Generics in Java can be both a class and a method. It refers to the ability of a programmer to “specify, with a single method declaration, a set of related methods, or with a single class declaration, a set of related types, respectively.” ([www.tutorialspoint.com](http://www.tutorialspoint.com)) This in effect gives programmers a way to write a generic method for example that can help sort an ArrayList for various Types. This is because we cannot write one sort method for instances for multiple Types like Integer or String without the Generic class.

Another benefit to Generics is that they provide compile time feedback instead of throwing errors at run time. This can help with catching errors with types.

When we are declaring a generic method or class the standard way of doing so is with using <E> or <T> as the type. An example of generic method could look like:

public static < E > void printArray( E[] inputArray )

<E> is our Type, we return void, printArray is our method name and our parameter is an <E> Type object array.

With this method we could have the ability to create Integer, Double or Char Arrays and pass them into the generic method.

When we are passing in values into our generic method it is important to note that we still cannot pass varying types into a single array. This is important to know as we must make sure we are properly using casting and or autoboxing to ensure the type values are appropriate for our needs.