**Generics**

* A class or interface whose declaration has one or more type parameter is a generic class or interface.
* Each generic type defines a set of **parameterized types**, which consist of the class or interface name followed by an angle-bracketed list of actual type parameters corresponding to the generic types **formal type parameters**. For example, List<String> (“read list of string”) is a parameterized type representing a list whose elements are of type String. (**String** is the **actual parameter** corresponding to the **formal type parameter** **E**)
* One of the primary motivation for generics is to specify what type of object a container holds, and to have that specification backed up by the compiler. So instead of Object, we’d like to use an unspecified type which can be decided at later time. For the benefit of using a typed / wildcard parameter instead of a raw one please sees C:\Users\Mohit\IdeaProjects\OCJP\src\generics\effective\_java\RawTypesSetIssue.java
* A generic class always requires a Type parameter declaration in the class definition, e.g. class GenericHolder**<T>.**

|  |  |
| --- | --- |
| Term | Example |
| Parameterized type | List<String> |
| Actual type parameter | String |
| Generic Type | List<E> |
| Formal type parameter | E |
| Unbounded wildcard type | List<?> |
| Raw Type | List |
| Bounded type parameter | <E extends Number> |
| Recursive type bound | <T extends Comparable<T>> |
| Bounded wildcard type | List<? extends Number> |
| Generic method | Static <E> List<E> asList<E[] a) |
| Type token | String.class |