RxJava2.0 操作符之 -- 算数操作符

作者：hiquanta
原文链接：https://ld246.com/article/1500534681202
来源网站：链滴

rxjava-math 模块的操作符 此类库只有1.x的支持，2.0代码没法写

averageInteger(□) — 求序列平均数并发射

averageLong(□) — 求序列平均数并发射

averageFloat(□) — 求序列平均数并发射

averageDouble(□) — 求序列平均数并发射

max(□) — 求序列最大值并发射

maxBy(□) — 求最大key对应的值并发射

min(□) — 求最小值并发射

minBy(□) — 求最小Key对应的值并发射

sumInteger(□) — 求和并发射

sumLong(□) — 求和并发射

sumFloat(□) — 求和并发射

sumDouble(□) — 求和并发射

Concat

不交错的发射两个或多个Observable的发射物

```
Observable observable1 = Observable.just(1, 2, 3, 4);
Observable observable2 = Observable.just(5, 6, 7, 8);
Observable.concat(observable1,observable2).subscribe(RxUtils.getObserver());
```

```
onSubscribe
Thread:Thread[main,5,main]
onNext:1
Thread:Thread[main,5,main]
onNext:2
Thread:Thread[main,5,main]
onNext:3
Thread:Thread[main,5,main]
```

```
onNext:4
Thread:Thread[main,5,main]
onNext:5
Thread:Thread[main,5,main]
onNext:6
Thread:Thread[main,5,main]
onNext:7
Thread:Thread[main,5,main]
onNext:8
Thread:Thread[main,5,main]
onComplete
```

Reduce

按顺序对Observable发射的每项数据应用一个函数并发射最的值

```
Observable.just(1, 2, 3, 4, 5).reduce(new BiFunction, Integer, Integer>() {
    public Integer apply(@NonNull Integer integer, @NonNull Integer integer2) throws Except
on {
        return integer + integer2;
  }
}).toObservable().subscribe(RxUtils.getObserver());
```

```
onSubscribe
Thread:Thread[main,5,main]
onNext:15
Thread:Thread[main,5,main]
onComplete
Thread:Thread[main,5,main]
```

Count

计算数据项的个数并发射结果

```
Observable.just(1,2,3,4)
        .count().toObservable()
        .subscribe(RxUtils.getObserver());
```

```
onSubscribe
Thread:Thread[main,5,main]
onNext:4
Thread:Thread[main,5,main]
onComplete
Thread:Thread[main,5,main]
```

collect

Collect操作符类似于Reduce，但是其目的不同，collect用将源Observable发射的数据给收集到一个数据结构里面，需使用两个参数：

- 一个产生收集数据结构的函数。
- 一个接收第一个函数产生的数据结构和源Observable发射的数据作为参数的函数。

```
List list=new ArrayList();
list.add(1);
list.add(2);
list.add(3);
list.add(4);
list.add(5);
Observable.fromIterable(list).collect(new Callable>() {
    public ArrayList call() throws Exception {
        return new ArrayList();
  }
}, new BiConsumer, Integer>() {
    public void accept(ArrayList integers, Integer integer) throws Exception {
        integers.add(integer);
  }
}).toObservable().subscribe(RxUtils.>getObserver());
```

```
  onSubscribe
  Thread:Thread[main,5,main]
  onNext:[1, 2, 3, 4, 5]
  Thread:Thread[main,5,main]
  onComplete
  Thread:Thread[main,5,main]
```