## 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.

**Code:-**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

class ArrayOp

{

public void integer()

{

Console.WriteLine("Enter Size of an Array:");

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

int[] arr = new int[size];

int[] arr2 = new int[size];

Console.WriteLine("Enter Elements to the Array:");

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

{

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

}

Array.Copy(arr, arr2, size);

Console.WriteLine("Copy Elements in Array :");

foreach (int array in arr)

{

Console.WriteLine(array);

}

Array.Sort(arr);

Console.WriteLine("Sort Elements in Array :");

foreach (int n1 in arr)

{

Console.WriteLine(n1);

}

Array.Clear(arr, 1, 3);

Console.WriteLine("Clear Elements in Array :");

foreach (int n2 in arr)

{

Console.WriteLine(n2);

}

Array.Reverse(arr);

Console.WriteLine("Reverse Elements in Array :");

foreach (int n3 in arr)

{

Console.WriteLine(n3);

}

}

public void String()

{

Console.WriteLine("Enter Size of an Array: ");

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

string[] str = new string[size];

string[] str2 = new string[size];

Console.WriteLine("Enter String to the array:");

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

{

str[i] = Console.ReadLine();

}

Array.Copy(str, str2, size);

Console.WriteLine("Copy String:");

foreach (string array in str)

{

Console.WriteLine(array);

}

Array.Sort(str);

Console.WriteLine("Sort String:");

foreach (string n1 in str)

{

Console.WriteLine(n1);

}

Array.Clear(str, 1, 2);

Console.WriteLine("Clear String:");

foreach (string n2 in str)

{

Console.WriteLine(n2);

}

Array.Reverse(str);

Console.WriteLine("Reverse String:");

foreach (string n3 in str)

{

Console.WriteLine(n3);

}

}

public static void Main(string[] args)

{

ArrayOp a = new ArrayOp();

a.integer();

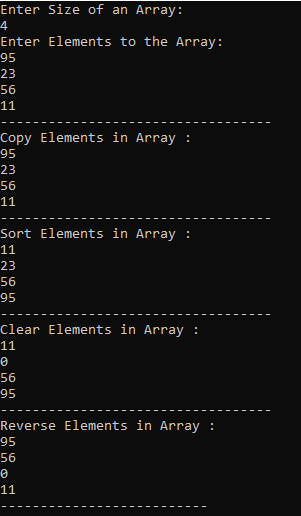
Console.WriteLine("--------------------------");

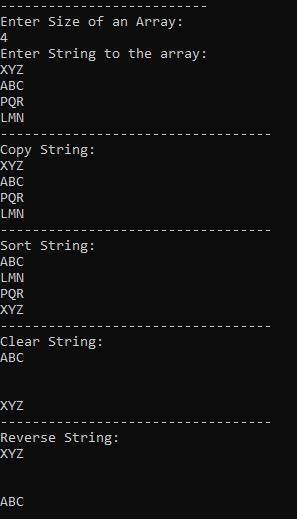
a.String();

}

}

}





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.Generic;

namespace ConsoleApp1

{

public class EmpManagementApp

{

public static void Main()

{

List<object> L1 = new List<object>();

L1.Add("1011");

L1.Add("Shruti");

L1.Add("45000");

Console.WriteLine("Before Adding Elements: ");

foreach (object i in L1)

{

Console.WriteLine(i);

}

object[] L2 = new object[3];

L2[0] = 1012;

L2[1] = "Shwaet";

L2[2] = 30000;

L1.InsertRange(3, L2);

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

Console.WriteLine("After adding elements :");

foreach (object i in L1)

{

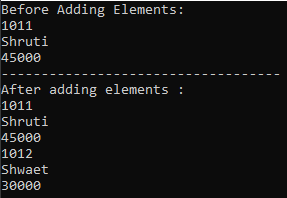
Console.WriteLine(i);

}

}

}

}



3) Write a console based program to create a linked list of Employee objects using the generic class List<>.Perform following operations on the list:

a. Add a new employee

b. Display the list of employees.

c. Total number of employees in the list

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

class EmpList

{

static void Main(string[] args)

{

Employee employee1 = new Employee()

{

EmpID = 1011,

EmpName = "Shruti",

EmpSalary = 20000

};

List<Employee> employees = new List<Employee>(1);

employees.Add(employee1);

foreach (Employee c in employees)

{

Console.WriteLine("ID={0}, Name={1}, Salary={2}", c.EmpID, c.EmpName, c.EmpSalary);

}

again:

Console.WriteLine("Add emoployee---Y or N");

string choice = Convert.ToString(Console.ReadLine());

if (choice.ToUpper() == "Y")

{

Employee emp = new Employee();

Console.WriteLine("Enter Employee Id : ");

emp.EmpID = Convert.ToInt32(Console.ReadLine());

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

emp.EmpName = Console.ReadLine();

Console.WriteLine("Enter Employee Salary : ");

emp.EmpSalary = Convert.ToInt32(Console.ReadLine());

employees.Add(emp);

}

else

{

Console.WriteLine("Total no.of Employees =" + employees.Count);

}

Console.WriteLine("Total no. of employees =" + employees.Count);

foreach (Employee c in employees)

{

Console.WriteLine("ID={0}, Name={1}, Salary={2}", c.EmpID, c.EmpName, c.EmpSalary);

}

}

}

class Employee

{

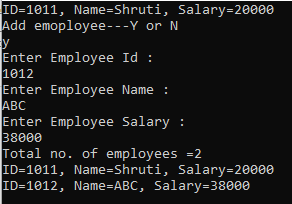
public int EmpID { get; set; }

public string EmpName { get; set; }

public int EmpSalary { get; set; }

}

}



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

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

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

class Stack2

{

public static void Main()

{

try

{

Stack<object> numbers = new Stack<object>();

numbers.Push("Eleven");

numbers.Push("10");

numbers.Push("abc");

numbers.Push("4");

numbers.Push("Two");

// A stack can be enumerated without disturbing its contents.

foreach (object number in numbers)

{

Console.WriteLine(number);

}

Console.WriteLine("Popping '{0}'", numbers.Pop());

Console.WriteLine("Peek to destack: {0}",

numbers.Peek());

Console.WriteLine("Popping '{0}'", numbers.Pop());

// Create a copy of the stack, using the ToArray method and the

// constructor that accepts an IEnumerable<T>.

Stack<object> stack2 = new Stack<object>(numbers.ToArray());

Console.WriteLine("\nContents of the first copy:");

foreach (string number in stack2)

{

Console.WriteLine(number);

}

// Create an array twice the size of the stack and copy the

// elements of the stack, starting at the middle of the

// array.

string[] array2 = new string[numbers.Count \* 2];

numbers.CopyTo(array2, numbers.Count);

// Create a second stack, using the constructor that accepts an

// IEnumerable(Of T).

Stack<object> stack3 = new Stack<object>(array2);

Console.WriteLine("\nContents of the second copy, with duplicates and nulls:");

foreach (object number in stack3)

{

Console.WriteLine(number);

}

Console.WriteLine("\nstack2.Contains(\"abc\") = {0}",

stack2.Contains("abc"));

Console.WriteLine("\nstack2.Clear()");

stack2.Clear();

Console.WriteLine("\nstack2.Count = {0}", stack2.Count);

}

catch (Exception ex)

{ Console.WriteLine(ex.Message); }

}

}

}

## 

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

//create a list

List<string> list = new List<string>();

//add elements in the list

list.Add(name);//here name is string which used to enter the employeename in the main method

Console.WriteLine("search employee name :- ");

string target = Console.ReadLine();

bool isexist = list.Contains(target);

if (isexist)

{

Console.WriteLine("Employee Found");

}

else

{

Console.WriteLine("Employee not found ");

}

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.

using System;

using System.Collections;

namespace ConsoleApp1

{

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[] arr = new Player[5];

public Team()

{

arr[0] = new Player("Dhoni", 42);

arr[1] = new Player("Virat", 31);

arr[2] = new Player("Rohit", 28);

arr[3] = new Player("Sachin", 45);

arr[4] = new Player("Hardik", 25);

}

public IEnumerator GetEnumerator()

{

foreach(Player player in arr)

{

Console.WriteLine("Player Name : {0} , Runs Scored : {1}", player.Name, player.Run);

}

return arr .GetEnumerator();

}

}

internal class IEnumerable

{

public static void Main(String[] args)

{

Team India = new Team();

India.GetEnumerator();

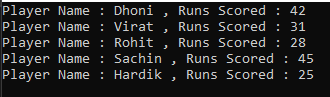
Console.ReadKey();

}

}

}

}



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

using System;

using System.Collections;

namespace ConsoleApp1

{

class Player

{

public string Name { get; set; }

public string Run { get; set;}

public Player(string name, string run )

{

Name = name;

Run = run;

}

class Team : IEnumerable

{

private Player[] arr = new Player[5];

public Team()

{

arr[0] = new Player("Dhoni", "42");

arr[1] = new Player("Virat", "31");

arr[2] = new Player("Rohit", "28");

arr[3] = new Player("Sachin", "45");

arr[4] = new Player("Hardik", "25");

}

public IEnumerator GetEnumerator()

{

foreach (Player p in arr)

{

yield return p.Name;

yield return p.Run;

}

}

}

internal class IEnumerable

{

public static void Main()

{

Team India = new Team();

foreach (string name in India)

{

Console.Write(name + " \n");

}

Console.WriteLine();

Console.ReadKey();

}

}

}

}



## 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.