## Objective

* To use non – generic and generic classes provided in collection framework.
* To implement in-built collection interfaces.
* To create a custom generic class.

## Assignments to be done in this session

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

Use System.Array class to perform following operations on them

Copy, Sort, Clear, Reverse

Accept input from user through Console.

**INPUT:**

using System;

namespace ArrayAssignment

{

class ArrayProgram

{

static void Main(string[] args)

{

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

int ArrSize = int.Parse(Console.ReadLine());

int[] arrayInt = new int[ArrSize];

string[] arrayStr = new string[ArrSize];

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

{

Console.Write("Enter {0} int type element in array: ",i);

arrayInt[i] = int.Parse(Console.ReadLine());

}

Console.WriteLine("") ;

Console.Write("Output before applying any method : ");

foreach (int val in arrayInt)

{

Console.Write(val + " ");

}

Console.WriteLine("\n");

int[] array2 = new int[ArrSize];

Array.Copy(arrayInt, array2, ArrSize);//Copy()

Console.WriteLine("");

Console.Write("After copying one array in array2 : ");

Console.Write("");

foreach (int value in array2)

{

Console.Write(value + " ");

}

Console.WriteLine("");

Console.WriteLine("Cleared array2: ");

Array.Clear(array2, 0, ArrSize);//clear array2

foreach (int val in array2)

{

Console.Write(val + " ");

}

Console.WriteLine();

Array.Reverse(arrayInt); //Reversing array

foreach (int val in arrayInt)

{

Console.Write(val + " ");

}

Array.Sort(arrayInt);//sorting array

Console.WriteLine("");

Console.Write("\nAfter Sorting array: ");

foreach (int val in arrayInt)

{

Console.Write(val + " ");

}

Console.WriteLine("");

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Console.WriteLine("\nEnter String in Array\n");

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

{

Console.Write("Enter {0} String type element in array: ", i);

arrayStr[i] = Console.ReadLine();

}

Console.WriteLine("");

Console.Write("Output of Array String before appplying any method: ");

foreach (string value in arrayStr)

{

Console.Write(value + " ");

}

Array.Reverse(arrayStr);

Console.WriteLine("");

Console.WriteLine("");

Console.Write("After Reversing String Array: ");

foreach (string value in arrayStr)

{

Console.Write(value + " ");

}

Console.WriteLine("");

Array.Sort(arrayStr);

Console.Write("Sorting String Array: ");

foreach (string value in arrayStr)

{

Console.Write(value + " ");

}

string[] arraystring2 = new string[ArrSize];

Array.Copy(arrayStr, arraystring2, ArrSize);

Console.WriteLine("");

Console.Write("After copying to Array String2 : ");

foreach (string value in arraystring2)

{

Console.Write(value + " ");

}

Console.WriteLine("");

Console.WriteLine("After Clearing String Array2 : ");

Array.Clear(arraystring2, 0, ArrSize);

foreach (string value in arrayStr)

{

Console.Write(value + " ");

}

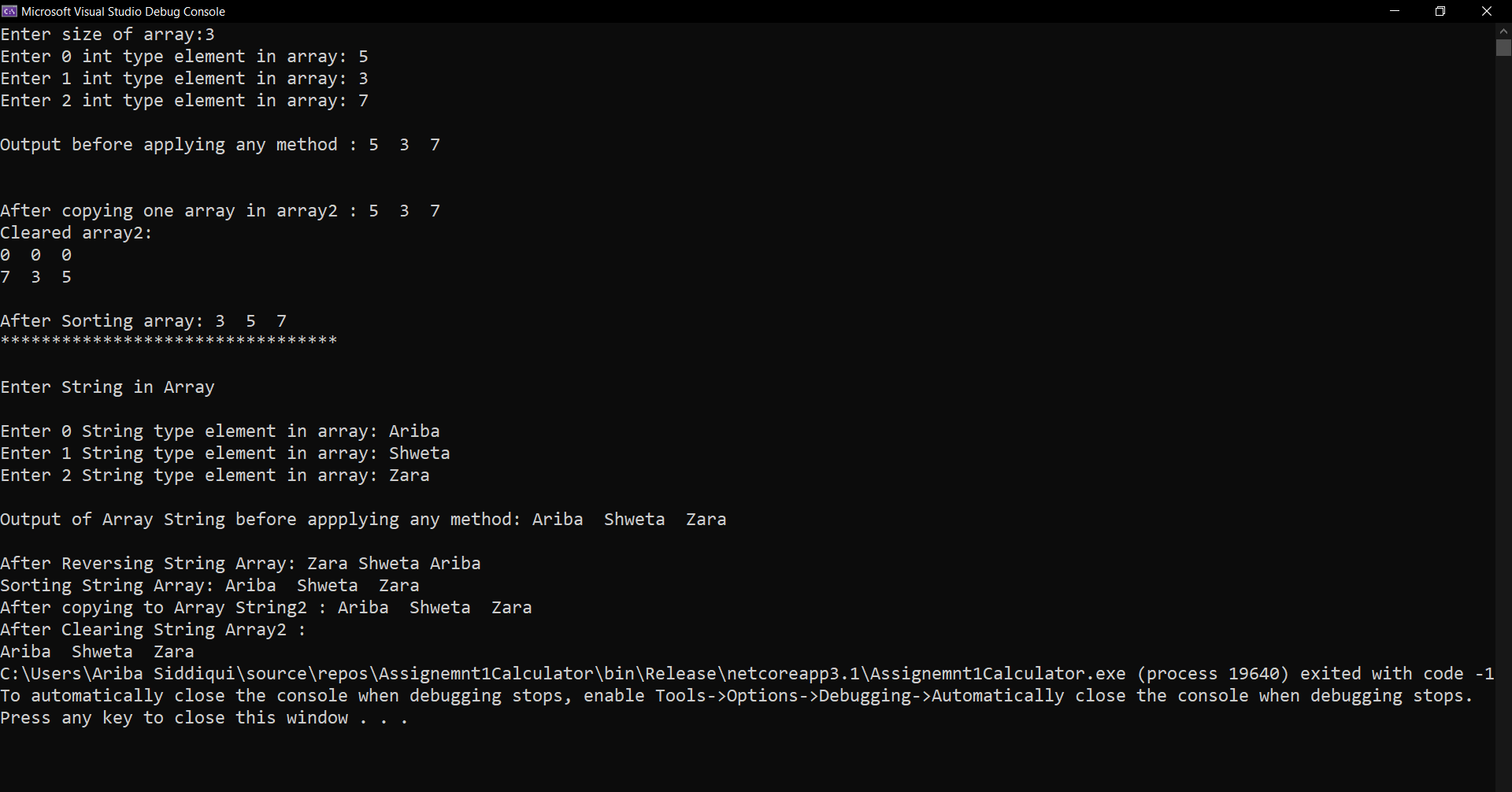
Console.ReadLine();

}

}

}

**OUTPUT:**



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.

**INPUT:**

using System;

using System.Collections;

namespace ArrayListExample

{

public class Employee

{

int EmpId;

string EmpName;

double Empsalary;

public Employee(int empid, string empname, double empsalary)

{

this.EmpId = empid;

this.EmpName = empname;

this.Empsalary = empsalary;

}

public override string ToString()

{

return

String.Format("Employee Id: {0} Employee Name: {1} Employee Salary: {2} ", EmpId,EmpName, Empsalary);

}

static void Main(string[] args)

{

ArrayList empdetail = new ArrayList();

empdetail.Add(new Employee(101, "Rahul", 600000));

empdetail.Add(new Employee(102, "Taiba", 500000));

empdetail.Add(new Employee(098, "Kiran", 300000));

empdetail.Add(new Employee(078, "Uzmaa", 800000));

Console.WriteLine("Employee Details by using array list");

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

foreach (Employee employee in empdetail)

{

Console.WriteLine(employee);

}

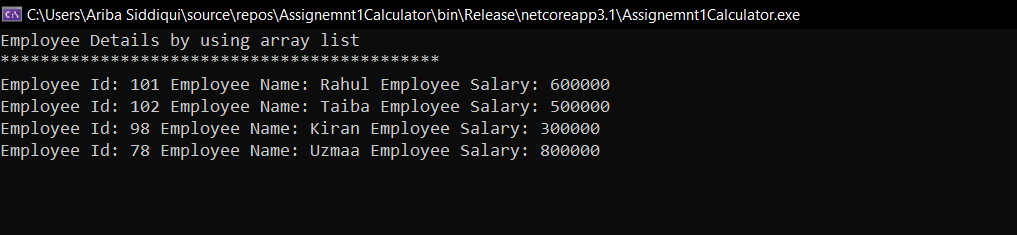
Console.ReadLine();

}

}

}

**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

**INPUT:**

using System;

using System.Collections.Generic;

using System.Linq;

namespace LinkedListGenExample

{

public class LinnkedListAssignment

{

static void Main(string[] args)

{

int i,n;

LinkedList<string> NewEmp = new LinkedList<string>();

//Adding Employee in Linked List

Console.Write("Enter No. of employee you want to add:");

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

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Console.WriteLine("Enter {0} elements in the list:", n);

for (i=0; i < n; i++)

{

Console.Write("Enter Employee in the list:", i);

string empList = Console.ReadLine();

NewEmp.AddFirst(empList);

}

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Console.WriteLine("List of an Employee:");

foreach (string employee in NewEmp)

{

Console.WriteLine(employee);

}

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Console.WriteLine("Total number of Employee : " + NewEmp.Count());

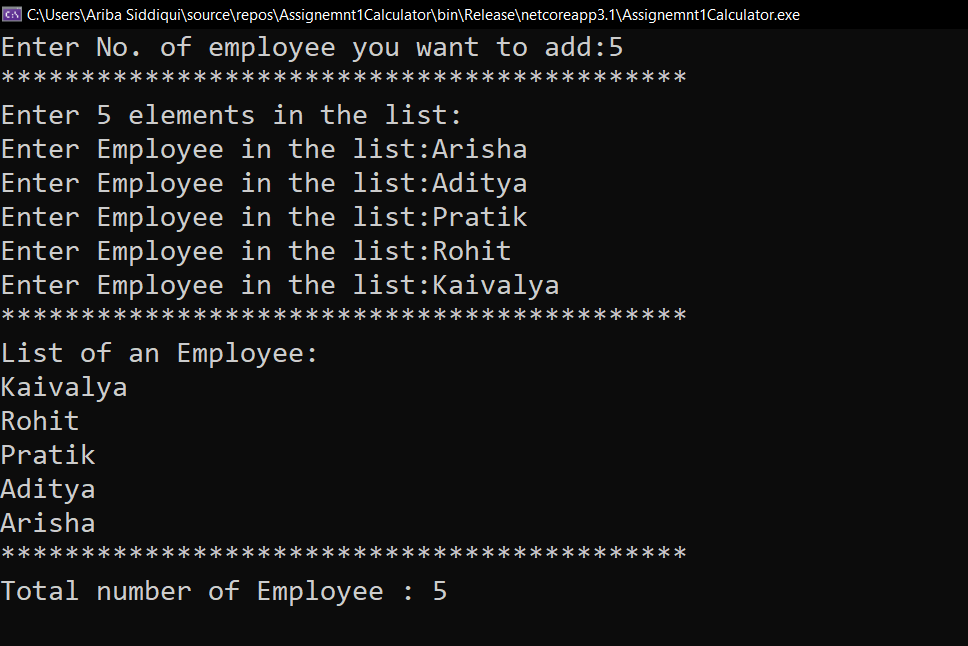
Console.ReadLine();

}

}

}

**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.

**INPUT:**

using System;

using System.Collections.Generic;

namespace StackObj

{

class MyStack

{

static void Main(string[] args)

{

Stack<Object> genstack = new Stack<Object>();

Console.WriteLine("Adding Element in Stack");

genstack.Push(66);

genstack.Push("Atif");

genstack.Push("Red");

genstack.Push(25.456);

foreach (Object ob in genstack)

{

Console.WriteLine(ob);

}

Console.WriteLine("");

Console.WriteLine("Pop Operation is Applied on {0}", genstack.Pop());

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Console.WriteLine("After pop");

foreach (Object ob in genstack)

{

Console.WriteLine(ob);

}

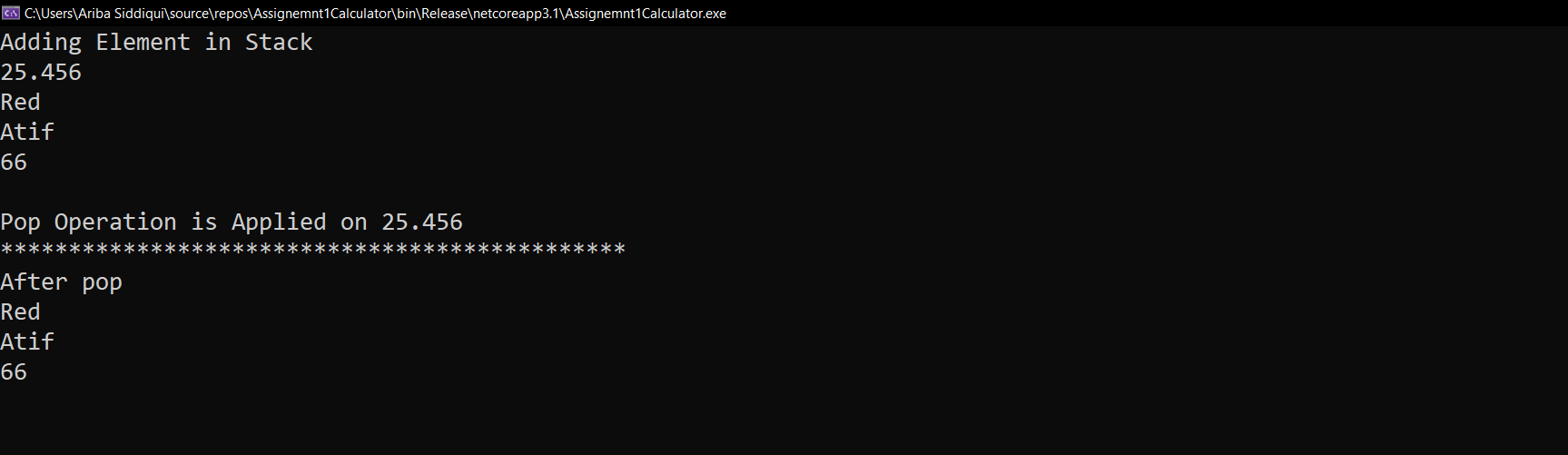
Console.ReadLine();

}

}

}

**OUTPUT:**



## Now try these to get a complete grip…

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

**INPUT:**

using System;

using System.Collections.Generic;

using System.Linq;

namespace LinkedListGenExample

{

public class LinnkedListAssignment

{

static void Main(string[] args)

{

int i, n;

LinkedList<string> NewEmp = new LinkedList<string>();

//Adding Employee in Linked List

Console.Write("Enter No. of employee you want to add:");

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

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Console.WriteLine("Enter {0} elements in the list:", n);

for (i = 0; i < n; i++)

{

Console.Write("Enter Employee in the list:", i);

string empList = Console.ReadLine();

NewEmp.AddFirst(empList);

}

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Console.WriteLine("List of an Employee:");

foreach (string employee in NewEmp)

{

Console.WriteLine(employee);

}

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Console.WriteLine("Total number of Employee : " + NewEmp.Count());

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Console.Write("Enter the of Employee you want to Search: ");

string search = Console.ReadLine();

String res = Convert.ToString(NewEmp.Contains(search));

Console.WriteLine("Contains Employee in List: " + res);

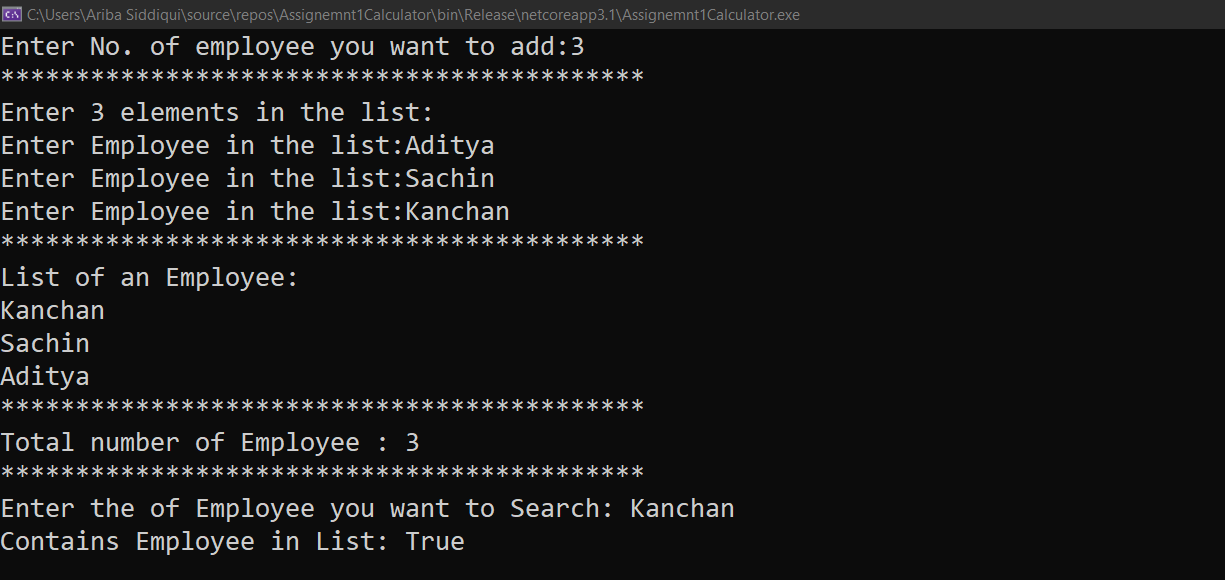
Console.ReadLine();

}

}

}

**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.

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

## By the way

* Generic classes offer better performance over non-generic counterparts as issues like casting and boxing are reduced.
* yield keyword is used iterate through a collection. This enhances the performance of an application.
* Custom Collection classes can be created by implementing the Collection Interfaces provided by .NET framework class library.