

# JDK8函数式编程（二）

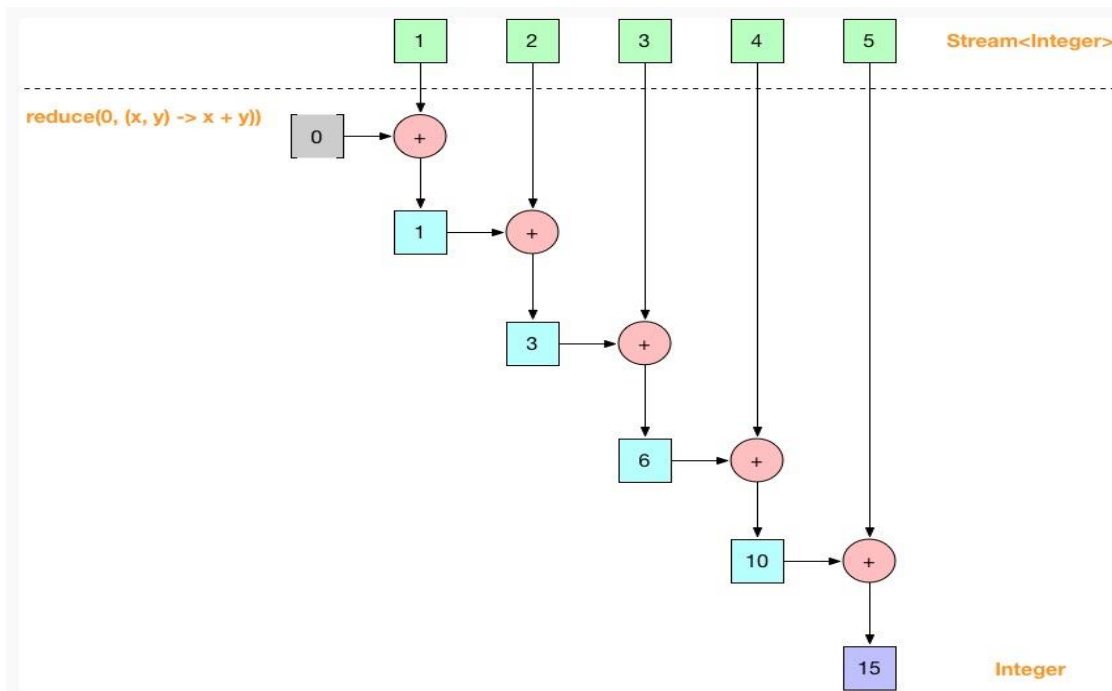
Tim

## 大纲

- Stream.collect()
- Collectors源码解析及相关实战
- Optional 与 functor
- map与flatMap的区别
- Function composition



## Reduce



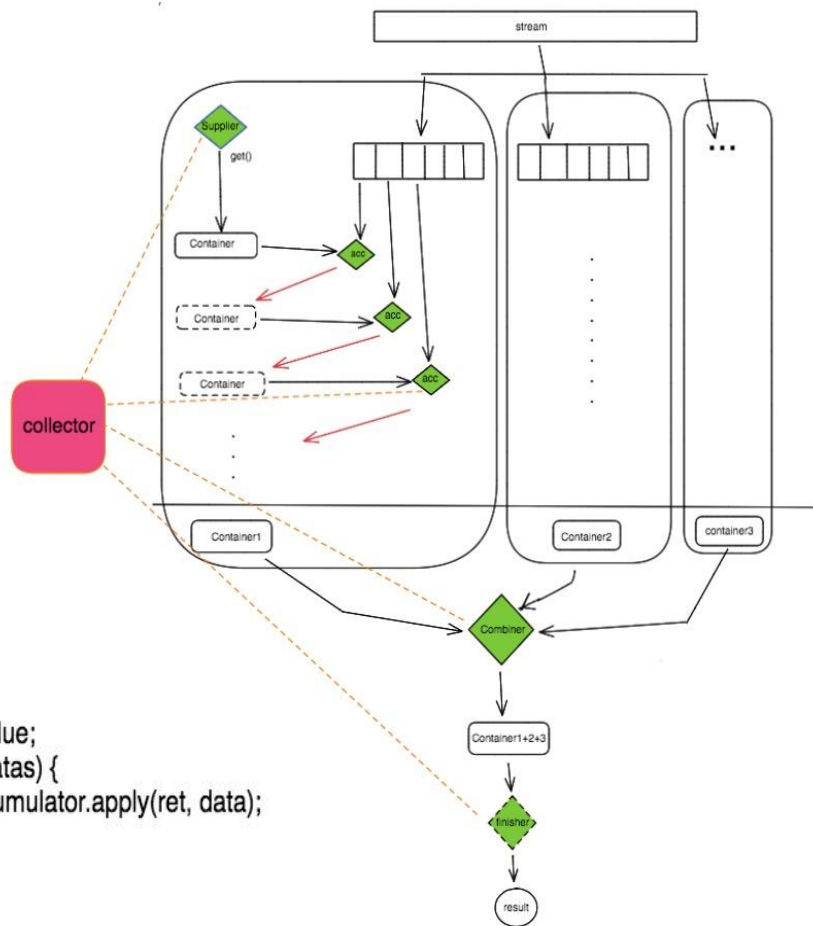
## Stream.collect()

- collect vs reduce?
  - reduce 操作不变数据
  - collect 操作可变数据
- collect(Supplier, Accumulator, Combiner)
- collect(Collector)

```
R container = supplier.get();
for (T data : datas){
    accumulator.accept(container, data);
}
return container
```

VS

```
R ret = initValue;
for (T data : datas) {
    ret = accumulator.apply(ret, data);
}
return ret;
```



## Collector

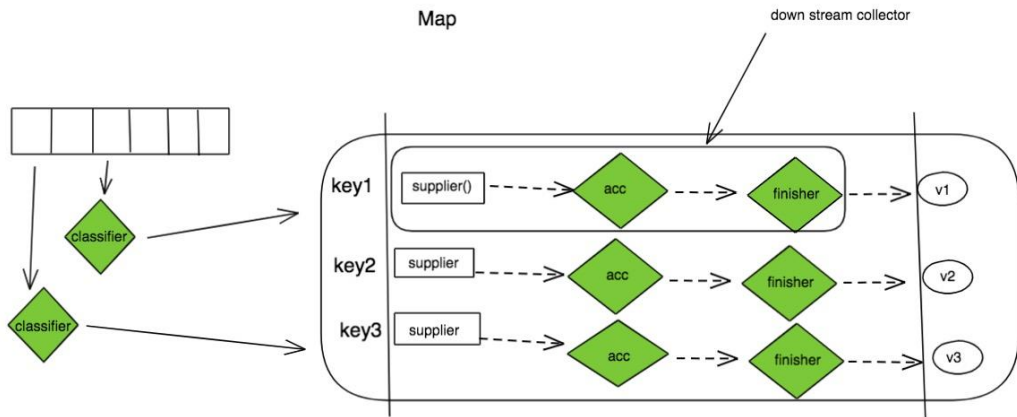
- Collector 要素
  - Supplier: 累积数据构造函数
  - Accumulator: 累积函数，同reduce
  - Combiner: 合并函数，并行处理场合下用，同reduce
  - Finisher: 对累积数据做最终转换
  - \*Characteristics: 特征（并发 / 无序 / 无finisher）
- Collector 需要满足：
  - 同一律： `Combiner.apply(acc, []) == acc`
  - 结合率：
    - `acc.accept(t1), acc.accept(t2)`                      `acc1.accept(t1) , acc2.accept(t2)`
    - `finisher.apply(acc)`                                      `== finisher.apply(combiner.apply(t1,t2))`

## Collectors API

- toList/to(Concurrent)Map/toSet/toCollection
- counting/averagingXX/joining/summingXX
- groupBy/partitioningBy
- mapping/reducing

## Collectors.groupBy

- `groupByKey(Function)` – 单纯分key存放成`Map<key, List>`，默认使用`HashMap`
- `groupByKey(Function, Collector)` - 分key后，对每个key的元素进行后续collect操作
- `groupByKey(Function, Supplier, Collector)` - 同上，允许自定义Map创建



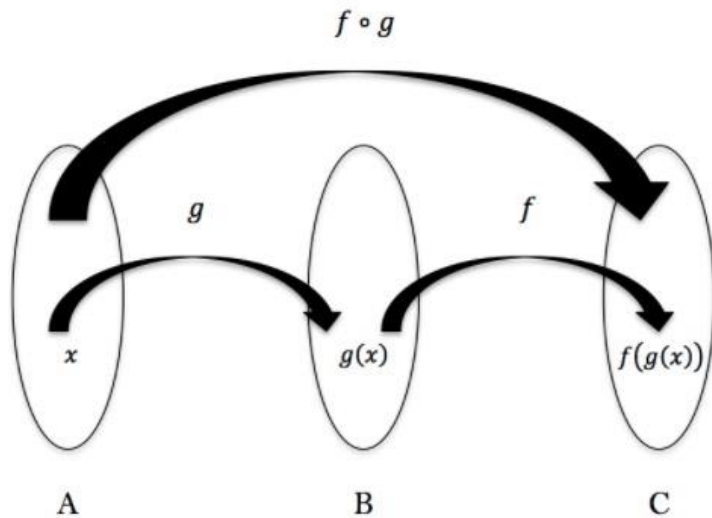
## Collect 实战

1. 先按编程语言再按程序等级分层，然后返回一个元组<平均工资，程序员列表>
2. 随机创建程序员列表
  - 创建指定个数长度的stream
  - 对每个数字使用构造函数创建对象
  - 对每个对象的各个field进行随机赋值
  - 过滤数据范围以外的元素
  - 按指定字段排序
  - 生产到list

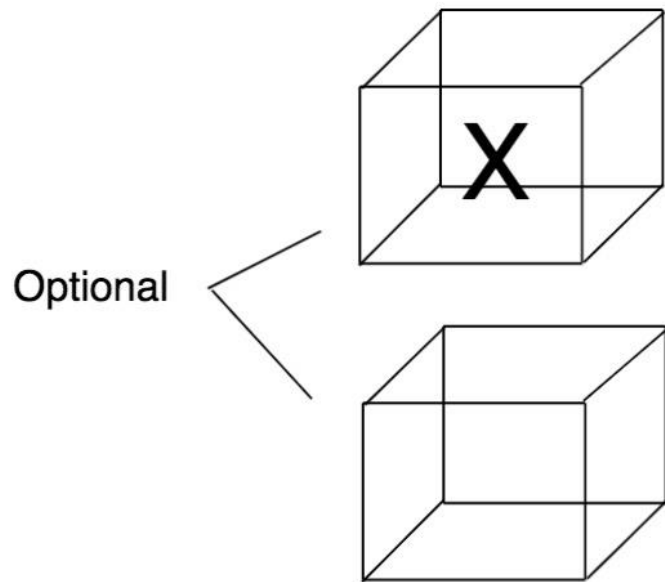


## Function composition

- `f1.andThen(f2)`  $\rightarrow$  `f2(f1())`
- `f1.compose(f2)`  $\rightarrow$  `f1(f2())`



Optional

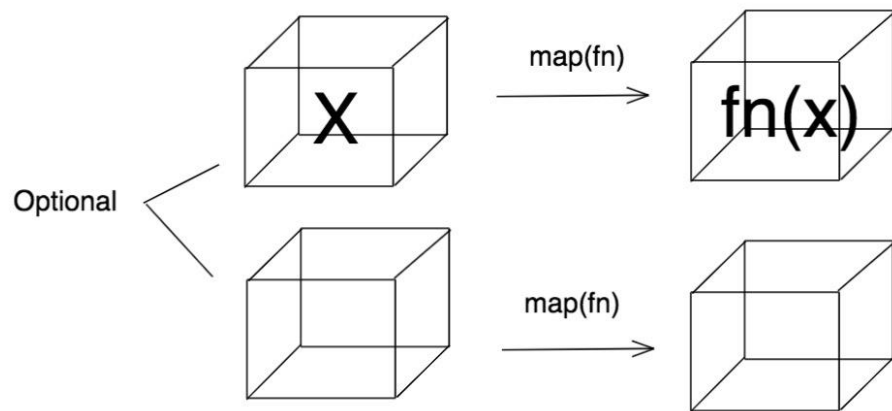


## Optional API

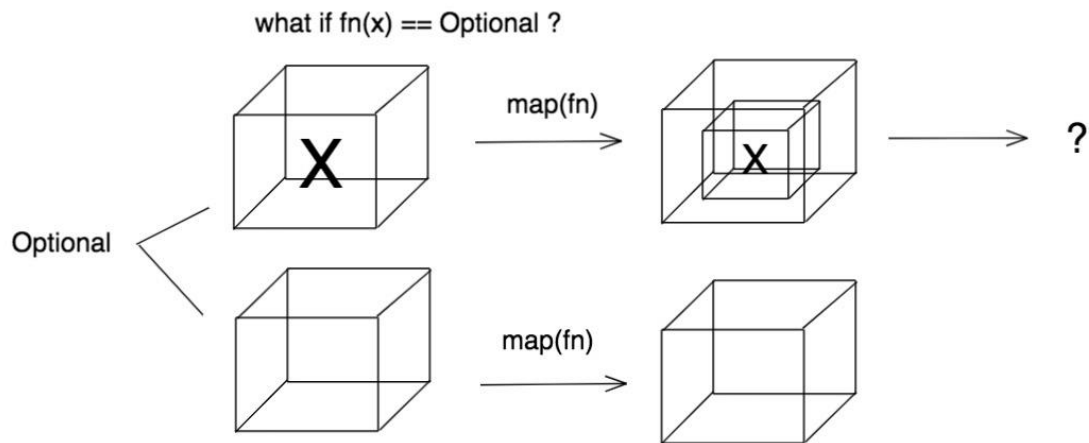
- `orElse(T)` → if `x != null` return `x` else return `T`
- `orElseGet(fn)` → if `x != null` return `x` else return `fn()`
- `ifPresent(fn)` → if `x != null` `fn()`



## map()

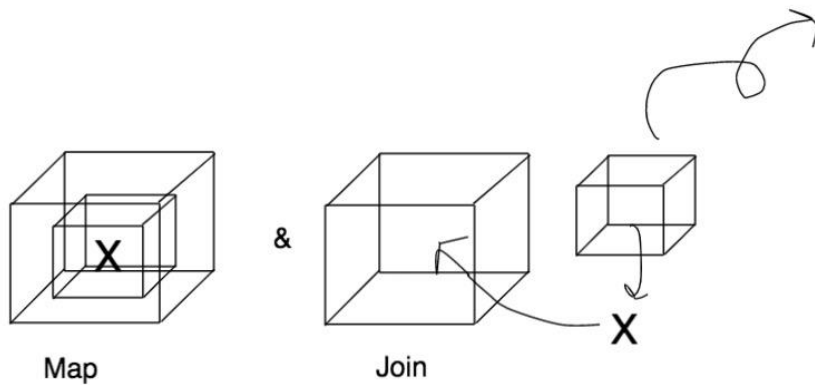


## 思考

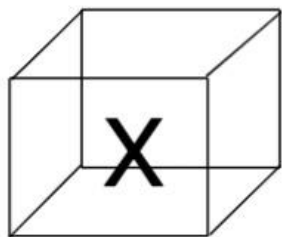


## FlatMap()

`flatMap(Fn) ==`



## Functor & Monad



Optional: null or T

Stream: 0..n

Either: A or B

Promise: has result or not yet

IO : IO operation

Q & A

