**Oriette Ramos**

**Lesson Plan 27**

**ISTA-220**

1. What is an enumerable collection?  
   **An enumerable collection is a collection that implements the System.Collections.IEnumerable interface.**
2. What properties and methods does the IEnumerable interface contain?  
   **The IEnumerable interface contains a single method called GetEnumerator.**
3. What properties and methods does the IEnumerator interface contain?  
   **The IEnumerator interface specifies the following property and methods:**
4. object Current { get; }
5. bool MoveNext();

void Reset();

1. What values does the MoveNext() method return? What does it do?  
   **The MoveNext() method returns either true or false. You call the MoveNext() method to move the pointer down to the next (first) item in the list; the MoveNext() method should return true if there actually is another item and false if there isn’t. .**
2. What values does the Reset() method return? What does it do?  
   **The Reset() method does not return a value. You use the Reset method to return the pointer back to before the first item in the list.**
3. Are IEnumerable and IEnumerator type safe? Why or why not? If not, how do you implement type safety?  
   **The Current property of the IEnumerator interface exhibits non-type-safe behavior in that it returns an object rather than a specific type. However, the Microsoft .NET Framework class library also provides the generic IEnumerator<T> interface, which has a Current property that returns a T instead. Likewise, there is also an IEnumerable<T> interface containing a GetEnumerator method that returns an Enumerator<T> object. Both of these interfaces are defined in the System.Collections.Generic namespace, and you should make use of these generic interfaces rather than the nongeneric versions when you define enumerable collections.**
4. Why don't recursive methods retain state when used with data structures like binary trees?  
   **Recursive algorithms, such as that used when walking a binary tree, do not lend themselves to maintaining state information between method calls in an easily accessible manner because the data within the recursive calls is lost when they lose scope.**
5. How do you define an enumerator?  
   **You define an enumerator be defining a class that implements the IEnumerator<T> interface and the type parameter must be a valid type for the object that the class enumerates, so it must be constrained to implement the IComparable<T> interface.**
6. What is an iterator?  
   **An iterator is a block of code that yields an ordered sequence of values. An iterator is not actually a member of an enumerable class; rather, it specifies the sequence that an enumerator should use for returning its values. In other words, an iterator is just a description of the enumeration sequence that the C# compiler can use for creating its own enumerator.**
7. What does yield do?  
   **The yield keyword indicates the value that should be returned by each iteration. If it helps, you can think of the yield statement as calling a temporary halt to the method, passing back a value to the caller.**