**S23: Collections: Non 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.

* **Non Generic Collections: -** System. Collections namespace used. Its general purpose data structure that work on object references it can handle any type of data. It defines the set of interface and classes. Not type safe. Not secure. Called loosely type
  + **Array List:** - it’s similar to array. Its dynamic array means the size of array is not fixed increase and decrease at run time. No need to specify the size of array list. Inserting element in between we used insert Method. Its not type safe. Using size ( ) Method we can check array list. It’s always single dimensions. Capacity Property is used to check the collection no. It implement IEnumerable, ICollection, IList and IClonable Interface. Keys are predefined. Use: when we want some information from data then using indexvalue.
  + **Sorted List:** - sorted list stores key value pair. Automatically arrange element in ascending order of key by default. It’s a class. IEnumerable, ICollection, IDictionary and IClonable interface. Element is access by its key. It used 2 array one for key and another for value. Duplicate keys not allowed. Value of same or different type can store bcoz of non-generic collection. Cannot add different data type key. But value can.
  + **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

* + **Hashtable: -** store key and value pair it retrieves the values comparing hash values or hash code. You can explicitly define keys. Use: when we want some information from data then using key. Key value will organized using hashcode. Gethashcode used to return the hash code from hashtable. And it’s an integer value. Internally each key allocate hashcode. It has implemented hashing algorithm. It’s faster than array and array list. And data will be retrieved randomly.

S23\_\_NonGenericCollections.cs

using System;

using System.Collections;

namespace AdvancedTopic\_\_Session

{

class S23\_\_NonGenericCollections

{

void ArrayData()

{

string[] city = { "Panvel", "Pune", "Nanded", "Latur", "Beed", "Mumbai", "Dubai", 122.ToString(), 2.89.ToString() };

Console.Write($"Length of Array is : {city.Length} \n\nCities Are: \t");

foreach (string cities in city)

Console.Write(cities + " , ");

}

public static void PrintArrayList()

{

S23\_\_NonGenericCollections array = new S23\_\_NonGenericCollections();

ArrayList arrayList = new ArrayList()

{102,"Imran","Shaikh",9000998877,8.33};

Console.WriteLine(arrayList.Capacity);

arrayList.Insert(2, "Pune");

arrayList.Add("Maharashtra");

Console.WriteLine("Array List are: --");

foreach (var arrayData in arrayList)

Console.Write(arrayData + " ");

array.PrintArrayLists();

array.ArrayData();

}

void PrintArrayLists()

{

ArrayList privateArray = new ArrayList()

{ 1, "Sahiba", "9899889900", 80.90, 400,2,"Abhilasha","9899889988", 89.90, 40000,3,"Amit","989985988", 89.99, 4000 };

Console.WriteLine("\n\nArray Data Is:- ");

foreach (object privateData in privateArray)

Console.Write(privateData + " ");

}

public void PrintSortedList()

{

SortedList sortedList = new SortedList()

{

{0, "Sahiba" } ,{ 1, "Rohan"},{ 2, "Keshav"},{3, "Santoshi" },{4, "Venkata" },{5, "Akhil" }

};

sortedList.Add(6, "");

sortedList.Add(7, 333);

sortedList.Add(8, "Imran");

//sortedList.Add("ee", "Abhilasha"); Error

//Accessing Specific Value

var value = sortedList[5];

Console.WriteLine("\nValue is : " + value);

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

for (int i = 0; i < sortedList.Count; i++)

Console.WriteLine($"Key: {sortedList.GetKey(i)}, Value:- {sortedList.GetByIndex(i)}");

foreach (DictionaryEntry data in sortedList)

Console.WriteLine($"Key: {data.Key}, Value:- {data.Value}");

}

public void HashTable()

{

Hashtable hashTable = new Hashtable()

{

{ "Id", 1123}, { "Name", "Imran" },

{"Roll Number", 100 } , { "Address", "Pune"},

{ "City", "Pune"},

};

hashTable.Add(102, 1123);

hashTable.Add(100.3, 1123.99);

if (hashTable.ContainsValue("Imran"))

Console.WriteLine("This Student is Alredy Exist");

else

Console.WriteLine("Not Present Please Add");

//Accessing Particular Value

var value = hashTable[102];

Console.WriteLine("\nValue is : " + value);

Console.WriteLine("------ Hashtable Data:- ");

foreach (object key in hashTable.Keys)

{

Console.WriteLine(key + ": " + hashTable[key]);

Console.WriteLine("HashCode is: " + key.GetHashCode());

}

}

public void Stack\_Queue\_Program()

{

Stack stack = new Stack();

stack.Push("Imran");

stack.Push("Amit");

stack.Push("Akansha");

stack.Push("Anirudh");

stack.Push("Shubham");

stack.Push("Imran");

stack.Push(112);

stack.Push(33.55);

stack.Push(true);

Console.WriteLine("Top Stack Element is:--" + stack.Peek() + "\n");

if (stack.Contains("Simran"))

Console.WriteLine("Present");

else

Console.WriteLine("Not Present");

Console.WriteLine("Stack Values Are:\*\*\*\*\*\*\*\*");

foreach (var stackData in stack)

Console.WriteLine(stackData);

Console.WriteLine("\nQue Data is :----");

Queue queue = new Queue();

queue.Enqueue("Anikesh");

queue.Enqueue("Pune");

queue.Enqueue(55);

queue.Enqueue(false);

queue.Enqueue(null);

queue.Enqueue("Anikesh");

queue.Enqueue(45.55);

foreach (object queData in queue)

Console.WriteLine(queData);

Console.WriteLine("\nDeQue Data:---");

Console.WriteLine(queue.Dequeue());

foreach (object queData in queue)

Console.WriteLine(queData);

}

}}

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();

}

}}