Generics

Generics introduces the concept of type parameters to .NET, which make it possible to design classes and methods that defer the specification of one or more types until the class or method is declared and instantiated by client code.

For example, by using a generic type parameter `T`, you can write a single class that other client code can use without incurring the cost or risk of runtime casts or boxing operations, as shown here:

```
// Declare the generic class.
public class GenericList<T>
{
    public void Add(T input) { }
}

class TestGenericList
{
    private class ExampleClass { }
    static void Main()
    {
        // Declare a list of type int.
        GenericList<int> list1 = new GenericList<int>();
        list1.Add(1);

        // Declare a list of type string.
        GenericList<string> list2 = new GenericList<string>();
        list2.Add("");

        // Declare a list of type ExampleClass.
        GenericList<ExampleClass> list3 = new GenericList<ExampleClass>();
        list3.Add(new ExampleClass());
    }
}
```

Generic classes and methods combine reusability, type safety, and efficiency in a way that their non-generic counterparts cannot.

Generics are most frequently used with collections and the methods that operate on them. The System.Collections.Generic namespace contains several generic-based collection classes.

The non-generic collections, such as ArrayList are not recommended and are maintained for compatibility purposes. For more information, see Generics in .NET.

Generics overview

- Use generic types to maximize code reuse, type safety, and performance.
- The most common use of generics is to create collection classes.
- The .NET class library contains several generic collection classes in the System.Collections.Generic namespace. The generic collections should be used whenever possible instead of classes such as ArrayList in the System.Collections namespace.
- You can create your own generic interfaces, classes, methods, events, and delegates.
- Generic classes may be constrained to enable access to methods on particular data types.
- Information on the types that are used in a generic data type may be obtained at run-time by using reflection.