# IEnumerable

- The IEnumerable interface represents a group of elements.
- It is the parent interface of all types of collections.
- It is the parent of ICollection interface, which is implemented by other interfaces such as IList, IDictionary etc.

System.Collections.Generic.IEnumerable<T>

```
public interface IEnumerable<out T> : IEnumerable
{
    IEnumerator<T>  GetEnumerator();
}
```

System.Collections.IEnumerable

```
public interface IEnumerable
{
    IEnumerator  GetEnumerator();
}
```

System.Collections.Generic.IEnumerable<T>

```
IEnumerable<T>  referenceVariable = new  List<T>( );
```

System.Collections.IEnumerable

```
IEnumerable  referenceVariable = new  ArrayList( );
```

# IEnumerator

- The IEnumerator interface is meant for readonly and sequential navigation of group of elements.
- IEnumerator is used by foreach loop internally.
- IEnumerable interface has a method called GetEnumerator that returns an instance of IEnumerator.
- IEnumerator by default starts with first element; MoveNext( ) method reads the next element; and the "Current" property returns the current element based on the current position.

**System.Collections.Generic.IEnumerator<T>**

```
public interface IEnumerator<out T> : IDisposable, IEnumerator
{
    T Current { get; }
}
```

**System.Collections.IEnumerator**

```
public interface IEnumerator
{
    object Current { get; }
    bool MoveNext();
    void Reset();
}
```

### System.Collections.Generic.IEnumerator<T>

```
IEnumerator<T>  referenceVariable = new  List<T>( ).GetEnumerator( );
```

### System.Collections.IEnumerator

```
IEnumerator  referenceVariable = new  ArrayList( ).GetEnumerator( );
```