Evaluating Java Programming Paradigms

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 Part of the Lesson

• Understand the key benefits & principles underlying the reactive programming paradigm
• Know the Java reactive streams API & popular implementations of this API
• Learn how Java reactive streams maps to key reactive programming principles
• Recognize how reactive programming compares with other Java paradigms
  • e.g., OO programming, & sync/async functional programming
Learning Objectives in this Part of the Lesson

• Understand the key benefits & principles underlying the reactive programming paradigm
• Know the Java reactive streams API & popular implementations of this API
• Learn how Java reactive streams maps to key reactive programming principles
• Recognize how reactive programming compares with other Java paradigms
• Be aware of the pros & cons of reactive streams platforms
Comparing Reactive Programming with Other Paradigms
Comparing Reactive Programming with Other Paradigms

- Reactive programming is one of several Java programming paradigms
Comparing Reactive Programming with Other Paradigms

- Reactive programming is one of several Java programming paradigms

<table>
<thead>
<tr>
<th>Single value</th>
<th>Multiple values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objects</td>
<td>Streams</td>
</tr>
</tbody>
</table>

byte[] downloadContent(URL url) {
    byte[] buf = new byte[BUFSIZ];
    ByteArrayOutputStream os =
        new ByteArrayOutputStream();
    InputStream is = url.openStream();
    for (int bytes;
        (bytes = is.read(buf)) > 0;)
        os.write(buf, 0, bytes); ...
Comparing Reactive Programming with Other Paradigms

- Reactive programming is one of several Java programming paradigms

```java
List<Image> imgs = getInput()
    .parallelStream()
    .filter(not(this::urlCached))
    .map(this::downloadImage)
    .flatMap(this::applyFilters)
    .collect(toList());
```
Comparing Reactive Programming with Other Paradigms

- Reactive programming is one of several Java programming paradigms

```java
CompletableFuture.supplyAsync(reduce).
.thenApply
   (BigFraction::toMixedString).
.thenAccept
   (System.out::println);
```
Comparing Reactive Programming with Other Paradigms

- Reactive programming is one of several Java programming paradigms

List<Image> imgs = getInput().stream()
    .map(checkUrlCachedAsync)
    .map(downloadImageAsync)
    .flatMap(applyFiltersAsync)
    .collect(toFuture())
    .thenApply(logResults)
    .join(); ...
Reactive programming is one of several Java programming paradigms.

Comparing Reactive Programming with Other Paradigms

- Single value
- Multiple values
- Synchronous
- Asynchronous

```java
List<Image> imgs = Observable.fromIterable(Options.instance().getUrlList())
    .parallel(parallelism)
    .runOn(scheduler)
    .map(downloadAndStoreImage)
    .sequential()
    .collect(toList())
    .blockingGet();
```
Pros & Cons of Java Reactive Streams Platforms
Pros & Cons of Java Reactive Streams Platforms

- Java reactive streams implementations apply reactive programming principles to achieve several benefits.
Pros & Cons of Java Reactive Streams Platforms

- Java reactive streams implementations apply reactive programming principles to achieve several benefits.
- Support concurrency with a minimal number of threads via a range of thread pools.
Pros & Cons of Java Reactive Streams Platforms

- Java reactive streams implementations apply reactive programming principles to achieve several benefits
  - Support concurrency with a minimal number of threads via a range of thread pools
  - Scale up performance with relatively few resources

See dzone.com/articles/spring-boot-20-webflux-reactive-performance-test
Pros & Cons of Java Reactive Streams Platforms

- Java reactive streams implementations apply reactive programming principles to achieve several benefits
  - Support concurrency with a minimal number of threads via a range of thread pools
  - Explicit synchronization and/or threading is rarely needed when applying these frameworks

Alleviates many accidental & inherent complexities of concurrency/parallelism
Java reactive streams implementations apply reactive programming principles to achieve several benefits:

- Support concurrency with a minimal number of threads via a range of thread pools.
- Explicit synchronization and/or threading is rarely needed when applying these frameworks.

These benefits are not unique to reactive streams, however!!
Pros & Cons of Java Reactive Streams Platforms

• However, reactive programming isn’t appropriate in all situations

See www.youtube.com/watch?v=z0a0N9OgaAA
Pros & Cons of Java Reactive Streams Platforms

- However, reactive programming isn’t appropriate in all situations

It’s essential to master the learning curve of reactive programming!
End of Evaluating Java Programming Paradigms