**Generics – Part\_01**

* **Generics:**

1. **Introduction**
2. **Generic classes.**
3. **Bounded Types.**
4. **Generic Methods & wild-card character (?)**
5. **Communication with non-generic code.**
6. **Conclusions.**

* **Introduction:**

The main objectives of Generics are to provide, type-safety and to resolve type-casting problems.

* **Case\_01: Type-Safety:**

Arrays are type-safe, that is we can give the guarantee for the type of elements present inside Array.

For example, if our programming requirement is to hold only String type of objects we can choose String Array (String[]) by mistake if we are trying to add any other type of Objects, we will get compile-time error. Hence, String[] will contain String object, due to this we can give the guarantee for the type of elements present inside array. Hence Arrays are safe to use with respect to type. That is, Arrays are type-safe.

Example:

String[] s = new String[10000];

s[0] = “durga”;

s[1] = “ravi”;

s[2] = new Integer(); // Invalid and it thows error

CE: incompatible types:

found: java.lang.Integer

required: java.lang.String

But collections are not type-safe, that is we can’t give the guarantee for the type of elements present inside collection.

For example, if our programming requirement is to hold only String type of objects, and we choose ArrayList, by mistake if we are trying to add any other type of object. We won’t get any compile-time error, but the program may fail at runtime.

Example:

ArrayList l = new ArrayList();

l.add(“durga”);

l.add(“ravi”);

l.add(new Integer(10));

String name1 = (String)l.get(0);

String name2 = (String) l.get(1);

String name3 = (String) l.get(2);

RE: ClassCastException

Hence, we can’t give the guarantee for the type of elements present inside collection, due to this collection are not safe to use, with respect to type. That is, collections are not type-safe.

* **Case\_02: type-casting:**

In the case of arrays at the time of retrieval it is not required to perform type-casting, because there is a guarantee for the type of elements present inside array.

Example:

String[] s = new String[10000];

S[0] = “durga”;

String name1 = s[0]; // type-casting not required.

But, in the case of collections at the time of retrieval compulsory we should perform type-casting because there is no guarantee for the type of elements present inside collection.

Example:

ArrayList l = new ArrayList();

l.add(“durga”);

String name1 = l.get(0); // Invalid : CE

Incompatible types

found: java.lang.object

required: java.lang.String

String name1 = (String) l.get(0); // type-casting is mandatory

Hence, type-casting is a bigger headache in collections.

To overcome above problems of collections, Sun people introduced generics concept in 1.5 version. Hence, the main objectives of generics are

1. To provide type-safety.
2. To resolve type-casting problems.