Reducing Data to Compute Statistics

José Paumard

PHD, JAVA CHAMPION, JAVA ROCK STAR

@JosePaumard https://github.com/JosePaumard

Agenda

Let us focus on the reduction step

How is a reduction computed

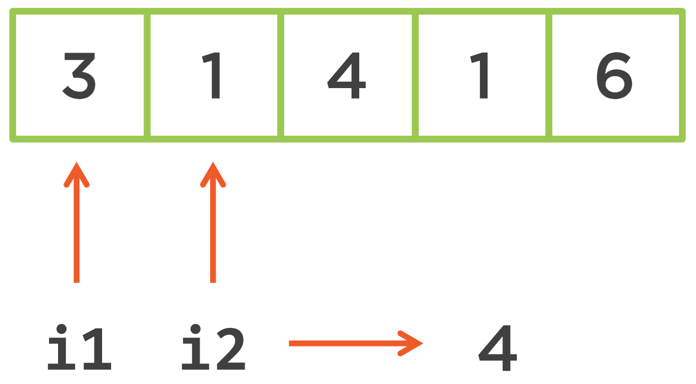
Reducing no data

And why is Optional needed

Reducing Data

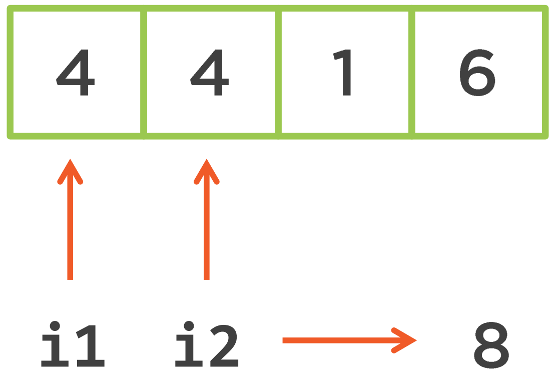
Reducing Numbers to a Sum

```
BinaryOperator<Integer> sum = (i1, i2) -> i1 + i2;
```



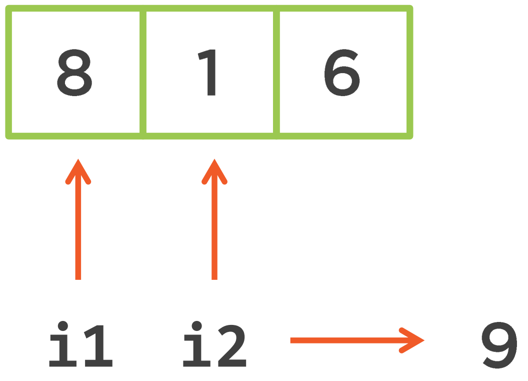
Reducing Numbers to a Sum

```
BinaryOperator<Integer> sum = (i1, i2) -> i1 + i2;
```



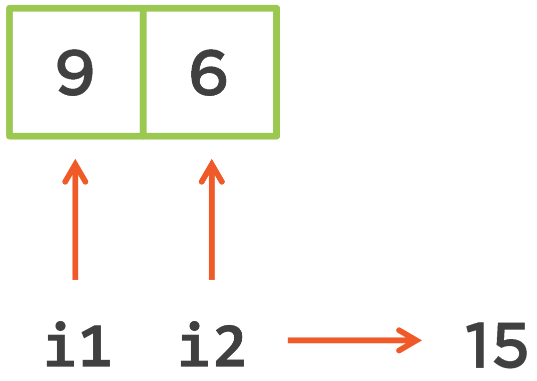
Reducing Numbers to a Sum

```
BinaryOperator<Integer> sum = (i1, i2) -> i1 + i2;
```



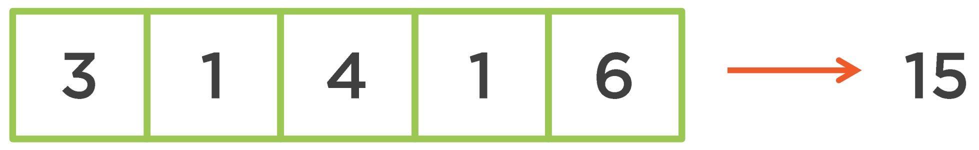
Reducing Numbers to a Sum

```
BinaryOperator<Integer> sum = (i1, i2) -> i1 + i2;
```



Reducing Numbers to a Sum

```
BinaryOperator<Integer> sum = (i1, i2) -> i1 + i2;
```



Reducing Numbers to a Sum

```
BinaryOperator<Integer> sum = (i1, i2) -> i1 + i2;
```

3	1	4	1	6

```
((((3 + 1) + 4) + 1) + 6)
```

Reducing Numbers to a Sum

```
BinaryOperator<Integer> avg = (i1, i2) -> (i1 + i2) / 2;
```

3	1	4	1	6

```
((((3 + 1)/2 + 4)/2 + 1)/2 + 6)/2 = 4
```

The result should be 3

Reducing a Singleton

Reducing a Singleton

```
BinaryOperator<Integer> sum = (i1, i2) -> i1 + i2;
```

$\boxed{3} \longrightarrow ?$

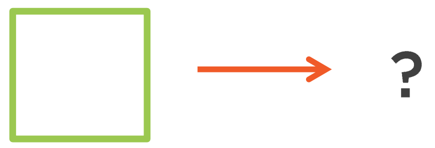
Reducing a Singleton

```java
BinaryOperator<Integer> sum = (i1, i2) -> i1 + i2;
```

$$\boxed{3} \longrightarrow 3$$

Reducing an Empty Stream

```
BinaryOperator<Integer> sum = (i1, i2) -> i1 + i2;
```



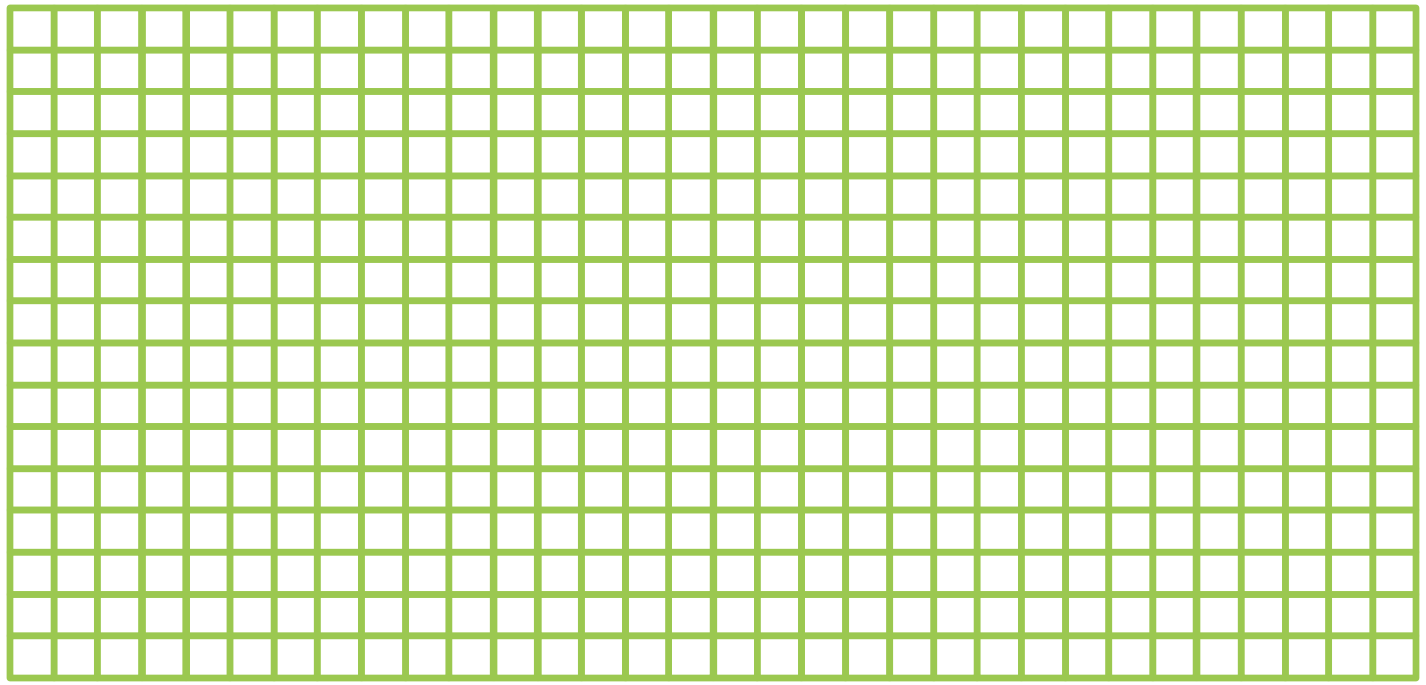
Reducing an Empty Stream

```
BinaryOperator<Integer> sum = (i1, i2) -> i1 + i2;
```

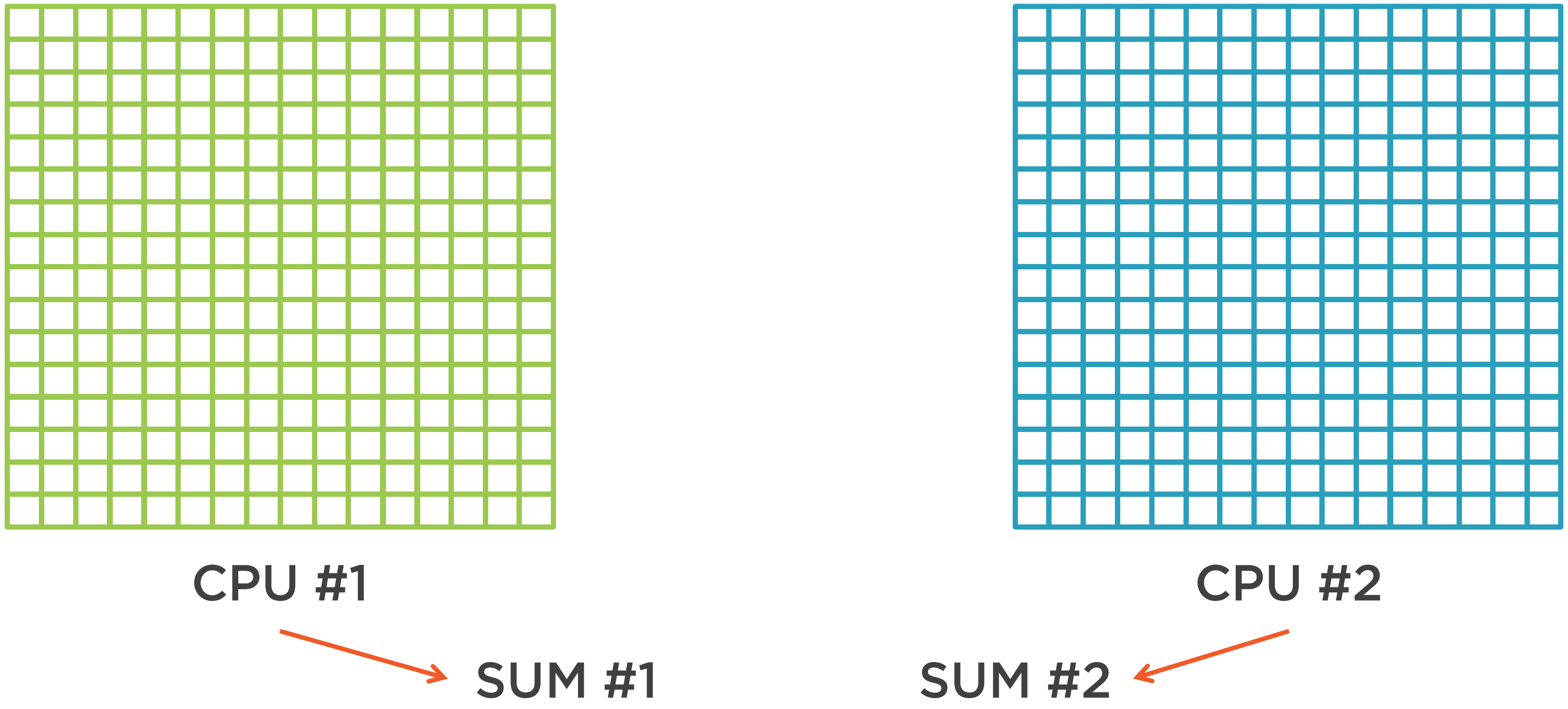
$\square \longrightarrow$ "I don’t know"

Reducing an Empty Stream

Reducing a Lot of Data



Reducing a Lot of Data



Reducing a Lot of Data

$A = A_1 \cup A_2$

$Sum(A_1) = s_1 \qquad Sum(A_2) = s_2$

$Sum(A) = Sum(A_1 \cup A_2) = Sum(s_1, s_2)$

$Red(A) = Red(A_1 \cup A_2) = Red(Red(A_1), Red(A_2))$

Reducing a Lot of Data

$$A = A_1 \cup \emptyset$$

Reducing a Lot of Data

$A = A \cup \emptyset$

$Red(A) = Red(A_1 \cup A_2) = Red(Red(A_1), Red(A_2))$

$Red(A) = Red(A \cup \emptyset) = Red(Red(A), Red(\emptyset))$

Which is true if and only if

$Red(\emptyset)$ is the identity element of the reduction

Reducing an Empty Stream

```
BinaryOperator<Integer> sum = (i1, i2) -> i1 + i2;
```

$\square \longrightarrow$ "I don’t know"

Reducing an Empty Stream

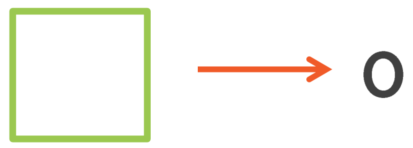
```
BinaryOperator<Integer> sum = (i1, i2) -> i1 + i2;
```

$\boxed{} \longrightarrow$ **The identity element of sum**

The reduction of an empty stream
is the identity element
of the reduction operation

Reducing an Empty Stream

```
BinaryOperator<Integer> sum = (i1, i2) -> i1 + i2;
```



Reducing an Empty Stream

```
BinaryOperator<Integer> sum = (i1, i2) -> i1 > i2 ? i1 : i2;
```

What is the identity element of max ?

Quick answer: there is no identity element for max…

Then what is the max of an empty stream?

Implementing the Reduction

Some reduction operators do not have any identity element

How can these be implemented?

```
List<Person> people = ...;

int sum =
    people.stream()
            .map(person -> person.getAge())
            .filter(age -> age > 20)
            .reduce(0, (i1, i2) -> i2 + i2);
```

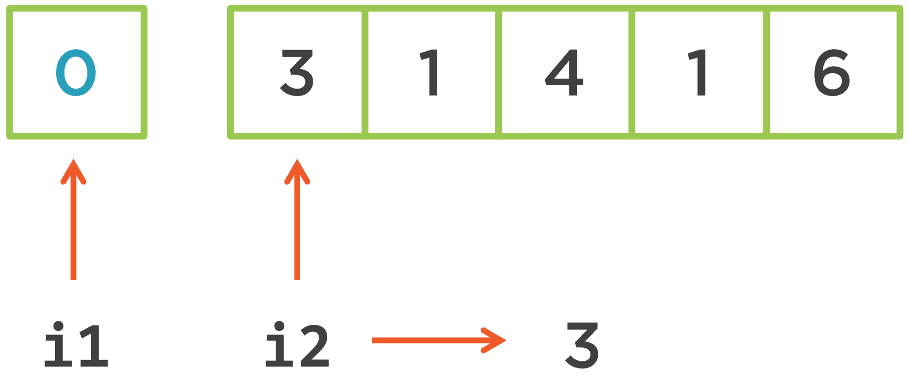
If the reduction operator has an identity element

Then it can be passed to the reduce method

If the processed stream is empty, then the identity element is returned

Implementation of the Reduce Method

```
BinaryOperator<Integer> sum = (i1, i2) -> i1 + i2;
```



```java
List<Person> people = ...;

Optional<Integer> optionalOfSum =
    people.stream()
          .map(person -> person.getAge())
          .filter(age -> age > 20)
          .reduce((i1, i2) -> i2 + i2);
```

If the reduction operator has no identity element

Or if no element is provided

Then it the reduce method wraps the result in an Optional object

```
Optional<Integer> optionalOfSum = ...;

optionalOfSum.get();          // Java 8
optionalOfSum.orElseThrow(); // Java 10
```

You can check if an optional holds a value with isPresent() or isEmpty()

And get this value with get() or orElseThrow()

Both throws a NoSuchElementException is the optional is empty

Handling Optional Objects

A reduction operator
with no identity element
returns an Optional

Identity Element of Reductions

Reductions that return Optional

reduce(BinaryOperator)

min()

max()

average()

Demo

Let us write some code!

See these reductions in action

And play with identity elements

Module Wrap Up

What did you learn?

How does the reduction step work

The importance of an identity element

How reduction is handled when there is no identity element

Why Optional has been added