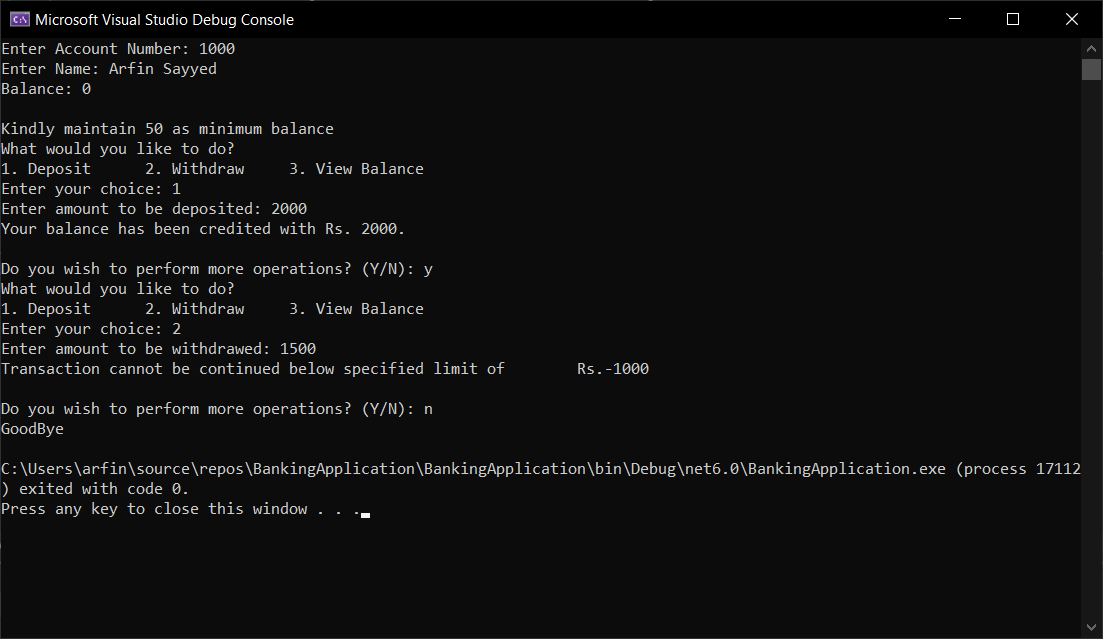
{

Console.WriteLine("Transaction cannot be continued as balance is insufficient/zero in Account");

}

}

OUTPUT:



**C# ASSIGNMENT 7**

1. Write a console application which will read text files from mentioned file system location. And list subdirectories from mentioned folder on file system using System.IO namespace and use DirectoryInfo, Directory, File and FileInfo Classes with all the methods present in these classes.

CODE:

using System;

using System.IO;

namespace FileIODemo

{

public class Program

{

public static void Main()

{

try

{

string path = @"C:\Users\arfin\Desktop\FileIO\";

string[] files = Directory.GetFiles(path);

string[] directories = Directory.GetDirectories(path);

Console.WriteLine($"Files in {path}\n");

foreach (string file in files)

{

string fileName = Path.GetFileName(file);

Console.WriteLine(fileName);

string filepath = Path.Combine(path, fileName);

FileInfo myfile = new FileInfo(filepath);

// Opening file to read

StreamReader sr = myfile.OpenText();

string data = "";

while ((data = sr.ReadLine()) != null)

{

Console.WriteLine(data);

}

Console.WriteLine("\n");

}

Console.WriteLine($"Subdirectories inside {path}\n");

foreach (string directory in directories)

{

DirectoryInfo directoryinfo = new DirectoryInfo(directory);

directoryinfo.GetDirectories();

string directoryName = directoryinfo.Name;

Console.WriteLine(directoryName);

}

}

catch(IOException e)

{

Console.WriteLine(e);

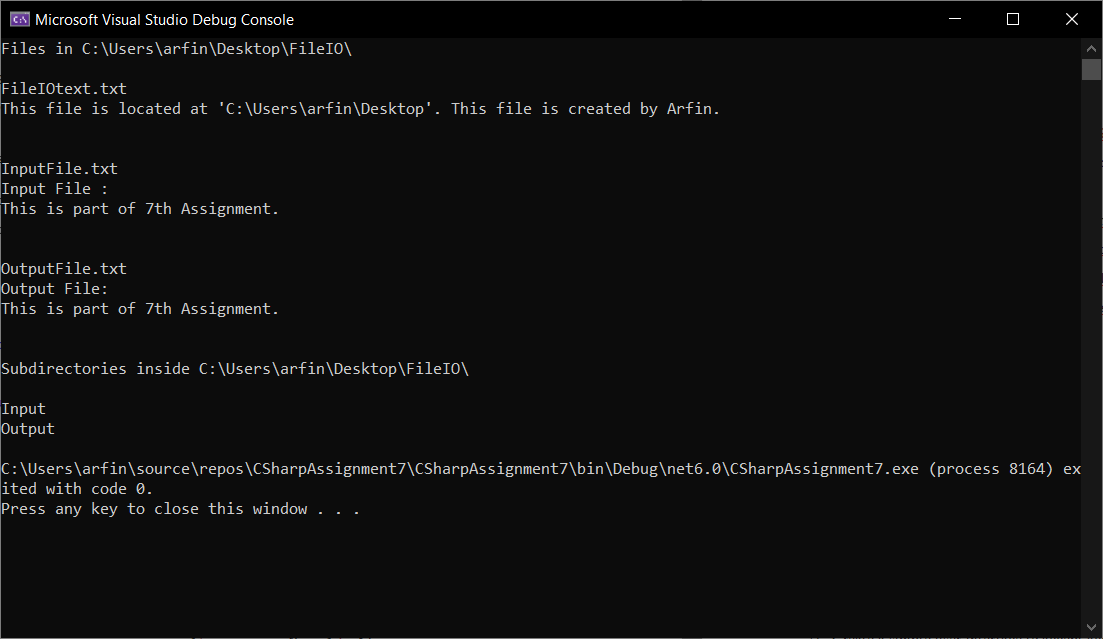
}

}

}

}

OUTPUT



1. Create a simple user interface to accept account related information of a customer.[ Account class from Lab session on Delegates and Events can be used]. Save the information about the customers in a file using StreamWriter and retrieve the information using StreamReader.

using System;

namespace BankingDomainApplication

{

public class Customer

{

public int AccountNumber;

public string CustomerName;

public void getData(int AccountNumber, string CustomerName)

{

this.AccountNumber = AccountNumber;

this.CustomerName = CustomerName;

}

public static void Main()

{

try

{

Customer MyCustomer = new Customer();

Console.Write("Enter Account Number: ");

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

Console.Write("Enter Name: ");

string CustomerName = Console.ReadLine();

MyCustomer.getData(AccountNumber, CustomerName);

string path = @"C:\Users\arfin\Desktop\FileIO\CustomerDetails.txt";

// Writing Customer Details

StreamWriter sw = File.CreateText(path);

sw.WriteLine($"Account number: {AccountNumber}");

sw.WriteLine($"Name of Customer: {CustomerName}");

sw.WriteLine("New Customer, Balance = 0");

sw.Close();

//Reading Customer Details

using (StreamReader file = new StreamReader(path))

{

int counter = 0;

string ln;

while ((ln = file.ReadLine()) != null)

{

Console.WriteLine(ln);

counter++;

}

}

}

catch (Exception e)

{

Console.WriteLine(e.GetType().Name);

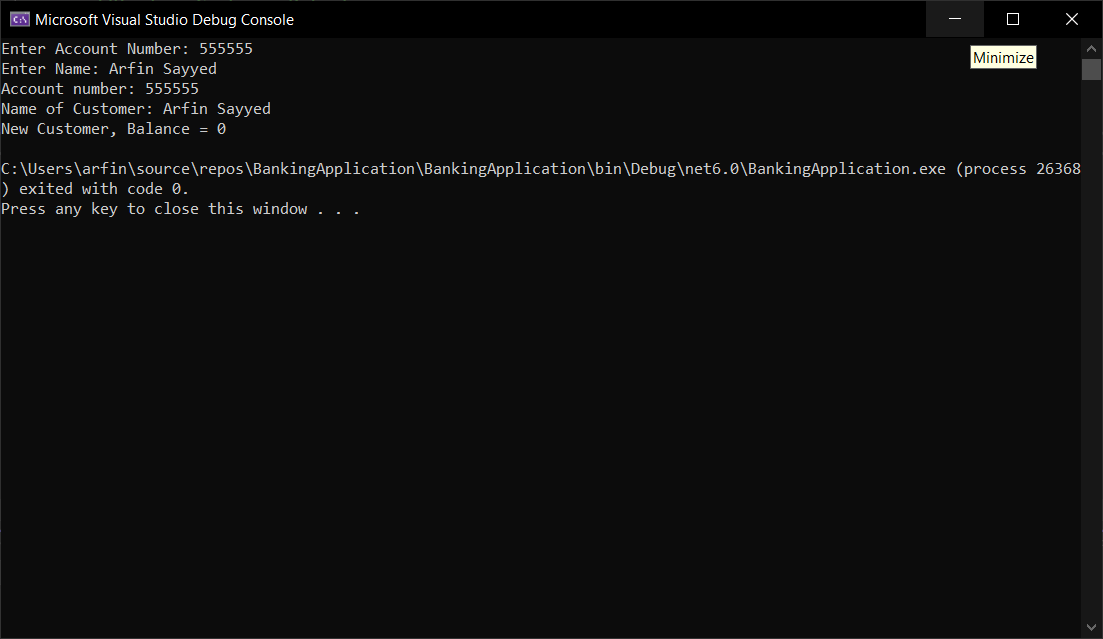
}

}

}

}

OUTPUT:



1. Make the Employee, MarketingExecutive and Manager class as Serializable

created in LitwareLib.dll .

1. Create a user interface to accept information about Manager(For simplicity accept only employee id , name and basic salary). Serialize the object using Binary Serialization and retrieve its information by deserializing the object.

using System;

using System.Runtime.Serialization;

using System.Runtime.Serialization.Formatters.Binary;

using LitwareLib;

class Program

{

public static void Main()

{

int empid;

string empname;

double salary;

Console.WriteLine("Welcome to Litware Organization");

Console.Write("Enter your Employee ID: ");

empid = Convert.ToInt32(Console.ReadLine());

Console.Write("Enter your Name: ");

empname = Console.ReadLine();

Console.Write("Enter your Salary: ");

salary = Convert.ToDouble(Console.ReadLine());

Employee employee = new Employee();

employee.GetEmployeeData(empid, empname, salary);

employee.CalculateSalary();

// Serializing Employee object.

FileStream f = new FileStream(@"C:\Users\arfin\Desktop\FileIO\EmployeeDetails.txt", FileMode.Open, FileAccess.Write);

BinaryFormatter b = new BinaryFormatter();

b.Serialize(f, employee);

f.Close();

//Deserializing Employee object.

FileStream fr = new FileStream(@"C:\Users\arfin\Desktop\FileIO\EmployeeDetails.txt", FileMode.Open, FileAccess.Read);

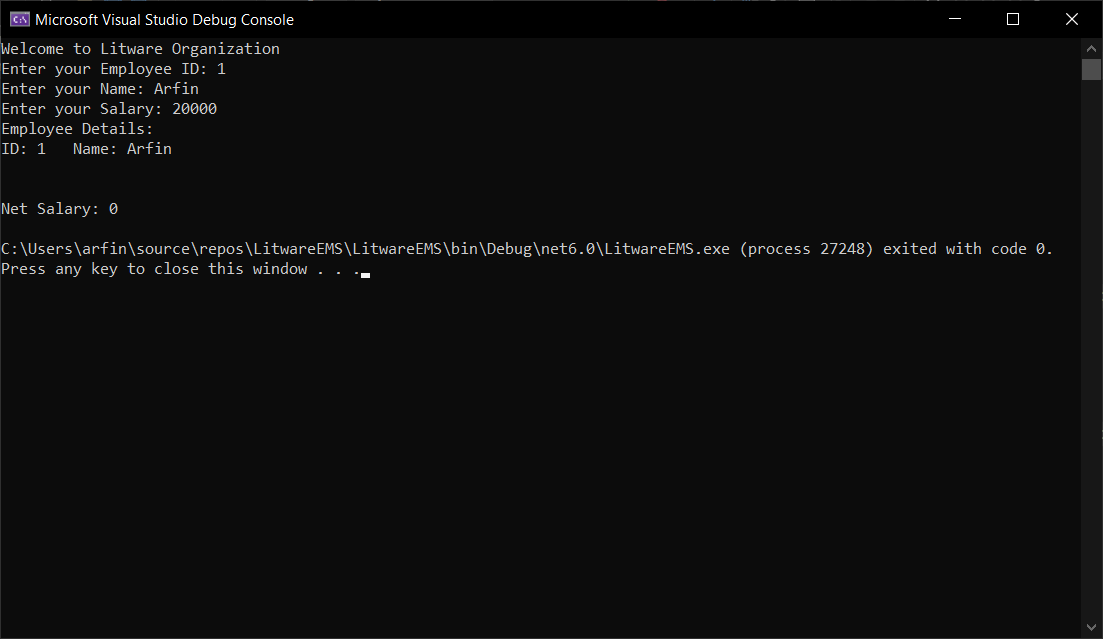
BinaryFormatter br = new BinaryFormatter();

br.Deserialize(fr);

fr.Close();

}

}



**C# ASSIGNMENT NO: 8**

* + 1. Write a console based application to dynamically load an assembly. Display the available types and their members. Prompt the user to invoke any method at runtime.

CODE:

using System;

using System.Reflection;

namespace Reflectionexample

{

class Employee

{

public int empno

{

get;

set;

}

public string empName

{

get;

set;

}

public double salary

{

get;

set;

}

public Employee()

{

empno = 0;

empName = string.Empty;

salary = 0;

}

// Parameterised Constructor

public Employee(int eno, string n, double sal)

{

empno = eno;

empName = n;

salary = sal;

}

public void displayData()

{

Console.WriteLine($"Employee Number : {empno}" );

Console.WriteLine($"Eployee Name : {empName}");

Console.WriteLine($"Employee Salary :{salary}");

}

}

class DynamicAssembly

{

public static void Main()

{

Assembly executing = Assembly.GetExecutingAssembly();

Type[] types = executing.GetTypes();

foreach (var item in types)

{

Console.WriteLine("Class : {0}", item.Name);

MethodInfo[] methods = item.GetMethods();

foreach (var method in methods)

{

Console.WriteLine("--> Method : {0}", method.Name);

ParameterInfo[] parameters = method.GetParameters();

foreach (var arg in parameters)

{

Console.WriteLine("----> Parameter : {0} Type : {1}",

arg.Name, arg.ParameterType);

}

}

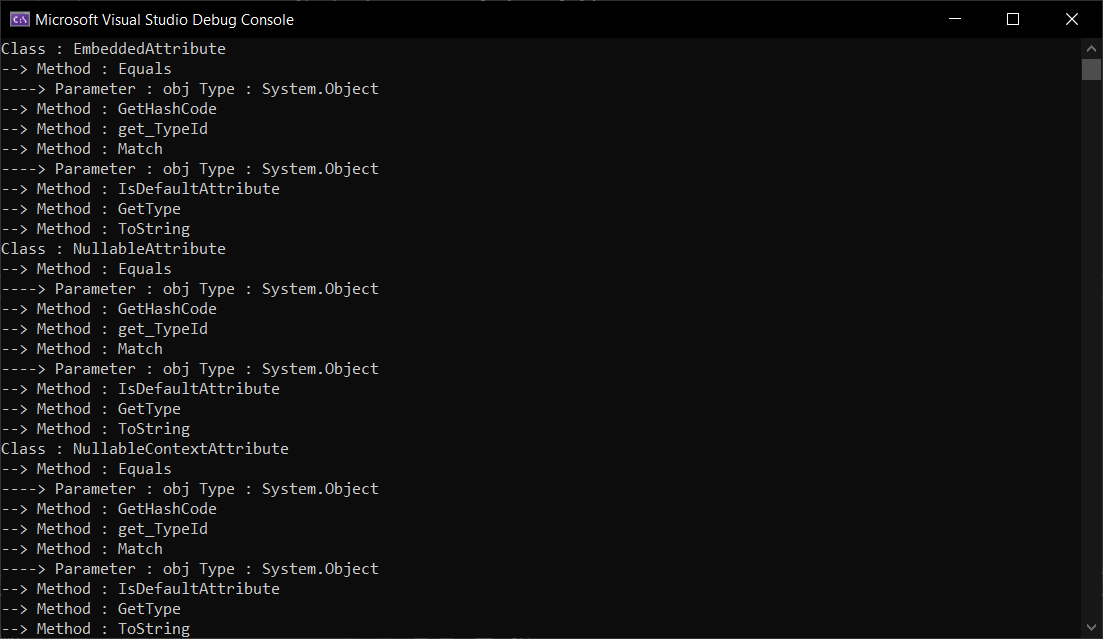
}

}

}

}

OUTPUT:



* + 1. Define a SoftwareAttribute which is having its base class as “Attribute” with following private members:
       - String ProjectName.
       - String Description.
       - String ClientName.
       - String StartedDate.
       - String EndDate.

1. Write properties for all these members.
2. Design a simple Console Application for testing refection concept of .NET Framework:
   * Define SoftwareAttribute as Custom Attribute
   * Write two classes HDFCAccount, ICICIAccount Apply

SoftwareAttribute to these classes.

* + Write Test class which will read attributes applied on each classes using reflection technique.

**CODE:**

using System;

using System.Reflection;

using System.Collections.Generic;

[AttributeUsage(AttributeTargets.All)]

public class SoftwareAttribute : Attribute

{

String ProjectName, Description, ClientName, StartedDate, EndDate;

public SoftwareAttribute(String p, String d, String c, String s, String e)

{

ProjectName = p;

Description = d;

ClientName = c;

StartedDate = s;

EndDate = e;

}

public static void AttributeDisplay(Type classType)

{

Console.WriteLine($"Methods of class {classType.Name}");

MethodInfo[] methods = classType.GetMethods();

for (int i = 0; i < methods.GetLength(0); i++)

{

object[] attributesArray = methods[i].GetCustomAttributes(true);

foreach (Attribute item in attributesArray)

{

if (item is SoftwareAttribute)

{

SoftwareAttribute attributeObject = (SoftwareAttribute)item;

Console.WriteLine("{0} - {1}, {2}, {3} , {4} ,{5} ", methods[i].Name, attributeObject.ProjectName, attributeObject.Description, attributeObject.ClientName, attributeObject.EndDate, attributeObject.EndDate);

}

}

}

}

}

class ICICI

{

double AccountNumber;

string Name;

double Bankbalance;

public ICICI(double a, string n, double b)

{

AccountNumber = a;

Name = n;

Bankbalance = b;

}

[SoftwareAttribute("Accessor", "gives the values of account number", "client name icici", "6th aug", "6th aug")]

public double getAccountNumber()

{

return AccountNumber;

}

[SoftwareAttribute("accessor", "gives the values of account holder name", "client name icici", "6th aug", "6th aug")]

public string getName()

{

return Name;

}

[SoftwareAttribute("accessor", "gives the values of Balance", "client name icici", "6th aug", "6th aug")]

public double getbankbalance()

{

return Bankbalance;

}

}

class HDFC

{

double AccountNumber;

string Name;

double Bankbalance;

public HDFC(double accountNumber, string name, double bankbalance)

{

AccountNumber = accountNumber;

Name = name;

Bankbalance = bankbalance;

}

[SoftwareAttribute("accessor", "gives the values of account number", "client name HDFC", "6th aug", "6th aug")]

public double getAccountNumber()

{

return AccountNumber;

}

[SoftwareAttribute("accessor", "gives the values of account holder name", "client name HDFC", "6th aug", "6th aug")]

public string getName()

{

return Name;

}

[SoftwareAttribute("accessor", "gives the values of account Balance", "client name HDFC", "6th aug", "6th aug")]

public double getbankbalance()

{

return Bankbalance;

}

}

class MyAttributes

{

public static void Main()

{

SoftwareAttribute.AttributeDisplay(typeof(ICICI));

Console.WriteLine();

SoftwareAttribute.AttributeDisplay(typeof(HDFC));

}

}

**OUTPUT:**