**S24: Collections: Generics Collection**

**Collections: -** introduced in 1.0. Set of related data that may not belong to same data type. It can dynamically modified at run time. It’s a dynamic array. Resize method of array destroy old and create new array. Can be resized at run-time. Value can be inserted or deleted in middle. Elements can be different type.(heterogeneous:- Different Data Type, homogenous:- Same Data Type). why used: some things common, auto resize,

* **Generic Collections: -** System. Collections.Generic namespace used. It provide generic implemetetion of standard data structure such as Linked List, Stack, Queue and dictionary. These are type safe. Faster than non-**generic.** This contain < > placeholder syntax.

* + **List< > :** - it’s a class and we can use with any variable e.g.:- string, integer, complex type, bunch of values. Value can be access by index. Its generic version of array List. Performs faster and less error as compare to array List. Auto size will be same as per array list. Only define data type can be stored. Capacity is used to check the size of list. Contain null and duplicate value. Insert () is used to insert values in list on particular position. Add will add the element at last. And add range(add collection data only) will add the particular collection of element.
  + **Dictionary< >:-** it store key value pair. It implements IDictionary interface. Key cant be null and unique. Values can be null or duplicate.value is access by using key. Elements are stored in form of objects. Keys are user defined. Insertion of same data type. Keyvalue pair is used to display or iterate the dictionary element. Also use var. similar to hashtable. Methods: add, Remove: remove element at specified key, ContainsKey: Check the key is exist or not it return bollean result. ContainsValue: check the value is exist or not. Clear:- Remove all the items from dictionary. TryGetValue:- return true and assign value to variable if key exist it also return bollean value.. It contain (“Key”, OutParameter).
  + **Stack< >:-** stack stores values in **Lifo**(Last in First Out) Style. It’s a pile(bunch) of object. It has **push** (adding on Top) and **pop & Peek**(Top Item and Remove) method. It’s in both generic and non-generic. Used to store temporary data. It doesn’t support indexer. It accept null and duplicate values in it. It store data in unique format. At run time it will autoresize. Capacity is used to check the no of element in stack. IEnumrable, ICollection and IClonable interface used. Used:- solving problems works on Recursion, single implemetetion, one pointer used to perform operation(Top), vertical collection. **>>**Top
  + **Queue< >:-** it represents in **FIFO(First in first out).** Used when u need to access item first. It contain adding item:- **Enque and** Removing Item and Returns(First) item:- **DeQueu** operation. It can resize automatically. IEnumrable, ICollection and IClonable interface used. Used: sequential Processing problems, complex implemetetion, two pointer used to perform operation (Front and Rear), horizontal collection.

Rear

Front

S24\_\_GenericCollections.cs

using OOPS\_Advanced\_\_Session;

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

namespace AdvancedTopic\_\_Session

{

class S24\_\_GenericCollections<int\_Data, string\_Data>

{

List<int\_Data> integerData;

List<string\_Data> stringData;

public S24\_\_GenericCollections()

{

integerData = new List<int\_Data>();

stringData = new List<string\_Data>();

}

public void AllMethods()

{

S24\_\_GenericCollections<int, string> string\_Integer = new S24\_\_GenericCollections<int, string>();

string\_Integer.AddToList(1, "Imran");

string\_Integer.AddToList(2, "Amit");

string\_Integer.AddToList(3, "Nijam");

string\_Integer.DisplayList();

Console.WriteLine("\nAftr Removing ");

string\_Integer.RemoveFromList(1, "Imran");

//stringdata.Remove("Shaikh");

string\_Integer.DisplayList();

Console.WriteLine("\n Inside Person Contact Method\*\*\*\*\*\*\*\*\*\*\*\*\*");

PersonContactManagement.CallingMethod();

Console.WriteLine("\*\*\*\*\*\*\*\*\*\* Inside the Que and Stack Problem \*\*\*\*\*\*\*\*\*\*\*");

Dictionary\_\_StackQue dictionary\_\_StackQue = new Dictionary\_\_StackQue();

dictionary\_\_StackQue.Push(56);

dictionary\_\_StackQue.Push(30);

dictionary\_\_StackQue.Push(70);

//dictionary\_\_StackQue.Push("Imran");

Console.WriteLine("Inserted Elements Are ");

dictionary\_\_StackQue.DisplayElements();

dictionary\_\_StackQue.Pop();

Console.WriteLine("After Pop.. ");

dictionary\_\_StackQue.DisplayElements();

dictionary\_\_StackQue.Peek();

Console.WriteLine();

Console.WriteLine("\*\*\*\*\*\*\* Executing Que \*\*\*\*\*\*\*\n");

dictionary\_\_StackQue.EndedQueue(56);

dictionary\_\_StackQue.EndedQueue(30);

dictionary\_\_StackQue.EndedQueue(70);

Console.WriteLine();

Console.WriteLine("Ended Que Elements Are: ");

dictionary\_\_StackQue.DisplayElements();

dictionary\_\_StackQue.DoubleQue();

Console.WriteLine();

Console.WriteLine("Doubled Que Elements Are:");

dictionary\_\_StackQue.DisplayElements();

}

void AddToList(int\_Data int\_Data, string\_Data string\_\_Data)

{

integerData.Add(int\_Data);

stringData.Add(string\_\_Data);

}

void RemoveFromList(int\_Data int\_Data, string\_Data string\_\_Data)

{

integerData.Remove(int\_Data);

stringData.Remove(string\_\_Data);

}

void DisplayList()

{

Console.WriteLine("\nInteger List Element:- ");

foreach (var element in this.integerData)

Console.WriteLine("\t{0}", element);

Console.WriteLine("\nString List Element:- ");

foreach (var element in this.stringData)

Console.WriteLine("\t{0}", element);

}

}

//object base list

class PersonContactManagement

{

public static void CallingMethod()

{

List<PersonContactClass> listPersonCity = new List<PersonContactClass>();

AddRecord(listPersonCity);

RetriveRecord(listPersonCity);

CheckTeenage(listPersonCity);

}

private static void AddRecord(List<PersonContactClass> listPersonCity)

{

listPersonCity.Add(new PersonContactClass("22223", "Amar", "Adarsh Nagar Pune", 8));

listPersonCity.Add(new PersonContactClass("44453", "Aniket", "Manipura Colony Mumbai", 8));

listPersonCity.Add(new PersonContactClass("45553", "Arti", "Kondhwa Khurd Latur", 44));

listPersonCity.Add(new PersonContactClass("88987", "Pratibha", "Pratibha Villa Mumbai", 18));

listPersonCity.Add(new PersonContactClass("22223", "Soham", "12 main strt Pune", 10));

listPersonCity.Add(new PersonContactClass("44453", "Vidyadhar", "120 main strt Beed", 19));

listPersonCity.Add(new PersonContactClass("45553", "Rehan", "Warali Mumbai", 22));

listPersonCity.Add(new PersonContactClass("88987", "Sumit", "112 main street Latur", 18));

Console.WriteLine();

}

public static void RetriveRecord(List<PersonContactClass> listPersonCity)

{

foreach (PersonContactClass Person in listPersonCity.FindAll(e => (e.Age < 10)).Take(2).ToList())

Console.WriteLine("Name:" + Person.Name + "\tAge:" + Person.Age);

}

public static void CheckTeenage(List<PersonContactClass> listPersonCity)

{

if (listPersonCity.Any(e => (e.Age >= 13 && e.Age <= 19)))

Console.WriteLine("Yes, We Have Some Tenneger ");

}

}

class Dictionary\_\_StackQue

{

public void Dictionary()

{

Dictionary<string, string> dictionary = new Dictionary<string, string>();

dictionary.Add("Number One", "This is Number 1");

dictionary.Add("Number Two", "This is Number 2");

dictionary.Add("Number Three", "This is Number 3");

dictionary.Add("Number Four", "This is Number 4");

dictionary.Add("Number Five", "This is Number 5");

//dictionary.Add(1, "This is Number 1");

//dictionary.Add(1,11);

//dictionary.Add("1",4);

//Accessing Particular Value

Console.WriteLine("Value of :-- " + dictionary["Number One"]);

//Console.WriteLine("Value of :-- " + dictionary[1]);

foreach (KeyValuePair<string, string> items in dictionary)

Console.WriteLine("Keys are: " + items.Key + "\tValues Are: " + items.Value);

if (dictionary.ContainsKey("Number One"))

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

else

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

//TryGetValue

string outValue;

dictionary.TryGetValue("Number Two", out outValue);

Console.WriteLine("Out Value is :" + outValue);

}

//QUE STACK

internal Node head;

internal void EndedQueue(int data)

{

Node node = new Node(data);

if (head == null)

head = node;

else

{

Node temp = head;

while (temp.next != null)

temp = temp.next;

temp.next = node;

Console.WriteLine("\n{0} Inserted in Q", node.data);

}

}

internal void Display()

{

Node temp = this.head;

if (temp == null)

{

Console.WriteLine("\n Que is Empty");

return;

}

Console.WriteLine("Data is : ");

while (temp != null)

{

Console.Write(temp.data + " ");

temp = temp.next;

}

Console.WriteLine("\n");

}

//Creating DeQ

internal void DoubleQue()

{

if (head == null)

Console.WriteLine("Que Empty");

else

Console.WriteLine("{0} Deque ", head.data);

head = head.next;

}

//STACK

private Node top;

public Dictionary\_\_StackQue()

{

this.top = null;

}

//Creating Stack

internal void Push(int value)

{

Node node = new Node(value);

if (this.top == null)

node.next = null;

else

node.next = this.top;

this.top = node;

Console.WriteLine("{0} push to stack", value);

}

internal void DisplayElements()

{

Node temp = this.top;

while (temp != null)

{

Console.WriteLine(temp.data + " ");

temp = temp.next;

}

}

//Pop And Peek

internal void Peek()

{

if (this.top == null)

{

Console.WriteLine("Stack is Empty");

return;

}

Console.WriteLine("{0} is top of stack", this.top.data);

}

internal void Pop()

{

if (this.top == null)

{

Console.WriteLine("stack is empty");

return;

}

Console.WriteLine("value Pop {0}", this.top.data);

this.top = this.top.next;

}

internal void IsEmpty()

{

while (this.top != null)

{

Peek();

Pop();

}

}

}

}

CREATE NEW CLASS FILE

using System;

using System.Collections.Generic;

using System.Text;

namespace OOPS\_Advanced\_\_Session

{

//List

public class PersonContactClass

{

public string SSN;

public string Name;

public string Address;

public int Age;

public PersonContactClass(string ssn, string name, string address, int age)

{

SSN = ssn;

Name = name;

Address = address;

Age = age;

}

}

//Stack Que

public class Node

{

public int data;

public Node next;

public Node(int data)

{

this.data = data;

}

}

}

OopsAdvanceSession.cs

using System;

namespace AdvancedTopic\_\_Session

{

class OopsAdvanceSession

{

static void Main(string[] args)

{

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

//S23\_\_NonGenericCollections arrayList = new S23\_\_NonGenericCollections();

//S23\_\_NonGenericCollections.PrintArrayList();

//arrayList.PrintSortedList();

//arrayList.HashTable();

//arrayList.Stack\_Queue\_Program();

//S24\_\_GenericCollections

S24\_\_GenericCollections<int, string> genericCollections = new S24\_\_GenericCollections<int, string>();

genericCollections.AllMethods();

Dictionary\_\_StackQue dictionary\_\_StackQue = new Dictionary\_\_StackQue();

dictionary\_\_StackQue.Dictionary();

}

}

}