链滴

函数式接口新体验

作者：woshiszz
原文链接：https://ld246.com/article/1569137589586
来源网站：链滴

闲空时看了看java8的新特性，发现lambda表达式的优美与简洁，瞬间起了兴趣，然后写了写几个dmo先入门。

函数式接口

一，Predicate<T> 接口中包含一个抽象方法，boolean test(T t) 。用于条件判断的场景：

```
public boolean checkString(String s, Predicate<String> p) {
return p.test(s);

}

@Test
public void main2() {
String a = "abcdefg";
        checkString(a, new Predicate<String>() {
          @Override
          public boolean test(String s) {
              return false;
          }
       });
       boolean b=checkString(a,(String s)-> {return a.length()>5;});
//判断字符串长度是否大于5
boolean b=checkString(a, s-> a.length()>5);
```

二，Supplier<T> （生产者） 接口包含一个无参方法：T get() 。用来获取一个泛型参数指定类型的象数据。由于这是一个函数式接口，这就意味着对应的Lambda表达式需要对外提供一个符合泛型类的对象数据。

```
 public String getString(Supplier<String> sup) {
```

```
return sup.get();
}

@Test
public void main3() {
    String s=getString(new Supplier<String>() {
        @Override
        public String get() {
            return "胡歌";
        }
    });
String s1 = getString(() -> "胡歌");
}
```

求数组最大的数

```
public int getMax(Supplier<Integer> sup) {
return sup.get();
}

@Test
public void main4() {
int[] arr = {1, 3, 5, 7, 2, 9};
int maxValue=getMax(()->{
int max=arr[0];
for (int i:arr) {
if (i > max) {
max = i;
}
}
return max;});
System.out.println(maxValue);
}
```

三，Consumer<T>（消费者）接口与Supplier接口相反，他目的是消费一个数据，数据类型由泛型定。抽象方法：void accept(T t)。具体消费什么，需要自定义（输出，计算……）

```
public void method(String name, Consumer<String> con) {
con.accept(name);
}

@Test
public void main5() {
method("赵丽媛",name->{
//对字符串反向输出
String thname = new StringBuffer(name).reverse().toString();
System.out.println(thname);
});
}
```

Consumer<T>中默认方法：andThen。如果方法的参数和返回值都是Consumer类型，在消费数据时候，首先做个A，然后做B，实现组合式消费。

```
public void method1(String s, Consumer<String> con1, Consumer<String> con2) {
con1.andThen(con2).accept(s);
```

```
}

@Test
public void main6() {
String a = "Hello";
method1(a,
(s)->{
//所有字符串变大写
System.out.println(s.toUpperCase());},
(s)->{
//所有字符串变小写
System.out.println(s.toLowerCase());
}
);
}
```

四，Function<T,R> 接口用来根据一个类型的数据得到另一个类型的数据，前者是前置条件，后者后置条件。抽象方法为 R apply(T t)，根据类型T的参数获取类型的R的结果。比如将String类型转换Integer类型。

```
public int change(String s, Function<String,Integer> fun) {
int in = fun.apply(s);
return in;
}

@Test
public void main7() {
String a = "12345";
//把String类型转化int类型
int b=change(a,(str)-> Integer.parseInt(str));
System.out.println(b);

}
```