Generics allow you to delay the specification of the data type of programming elements in a class or a method, until it is actually used in the program. In other words, generics allow you to write a class or method that can work with any data type.

You write the specifications for the class or the method, with substitute parameters for data types. When the compiler encounters a constructor for the class or a function call for the method, it generates code to handle the specific data type.

You have learned about the [collection](http://www.tutorialsteacher.com/csharp/csharp-collection) in the previous section, e.g. [ArrayList](http://www.tutorialsteacher.com/csharp/csharp-arraylist), BitArray, [SortedList](http://www.tutorialsteacher.com/csharp/csharp-sortedlist), [Queue](http://www.tutorialsteacher.com/csharp/csharp-queue), [Stack](http://www.tutorialsteacher.com/csharp/csharp-stack) and [Hashtable](http://www.tutorialsteacher.com/csharp/csharp-hashtable). These types of collections can store any type of items. For example, ArrayList can store items of different data types:

Example: C# Collection

ArrayList arList = new ArrayList();

arList.Add(1);

arList.Add("Two");

arList.Add(true);

arList.Add(100.45);

arList.Add(DateTime.Now);

The limitation of these collections is that while retrieving items, you need to cast into the appropriate data type, otherwise the program will throw a runtime exception. It also effects on performance, because of boxing and unboxing.

To overcome this problem, C# includes generic collection classes in the **System.Collections.Generic** namespace.

Features of Generics

Generics is a technique that enriches your programs in the following ways:

* It helps you to maximize code reuse, type safety, and performance.
* You can create generic collection classes. The .NET Framework class library contains several new generic collection classes in the System.Collections.Generic namespace. You may use these generic collection classes instead of the collection classes in the System.Collections namespace.
* You can create your own generic interfaces, classes, methods, events, and delegates.
* You may create generic classes constrained to enable access to methods on particular data types.
* You may get information on the types used in a generic data type at run-time by means of reflection.