**Generics in collection or Arrays:**

Java new generic collection allows us to have only one type of object in collection. Now it is type safe so typecasting is not required at run time.

Non-Generic ArrayList

ArrayList al=new ArrayList();

Generic example of creating ArrayList

ArrayList<Integer> al=new Arraylist<Integer>();

Generic in Java is added to provide compile time type-safety of code and removing risk of  ClassCastException at [runtime](http://javarevisited.blogspot.sg/2012/03/what-is-static-and-dynamic-binding-in.html) which was quite frequent error in Java code, for those who doesn’t know what is type-safety at compile time, it’s just a check by compiler that correct Type is used in correct place and there should not be any ClassCastException.

Though Generics may look very complex because of its mysterious angle bracketing <> and various wild cards on Java generics  
  
Read more: <https://javarevisited.blogspot.com/2011/09/generics-java-example-tutorial.html#ixzz5mCcTkgDD>  
  
For example, HashSet of String will only contain String object and if you try to put Integer or any other object, the compiler will complain. Before Java 5 same code will pass compile time check but will fail at runtime which is worse. Generics allows Java programmer to write more robust and **type-safe code**. In my opinion *generics in Java* is much overdue feature given popularity of [Collection framework in java](http://javarevisited.blogspot.com/2011/11/collection-interview-questions-answers.html)and its limitation around handling type-safety.

Arrays don't support Generics in Java so you can not create Arrays like T[] which makes gentrifying an existing class hard if you are using arrays.

1) One limitation of Generics in Java is that it can not be applied to primitive type, for example, you can not create pass primitives in angle bracket that will result in compilation error, for Example, ArrayList<int> will result in compilation error,