**SECTION A: Collection Overview**

1. Understand the System.Collections namespace.
2. Explain collection types.
3. Explain the interfaces used in collection types.

* Namespace -- used to define collection of classes and interfaces.

Ex: System.Collections is a namespace holding groups of interfaces and classes that define objects: lists, arrays, queues, stacks.

* Collection --- group of objects/data sharing same characteristics.

CollectionBase class is defined from System.Collection namespace. Holds:

COLLECTION CLASSES:

* 1. ArrayList ---- implements several interfaces. Size is dynamic and changeable when new items are added.
  2. Hashtable: set of keys that reference value pairs in data lists.
  3. Queue: symbolizes first-in, first-out object collection. (FIFO)
  4. Stack: signifies last-in, first-out collection. (LIFO)

COLLECTION INTERFACE:

* Contains templates for a specific collection, that describes how data is collected and stored. ILIST and IDictionary inherit the ICollection interface.
  + 1. ICollection – an interface for collections of similar objects, and which all collection classes can access.
    2. IList – successor of ICollection and is the base interface of all lists that hold the same types of data. Lists can be fixed, variable lengths, or read-only.
    3. IDictionary --- the base interface for collection classes that use key/value-based collections.

COLLECTION ORGANIZATION:

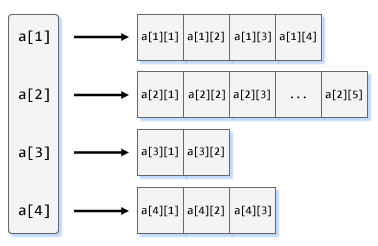
1. Ordered collection – items are stored in specific order. User must retrieve the correct data by reaching its specific spot.
2. Indexed collection --- items are identifiable with a numeric index.
3. Keyed collection ---key values identify items.

**SECTION B: ArrayList**

1. Understand what an Arraylist is and what it does.
2. Create an Arraylist.
3. Store and retrieve items in an arraylist.
4. Implement an Arraylist in a program.

ARRAY:

* Array: container holding fixed ordered quantity of objects.
* Index-based collection
* Fixed size; and one data type.
* ReDim keyword: to resize the array. Dimension cannot be changed.
* Preserve keyword: to preserve content when resizing.
* Single dimensional array: data is stored in one sequence.
* Multi-dimensional array: collection of single dimensional arrays. In the first a[1], an entire array is stored.



|  |
| --- |
|  |
| Multi Dimensional Array |

ARRAYLIST:

* A class in .NET that holds items in the order they are added.
* Index-based collection.
* Dynamic size; and more than one data type.

ARRAYLIST PROPERTIES:

* Capacity: maximum elements it can hold
* Count: actual number it can hold
* IsFixedSize: shows if size is fixed or dynamic
* Item: helps identify value in arraylist.

ARRAYLIST METHODS:

* **Add:**  to add a value at the end of the arraylist.
* **Insert:** puts an element at a specified location in the arraylist.
* **AddRange:** more than one element can be added at once.
* **InsertRange:** many elements can be added at a specified place in the arraylist.
* **Clear:** removes all elements
* **Contains:**  checks to see if a value is in the arraylist.
* **Remove:**  specific element can be removed
* **RemoveAt:** In removeat method, an element at a specific location (index) can be removed from a particular index.
* **RemoveRange:** Through this method, a set of elements within a range can be removed from the ArrayList.
* **ToArray:**  to copy the elements of the System.Collections.ArrayList to a new array of System.Object type.

TO ESTABLISH:

Dim arrayname As New System.Collections.ArrayList()

Dim fruits As New System.Collections.ArrayList()

ADD:

fruits.Add(“apple”) ------------------🡪 fruits(0)  
fruits.Add(“orange”) ---------------🡪 fruits(1)

* Item Property Indexer: retrieves item property index number given to an element.

REMOVE:

fruits.remove(“apple”) -------🡪 use if unsure of index number because of large data.   
fruits.removeat(0)

COUNT: fruits.Count -------🡪 returns current size

**SECTION C: Stacks and Queues:**

1. Define and use Stack.
2. Define and use Queue.
3. Store and retrieve items in Stack and Queue.

Dim Stackobject As New Stack()

Dim Queueobject As New Queue()

STACK: (class)

* Stack **can hold any type of data casted to an object.**
* EXCEPTIONS:
  + Overflow: amount of elements is greater than limit given.
  + Underflow: occurs when trying to pop an empty stack.
* Insertion and deletion happen at same end (top) (stack of books) LIFO
* Stackobject.Pop(item) ---- to remove item from top
* Stackobject.Push(item) ---- to add item on top

QUEUE: (class)

* A data structure organized with FIFO principle.
* Insertion and deletion happens at different ends (people in line) FIFO
* Queueobject.Dequeue(item) ------ to remove item from front.
* Queueobject.Enqueue(item) ----- to add item to the rear

**SECTION D: Hashtable**

1. Understand Hashing.
2. Define a Hashtable.
3. Store and retrieve items in a Hashtable.
4. Understand where a Hashtable must be used.

DEFINITION:

* Hashtable is a **class** and **data** **structure** that uses key to store, search, retrieve, insert, and delete data in an array.
* Hashtable: defined by any hash function (linear, or non-linear): h(k) = x, where h = hash function, k = key variable, x = equals data value.

Ex: h(1) = 1 means key = 1 produces INTEGER hash code = 1.

* System.Collections.Hashtable

HASHING:

* Method: enter target value with key to go straight to target. Ex: key = 8 for target = 80.

DECLARATION:

Dim HashTableObject As New Hashtable()

ADD:

HashTableObject.Add(key, value)

RETRIEVE:

HashTableObject(key)

LINEAR:

* Data does not have to be ordered.
* Method: Scans data from start to finish.
* Not efficient, time-consuming.

BINARY:

* Data must be in order
* Method:
  1. Starts from array center.
  2. If center value is less than data value, lesser values are ignored -🡪 program picks a new center in the latter half of array.
  3. If center value is greater than data value, greater values are ignored 🡪 new center in first half of the array.
  4. Same method continues.
* Efficient because some values are eliminated at each step.
* An unhandled exception of type 'System.InvalidCastException' occurred in QuizHashtable.exe
* Additional information: Unable to cast object of type 'System.Collections.DictionaryEntry' to type 'System.Collections.Hashtable'.
* 