Learning Objectives in this Part of the Lesson

• Recognize key Observable operators
  • Factory method operations
  • Transforming operators
  • Concurrency & scheduler operators
• Error handling operators
• Combining operators
  • This operator creates a Maybe by accumulating elements in an Observable stream
    • e.g., reduce()
Key Combining Operators in the Observable Class
Key Combining Operators in the Observable Class

• The reduce() operator
  • Reduce this Observable’s values into a single object of the same type as the emitted items

```
Maybe<T> reduce
  (BiFunction<T, T, T> reducer)
```

See reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/core/Observable.html#reduce
Key Combining Operators in the Observable Class

• The reduce() operator
  • Reduce this Observable’s values into a single object of the same type as the emitted items
  • Reduction is performed using a BiFunction param

\[
\text{Maybe<T> reduce (BiFunction<T, T, T> reducer)}
\]

See reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/functions/BiFunction.html
Key Combining Operators in the Observable Class

- The reduce() operator
  - Reduce this Observable’s values into a single object of the same type as the emitted items
  - Reduction is performed using a BiFunction param
    - This param is passed the intermediate result of the reduction & the current value

```java
Maybe<T> reduce
    (BiFunction<T, T, T> reducer)
```

Observable of Integers from 1..4

```
0 + 1 + 2 + 3 + 4
0 + 1 + 3 + 6 + 10
```

This value is initialized to zero (0) for Integer
Key Combining Operators in the Observable Class

- The reduce() operator
  - Reduce this Observable’s values into a single object of the same type as the emitted items
    - Reduction is performed using a BiFunction param
      - This param is passed the intermediate result of the reduction & the current value
        - It returns the next intermediate value of the reduction

```java
Maybe<T> reduce
  (BiFunction<T, T, T> reducer)
```

Observable of Integers from 1..4

```
0 + 1 + 2 + 3 + 4 = 10
```
Key Combining Operators in the Observable Class

• The reduce() operator
  • Reduce this Observable’s values into a single object of the same type as the emitted items
  • Reduction is performed using a BiFunction param
    • This param is passed the intermediate result of the reduction & the current value
      • It returns the next intermediate value of the reduction
    • This process repeats for each pair of values

Maybe\<T>\ reduce
(BiFunction\<T, T, T> reducer)

Observable of Integers from 1..4

0
1
2
3
4

0 +
1 +
2 +
3 +
4 +

1 +
3 +
6 +
10 +
Key Combining Operators in the Observable Class

- The reduce() operator
  - Reduce this Observable’s values into a single object of the same type as the emitted items
  - Reduction is performed using a BiFunction param
    - This param is passed the intermediate result of the reduction & the current value
      - It returns the next intermediate value of the reduction
    - This process repeats for each pair of values

Maybe\(<T>\) reduce
(BiFunction\(<T, T, T>\) reducer)

Observable of Integers from 1..4

```
0 + 1 -> 1 + 2 -> 3 + 3 -> 6 + 4 -> 10
```

```java
Maybe<Integer> reduce(BiFunction<Integer, Integer, Integer> reducer)
```
• The reduce() operator
• Reduce this Observable’s values into a single object of the same type as the emitted items
  • Reduction is performed using a BiFunction param
  • This param is passed the intermediate result of the reduction & the current value
    • It returns the next intermediate value of the reduction
  • This process repeats for each pair of values
Key Combining Operators in the Observable Class

• The `reduce()` operator
  • Reduce this Observable’s values into a single object of the same type as the emitted items
  • Reduction is performed using a `BiFunction` param
    • This param is passed the intermediate result of the reduction & the current value
      • It returns the next intermediate value of the reduction
  • This process repeats for each pair of values

```java
Maybe<T> reduce
    (BiFunction<T, T, T> reducer)
```

Observable of Integers from 1..4

```
0 + 1 + 2 + 3 + 4 = 10
```
Key Combining Operators in the Observable Class

• The reduce() operator
  • Reduce this Observable’s values into a single object of the same type as the emitted items
  • Reduction is performed using a BiFunction param
    • This param is passed the intermediate result of the reduction & the current value
      • It returns the next intermediate value of the reduction
  • This process repeats for each pair of values

```java
Maybe<T> reduce
  (BiFunction<T, T, T> reducer)
```

Observable of Integers from 1..4

```
0
1
2
3
4
```

```
+  +  +  +
1  2  3  4
```

```
1
3
6
10
```

```
+  +  +  +
1  2  3  4
```

```
+  +  +  +
1  2  3  4
```

```
+  +  +  +
1  2  3  4
```

```
+  +  +  +
1  2  3  4
```

```
+  +  +  +
1  2  3  4
```

```
+  +  +  +
1  2  3  4
```

```
+  +  +  +
1  2  3  4
```

```
+  +  +  +
1  2  3  4
```

```
+  +  +  +
1  2  3  4
```

```
+  +  +  +
1  2  3  4
```

```
+  +  +  +
1  2  3  4
```

```
+  +  +  +
1  2  3  4
```

```
+  +  +  +
1  2  3  4
```

```
+  +  +  +
1  2  3  4
```

```
+  +  +  +
1  2  3  4
```

```
+  +  +  +
1  2  3  4
```

```
+  +  +  +
1  2  3  4
```

```
+  +  +  +
1  2  3  4
```

```
+  +  +  +
1  2  3  4
```

```
+  +  +  +
1  2  3  4
```

```
+  +  +  +
1  2  3  4
```

```
+  +  +  +
1  2  3  4
```

```
+  +  +  +
1  2  3  4
```

```
+  +  +  +
1  2  3  4
```

```
+  +  +  +
1  2  3  4
```

```
+  +  +  +
1  2  3  4
```
Key Combining Operators in the Observable Class

• The reduce() operator
  • Reduce this Observable’s values into a single object of the same type as the emitted items
    • Reduction is performed using a BiFunction param
      • This param is passed the intermediate result of the reduction & the current value
        • It returns the next intermediate value of the reduction
      • This process repeats for each pair of values
Key Combining Operators in the Observable Class

- The reduce() operator
  - Reduce this Observable’s values into a single object of the same type as the emitted items
    - Reduction is performed using a BiFunction param
  - The final result is emitted from the final call as the sole item of a Maybe

\[ \text{Maybe}<T> \ reduce (\text{BiFunction}<T, T, T> \ \text{reducer}) \]

See [reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/core/Maybe.html](http://reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/core/Maybe.html)
Key Combining Operators in the Observable Class

• The reduce() operator
  • Reduce this Observable’s values into a single object of the same type as the emitted items
    • Reduction is performed using a BiFunction param
  • The final result is emitted from the final call as the sole item of a Maybe
    • An empty Maybe will be returned if the Observable emits no items

```java
Maybe<T> reduce
  (BiFunction<T, T, T> reducer)
```
Key Combining Operators in the Observable Class

• The reduce() operator
  • Reduce this Observable’s values into a single object of the same type as the emitted items
    • Reduction is performed using a BiFunction param
  • The final result is emitted from the final call as the sole item of a Maybe
    • An empty Maybe will be returned if the Observable emits no items
  • The internally accumulated value is discarded upon cancellation or error

```typescript
Maybe<T> reduce
(BiFunction<T, T, T> reducer)
```
The reduce() operator

- Reduce this Observable’s values into a single object of the same type as the emitted items
- Upstream must signal `onComplete()` before accumulator can be emitted

```java
return Observable
    .fromArray(bigFractions)
    ...
    .flatMap(bf ->
        multiplyFractions(bf, Schedulers.computation())
    )
    .reduce(BigFraction::add)
    ...
```

Sum the results of async multiplications

See Reactive/Observable/ex3/src/main/java/ObserableEx.java
Key Combining Operators in the Observable Class

- The `reduce()` operator
  - Reduce this Observable’s values into a single object of the same type as the emitted items
  - Upstream must signal `onComplete()` before accumulator can be emitted
  - Sources that are infinite & never complete will never emit anything through this operator
Key Combining Operators in the Observable Class

• The reduce() operator
  • Reduce this Observable’s values into a single object of the same type as the emitted items
  • Upstream must signal onComplete() before accumulator can be emitted
    • Sources that are infinite & never complete will never emit anything through this operator
      • An infinite source may lead to a fatal OutOfMemoryError

See docs.oracle.com/javase/8/docs/api/java/lang/OutOfMemoryError.html
Key Combining Operators in the Observable Class

- The reduce() operator
  - Reduce this Observable’s values into a single object of the same type as the emitted items
  - Upstream must signal onComplete() before accumulator can be emitted
- Project Reactor’s Flux.reduce() operator works the same

```java
return Flux
  .fromArray(bigFractions)
  .flatMap(bf -> multiplyFractions(bf, Schedulers.parallel()))
  .reduce(BigFraction::add)
...```

*Sum results of async multiplications*

See [projectreactor.io/docs/core/release/api/reactor/core/publisher/Flux.html#reduce](projectreactor.io/docs/core/release/api/reactor/core/publisher/Flux.html#reduce)
Key Combining Operators in the Observable Class

- The reduce() operator
  - Reduce this Observable’s values into a single object of the same type as the emitted items
  - Upstream must signal onComplete() before accumulator can be emitted
  - Project Reactor’s Flux.reduce() operator works the same
  - Similar to the Stream.reduce() method in Java Streams

```java
int result = List.of(1, 2, 3, 4, 5, 6).stream()
  .reduce(0, Math::addExact);
```

See [docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html#reduce](docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html#reduce)
End of Key Combining Operators in the Observable Class (Part 2)