Any Interface which has only one abstract method then it is called as functional interface

To give extra information about functional interface to compiler, we use annotation @FunctionalInerface

Using Functional Interface

```
@FunctionalInterface
public interface MyFunctionalInterface {
        void m1();
        default void m2() {
                m5();
                System.out.println("in m2
method");
        }
        static void m3() {
                m5();
                System.out.println("in static
m3 method");
        }
        private static void m5() {
                System.out.println("in m5
method");
        }


}
```

```
public class TestInterface{
  public static void main(String[] args) {
MyFunctionalInterface ob=()->{
    System.out.println("In m1 in
MyTestClass");

}
ob.m1();
}
}
```

```
import
com.demo.interfaces.MyFunctionalInterface;

public class MyTestClass implements
MyFunctionalInterface {

        @Override
        public void m1() {
                System.out.println("In m1 in
MyTestClass");

        }

}
```

```
public class TestInterface {
        public static void main(String[] args)
{
                MyFunctionalInterface
ob=new MyTestClass();
                ob.m1();
}
```

```
//Annonymous class
MyFunctionalInterface f2=new
MyFunctionalInterface(){
    public void m1(){
   System.out.println("In m1 in
MyTestClass");
//more abstract functions also can be
implemented
```

}
f2.m1()
}	

Generics Example

public interface CompareInt {
        int findMax(**int** x,**int** y);
 	
}
	public interface CompareString {
	        String findMax(String x,String y);
	}

public interface MyCompare<T>{
    T findMax(T x,T y);
}
	Generics

MyCompare<Integer> c1=(a,b)->{
    return a>b?a:b
}
	MyCompare<String> c1=(a,b)->{
	    return a.length()>b.length()?a:b
	}

public interface MyAddInterface<T,F>{
     T add(F x,F,y)
}	

MyAddInterface<Integer,Integer> f1=(a,b)->{
   return a+b
}
System.out.println("Addition : "+f1.add(12,13));
	MyAddInterface<Integer,String> f1=(a,b)->{
	   return a.length()+b.length()
	}
	System.out.println("Addition : "+f1.add(12,13));