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

Modern Java

- Imperative: Procedural (e.g., C, FORTRAN) and Object-Oriented (e.g., C++, Java, C#)
- Declarative: Functional (e.g., ML, Haskell) and Logic (e.g., Prolog)
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
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”

```java
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
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”

```java
List<Image> images = urls
    .parallelStream()
    .filter(not(this::urlCached))
    .map(this::downloadImage)
    .flatMap(this::applyFilters)
    .collect(toList());
```
Likewise, Java’s object-oriented features help to structure a program’s software architecture.
Combining Object-Oriented & Functional Programming in Java

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

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

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

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

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

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](http://www.infoq.com/articles/How-Functional-is-Java-8)
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
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](http://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.

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

See [www.youtube.com/playlist?list=PLZ9NgFYEMxp6IM0Cddzr_qjqfiGC2pq1a](http://www.youtube.com/playlist?list=PLZ9NgFYEMxp6IM0Cddzr_qjqfiGC2pq1a)
End of Recognize How Java Combines Object-Oriented & Functional Programming