1. Use of lambda

**public** **interface** UserDefined1<T> {

**public** **void** apply(T t);

}

The interface can be generic or non generic

List<String> list = Arrays.*asList*("a","b","c");

UserDefined1<List<String>> obj1 = (l)->{for(String s : l) System.out.println(s)};

obj1.apply(list);

1. Use of colon

Reference link : <https://netjs.blogspot.in/2015/06/method-reference-in-java-8.html>

Create a class ColonClassImpl and write a method of same signature like

apply. Return type does not matter. As here input argument of apply() is

generic, so any type can be allowed, but if apply(List<Integer> list) is there then it can be a

type mismatch compilation error

**class** ColonClassImpl{

**public** **void** method(List<String> list){

**for**(String s : list){

**if**(s.equalsIgnoreCase("a"))

System.***out***.println("A");

}

}

}

ColonClassImpl implObj = new ColonClassImpl();

UserDefined1< List<String>> obj2 = implObj :: method;