**Generics**

Generics were added to version 2.0 of the C# language and the common language runtime (CLR). Generics introduce to the .NET Framework the concept of type parameters, 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.

## **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 Framework class library contains several new generic collection classes in the [System.Collections.Generic](https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic) namespace. These should be used whenever possible instead of classes such as [ArrayList](https://docs.microsoft.com/en-us/dotnet/api/system.collections.arraylist) in the [System.Collections](https://docs.microsoft.com/en-us/dotnet/api/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.

# Benefits of Generics (C# Programming Guide)

Generics provide the solution to a limitation in earlier versions of the common language runtime and the C# language in which generalization is accomplished by casting types to and from the universal base type [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object). By creating a generic class, you can create a collection that is type-safe at compile-time.

The limitations of using non-generic collection classes can be demonstrated by writing a short program that uses the [ArrayList](https://docs.microsoft.com/en-us/dotnet/api/system.collections.arraylist) collection class from the .NET Framework class library. [ArrayList](https://docs.microsoft.com/en-us/dotnet/api/system.collections.arraylist) is a highly convenient collection class that can be used without modification to store any reference or value type.

The following are the benefits of generics:

* There is no need for casting for accessing the elements of the data.
* Code is not duplicated for multiple types of data.
* Generics can hold the data with the same type and we can decide what type of data that the collection holds.
* You can create your own generic interface, classes, method, events and delegate.

Generics allow you to write a class or method that can work with any data type. Arrays are good for many tasks, but C# v2.0 introduced a new feature called generics. Among many benefits, one huge benefit is that generics allow us to create collections that allow us to do more than allowed by an array.