**Java Generics**

Generics were added by JDK 5. By using generics you can define an algorithm once, and you can apply it on any kind of datatype without any additional effort.

At very high level, generics are nothing but parameterized types. Generics helps us to create a single class, which can be useful to operate on multiple data types. A class, interface or a method that operates on a parameterized type is called **generics class**, **interface or method.** Generics adds type safty. Remember that generics only works on objects, not primitive types.

Example-

*package com.netparam.generics;*

*public class MySimpleGenerics {*

*public static void main(String a[]){*

*//we are going to create SimpleGeneric object with String as type parameter*

*SimpleGeneric<String> sgs = new SimpleGeneric<String>("NETPARAM");*

*sgs.printType();*

*//we are going to create SimpleGeneric object with Boolean as type parameter*

*SimpleGeneric<Boolean> sgb = new SimpleGeneric<Boolean>(Boolean.TRUE);*

*sgb.printType();*

*}*

*}*

*/\*\**

*\* Here T is a type parameter, and it will be replaced with*

*\* actual type when the object got created.*

*\*/*

*class SimpleGeneric<T>{*

*//declaration of object type T*

*private T objReff = null;*

*//constructor to accept type parameter T*

*public SimpleGeneric(T param){*

*this.objReff = param;*

*}*

*public T getObjReff(){*

*return this.objReff;*

*}*

*//this method prints the holding parameter type*

*public void printType(){*

*System.out.println("Type: "+objReff.getClass().getName());*

*}*

*}*

**Output-**

Type java.lang.String

Type java.lang.Boolean