Java 8 的 java.util.function 包里面有很多内置的函数式接口。

Predicate : Predicate 接口包含一个抽象方法 test，test 方法会对输入的参数做一个评判，输出一个布尔值作为评判结果，至于评判的 criteria 交给具体的实现去做。
Predicate 的函数原型如下:

```
public interface Predicate
{
   public boolean test(T  t);
}
```

BinaryOperator: 见名知意，BinaryOperator 是一个二元输入的函数式接口，它包含一个 apply 抽象方法，接收两个相同类型的参数，返回一个与输入类型相同的结果。

```
public interface BinaryOperator
{
   public T apply(T x, T y);
}
```

Function: Function 接口有一个抽象方法 apply，接收一个类型T的输入，返回一个类型 R 的结果。这个名字也很体现用途，Function 是函数的意思，函数的作用就是将定义域映射到值域的操作。这里 T 就是定义域，R 是值域。映射关系交给具体实现去定义。

```
public interface Function
{
   public R apply(T t);
}
```

示例程序：

```
// A simple program to demonstrate the use
// of predicate interface
import java.util.*;
import java.util.function.Predicate;

class Test
{
   public static void main(String args[])
   {
      // create a list of strings
      List<String> names =
         Arrays.asList("Geek","GeeksQuiz","g1","QA","Geek2");

      // declare the predicate type as string and use
      // lambda expression to create object
      Predicate<String> p = (s)->s.startsWith("G");

      // Iterate through the list
```

```
        for (String st:names)
        {
            // call the test method
            if (p.test(st))
                System.out.println(st);
        }
    }
}

import java.util.Arrays;
import java.util.List;
import java.util.function.Predicate;

public class Java8Tester {
   public static void main(String args[]){
      List<Integer> list = Arrays.asList(1, 2, 3, 4, 5, 6, 7, 8, 9);

      // Predicate<Integer> predicate = n -> true
      // n 是一个参数传递到 Predicate 接口的 test 方法
      // n 如果存在则 test 方法返回 true

      System.out.println("输出所有数据:");

      // 传递参数 n
      eval(list, n->true);

      // Predicate<Integer> predicate1 = n -> n%2 == 0
      // n 是一个参数传递到 Predicate 接口的 test 方法
      // 如果 n%2 为 0 test 方法返回 true

      System.out.println("输出所有偶数:");
      eval(list, n-> n%2 == 0 );

      // Predicate<Integer> predicate2 = n -> n > 3
      // n 是一个参数传递到 Predicate 接口的 test 方法
      // 如果 n 大于 3 test 方法返回 true

      System.out.println("输出大于 3 的所有数字:");
      eval(list, n-> n > 3 );
   }

   public static void eval(List<Integer> list, Predicate<Integer> predicate) {
      for(Integer n: list) {

         if(predicate.test(n)) {
            System.out.println(n + " ");
```

```
        }
    }
  }
}
```

java.util.function 它包含了很多类，用来支持 Java的 函数式编程，该包中的函数式接口有：

序号	接口 & 描述
1	**BiConsumer<T,U>** 代表了一个接受两个输入参数的操作，并且不返回任何结果
2	**BiFunction<T,U,R>** 代表了一个接受两个输入参数的方法，并且返回一个结果
3	**BinaryOperator<T>** 代表了一个作用于于两个同类型操作符的操作，并且返回了操作符同类型的结果
4	**BiPredicate<T,U>** 代表了一个两个参数的boolean值方法
5	**BooleanSupplier** 代表了boolean值结果的提供方
6	**Consumer<T>** 代表了接受一个输入参数并且无返回的操作
7	**DoubleBinaryOperator** 代表了作用于两个double值操作符的操作，并且返回了一个double值的结果。
8	**DoubleConsumer** 代表一个接受double值参数的操作，并且不返回结果。
9	**DoubleFunction<R>** 代表接受一个double值参数的方法，并且返回结果
10	**DoublePredicate** 代表一个拥有double值参数的boolean值方法
11	**DoubleSupplier** 代表一个double值结果的提供方
12	**DoubleToIntFunction** 接受一个double类型输入，返回一个int类型结果。
13	**DoubleToLongFunction** 接受一个double类型输入，返回一个long类型结果
14	**DoubleUnaryOperator** 接受一个参数同为类型double,返回值类型也为double 。
15	**Function<T,R>** 接受一个输入参数，返回一个结果。
16	**IntBinaryOperator** 接受两个参数同为类型int,返回值类型也为int 。
17	**IntConsumer** 接受一个int类型的输入参数，无返回值 。
18	**IntFunction<R>** 接受一个int类型输入参数，返回一个结果 。
19	**IntPredicate** 接受一个int输入参数，返回一个布尔值的结果。
20	**IntSupplier** 无参数，返回一个int类型结果。

21	**IntToDoubleFunction** 接受一个int类型输入，返回一个double类型结果 。
22	**IntToLongFunction** 接受一个int类型输入，返回一个long类型结果。
23	**IntUnaryOperator** 接受一个参数同为类型int,返回值类型也为int 。
24	**LongBinaryOperator** 接受两个参数同为类型long,返回值类型也为long。
25	**LongConsumer** 接受一个long类型的输入参数，无返回值。
26	**LongFunction<R>** 接受一个long类型输入参数，返回一个结果。
27	**LongPredicate** R接受一个long输入参数，返回一个布尔值类型结果。
28	**LongSupplier** 无参数，返回一个结果long类型的值。
29	**LongToDoubleFunction** 接受一个long类型输入，返回一个double类型结果。
30	**LongToIntFunction** 接受一个long类型输入，返回一个int类型结果。
31	**LongUnaryOperator** 接受一个参数同为类型long,返回值类型也为long。
32	**ObjDoubleConsumer<T>** 接受一个object类型和一个double类型的输入参数，无返回值。
33	**ObjIntConsumer<T>** 接受一个object类型和一个int类型的输入参数，无返回值。
34	**ObjLongConsumer<T>** 接受一个object类型和一个long类型的输入参数，无返回值。
35	**Predicate<T>** 接受一个输入参数，返回一个布尔值结果。
36	**Supplier<T>** 无参数，返回一个结果。
37	**ToDoubleBiFunction<T,U>** 接受两个输入参数，返回一个double类型结果
38	**ToDoubleFunction<T>** 接受一个输入参数，返回一个double类型结果
39	**ToIntBiFunction<T,U>** 接受两个输入参数，返回一个int类型结果。
40	**ToIntFunction<T>** 接受一个输入参数，返回一个int类型结果。
41	**ToLongBiFunction<T,U>** 接受两个输入参数，返回一个long类型结果。
42	**ToLongFunction<T>** 接受一个输入参数，返回一个long类型结果。
43	**UnaryOperator<T>** 接受一个参数为类型T,返回值类型也为T。