**Collection in C#**

Microsoft has introduced a concept called collections that consists of set of classes that provides array functionality in a superior manner.  
  
There are the following 2 types of collections:

1. **Non-Generic**
2. **Generic**

**Non-generic                          Generic**

  ArrayList     ------------->          List

  HashTable  ------------->          Dictionary

  SortedList   ------------->          SortedList

  Stack           ------------->          Stack

  Queue         ------------->          Queue  
  
**1. Non-Generic**

1. Each element can represent a value of a different type.
2. Array Size is not fixed.
3. Elements can be added / removed at runtime.

**ArrayList**

1. Arraylist is a class that is similar to an array, but it can be used to store values of various types.
2. An Arraylist doesn't have a specific size.
3. Any number of elements can be stored.

**HashTable**

HashTable is similar to arraylist but represents the items as a combination of a key and value.

**SortedList:**

1. Is a class that has the combination of arraylist and hashtable.
2. Represents the data as a key and value pair.
3. Arranges all the items in sorted order.

# **C# - Stack Class**

It represents a last-in, first out collection of object. It is used when you need a last-in, first-out access of items. When you add an item in the list, it is called pushing the item and when you remove it, it is called popping the item

**Queue:**   
It represents a first-in, first out collection of object. It is used when you need a first-in, first-out access of items. When you add an item in the list, it is called **enqueue**, and when you remove an item, it is called **deque**.

## **Iterators**

An iterator is used to perform a custom iteration over a collection. An iterator can be a method or a get accessor. An iterator uses a [yield return](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/yield) statement to return each element of the collection one at a time.

# **Generic Collections**

Generic Collections work on the specific type that is specified in the program whereas non-generic collections work on the object type.

1. Specific type
2. Array Size is not fixed
3. Elements can be added / removed at runtime.

**List:**

You have already learned about ArrayList in the previous section. An ArrayList resizes automatically as it grows. The List<T> collection is the same as an ArrayList except that List<T> is a generic collection whereas ArrayList is a non-generic collection.

### **Points to Remember :**

1. List<T> stores elements of the specified type and it grows automatically.
2. List<T> can store multiple null and duplicate elements.
3. List<T> can be assigned to **IList<T>** or **List<T>** type of variable. It provides more helper method When assigned to List<T> variable
4. List<T> can be access using indexer, for loop or foreach statement.
5. LINQ can be use to query List<T> collection.
6. List<T> is ideal for storing and retrieving large number of elements.

**Dictionary:**

A Dictionary class is a data structure that represents a collection of keys and values pair of data. The key is identical in a key-value pair and it can have at most one value in the dictionary, but a value can be associated with many different keys.

This class is defined in the System.Collections.Generic namespace, so you should import or using System.Collections.Generic namespace.

1. A Dictionary stores Key-Value pairs where the key must be unique.
2. Before adding a KeyValuePair into a dictionary, check that the key does not exist using the ContainsKey() method.
3. Use the TryGetValue() method to get the value of a key to avoid possible runtime exceptions.
4. Use a foreach or for loop to iterate a dictionary.
5. Use dictionary indexer to access individual item.
6. Use custom class that derives IEqualityComparer to compare object of custom class with Contains() method.

Ref:

<http://www.c-sharpcorner.com/UploadFile/736bf5/collection-in-C-Sharp/>