**Understanding basic of JAVA Generics**

When somebody asks what is the need of generics in Java?

Two points that should come to mind immediately is

1. Type safety
2. Type casting

How type safety??? Let us have a look.

Assume you are working on a requirement to prepare a list of all the employee names working as QA, which is to be used by another guy working in the same project.

You came up with the idea of storing all the names in an array as it is type safe.

String array[]=new String[100]; (Arrays are type safe as it allows only similar type of elements to be stored, in an array of strings you cannot have an integer)

But then you realized that arrays are not growable they are static in size, so you dropped the idea of taking array and continued coding taking an ArrayList.

ArrayList list=new Arraylist();

list.add(“abc”);

list.add(“xyz”);

list.add(new Integer(1)); If by mistake you added an integer and compiled the code, during compile time you did not get any error WOW!!!. You checked in the code and asked the other guy in your project to access your method to get the list. You will not get any compile time error as collections allows to have heterogeneous datatypes.

When the other guy was trying to get the names from your list with his code which looks something like this.

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

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

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

The first two lines of code were fine, but he got the error at the last line because it returned an integer instead of a string and you cannot directly type cast an integer to string.

As collections are not type safe without generics so the concept was introduced to make collections type safe.

So for the above example ArrayList should be like ArrayList<String> list=new ArrayList<String>();

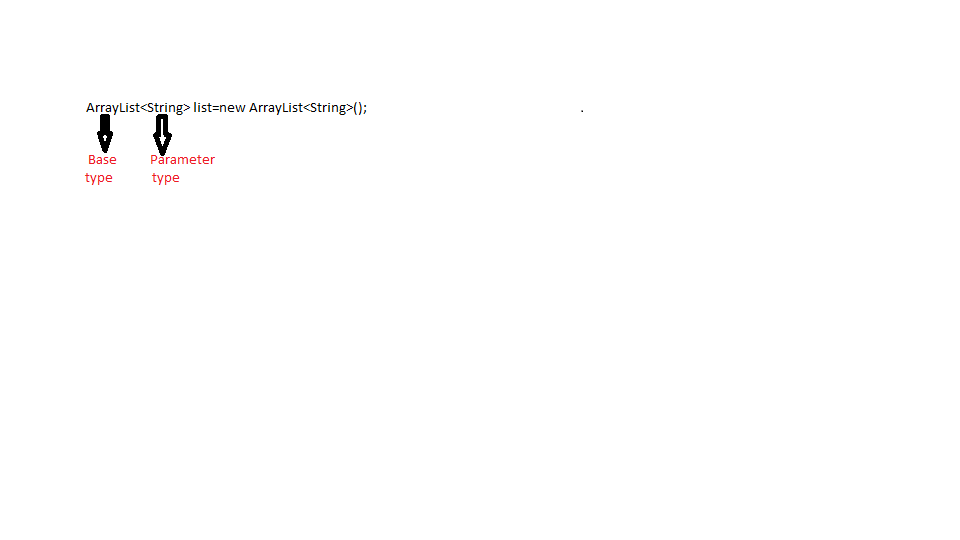
Type casting….

As in the above example we have seen every time we are fetching the name from the list (String name1=(String)list.get(0);) we are type casting it to the string. The reason is Array list can have an element of any type, so we must cast it to the type we need.

This is another headache to use collections without generics. To overcome the difficulty of type safety and casting generics were introduced.

The key takes away from the above discussion is.

1. Generics helps us to overcome the problem of type safety and type casting.
2. Non generic version of any collections object like ArrayList list=new ArrayList(); is never type safe, whereas generic version like ArrayList<String> list=new ArrayList<String>(); is type safe.
3. Non generic version of any collections object like ArrayList list=new ArrayList(); always requires type casting at the time of retrieval, whereas generic version like ArrayList<String> list=new ArrayList<String>(); does not requires type casting.

**Generics Intersection**

First Look….

If you remember run time polymorphism where a parent class reference can be used to hold child class object, using the concept the above example can be written as List<String> list=new ArrayList<String>();

Or Collection<String> list=new ArrayList<String>();. The idea behind discussing this concept is that we can use polymorphism to make BaseType as the parent reference of the child class object it is holding, but this concept cannot be applied to the Parameter type.

Let us say ArrayList<Object> list=new ArrayList<String>() is invalid.

Second Look…

The parameter type of a generics must be any class or interface, it should not be any primitive data type.

For eg- ArrayList<String> list=new ArrayList<String>(); // Syntax is perfect as the parameter type is a class.

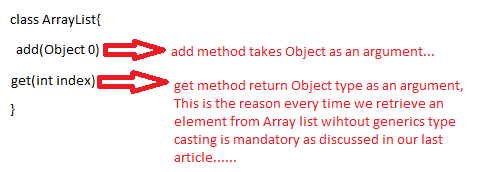
ArrayList<char> list=new ArrayList<char>(); // This syntax will throw a compile time error as we are trying to pass a primitive data type as a parameter type.

Always great to compile and check the error!!! 😊

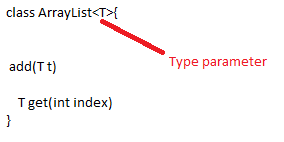
**Little deep**

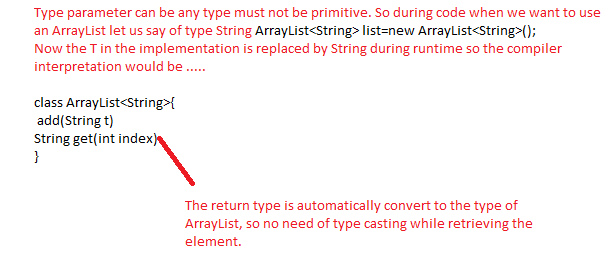
Generics in Java were introduced in version 5.

In our examples from previous article we took array list. Let us see how the actual implementation of ArrayList class was before version 5.



In version 5 ArrayList class implementation changes looks like.





This is a basic introduction to generics what to deep dive into it follow my next articles on generics in detail.

Happy to help!!!