Contents

[Generic Methods 1](#_Toc163224540)

[Bounded Type Parameters 2](#_Toc163224541)

[**Example** 3](#_Toc163224542)

[Generic Classes 3](#_Toc163224543)

[**Example** 4](#_Toc163224544)

# Generics

It would be nice if we could write a single sort method that could sort the elements in an Integer array, a String array, or an array of any type that supports ordering.

Java Generic methods and generic classes enable programmers to specify, with a single method declaration, a set of related methods, or with a single class declaration, a set of related types, respectively.

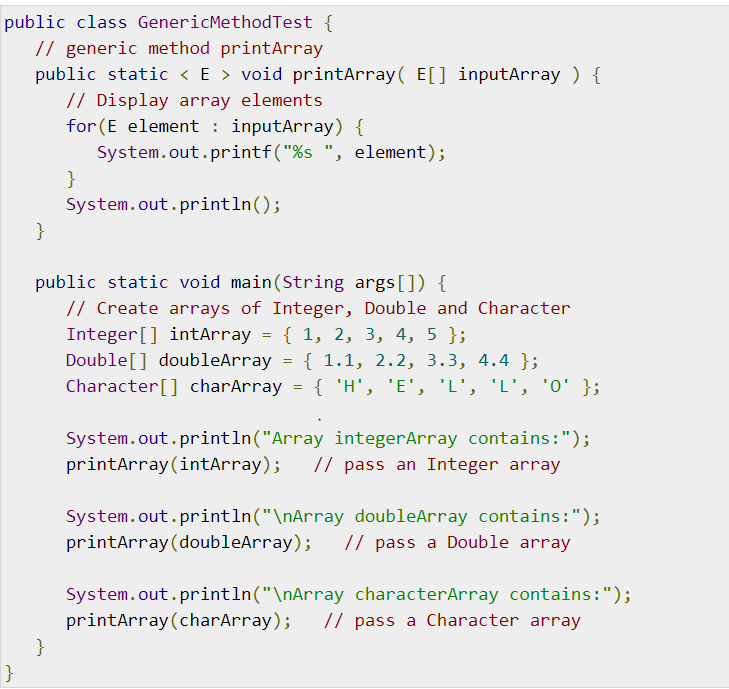
Generics also provide compile-time type safety that allows programmers to catch invalid types at compile time.

Using Java Generic concept, we might write a generic method for sorting an array of objects, then invoke the generic method with Integer arrays, Double arrays, String arrays and so on, to sort the array elements.

# Generic Methods

You can write a single generic method declaration that can be called with arguments of different types. Based on the types of the arguments passed to the generic method, the compiler handles each method call appropriately. Following are the rules to define Generic Methods −

* All generic method declarations have a type parameter section delimited by angle brackets (< and >) that precedes the method's return type ( < E > in the next example).
* Each type parameter section contains one or more type parameters separated by commas. A type parameter, also known as a type variable, is an identifier that specifies a generic type name.
* The type parameters can be used to declare the return type and act as placeholders for the types of the arguments passed to the generic method, which are known as actual type arguments.
* A generic method's body is declared like that of any other method. Note that type parameters can represent only reference types, not primitive types (like int, double and char).



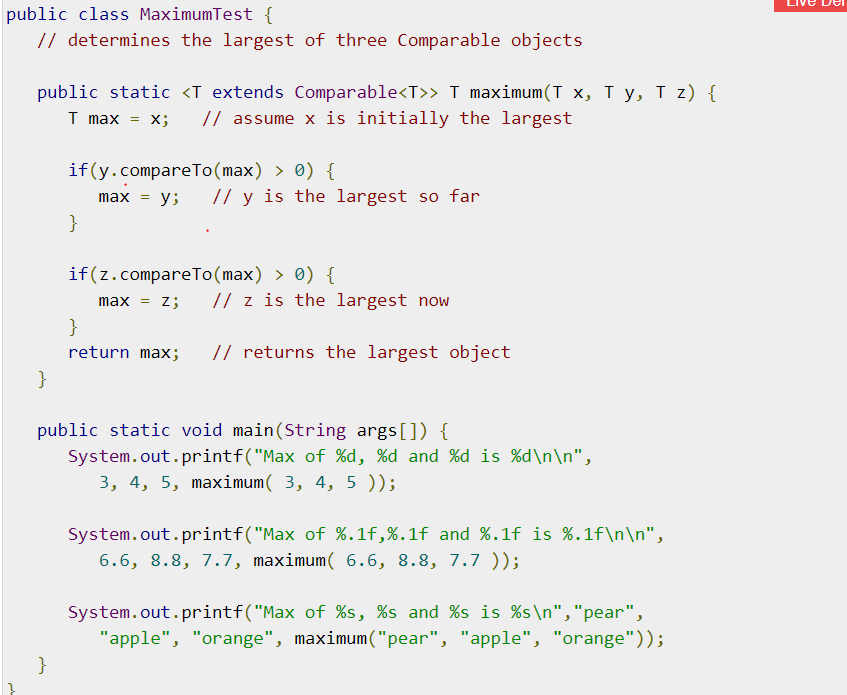
# Bounded Type Parameters

There may be times when you'll want to restrict the kinds of types that are allowed to be passed to a type parameter. For example, a method that operates on numbers might only want to accept instances of Number or its subclasses. This is what bounded type parameters are for.

To declare a bounded type parameter, list the type parameter's name, followed by the extends keyword, followed by its upper bound.

### Example

Following example illustrates how extends is used in a general sense to mean either "extends" (as in classes) or "implements" (as in interfaces). This example is Generic method to return the largest of three Comparable objects −



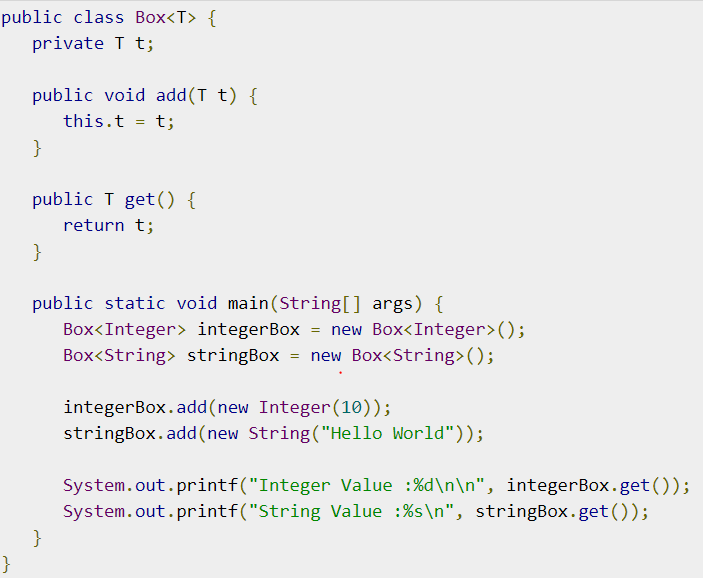
# Generic Classes

A generic class declaration looks like a non-generic class declaration, except that the class name is followed by a type parameter section.

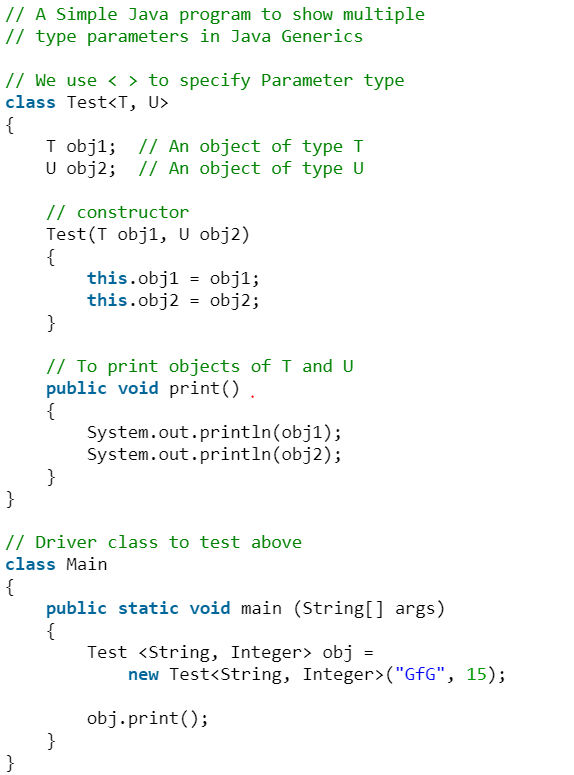
As with generic methods, the type parameter section of a generic class can have one or more type parameters separated by commas. These classes are known as parameterized classes or parameterized types because they accept one or more parameters.

### Example

Following example illustrates how we can define a generic class −



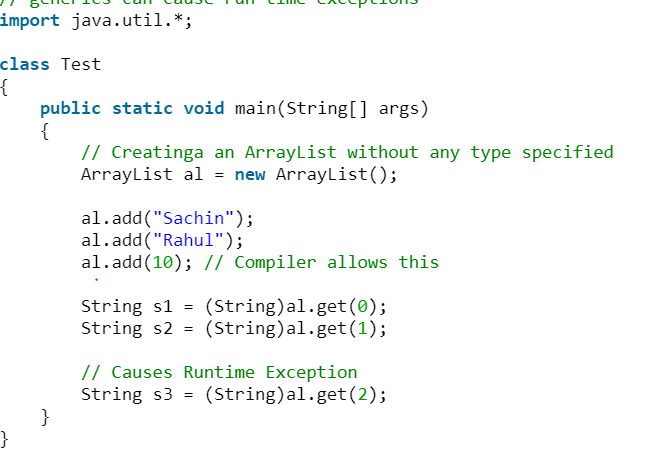
We can also pass multiple Type parameters in Generic classes.



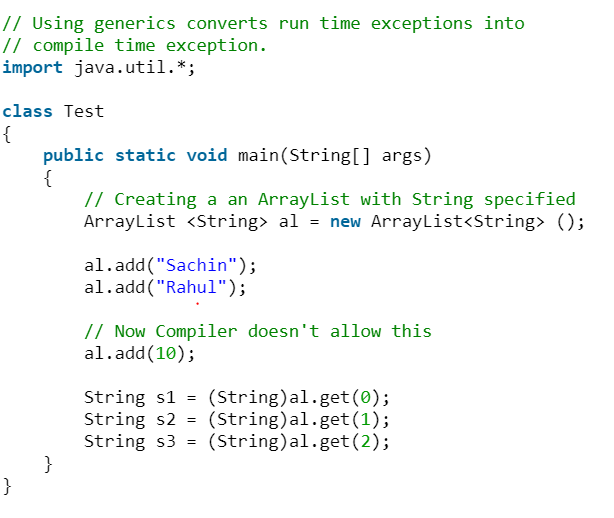
# Advantages of Generics:

Programs that uses Generics has got many benefits over non-generic code.

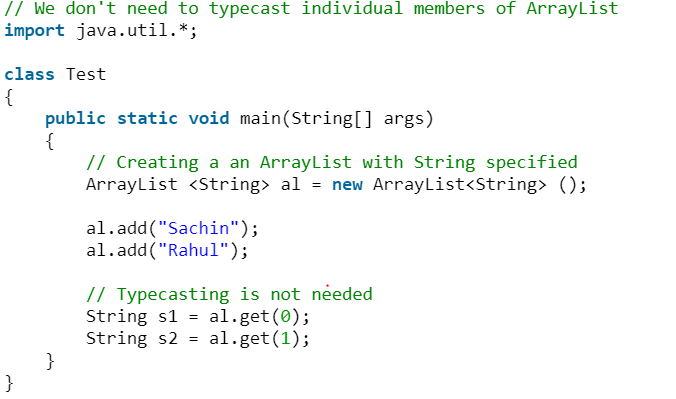
1. Code Reuse: We can write a method/class/interface once and use for any type we want.
2. Type Safety : Generics make errors to appear compile time than at run time (It’s always better to know problems in your code at compile time rather than making your code fail at run time). Suppose you want to create an ArrayList that store name of students and if by mistake programmer adds an integer object instead of string, compiler allows it. But, when we retrieve this data from ArrayList, it causes problems at runtime



How generics solve this problem?  
At the time of defining ArrayList, we can specify that this list can take only String objects.



1. Individual Type Casting is not needed: If we do not use generics, then, in the above example every-time we retrieve data from ArrayList, we have to typecast it. Typecasting at every retrieval operation is a big headache. If we already know that our list only holds string data then we need not to typecast it every time.



1. Implementing generic algorithms: By using generics, we can implement algorithms that work on different types of objects and at the same they are type safe too.