```
Generics
========
   a. To promote typesafety and to avoid type casting problems


Type parameter at class level
   class Test<T>{

  }

a. <T extends X>
b. <T extends X&Y>
c. <T extends X&Y&Z>

X,Y,Z can beinterface
X,Y  can be interface
X -> it should be class name ,Y,Z is interface name


 method level Type parameter
    1. methodOne( ArrayList<String> t)
    2. methodOne( ArrayList<? extends X> t)
    3. methodOne(ArrayList< ? super X> t)

return type of method Type Parameter
   1. public <T> void m1(T t)
   2. public <T extends X&Y> void m1(T t)
   3. public <T extneds X&Y&Z> void m1(T t)

Generics concept is applicable only at the compiler level, not at JVM
level(runtime)
Even if we have code in generics in .classfile the syntax of generics will be
removed
      eg:  ArrayList<String> al  = new ArrayList<String>();
              ArrayList<String> al = new ArrayList<>();
                              |
                              |
                              |
               ArrayList al =new ArrayList();


class Test{
      public void m1(Arraylist<String> al){  ======> m1(ArrayList al)

      }
      public void m1(ArrayList<Double> al ){  ========> m1(ArrayList al)

      }
}
this mechanism is called "type erasure"


Basically my question is what is the difference between Arrays.asList() & List.of()
methods to create objects in collaboration?
     Arrays.asList() --> gives a copy of Array to read as List(only for reading
purpose)
     List.of() ---> creates a list which are immutable.
```