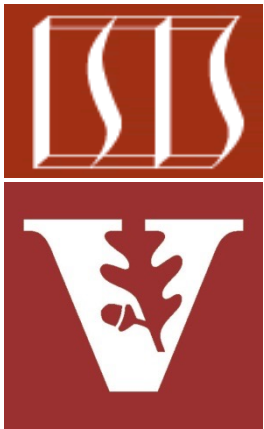


Analyzing the TaskGang Class



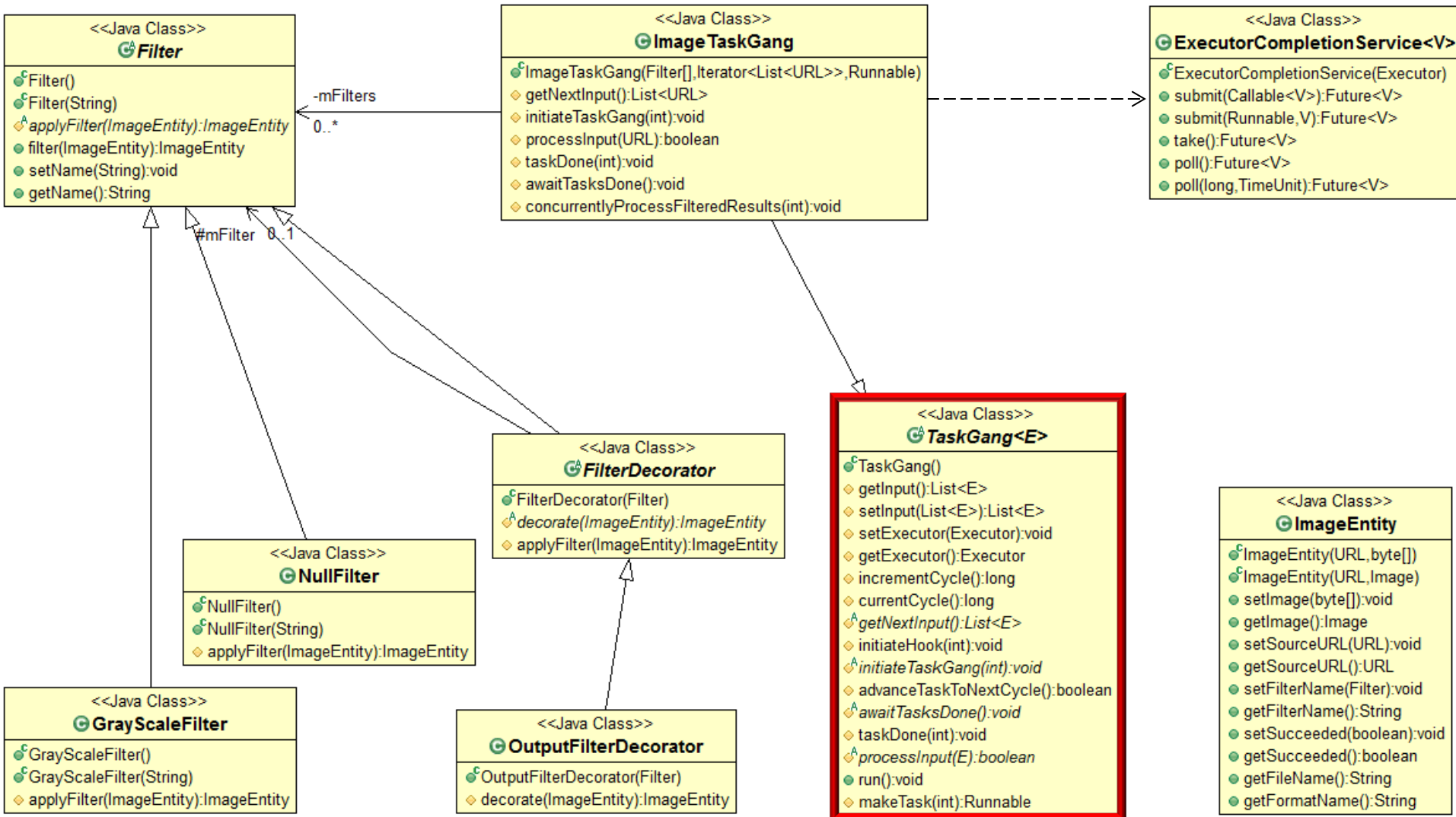
Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

**Institute for Software
Integrated Systems
Vanderbilt University
Nashville, Tennessee, USA**



Learning Objectives in this Part of the Lesson

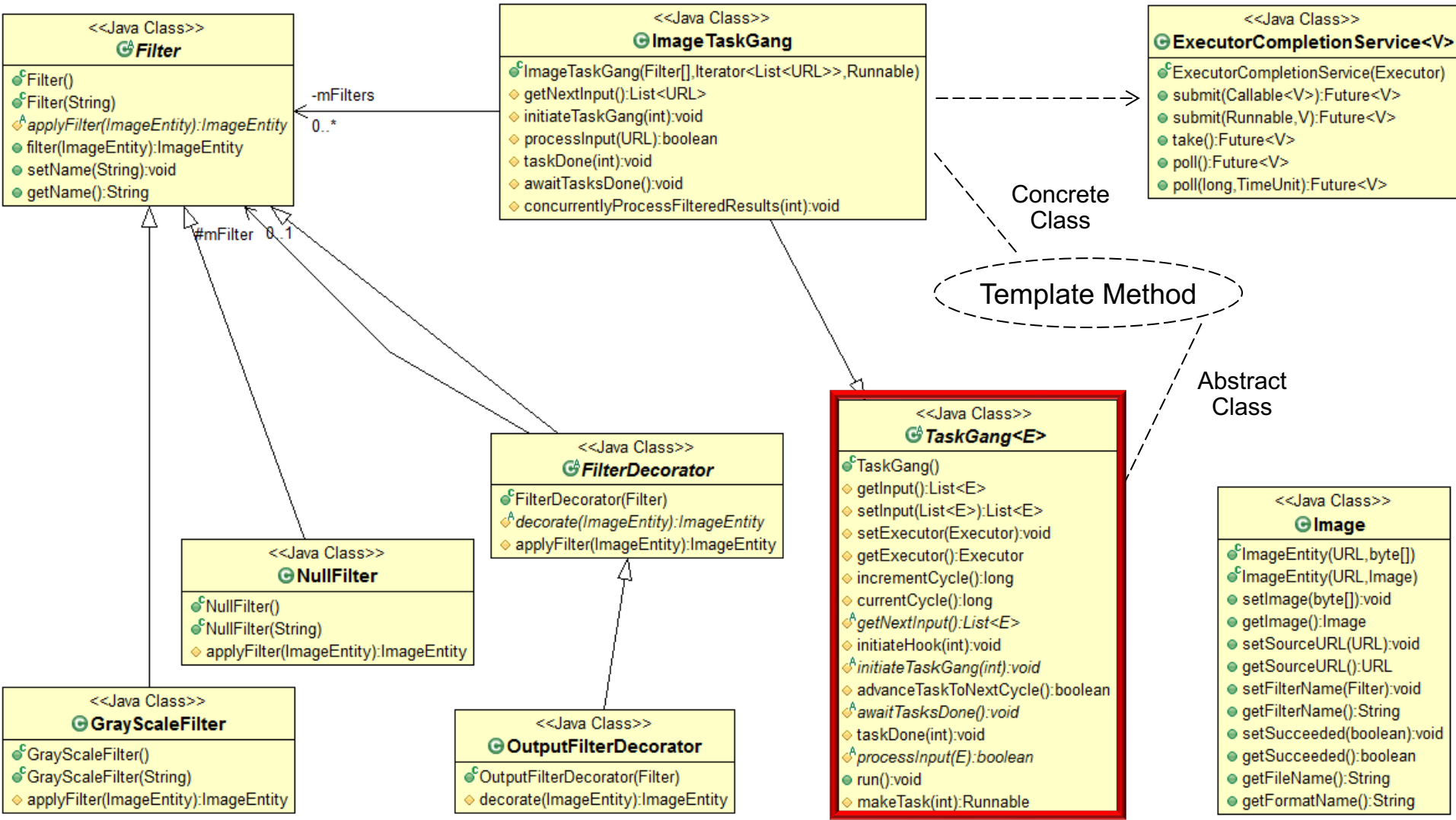
- Understand the pattern-oriented software implementation of the TaskGang class, which is an abstract super class



See [ImageTaskGang/src/main/java/livelessons/tasks/TaskGang.java](https://github.com/leecostello/image-task-gang/blob/master/src/main/java/livelessons/tasks/TaskGang.java)

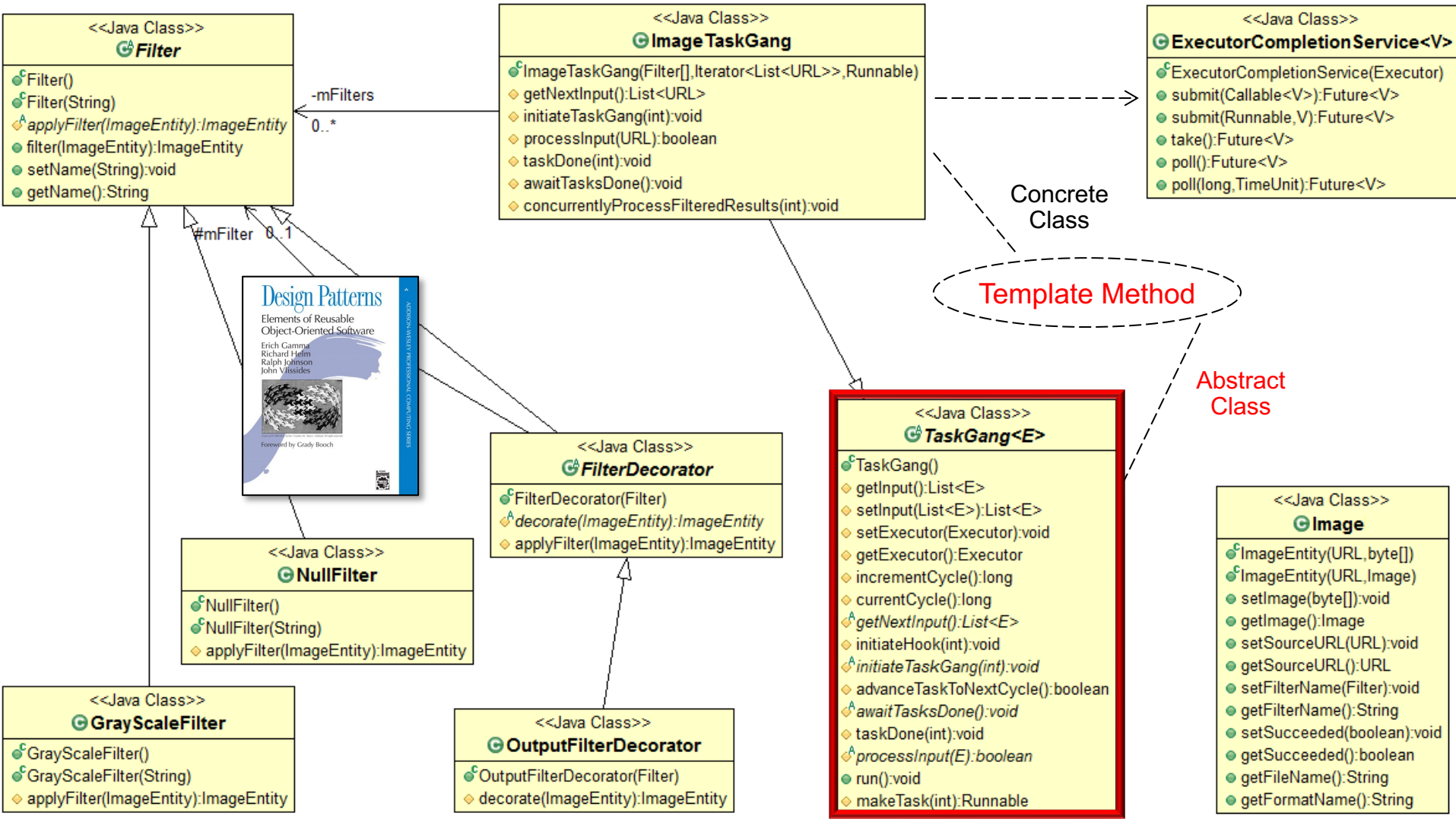
Analysis of the TaskGang Class Source Code

Analysis of the TaskGang Class



Defines a framework for spawning & running a gang of tasks that concurrently process input from a generic list

Analysis of the TaskGang Class



This class plays the role of the "Abstract Class" in the *Template Method* pattern

Analysis of the TaskGang Class

```
7  /**
8   * Defines a framework for spawning and running a "gang" of tasks that
9   * concurrently process input from a generic {@link List} of elements
10  * {@code E} for one or more cycles.
11  */
12  public abstract class TaskGang<E>
13      implements Runnable {
14      /**
15       * The input {@link List} that's processed, which can be
16       * initialized via the {@code makeInputList()} factory method.
17       */
18      private volatile List<E> mInput = null;
19
20      /**
21       * Executes submitted Runnable tasks in a {@link Thread} pool.
22       */
23      private Executor mExecutor = null;
24
25      /**
26       * Keeps track of which cycle is currently active.
27       */
28      private final AtomicLong mCurrentCycle = new AtomicLong(initialValue: 0);
```

See [ImageTaskGang/src/main/java/livelessons/tasks/TaskGang.java](#)

End of Analysis of the ImageTaskGang Class