Collections and Generics

- Introduction to Collections
- Introduction to Generics

Collections

- The .NET Framework provides specialized classes for data storage and retrieval.
- These classes provide support for stacks, queues, lists, hash tables etc.
- Most collection classes implement the specific interfaces.

Collection Namespaces

- Collection classes are defined as part of the
  - System.Collections

- Most collection classes derive from the following interfaces
- ICollection,
- IComparer,
- IEnumerable,
- IList,
- IDictionary,
- IDictionaryEnumerator.

IEnumerable Interface

- An enumerator is an object that provides a forward, read-only cursor for a set of items.
- The IEnumerable interface has one method called the GetEnumerator() method.
- This method returns an object that implements the IEnumerator interface.

IEnumerable Interface

- The code snippet below illustrates how an enumerator can be used to iterate through a list or collection of items.

```
String[] names = new String[2] {"GNR", "Pune"};
IEnumerator e =names.GetEnumerator();
e.MoveNext();
Console.WriteLine(e.Current);
```

- Note that the GetEnumerator() method returns an enumerator object each time it is called.

ICollection

- The following are the classes that are derived from the ICollection interface.

- System.Collections.Stack
- System.Collections.Queue
- System.Collections.BitArray
- System.Collections.Specialized.NameValueCollection

IDictionary

- The IDictionary interface represents collections that have name value pairs.

- The collections that inherit the IDictionary interface include:

- System.Collections.SortedList
- System.Collections.Hashtable
- System.Collections.Specialized.HybridDictionary
- System.Collections.Specialized.ListDictionary

IList

- The IList interface represents collections that only have value. The following are the classes that extend this interface.

- System.Array
- System.Collections.ArrayList
- System.Collections.Specialized.StringCollection

ArrayList

- The ArrayList class is a dynamic array of heterogeneous objects.

- Methods of ArrayList class :
- Add()
- Rmove()
- Sort()
- Reverse()

- Properties of ArrayList Class :
- Count
- Capacity

StringCollection

- The StringCollection class implements the IList interface and is like an ArrayList of strings.
- The following code example shows how we can work with a StringCollection class.

```
using System.Collections.Specialized;
StringCollection stringList = new StringCollection();
stringList.Add("PUNE");
stringList.Add("GNR");
stringList.Add("HYD");
```

StringDictionary

- Similar to the StringCollection class we have the StringDictionary class, which is just a Hashtable that has its keys as strings only.
- Remember that a Hashtable can contain any object type in its key.
- The following code shows how we can work with a StringDictionary class.

```
StringDictionary stringList = new StringDictionary();
stringList.Add("G", "Gandhinagar");
stringList.Add("P", "Pune");
stringList.Add("H", "Hyderabad");
```

Stack

- The Stack class is one that provides a Last-in-First-out (LIFO) collection of items of the System.Object type.
- The last added item is always at the top of the Stack and is also the first one to be removed.
- The following code sample shows how we can use a Stack class for LIFO operation on its collection of items.

```
Stack stackObject = new Stack();
 stackObject.Push("GNR");
 stackObject.Push("PUN");
 stackObject.Push("HYD");
```

- The Push() method is responsible for storing items in the Stack and the method Pop() removes them one at a time from the top of the Stack.

Queue

- The Queue is a data structure that provides a First-in-First-out (FIFO) collection of items of the System.Object type.
- The newly added items are stored at the end or the rear of the Queue and items are deleted from the front of the Queue.
- The following code shows how the Queue class can be used.

```
Queue queueObject = new Queue();
queueObject.Enqueue("GNR");
queueObject.Enqueue("PUN");
queueObject.Enqueue("HYD");
```

- The Enqueue() method is responsible for storing items at the rear of the Queue and the method Dequeue() removes them one at a time from the front of the Queue.

Hashtable

- The Hashtable provides a faster way of storage and retrieval of items of the object type.
- The Hashtable class provides support for key based searching.
- These keys are unique hash codes that are unique to a specific type.
- The following code snippet shows how we can use a Hashtable class.

```
Hashtable hashTable = new Hashtable();
hashTable.Add(1, "GNR");
hashTable.Add(2, "HYD");
hashTable.Add(3, "PUN");
```

SortedList

- The SortedList class allows items of the System.Object type to be placed in the collection using key value pairs and, at the same time, supports sorting.

- The following code shows how we can use a SortedList.

```
SortedList sortedList = new SortedList();
sortedList.Add(1, "GNR");
sortedList.Add(3, "PUN");
sortedList.Add(2, "HYD");
```

Selecting Collection class

- Do you need a sequential list where the element is typically discarded after its value is retrieved?
- If yes, consider using the Queue class that provides first-in, first-out (FIFO) behavior.
- Consider using the Stack class that provides last in first-out (LIFO) behavior.

Selecting Collection class (contd)

- Do you need to access each element by index or key?
- If yes go for the following classes :

- ArrayList
- StringCollection
- HashTable
- SortedList

Collection and Object Intializers

- Instead of adding items to the collection using add method, you can also add many values to your collection in a single line of code using collection initializers.

```
//Collection initializers
List<int> myList = new List<int> { 1, 2, 3, 4 };

//using object initializers
Car myCar1 = new Car() { Make = "Hyundai", Model = "i20" };
Car myCar2 = new Car() { Make = "Honda", Model = "City" };
```

Generic

- Generic in C# allow us to use Generic Version of Collection classes.
- The Generic Collection restrict the Type of Object list that Collection holds to one Type.
- This prevent useless overheads of Boxing and Unboxing that is faced by List, which can hold any Type of Object list derived from Object Class.

Namespace for Generics

- Generic Collection classes are defined as part of the
  - System.Collections.Generic namespace.

Advantages of Generic

- The following are the features of Generic Collection.

- Generic Collection provide Type safety feature.
- Restricted to Type specific members.
- Reduce number of overhead relevant to boxing and unboxing.
- Generic Collections are faster than Non Generic collections such as ArrayList.
- No need to resize manually, it grows dynamically.

List

- The List class is the generic Class that correspond to non-Generic ArrayList class.
- The List Class have method such as contains, IndexOf, LastIndexOf, Remove, Equals, Sort, CompareTo etc.

- The features of Generic List Class are as follows:

- Their size can be dynamically increased.
- Object List in the Collection are not necessarily in the sorted form.
- The List are better in perfomance and are Type Safe.
- The indexes in the List are zero-based.

Questions ?

Thank You