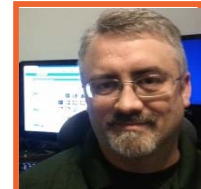


Reactive Programming in Java 8 with RxJava

Functional Reactive Concepts

Russell Elledge
@russellelledge



pluralsight
hardcore dev and IT training



Course Agenda

Functional Reactive
Concepts

“Observable” Creation, Composition, and Filtering

Observable Transformations and Conditional Operations

Connectable Observables, Resource Management,
and Publish/Subscribe Subjects

Implementation Patterns

Module Agenda

Functional Reactive Concepts

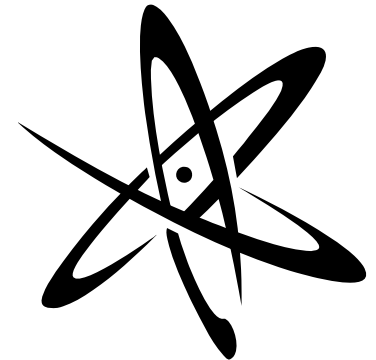
Functional Programming
Concepts



$y = f(x)$ Pure Functions
Functions as First Class Citizens
High Order Functions

The Reactive Manifesto
Event Driven
Scalable
Resilient
Responsive

Reactive Programming
Concepts





Functional Programming – Pure Functions

Demonstration



Functional Programming – Pure Functions

- **Pure Functions** always return the same result for a given set of parameter values.
 - No side effects caused by Class or Instance state.
 - No side effects caused by I/O devices.
 - No time related side effects.
- **Code Example**
 - `f1(5)` will always equal 10...it is a pure function.
 - `f2(5)` returns something different based on information that is out of the callers control.

Functional Programming – Functions as First Class Citizens

Demonstration

Functional Programming – Functions as First Class Citizens

- **Definition:** The ability to store a function as a variable and pass that function as a parameter.
- **Code Example**
 - Variables may contain functions
 - Functions can have defined types, making them valid parameters

Functional Programming – High Order Functions

Demonstration

Functional Programming – High Order Functions

- **Definition:** A high order function is a function that can return a function.
- **Uses related to reactive programming...**
 - Composition
 - Lazy execution

Reactive Programming

The Reactive Manifesto
<http://www.reactivemanifesto.org>



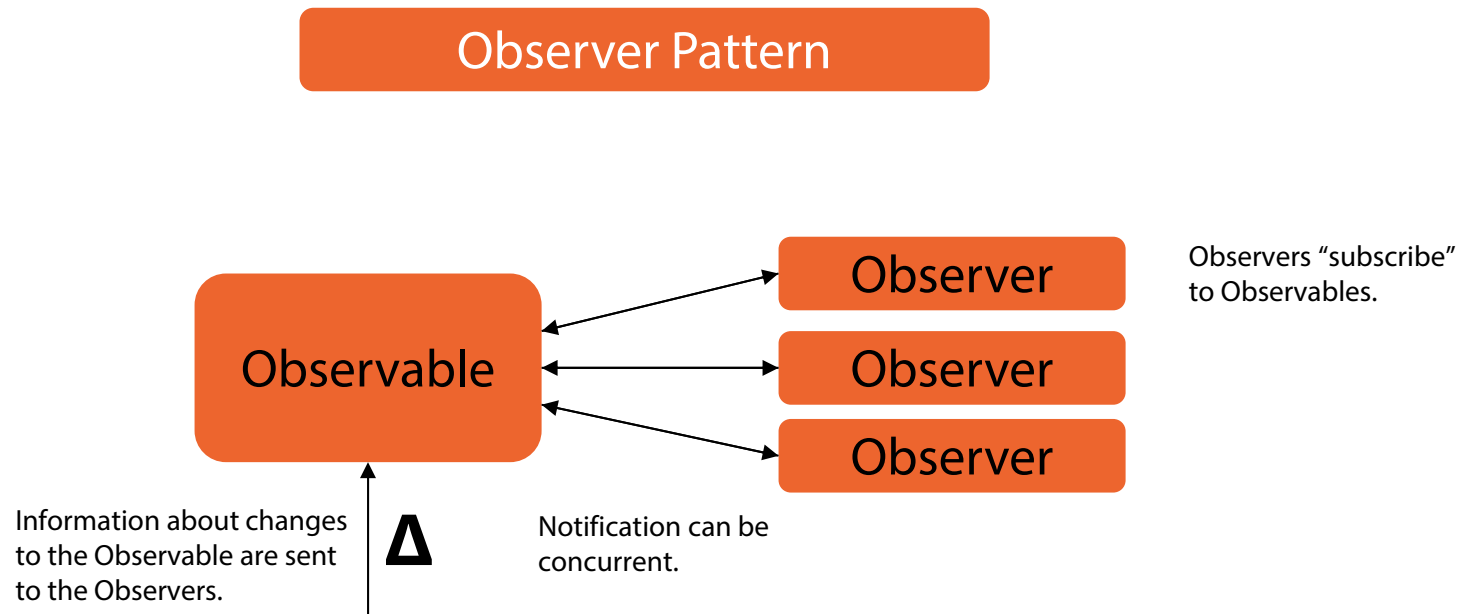
Event Driven

Scalable

Resilient

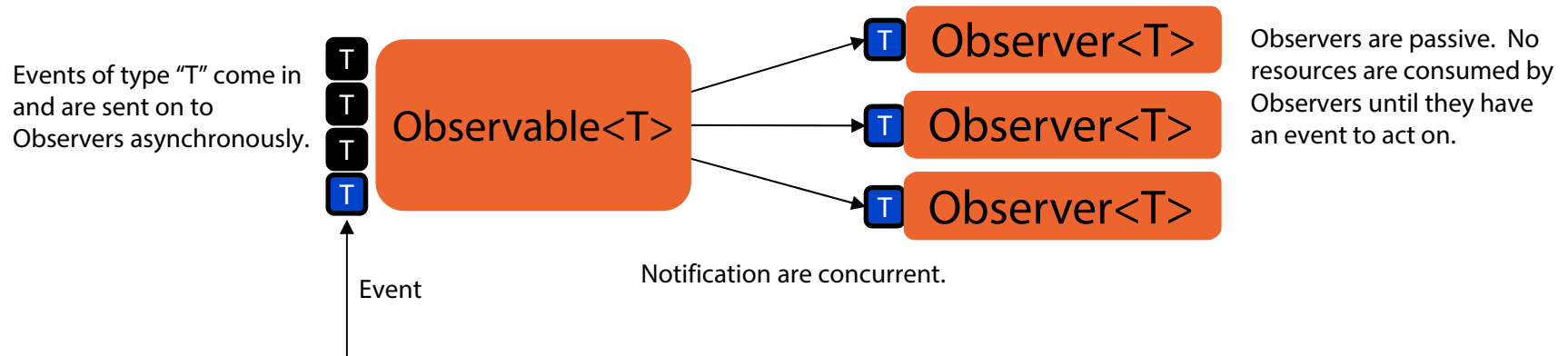
Responsive

Reactive Programming – Event Driven

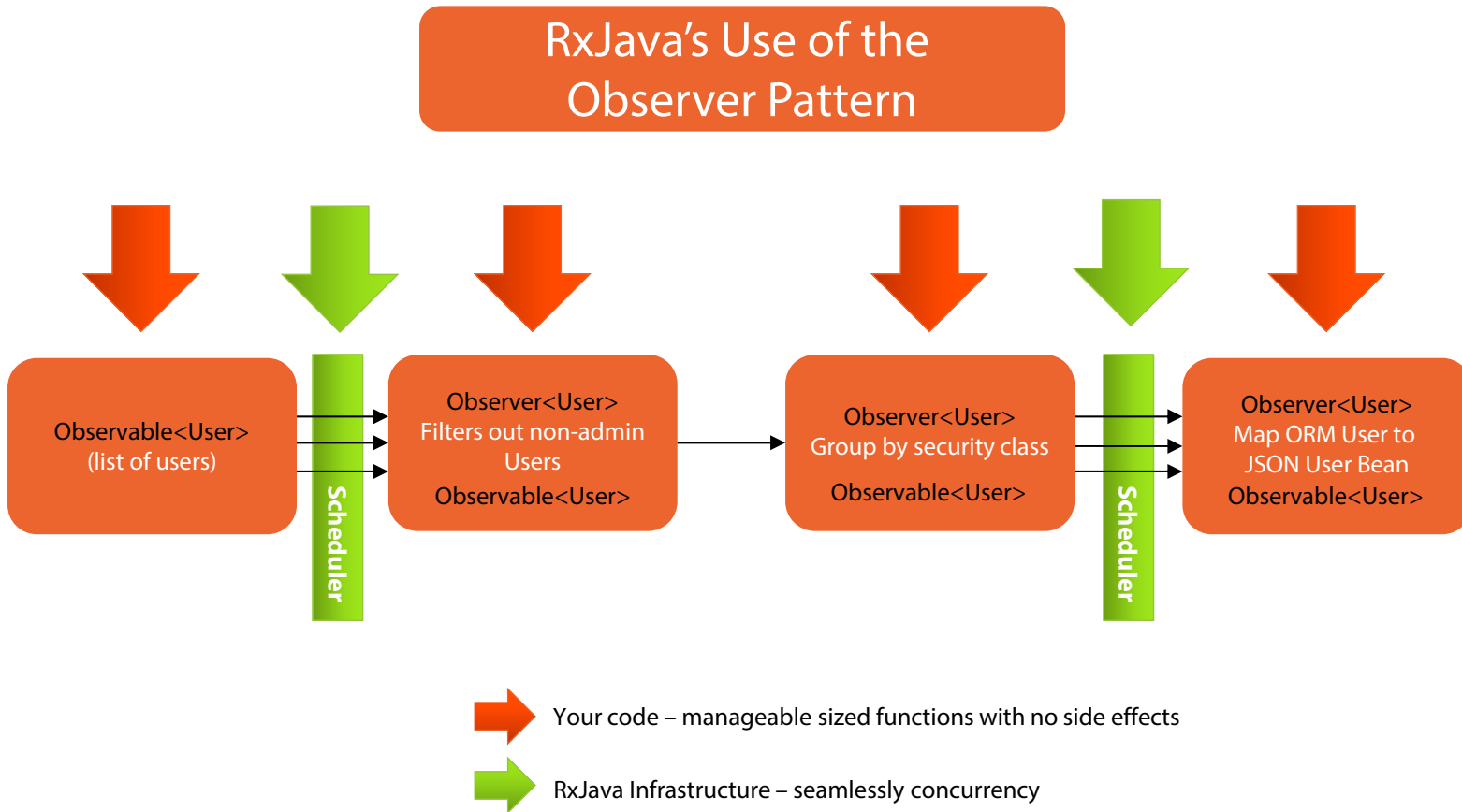


Reactive Programming – Event Driven

RxJava's Use of the Observer Pattern



Reactive Programming – Event Driven



Reactive Programming – Event Driven

Q: What software pattern does the RxJava heavily leverage?

A: The Observer Pattern

Q: In RxJava, changes to an Observable are called what?

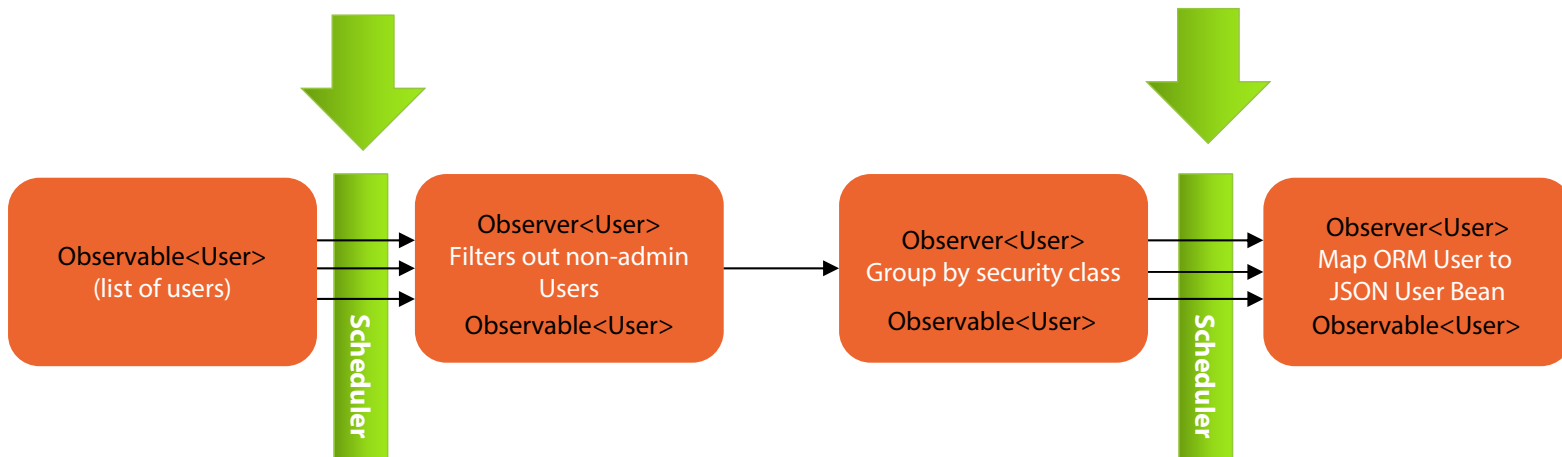
A: Events



Q: Are Observers actively running, or completely passive when not responding to events?

A: Passive

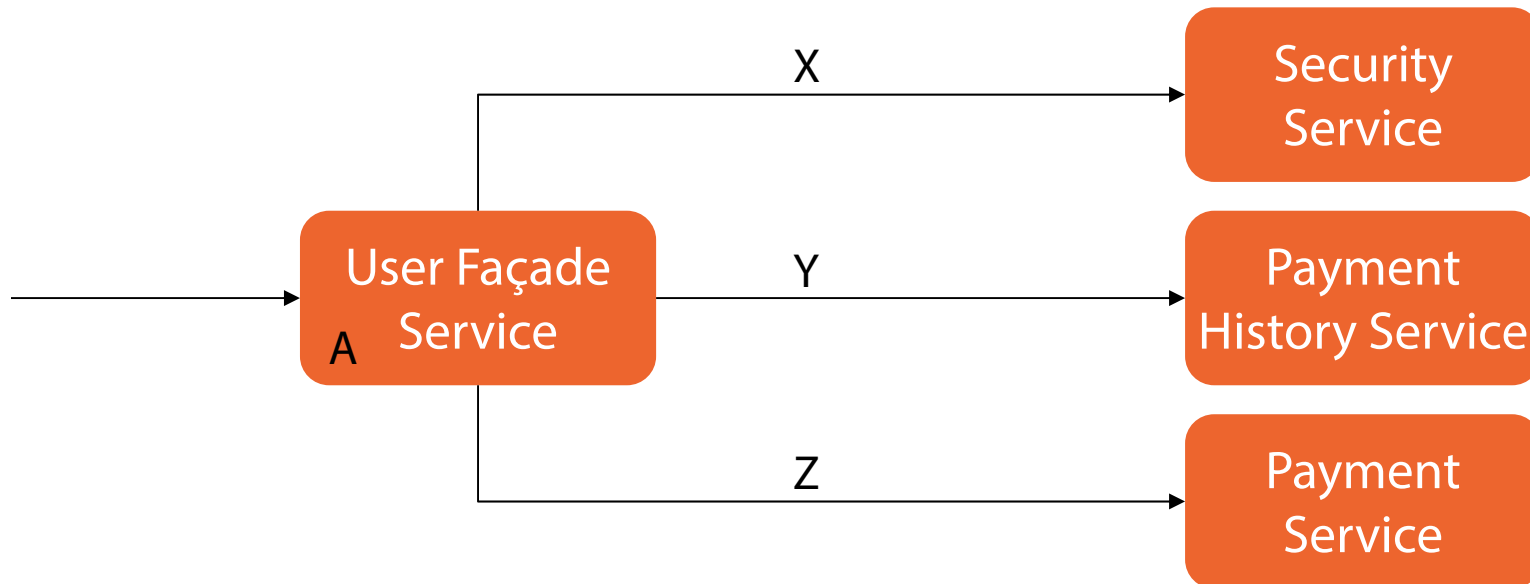
Reactive Programming – Scalable

RxJava's Handles the Threading Details



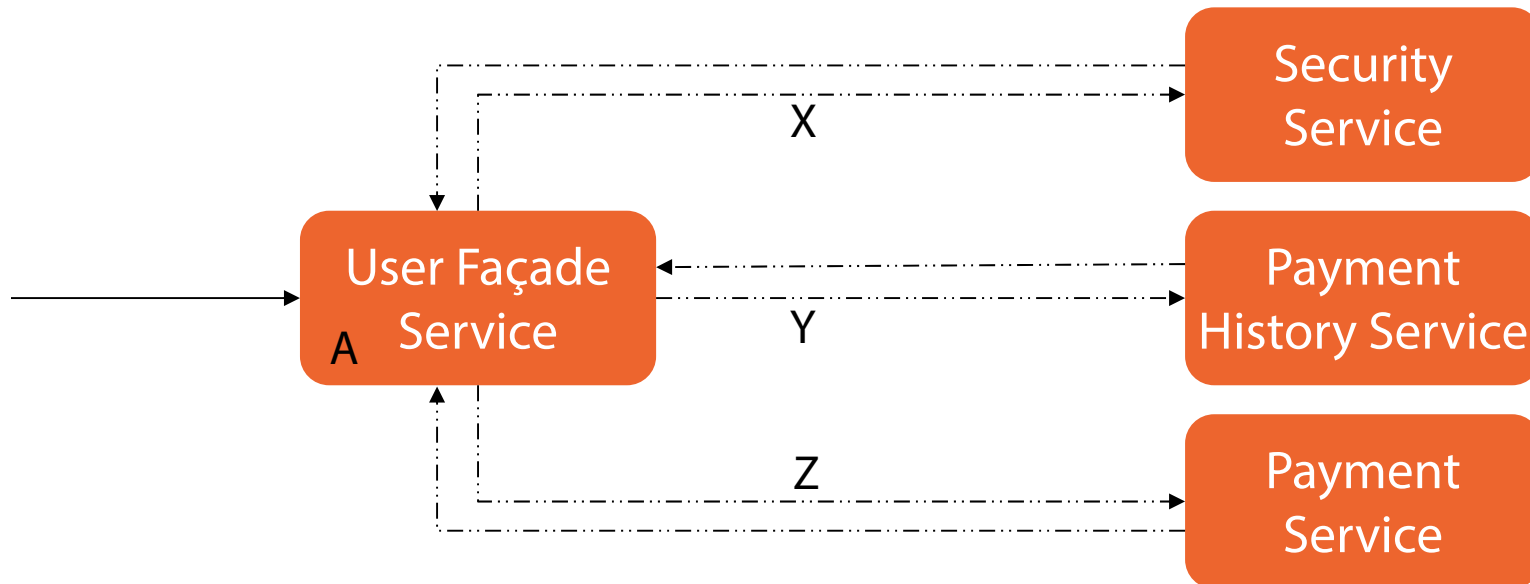
-  Your code – manageable sized functions with no side effects
-  RxJava Infrastructure – seamlessly concurrency

Reactive Programming - Scalable



Total Execution Time = $A + X + Y + Z$

Reactive Programming - Scalable



$$\text{Total Execution Time} = \text{MAX}(X, Y, Z) + A$$

Reactive Programming – Scalable

Q: Which is faster, serial execution of tasks or parallel execution of tasks?

A: Parallel

Q: When implementing RxJava Observers, what programming style is best?

A: Functional

Q: True or False? As a developer using RxJava, you must handle the creation and management of all threads?

A: False

Reactive Programming - Resilient



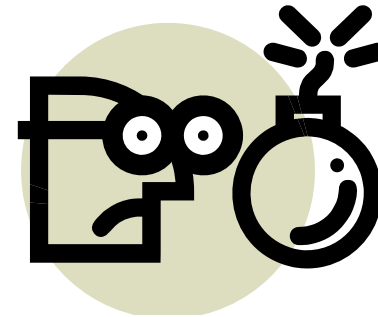
Graceful Error Handling

Handle more than just the “happy path”
Attempt to provide value even in the
face of failure

Manage Failure

Bulkheads – Prevent failed components
from affecting healthy ones.

Event driven structure can help create components
that are location transparent.



Reactive Programming – Resilience

Q: What technique can be used to ensure that failing components do not affect healthy ones?

A: Bulkheads

Q: What goal should we always have, even when something in our application has failed?

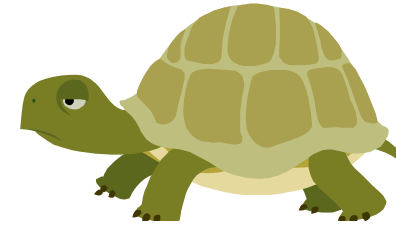
A: Provide value to the user

Q: Reactive Programming's event driven structure helps components to have what attribute?

A: Location Transparency

Reactive Programming - Responsive

Slow Response is considered a failure.



Poor performance is failure

Define and implement to the desired level of performance

Observable Data Models



Automatic UI update via background processes

Users generate streams of events by their actions

Reactive Programming – Responsive

Q: True or False? The Reactive Manifesto states that calls to dependent systems that run slowly should be considered a failure.

A: True. The performance of a system should be defined to ensure a good user experience. Responses that are too slow should be treated as failures.

Q: In a Reactive Application, components and systems are connected using what model?

A: Event Streams

Q: What implementation pattern is often used to create a responsive UI?

A: MVVM – Model, View, View Model