Overview of Concurrency Patterns in Android & Java Frameworks (Part 3)

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

Institute for Software Integrated Systems
Vanderbilt University
Nashville, Tennessee, USA
Learning Objectives in this Part of the Lesson

- Understand how *patterns* improve the structure & functionality of Java & Android concurrency frameworks used by apps & services
- Recognize key POSA patterns used in these concurrency frameworks
- Recognize key GoF patterns used in these concurrency frameworks
Gang-of-Four Patterns in Android Concurrency Frameworks
GoF Patterns in Android & Java Concurrency Frameworks

- Android & Java’s concurrency frameworks are also designed, implemented, & integrated in accordance with many GoF patterns

See en.wikipedia.org/wiki/Design_Patterns
GoF Patterns in Android & Java Concurrency Frameworks

- Android & Java’s concurrency frameworks are also designed, implemented, & integrated in accordance with many GoF patterns

<table>
<thead>
<tr>
<th>Android/Java Mechanism</th>
<th>Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>HandlerThread</td>
<td>Template Method</td>
</tr>
<tr>
<td>AsyncTask framework</td>
<td>Strategy</td>
</tr>
<tr>
<td>ThreadFactory</td>
<td>Factory Method</td>
</tr>
</tbody>
</table>
GoF Patterns in Android & Java Concurrency Frameworks

- **Template Method** – Provide a skeleton of an algorithm in a method, deferring some steps to subclasses

```
AbstractClass

templateMethod()
primitivOperation1()
primitivOperation2()
primitivOperation3()
...
```

```
ConcreteClass

primitivOperation1()
primitivOperation2()
...
```

See [en.wikipedia.org/wiki/Template_method_pattern](en.wikipedia.org/wiki/Template_method_pattern)
GoF Patterns in Android & Java Concurrency Frameworks

- **Template Method** – Provide a skeleton of an algorithm in a method, deferring some steps to subclasses

```java
class HandlerThread {
    void run() {
        onLooperPrepared();
    }
}

class DownloadTask {
    void run() {
        onLooperPrepared();
    }
}
```

See upcoming lesson on the “Android Looper”
GoF Patterns in Android & Java Concurrency Frameworks

- **Strategy** – Define a family of algorithms, encapsulate each one, & make them interchangeable to let clients & algorithms vary independently

GoF Patterns in Android & Java Concurrency Frameworks

- **Strategy** – Define a family of algorithms, encapsulate each one, & make them interchangeable to let clients & algorithms vary independently

```
AsyncTask
executeOnExecutor() ...
```

```
CustomExecutor
execute()
```

```
SERIAL_EXECUTOR
execute()
```

```
THREAD_POOL_EXECUTOR
execute()
```

See upcoming lesson on the “AsyncTask Framework”
GoF Patterns in Android & Java Concurrency Frameworks

- **Factory Method** – Provide an interface for creating an object, but leave the choice of object’s concrete type to a subclass

```java
Product

ConcreteProduct

Creator

newProduct()

ConcreteCreator

newProduct()

return new ConcreteProduct;
```

See [en.wikipedia.org/wiki/Factory_method_pattern](en.wikipedia.org/wiki/Factory_method_pattern)
GoF Patterns in Android & Java Concurrency Frameworks

• **Factory Method** – Provide an interface for creating an object, but leave the choice of object’s concrete type to a subclass

```java
// Factory Method

return new Thread(r, "AsyncTask #" + mCount.getAndIncrement());
```
Java concurrency mechanisms are used in Android’s concurrent frameworks

**package** java.util.concurrent

Utility classes commonly useful in concurrent programming. This package includes a few small standardized extensible frameworks, as well as some classes that provide useful functionality and are otherwise tedious or difficult to implement. Here are brief descriptions of the main components. See also the java.util.concurrent.locks and java.util.concurrent.atomic packages.

**package** java.util.concurrent.locks

Interfaces and classes providing a framework for locking and waiting for conditions that is distinct from built-in synchronization and monitors. The framework permits much greater flexibility in the use of locks and conditions, at the expense of more awkward syntax.

The Lock interface supports locking disciplines that differ in semantics (reentrant, fair, etc.), and that can be used in non-block-structured contexts including hand-over-hand and lock reordering algorithms. The main implementation is ReentrantLock.
GoF Patterns in Android & Java Concurrency Frameworks

- Java concurrency mechanisms are used in Android’s concurrent frameworks
- We’ll also discuss other GoF & POSA patterns throughout these lessons
End of Overview of Concurrency Patterns in Android & Java Frameworks (Part 3)