Recognize How Java Combines Object-Oriented & Functional Programming

Douglas C. Schmidt

d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science

Institute for Software Integrated Systems

Vanderbilt University

Nashville, Tennessee, USA

Learning Objectives in this Lesson

- Recognize the benefits of combining object-oriented & functional programming in Java

Again, we show modern Java code fragments we'll cover in more detail later

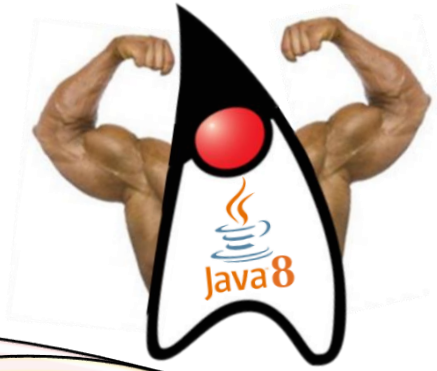
Learning Objectives in this Lesson

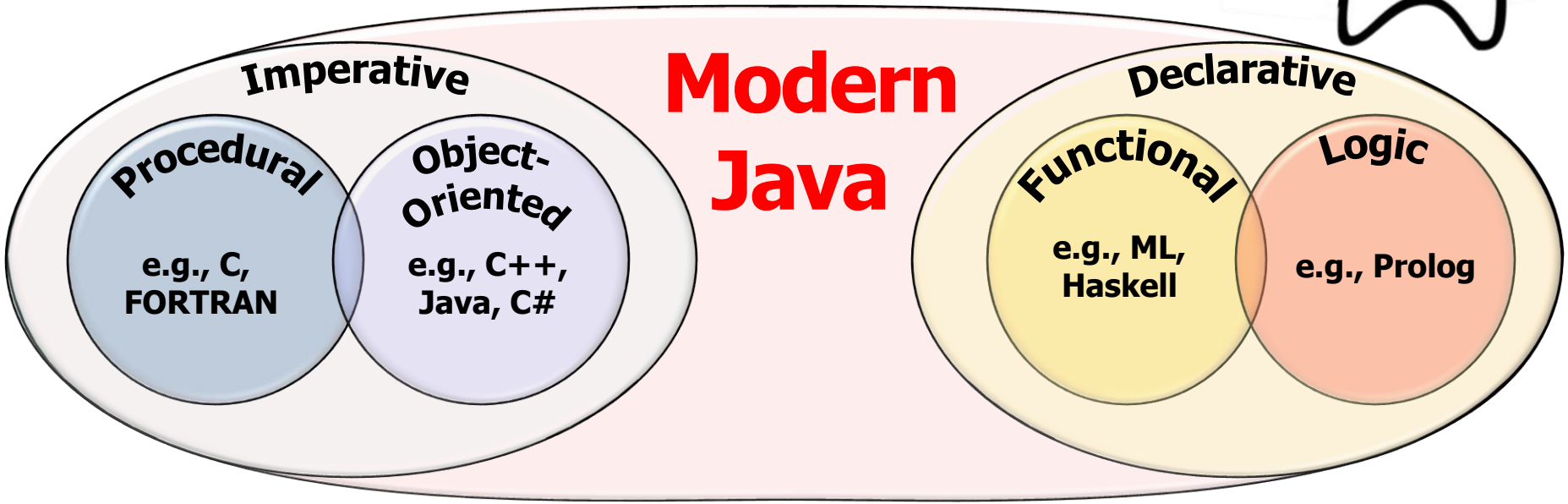
- Recognize the benefits of combining object-oriented & functional programming in Java
- Understand when, why, & how to use mutable state with Java

Combining Object-Oriented & Functional Programming in Java

Combining Object-Oriented & Functional Programming in Java

- Java's combination of functional & object-oriented paradigms is powerful!





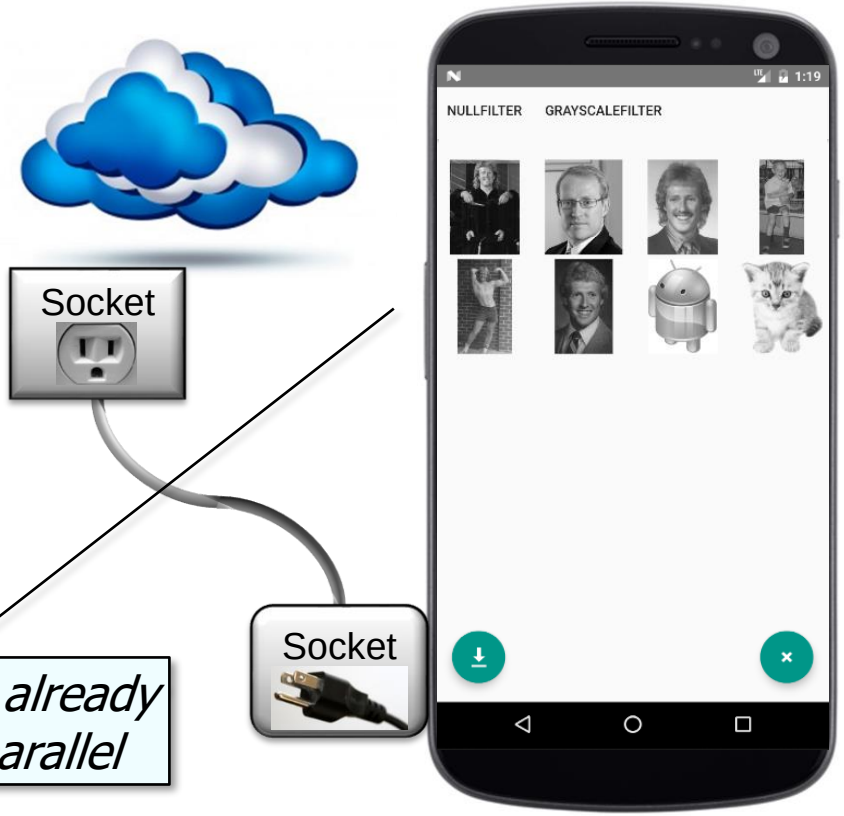
Combining Object-Oriented & Functional Programming in Java

- Java's functional features help close the gap between a program's "domain intent" & its computations

See www.toptal.com/software/declarative-programming

Combining Object-Oriented & Functional Programming in Java

- Java's functional features help close the gap between a program's "domain intent" & its computations, e.g.,
  - Domain intent defines "what"



Process a list of URLs to images that aren't already cached & transform/store the images in parallel

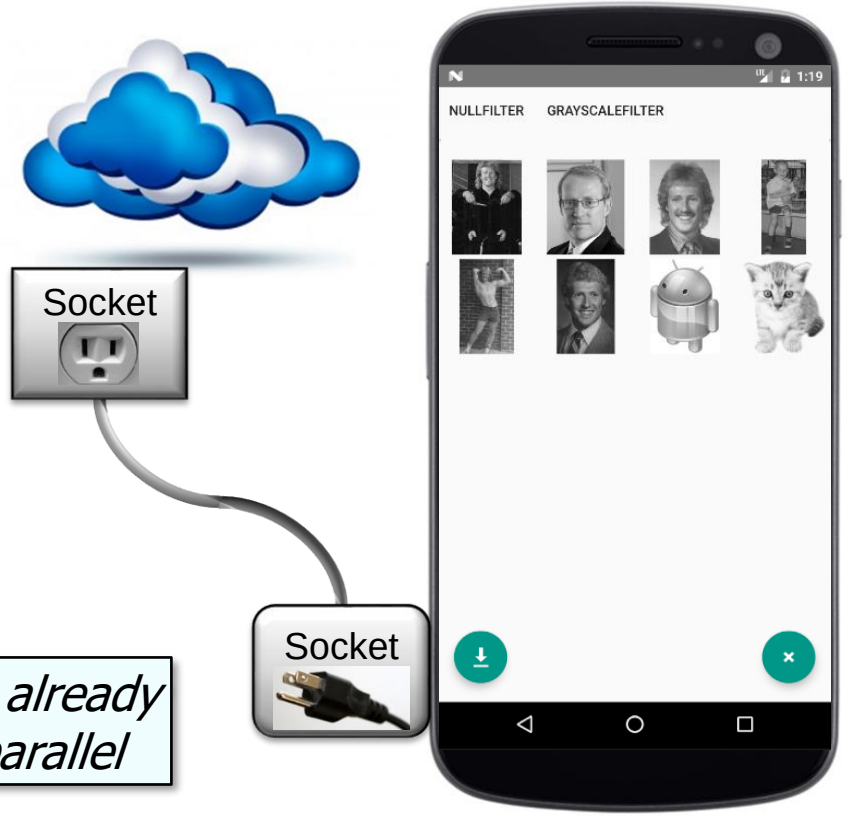
See github.com/douglascraigschmidt/LiveLessons/tree/master/ImageStreamGang

Combining Object-Oriented & Functional Programming in Java

- Java's functional features help close the gap between a program's "domain intent" & its computations, e.g.,
  - Domain intent defines "what"
  - Computations define "how"

```
List<Image> images = urls
   .parallelStream()
   .filter(not(this::urlCached))
   .map(this::downloadImage)
   .flatMap(this::applyFilters)
   .collect(toList());
```

Process a list of URLs to images that aren't already cached & transform/store the images in parallel

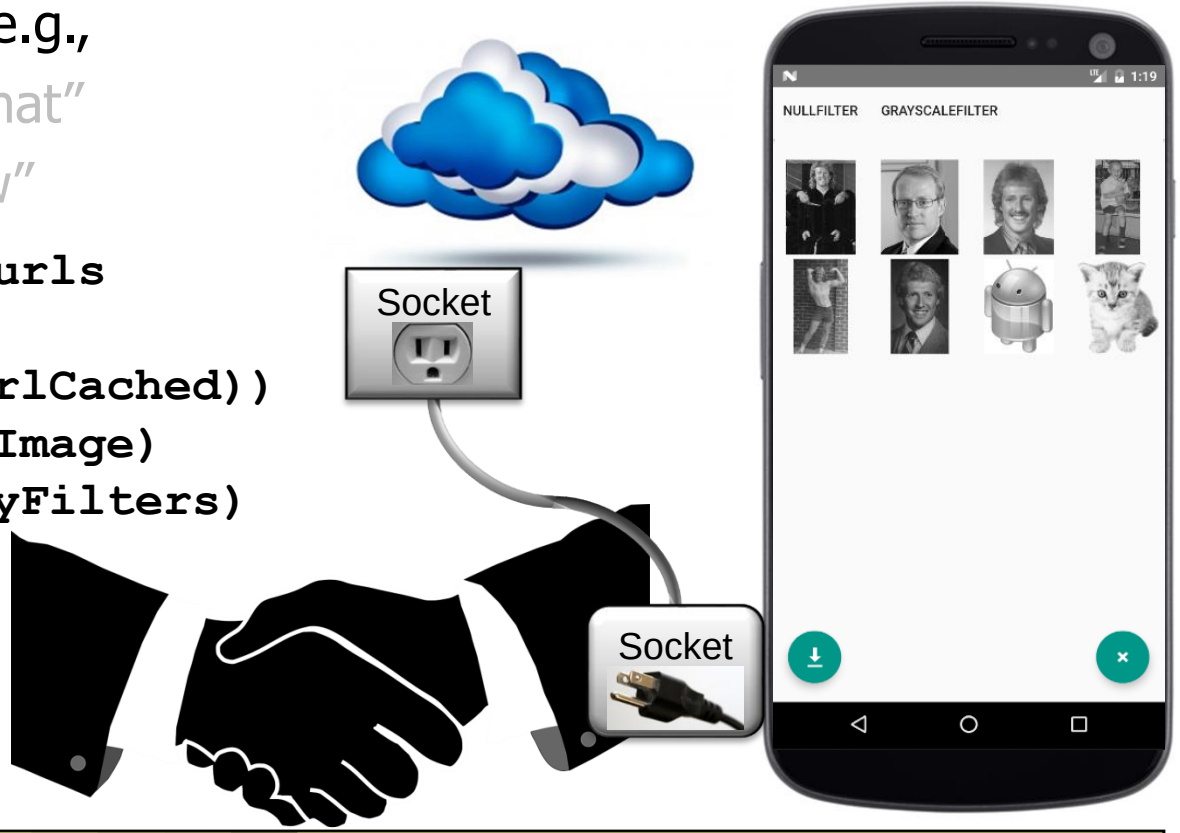


See github.com/douglascraigschmidt/LiveLessons/tree/master/ImageStreamGang

Combining Object-Oriented & Functional Programming in Java

- Java's functional features help close the gap between a program's "domain intent" & its computations, e.g.,
  - Domain intent defines "what"
  - Computations define "how"

```
List<Image> images = urls
   .parallelStream()
   .filter(not(this::urlCached))
   .map(this::downloadImage)
   .flatMap(this::applyFilters)
   .collect(toList());
```

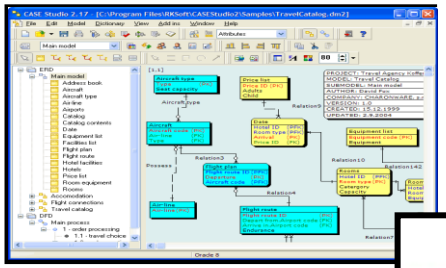


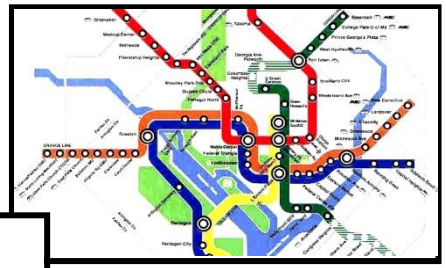
Java functional programming features connect domain intent & computations

Combining Object-Oriented & Functional Programming in Java

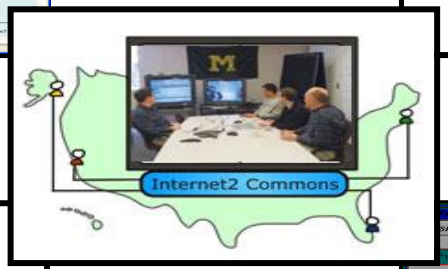
- Likewise, Java's object-oriented features help to structure a program's software architecture

Logical View



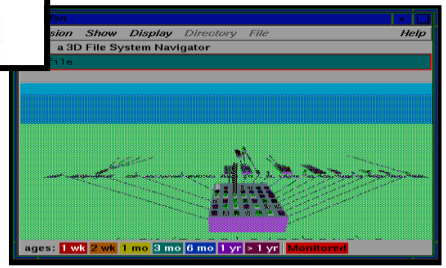


Process View



Physical View

Use Case View



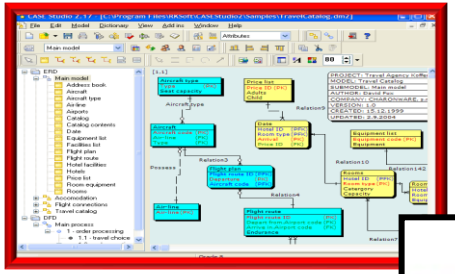
Development View

See en.wikipedia.org/wiki/Software_architecture

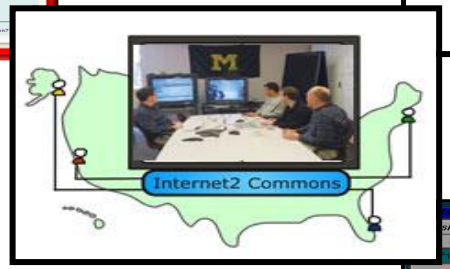
Combining Object-Oriented & Functional Programming in Java

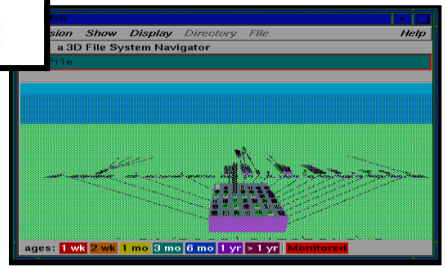
- Likewise, Java's object-oriented features help to structure a program's software architecture

Logical View



Process View





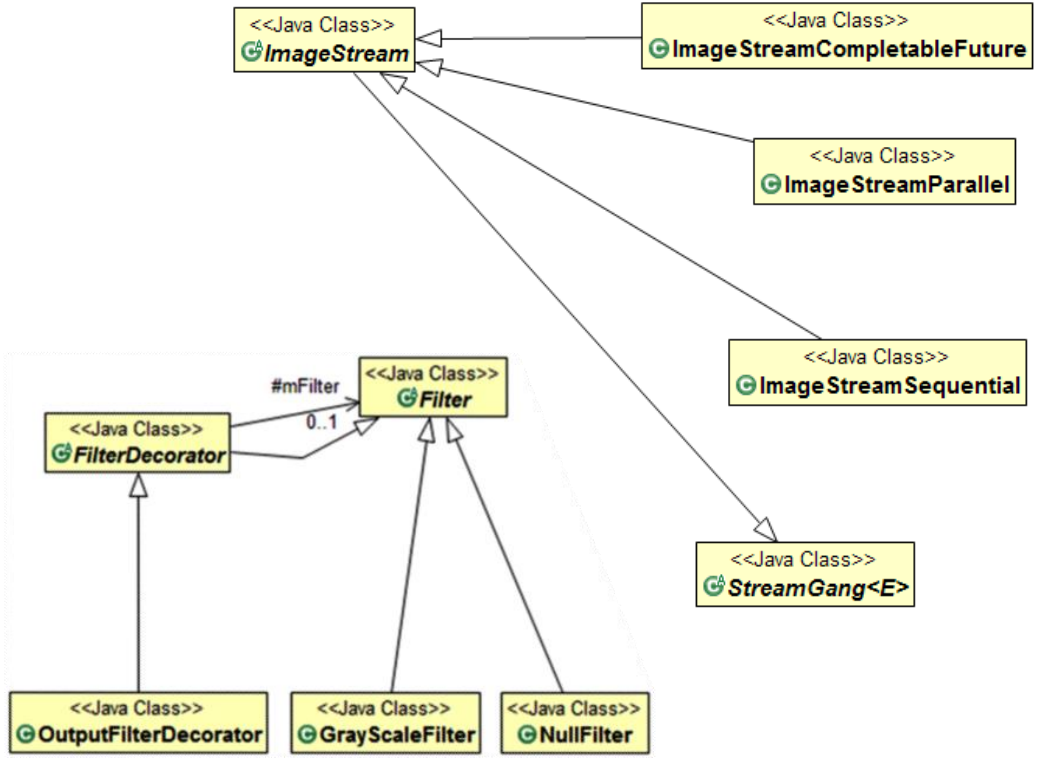
Development View

This view depicts sub-systems, packages, & classes that exhibit architecturally important structure & behavior

See sce.uhcl.edu/helm/rationalunifiedprocess/process/workflow/ana_desi/co_lview.htm

Combining Object-Oriented & Functional Programming in Java

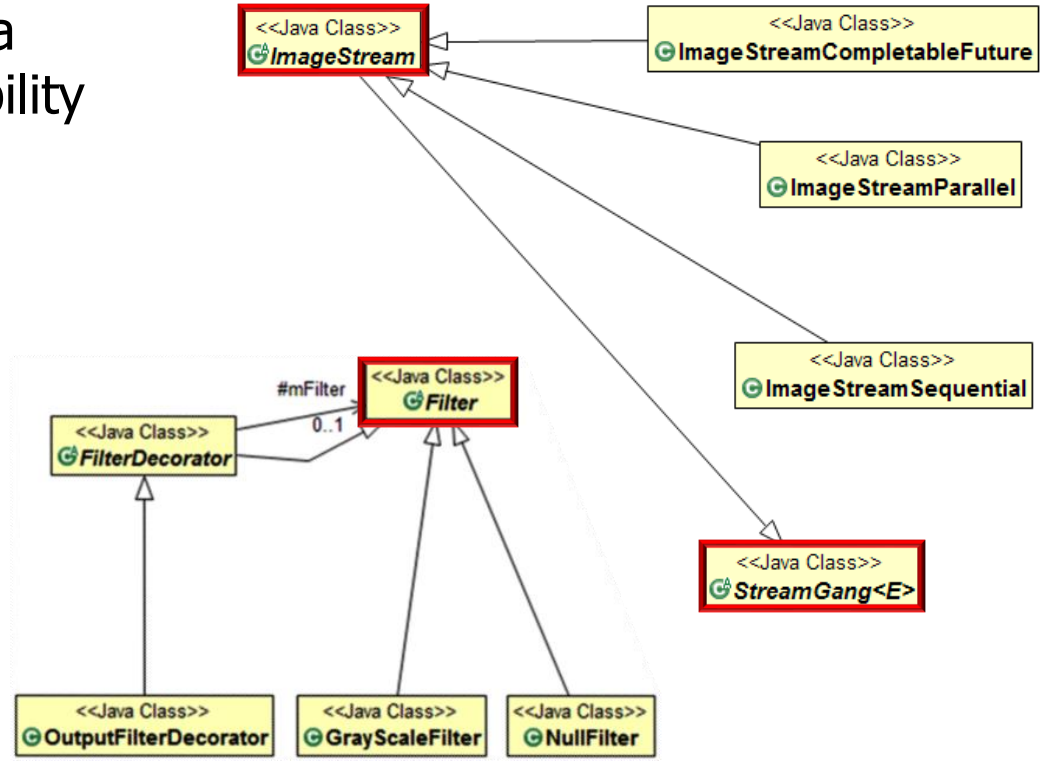
- e.g., consider the ImageStreamGang program



See github.com/douglascraigschmidt/LiveLessons/tree/master/ImageStreamGang

Combining Object-Oriented & Functional Programming in Java

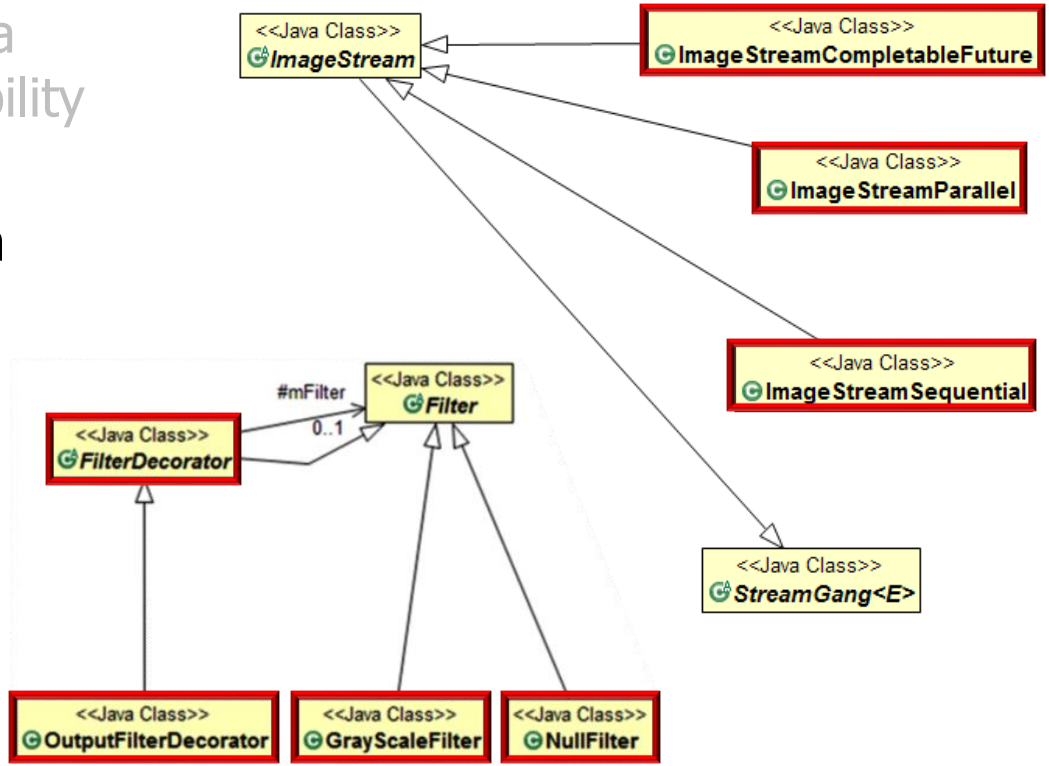
- e.g., consider the ImageStreamGang program
  - Common super classes provide a reusable foundation for extensibility



See www.dre.vanderbilt.edu/~schmidt/PDF/Commonality_Variability.pdf

Combining Object-Oriented & Functional Programming in Java

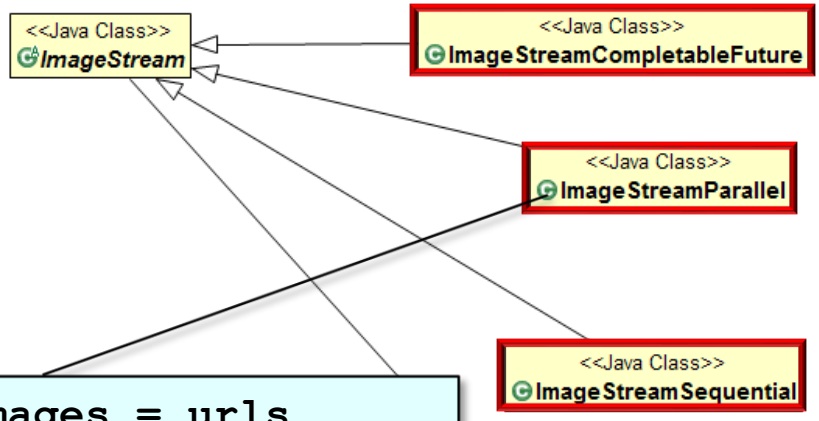
- e.g., consider the ImageStreamGang program
  - Common super classes provide a reusable foundation for extensibility
  - Subclasses extend the common classes to create various custom implementation strategies



See www.dre.vanderbilt.edu/~schmidt/PDF/Commonality_Variability.pdf

Combining Object-Oriented & Functional Programming in Java

- e.g., consider the ImageStreamGang program
  - Common super classes provide a reusable foundation for extensibility
  - Subclasses extend the common classes to create various custom implementation strategies
- Java's FP features are most effective when used to simplify computations within the context of an OO software architecture

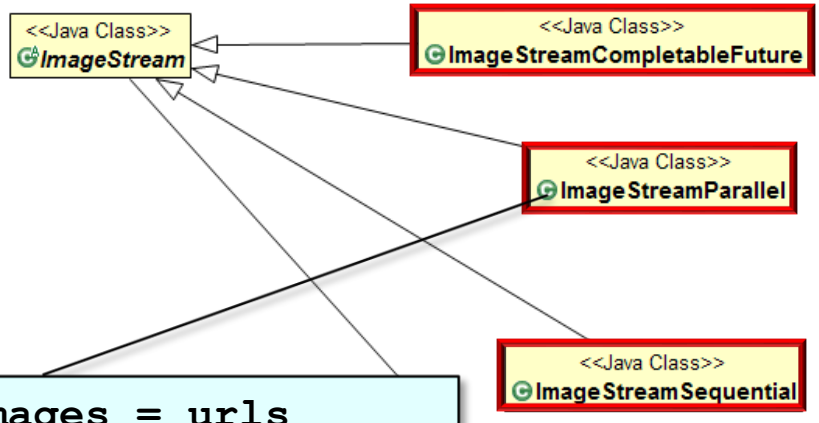



```
List<Image> images = urls
   .parallelStream()
   .filter(not(this::urlCached))
   .map(this::downloadImage)
   .flatMap(this::applyFilters)
   .collect(toList());
```

See www.drdobbs.com/jvm/lambda-expressions-in-java-8/240166764

Combining Object-Oriented & Functional Programming in Java

- e.g., consider the ImageStreamGang program
  - Common super classes provide a reusable foundation for extensibility
  - Subclasses extend the common classes to create various custom implementation strategies
- Java's FP features are most effective when used to simplify computations within the context of an OO software architecture
  - Especially concurrent & parallel computations



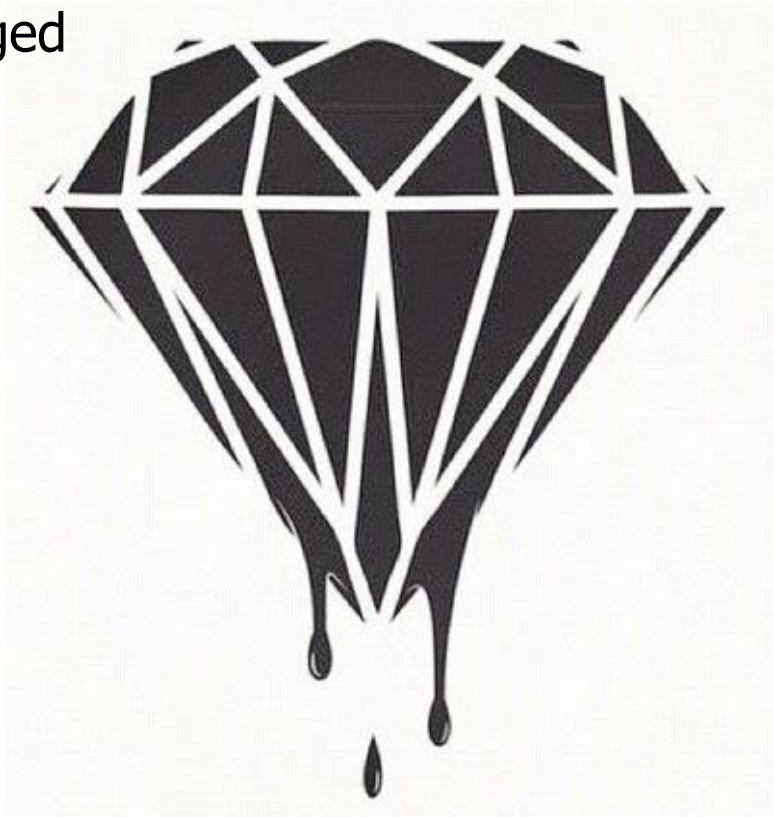

```
List<Image> images = urls
   .parallelStream()
   .filter(not(this::urlCached))
   .map(this::downloadImage)
   .flatMap(this::applyFilters)
   .collect(toList());
```

See docs.oracle.com/javase/tutorial/collections/streams/parallelism.html

When, Why, & How to Use Mutable State in Java

When, Why, & How to Use Mutable State in Java

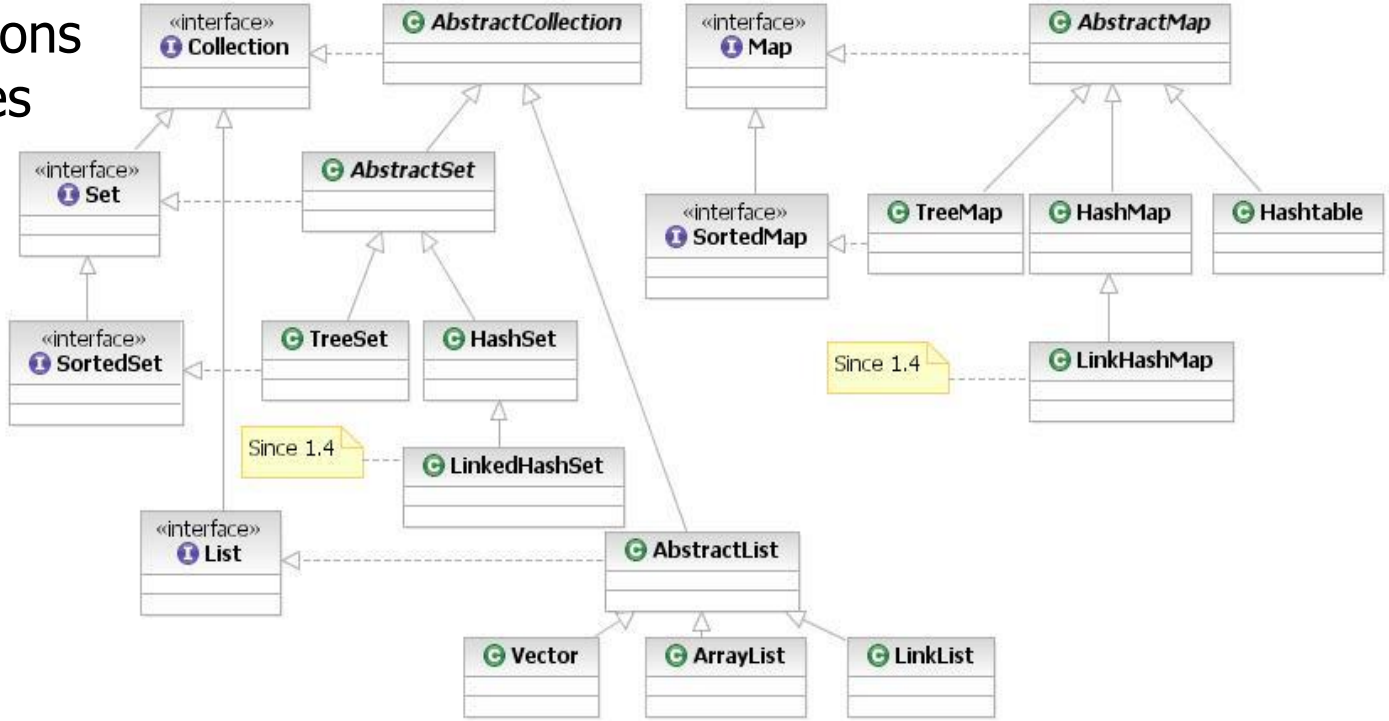
- Since Java is a hybrid language, there are situations in which mutable changes to shared state are allowed/encouraged



See www.infoq.com/articles/How-Functional-is-Java-8

When, Why, & How to Use Mutable State in Java

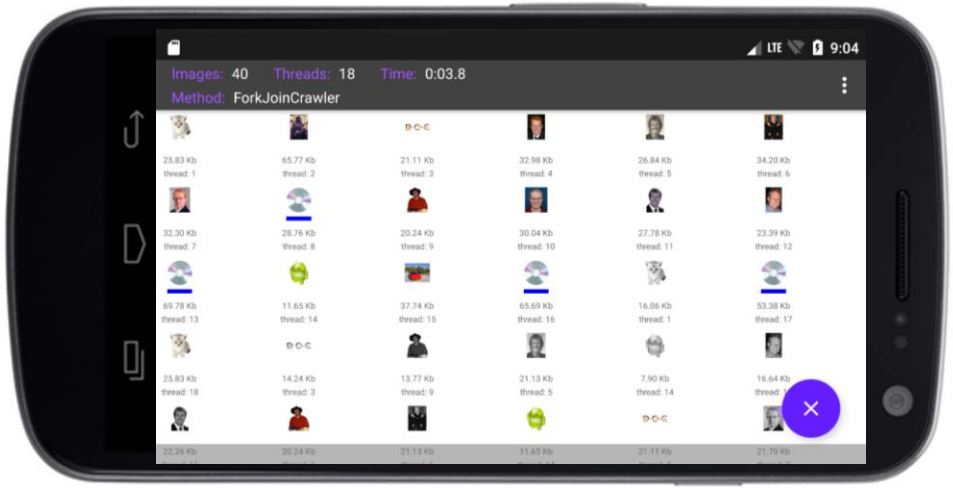
- Since Java is a hybrid language, there are situations in which mutable changes to shared state are allowed/encouraged
  - e.g., Java collections framework classes



See docs.oracle.com/javase/8/docs/technotes/guides/collections

When, Why, & How to Use Mutable State in Java

- However, you're usually better off by minimizing/avoiding the use of shared mutable state in *your* programs!!



See henrikeichenhardt.blogspot.com/2013/06/why-shared-mutable-state-is-root-of-all.html

When, Why, & How to Use Mutable State in Java

- If you *do* share mutable state in your programs then make sure you add the necessary synchronizers and/or use concurrent/synchronized collections

Category	Definition
Atomic operations	An action that effectively happens all at once or not at all
Mutual exclusion	Allows concurrent access & updates to shared mutable data without race conditions
Coordination	Ensures computations run properly, e.g., in the right order, at the right time, under the right conditions, etc.
Barrier synchronization	Ensures that any thread(s) must stop at a certain point & cannot proceed until all other thread(s) reach this barrier

See www.youtube.com/playlist?list=PLZ9NgFYEMxp6IM0Cddzr_qjqfiGC2pq1a

End of Recognize How Java Combines Object-Oriented & Functional Programming