Collections und Generics

Proseminar *Objektorientiertes Programmieren mit .NET und C#*

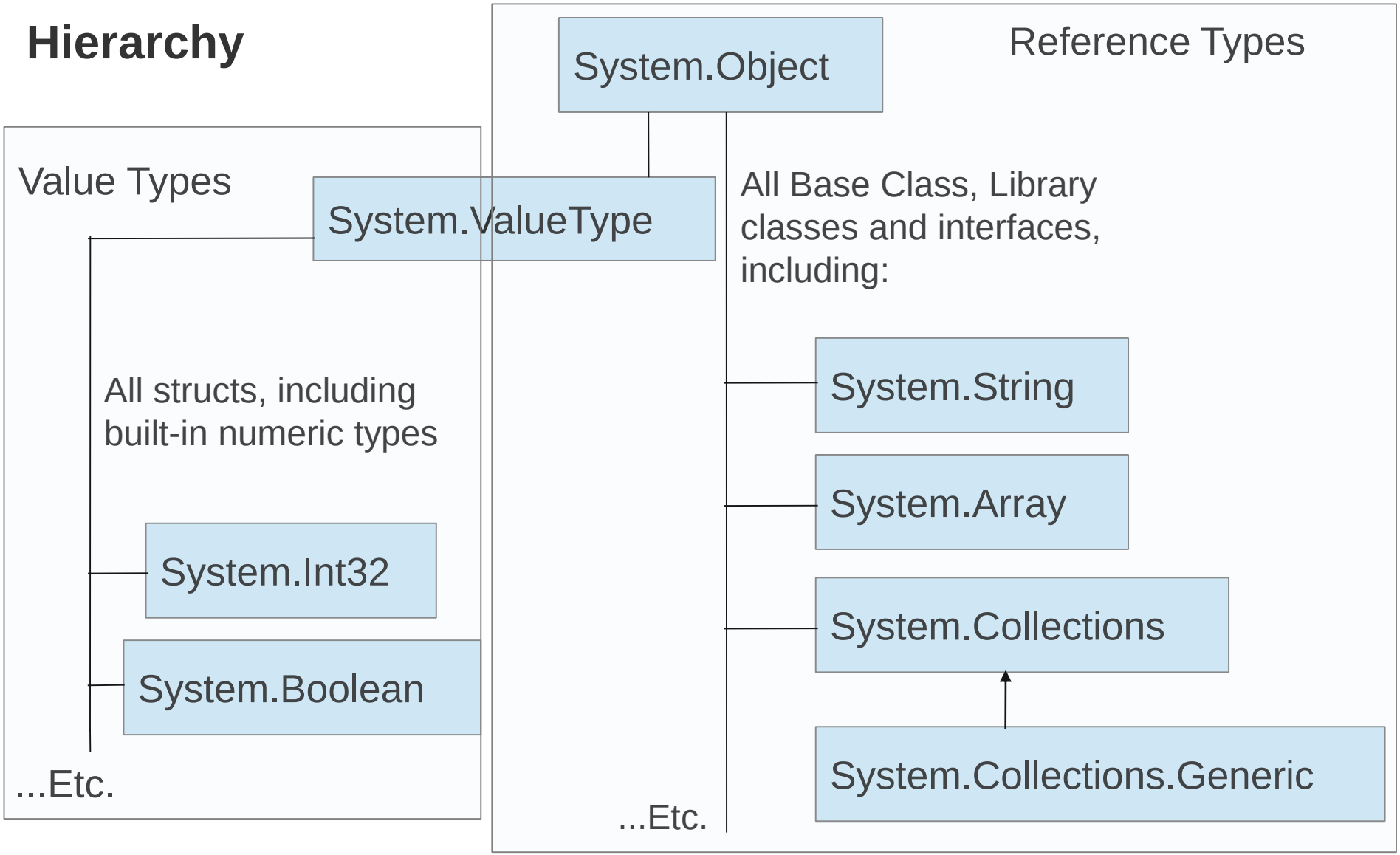
Nadim Yonis

Institut für Informatik
Software & Systems Engineering

Agenda

- Collections
- Standard Collections
- Comparer
- Generics
- Generic Collections
- Iterators

Hierarchy


System.Object
Reference Types
Value Types
System.ValueType
All Base Class, Library classes and interfaces, including:
All structs, including built-in numeric types
System.String
System.Array
System.Int32
System.Collections
System.Boolean
System.Collections.Generic
...Etc.
...Etc.

Collections

- Container
- Usually contains elements of the same type
- Dynamic

Collections

- Container
- Usually contains elements of the same type
- Dynamic

- Standard Collections

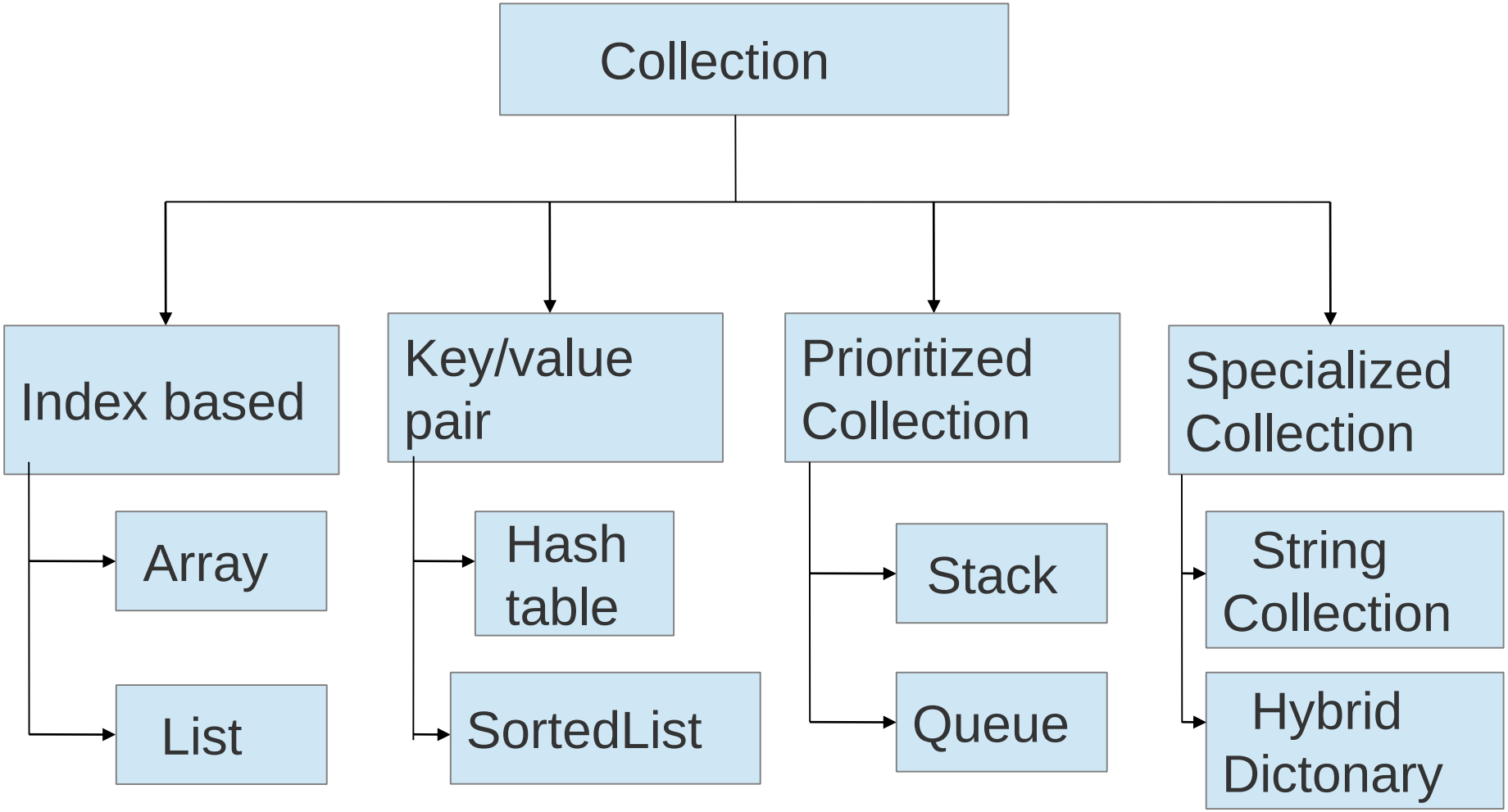
Collections

- Container
- Usually contains elements of the same type
- Dynamic

- Standard Collections

- Generic Collections

Collections



System.Collections Namespace

msdn

.NET Framework 2.0 | Other Versions | 30 out of 40 rated this helpful - Rate this topic

The **System.Collections** namespace contains interfaces and classes that define various collections of objects, such as lists, queues, bit arrays, hash tables and dictionaries.

Classes

Class	Description
ArrayList	Implements the IList interface using an array whose size is dynamically increased as required.
BitArray	Manages a compact array of bit values, which are represented as Booleans, where **true** indicates that the bit is on (1) and **false** indicates the bit is off (0).
CaseInsensitiveComparer	Compares two objects for equivalence, ignoring the case of strings.
CaseInsensitiveHashCodeProvider	Supplies a hash code for an object, using a hashing algorithm that ignores the case of strings.
CollectionBase	Provides the **abstract** base class for a strongly typed collection.
Comparer	Compares two objects for equivalence, where string comparisons are case-sensitive.
DictionaryBase	Provides the **abstract** base class for a strongly typed collection of key/value pairs.
Hashtable	Represents a collection of key/value pairs that are organized based on the hash code of the key.
Queue	Represents a first-in, first-out collection of objects.
ReadOnlyCollectionBase	Provides the **abstract** base class for a strongly typed non-generic read-only collection.
SortedList	Represents a collection of key/value pairs that are sorted by the keys and are accessible by key and by index.
Stack	Represents a simple last-in-first-out (LIFO) non-generic collection of objects.

System.Collections Namespace

msdn

.NET Framework 2.0 | Other Versions | 30 out of 40 rated this helpful - Rate this topic

The **System.Collections** namespace contains interfaces and classes that define various collections of objects, such as lists, queues, bit arrays, hash tables and dictionaries.

Classes

	Class	Description
	ArrayList	Implements the IList interface using an array whose size is dynamically increased as required.
	BitArray	Manages a compact array of bit values, which are represented as Booleans, where **true** indicates that the bit is on (1) and **false** indicates the bit is off (0).
	CaseInsensitiveComparer	Compares two objects for equivalence, ignoring the case of strings.
	CaseInsensitiveHashCodeProvider	Supplies a hash code for an object, using a hashing algorithm that ignores the case of strings.

ArrayList Class

msdn

.NET Framework 2.0 | Other Versions | 31 out of 79 rated this helpful - Rate this topic

Implements the IList interface using an array whose size is dynamically increased as required.

Namespace: System.Collections
Assembly: mscorlib (in mscorlib.dll)

Syntax

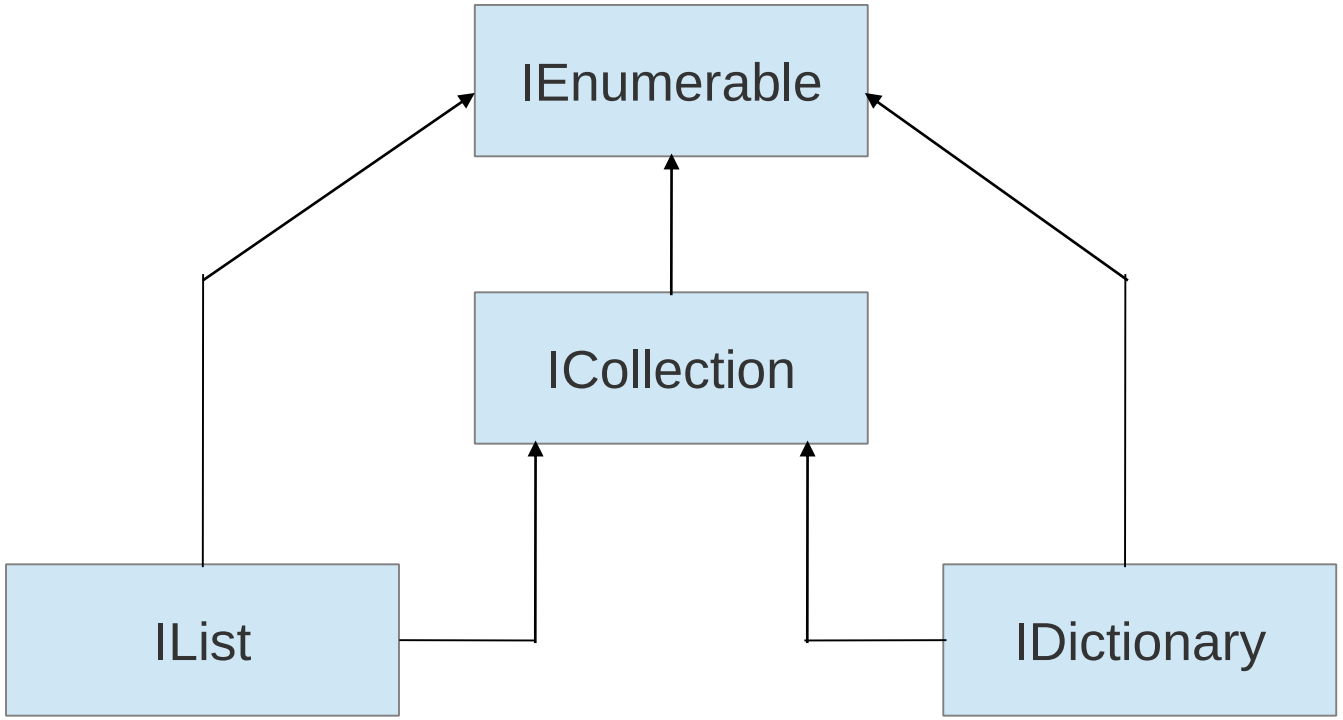
C# | C++ | VB

```
[SerializableAttribute]
[ComVisibleAttribute(true)]
public class ArrayList : IList, ICollection, IEnumerable,
        ICloneable
```

Interfaces

Interface	Purpose
ICollection	Defines size, enumerators and synchronized methods for all collections
IEnumerable	Exposes the enumerator, which supports a simple iteration over the collection
IList	Represents a collection of objects that can be individually accessed by index
IDictionary	Represents a collection of key/value pairs

Hierarchy



ArrayList

ArrayList

```
ArrayList liste = new ArrayList();
int index = liste.Add("Conie");
```

```
ArrayList liste = new ArrayList() {"Aachen", "Bonn",
                                   "Köln", "Düsseldorf" };
```

ArrayList

```
ArrayList liste = new ArrayList();
int index = liste.Add("Conie");
```

```
ArrayList liste = new ArrayList() {"Aachen", "Bonn",
                                   "Köln", "Düsseldorf" };
```

```
public interface IComparable {
  int CompareTo(object obj);
}
```

ArrayList

```
ArrayList liste = new ArrayList();
int index = liste.Add("Conie");
```

```
ArrayList liste = new ArrayList() {"Aachen", "Bonn",
                                   "Köln", "Düsseldorf" };

liste.Sort();
```

```
public interface IComparable {
  int CompareTo(object obj);
}
```

Interface	Purpose
ICollection	Defines size, enumerators and synchronized methods for all collections
IEnumerable	Exposes the enumerator, which supports a simple iteration over the collection
IList	Represents a collection of objects that can be individually accessed by index
IDictionary	Represents a collection of key/value pairs
IComparer	Exposes a method that compares two objects

Comparer

```
public class Demo : IComparable {
  public int Value {get; set;}
  public Demo(int value) {
    Value = value;
  }
  public int CompareTo(object obj) {
    if (((Demo)obj).Value < Value)
      return 1;
    else if (((Demo)obj).Value == Value)
      return 0;
    return -1;
  }
}
```

Standard vs Generic Collections

Standard Collections manage objects

Using Generic Collections is generally recommended

- Immediate benefit of type safety
- Better Performance when elements are value types (no need for boxing)
- Code is easier to read

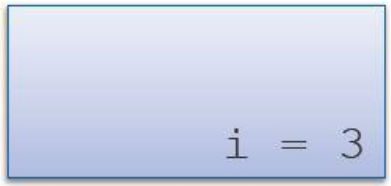
Boxing/Unboxing

Werttypen können mittels Cast in Referenztypen gepackt werden.

```
object o = (object)3;
```

Boxing

```
int i = (int)o;
```



Unboxing

Unterschied: Verweis- und Werttypen

- **Werttypen**
  - Ein Wert-Typ wird auf dem **Stack** gespeichert. Eine Instanziierung ist für eine Instanz eines Wert-Typs nicht notwendig. Eine Zuweisung erzeugt eine **Kopie des Wertes**, bei Vergleichen wird auf **Wertidentität** geprüft.

- **Verweistypen**
  - Ein Verweistyp wird auf dem **Heap** gespeichert, auf dem Stack gibt es einen Verweis auf die Speicheradresse im Heap. Eine Instanziierung ist erforderlich, um eine Instanz nutzen zu können. Eine Zuweisung erzeugt nur eine **Kopie des Zeigers**, bei Vergleichen wird auf die **Identität der Zeiger** geprüft.

Generics

- Java
  - Generics only exist in source code
  - Java Compiler 'Erases' the generic type information

- .NET
  - Generics are implemented at runtime
  - Generic type information is not lost

System.Collections.Generic Namespace

.NET Framework 2.0 | Other Versions | 56 out of 168 rated this helpful - Rate this topic

The **System.Collections.Generic** namespace contains interfaces and classes that define generic collections, which allow users to create strongly typed collections that provide better type safety and performance than non-generic strongly typed collections.

Classes

Class	Description
Comparer	Provides a base class for implementations of the IComparer generic interface.
Dictionary	Represents a collection of keys and values.
Dictionary.KeyCollection	Represents the collection of keys in a Dictionary. This class cannot be inherited.
Dictionary.ValueCollection	Represents the collection of values in a **Dictionary**. This class cannot be inherited.
EqualityComparer	Provides a base class for implementations of the IEqualityComparer generic interface.
KeyedByTypeCollection	Provides a collection whose items are types that serve as keys.
KeyNotFoundException	The exception that is thrown when the key specified for accessing an element in a collection does not match any key in the collection.
LinkedList	Represents a doubly linked list.
LinkedListNode	Represents a node in a LinkedList. This class cannot be inherited.
List	Represents a strongly typed list of objects that can be accessed by index. Provides methods to search, sort, and manipulate lists.
Queue	Represents a first-in, first-out collection of objects.

...Etc.

EqualityComparer	Provides a base class for implementations of the IEqualityComparer generic interface.
KeyedByTypeCollection	Provides a collection whose items are types that serve as keys.
KeyNotFoundException	The exception that is thrown when the key specified for accessing an element in a collection does not match any key in the collection.
LinkedList	Represents a doubly linked list.
LinkedListNode	Represents a node in a LinkedList. This class cannot be inherited.
List	Represents a strongly typed list of objects that can be accessed by index. Provides methods to search, sort, and manipulate lists.
Queue	Represents a first-in, first-out collection of objects.

List Generic Class

msdn

.NET Framework 2.0 | Other Versions | 58 out of 186 rated this helpful - Rate this topic

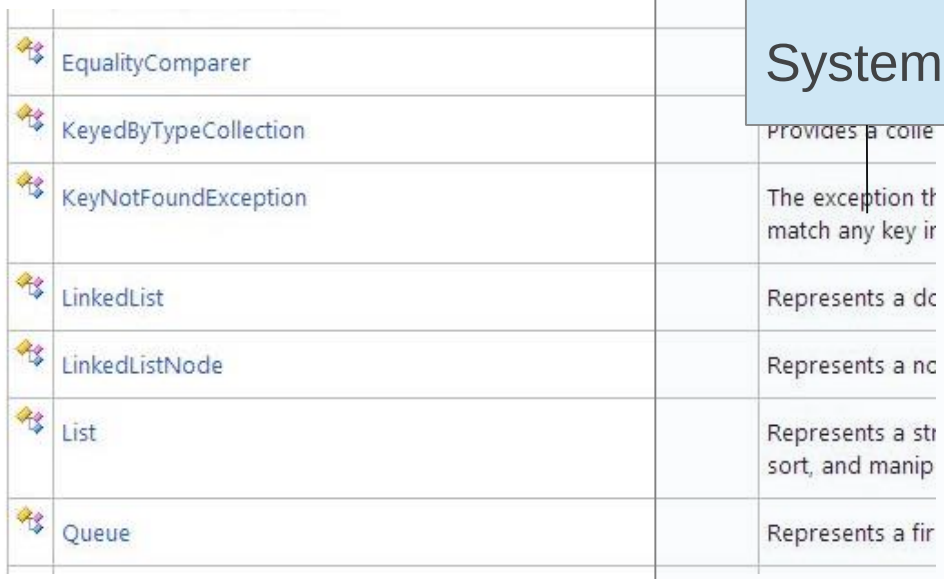
Represents a strongly typed list of objects that can be accessed by index. Provides methods to search, sort, and manipulate lists.

Namespace: System.Collections.Generic
Assembly: mscorlib (in mscorlib.dll)

Syntax

C# | C++ | VB

```
[SerializableAttribute]
public class List<T> : IList<T>, ICollection<T>,
        IEnumerable<T>, IList, ICollection, IEnumerable
```



List Generic Class

.NET Framework 2.0 | Other Versions | 58 out of 186 rated this helpful - Rate this

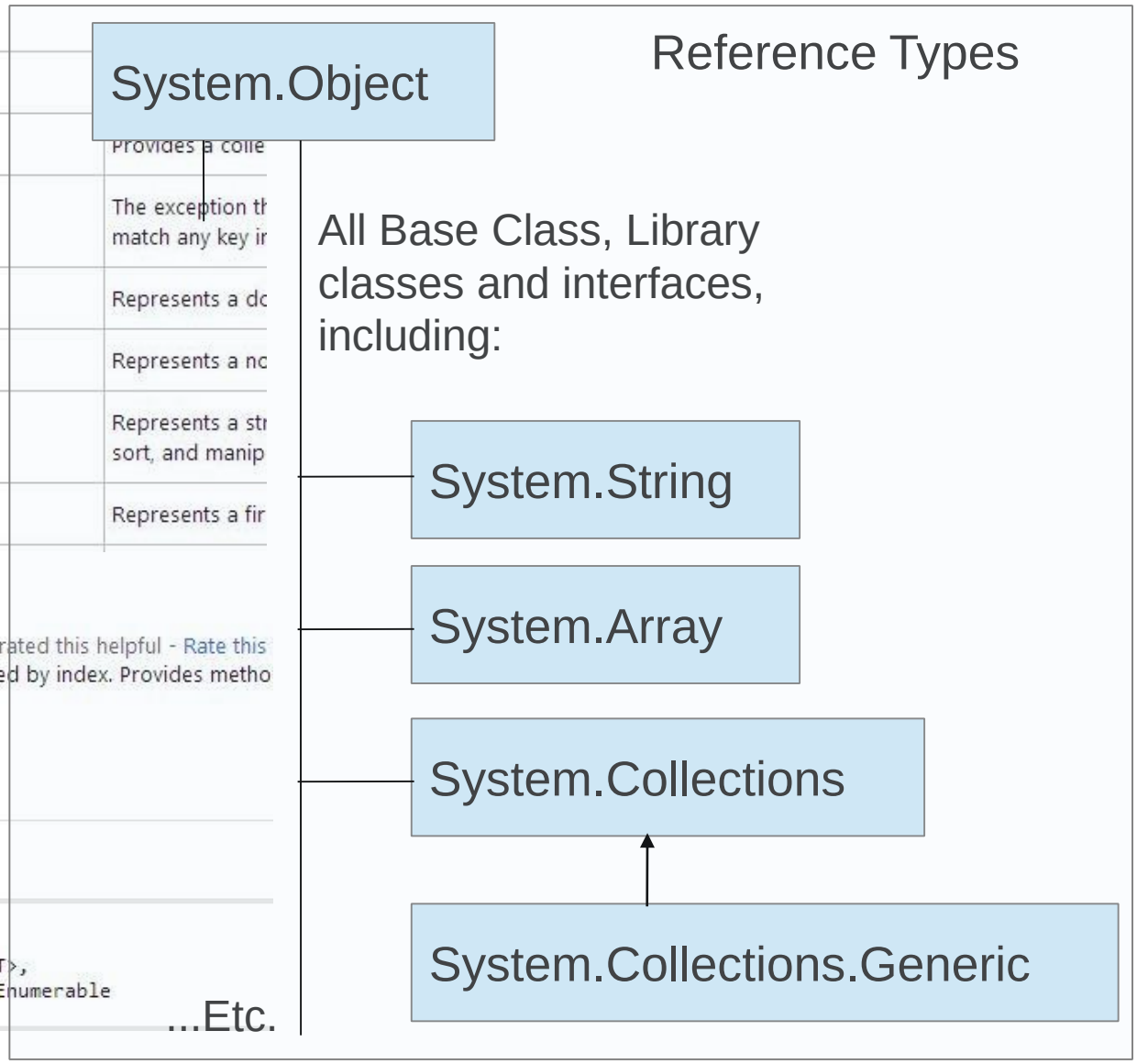
Represents a strongly typed list of objects that can be accessed by index. Provides metho

Namespace: System.Collections.Generic
Assembly: mscorlib (in mscorlib.dll)

Syntax

C# | C++ | VB

```
[SerializableAttribute]
public class List<T> : IList<T>, ICollection<T>,
        IEnumerable<T>, IList, ICollection, IEnumerable
```



Interfaces

Interface	Purpose
ICollection<T>	Defines size, enumerators and synchronized methods for all collections
IEnumerable<T>	Exposes the enumerator, which supports a simple iteration over the collection
IList<T>	Represents a collection of objects that can be individually accessed by index
IDictionary<TKey,TValue>	Represents a collection of key/value pairs

Iterators

- Method or operator that enables developers to support foreach iteration in a class or struct without having to implement the IEnumerable interface

Iterators

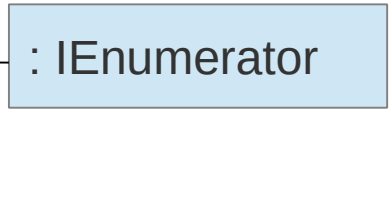
```
public class Months {
  string[] months = { "Januar", "Februar", "März", "April",
                      "Mai", "Juni", "Juli", "August",
                      "September", "Oktober", "November", "Dezember"};
}
```

Iterators

```
public class Months {
  string[] months = { "Januar", "Februar", "März", "April",
                      "Mai", "Juni", "Juli", "August",
                      "September", "Oktober", "November", "Dezember"};
}

Months monate = new Months();
foreach(string temp in monate) {
  Console.WriteLine(temp);
}
```

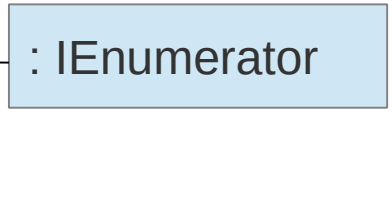
Iterators




```
public class Months {
  string[] months = { "Januar", "Februar", "März", "April",
                      "Mai", "Juni", "Juli", "August",
                      "September", "Oktober", "November", "Dezember"};
}

Months monate = new Months();
foreach(string temp in monate) {
  Console.WriteLine(temp);
}
```

Iterators

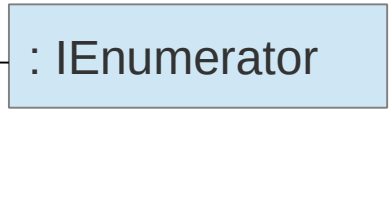



```
public class Months {
  string[] months = { "Januar", "Februar", "März", "April",
                      "Mai", "Juni", "Juli", "August",
                      "September", "Oktober", "November", "Dezember"};
}

Months monate = new Months();
foreach(string temp in monate) {
  Console.WriteLine(temp);
}

IEnumerator GetEnumerator();
```

Iterators




```
public class Months {
  string[] months = { "Januar", "Februar", "März", "April",
                      "Mai", "Juni", "Juli", "August",
                      "September", "Oktober", "November", "Dezember"};
}

Months monate = new Months();
foreach(string temp in monate) {
  Console.WriteLine(temp);
}


IEnumerator GetEnumerator();
```

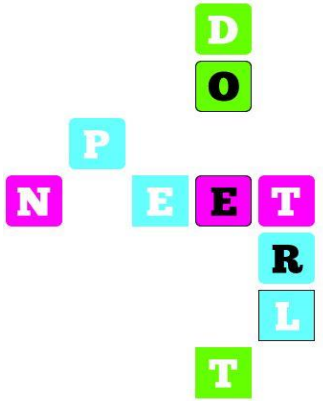
ToDo: Implement methods MoveNext and Reset and property Current

Yield

```
class Program {
  static void Main(string[] args) {
    Months months = new Months();
    foreach(string temp in months)
      Console.WriteLine(temp);
    Console.ReadLine();
  }
}
public class Months : IEnumerable {
  string[] month = { "Januar", "Februar", "März", "April",
              "Mai", "Juni", "Juli", "August", "September",
              "Oktober", "November", "Dezember"};
  // Methode der Schnittstelle 'IEnumerable'
  public IEnumerator GetEnumerator() {
    for (int i = 0; i < month.Length; i++)
      yield return month[i];
  }
}
```

References

- http://msdn.microsoft.com/
- Jay Hilyard & Stephen Teilhet, 'C# Kochbuch', 2006
- Jesse Liberty, 'Programmieren mit C#', 2005
- Krzystof Cwalina & Brad Abrams, 'Richtlinien für das Framework-Design', 2007
- http://dotnetpearls.com/



Proseminar *Objektorientiertes Programmieren mit .NET und C#*

Nadim Yonis

Institut für Informatik
Software & Systems Engineering