

Using Functional Interface

<pre> @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"); } } </pre>	<pre> public class TestInterface{ public static void main(String[] args) { MyFunctionalInterface ob=()->{ System.out.println("In m1 in MyTestClass"); } ob.m1(); } } </pre>
<pre> import com.demo.interfaces.MyFunctionalInterface; public class MyTestClass implements MyFunctionalInterface { @Override public void m1() { System.out.println("In m1 in MyTestClass"); } } </pre>	<pre> //Anonymous class MyFunctionalInterface f2=new MyFunctionalInterface(){ public void m1(){ System.out.println("In m1 in MyTestClass"); } } //more abstract functions also can be implemented </pre>
<pre> public class TestInterface { public static void main(String[] args) { MyFunctionalInterface ob=new MyTestClass(); ob.m1(); } } </pre>	

Generics Example

<pre>public interface CompareInt { int findMax(int x,int y); }</pre>	<pre>public interface CompareString { String findMax(String x,String y); }</pre>
<pre>public interface MyCompare<T>{ T findMax(T x,T y); }</pre>	Generics
<pre>MyCompare<Integer> c1=(a,b)->{ return a>b?a:b }</pre>	<pre>MyCompare<String> c1=(a,b)->{ return a.length()>b.length()?a:b }</pre>

<pre>public interface MyAddInterface<T,F>{ T add(F x,F,y) }</pre>	
<pre>MyAddInterface<Integer,Integer> f1=(a,b)->{ return a+b } System.out.println("Addition : "+f1.add(12,13));</pre>	<pre>MyAddInterface<Integer,String> f1=(a,b)->{ return a.length()+b.length() } System.out.println("Addition : "+f1.add(12,13));</pre>