```java
import java.util.HashMap;
import java.util.Map;
import java.util.function.BiConsumer;

public class BiConsumerExample {
    public static void main(String[] args) {
        Map<Integer,String> map = new HashMap<>();
        map.put(1, "A");
        map.put(2, "B");
        map.put(3, "C");
        BiConsumer<Integer,String> biConsumer = (key, value) -> System.out.println("Key:"+ key+" Value:"+
value);
        map.forEach(biConsumer);
    }
}
```

```java
import java.util.HashMap;
import java.util.Map;
import java.util.function.BiConsumer;
import java.util.function.BiFunction;

public class BiFunctionExample {
    public static void main(String[] args) {
        BiFunction<Integer, Integer, String> biFunction = (num1, num2) -> "Result:" +(num1 + num2);
        System.out.println(biFunction.apply(20,25));
    }
}
```

```java
import java.util.function.BiPredicate;

public class BiPredicateExample {
    public static void main(String[] args){
        BiPredicate<Integer, String> condition = (i, s)-> i>20 && s.startsWith("R");
        System.out.println(condition.test(10,"Ram"));
        System.out.println(condition.test(30,"Shyam"));
        System.out.println(condition.test(30,"Ram"));
    }
}
```

```java
public class Lession0 {
    interface NumericTest {
        boolean computeTest(int n);
    }

    class IsEven implements NumericTest {
        public boolean computeTest(int n) {
            return (n % 2) == 0;
        }
    }

    class IsNegative implements NumericTest {
        public boolean computeTest(int n) {
            return n < 0;
        }
    }

    public void lession0Test() {
        NumericTest isEven = new IsEven();
        NumericTest isNegative = new IsNegative();

        // Output: false
        System.out.println(isEven.computeTest(5));

        // Output: true
        System.out.println(isNegative.computeTest(-5));
    }
}
```

```java
public class Lession1 {
    interface NumericTest {
        boolean computeTest(int n);
    }

    public static void main(String args[]) {
        NumericTest isEven = (n) -> (n % 2) == 0;
        NumericTest isNegative = (n) -> (n < 0);

        // Output: false
        System.out.println(isEven.computeTest(5));

        // Output: true
        System.out.println(isNegative.computeTest(-5));
    }
}
```

```java
public class Lession2 {
    interface MyGreeting {
        String processName(String str);
    }

    public static void main(String args[]) {
        MyGreeting morningGreeting = (str) -> "Good Morning " + str + "!";
        MyGreeting eveningGreeting = (str) -> "Good Evening " + str + "!";

        // Output: Good Morning Luis!
        System.out.println(morningGreeting.processName("Luis"));

        // Output: Good Evening Jessica!
        System.out.println(eveningGreeting.processName("Jessica"));
    }
}
```

```java
public class Lession3 {
    interface MyString {
        String myStringFunction(String str);
    }

    public static void main (String args[]) {
        // Block lambda to reverse string
        MyString reverseStr = (str) -> {
            String result = "";

            for(int i = str.length()-1; i >= 0; i--)
                result += str.charAt(i);

            return result;
        };

        // Output: omeD adbmaL
        System.out.println(reverseStr.myStringFunction("Lambda Demo"));
    }
}
```

```java
public class Lession4 {
    interface MyGeneric<T> {
        T compute(T t);
    }

    public static void main(String args[]){

        // String version of MyGenericInteface
        MyGeneric<String> reverse = (str) -> {
            String result = "";

            for(int i = str.length()-1; i >= 0; i--)
                result += str.charAt(i);

            return result;
        };

        // Integer version of MyGeneric
        MyGeneric<Integer> factorial = (Integer n) -> {
            int result = 1;

            for(int i=1; i <= n; i++)
                result = i * result;

            return result;
        };

        // Output: omeD adbmaL
        System.out.println(reverse.compute("Lambda Demo"));

        // Output: 120
        System.out.println(factorial.compute(5));

    }
}
```

```java
public class Lession5 {
    interface MyString {
        String myStringFunction(String str);
    }

    public static String reverseStr(MyString reverse, String str){
        return reverse.myStringFunction(str);
    }

    public static void main (String args[]) {
        // Block lambda to reverse string
        MyString reverse = (str) -> {
            String result = "";

            for(int i = str.length()-1; i >= 0; i--)
                result += str.charAt(i);

            return result;
        };

        // Output: omeD adbmaL
        System.out.println(reverseStr(reverse, "Lambda Demo"));
    }
}
```

```java
import java.util.function.Function;

public class Lession6 {
    //API which accepts an implementation of
    //Function interface
    static void modifyTheValue(int valueToBeOperated,
                               Function<Integer, Integer> function){

        int newValue = function.apply(valueToBeOperated);
        /*
         * Do some operations using the new value.
         */
        System.out.println(newValue);
    }

    public static void main(String[] args) {
        int incr = 20;
        int myNumber = 10;
        modifyTheValue(myNumber, val-> val + incr);

        myNumber = 15;
        modifyTheValue(myNumber, val-> val * 10);
        modifyTheValue(myNumber, val-> val - 100);
        modifyTheValue(myNumber, val-> "somestring".length() + val - 100);
    }
}
```