**Generics**

Generics allow us to create classes, to design them, in a general way, without really worrying about the specific details of elements it might contain. Java Array’s list is an example of a generic class. You can apply Arrays list for any type of object, because many methods on that class are applied to any type.

Java supports generic types, such as classes, records, Interfaces. It also supports generic methods.

For example:

Regular class: class ITellYou {

Private String field;

}

Generic class: class YouTellMe <T> {

Private T field;

}

Another example of Generic class: ArrayList< String>listOfString; where ArrayList is the reference type, <String> is the parameter, listOfString is variable name.

**Generic Type Parameter**

We can have mnore than one parameter <T, S, U> are the conventions. A few letters are reserved for special use cases.

**E** for Element. Use extensively by Java collections framework

**K** for Key. Used for mapped types

**N** for number

**T** for type

**V** for value

**S**, **U**, **V** 2nd, 3rd and 4th types.

**You cannot use primitive data type with generics, use wrapper class instead**