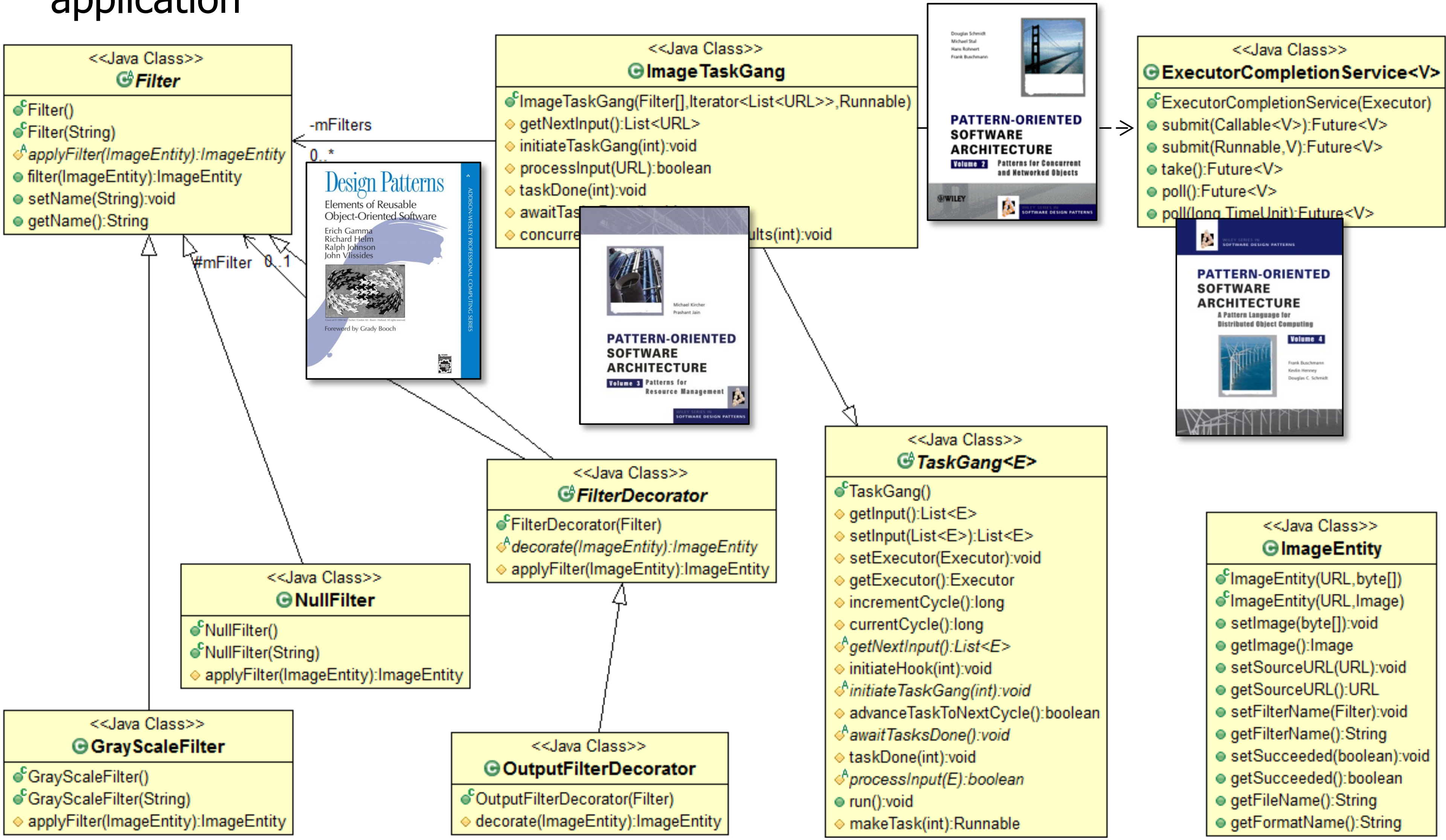
Pattern-Oriented Software Architecture of the ImageTask Gang Application

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 architecture of ImageTaskGang application



<<Java Class>>
Filter

- Filter()
- Filter(String)
- *applyFilter(ImageEntity):ImageEntity*
- filter(ImageEntity):ImageEntity
- setName(String):void
- getName():String

<<Java Class>>
ImageTaskGang

- ImageTaskGang(Filter[],Iterator<List<URL>>,Runnable)
- getNextInput():List<URL>
- initiateTaskGang(int):void
- processInput(URL):boolean
- taskDone(int):void
- awaitTas
- concurre lts(int):void

<<Java Class>>
ExecutorCompletionService<V>

- ExecutorCompletionService(Executor)
- submit(Callable<V>):Future<V>
- submit(Runnable,V):Future<V>
- take():Future<V>
- poll():Future<V>
- poll(long,TimeUnit):Future<V>

<<Java Class>>
FilterDecorator

- FilterDecorator(Filter)
- *decorate(ImageEntity):ImageEntity*
- applyFilter(ImageEntity):ImageEntity

<<Java Class>>
TaskGang<E>

- TaskGang()
- getInput():List<E>
- setInput(List<E>):List<E>
- setExecutor(Executor):void
- getExecutor():Executor
- incrementCycle():long
- currentCycle():long
- *getNextInput():List<E>*
- initiateHook(int):void
- *initiateTaskGang(int):void*
- advanceTaskToNextCycle():boolean
- *awaitTasksDone():void*
- taskDone(int):void
- *processInput(E):boolean*
- run():void
- makeTask(int):Runnable

<<Java Class>>
NullFilter

- NullFilter()
- NullFilter(String)
- applyFilter(ImageEntity):ImageEntity

<<Java Class>>
ImageEntity

- ImageEntity(URL,byte[])
- ImageEntity(URL,Image)
- setImage(byte[]):void
- getImage():Image
- setSourceURL(URL):void
- getSourceURL():URL
- setFilterName(Filter):void
- getFilterName():String
- setSucceeded(boolean):void
- getSucceeded():boolean
- getFileName():String
- getFormatName():String

<<Java Class>>
GrayScaleFilter

- GrayScaleFilter()
- GrayScaleFilter(String)
- applyFilter(ImageEntity):ImageEntity

<<Java Class>>
OutputFilterDecorator

- OutputFilterDecorator(Filter)
- decorate(ImageEntity):ImageEntity

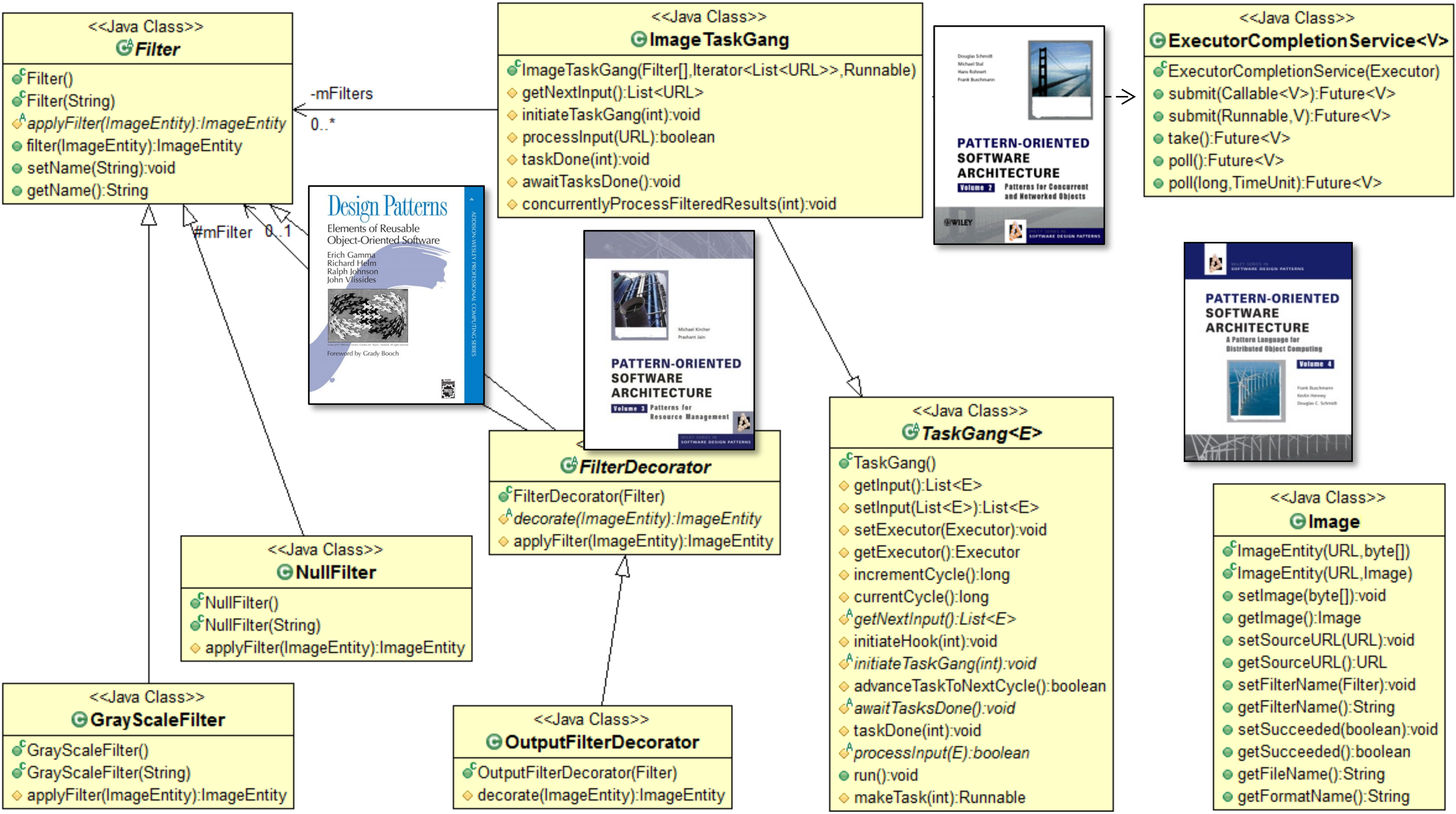
-mFilters
0..*
#mFilter
0..1
Design Patterns
Elements of Reusable Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides
Foreword by Grady Booch
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES
PATTERN-ORIENTED SOFTWARE ARCHITECTURE
Volume 2
Patterns for Concurrent and Networked Objects
WILEY
PATTERN-ORIENTED SOFTWARE ARCHITECTURE
Volume 3
Patterns for Resource Management
PATTERN-ORIENTED SOFTWARE ARCHITECTURE
A Pattern Language for Distributed Object Computing
Volume 4

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

Pattern-Oriented Software Architecture of the Image TaskGang Application

Overview of Pattern-Oriented Framework Solution

- "Gang-of-Four" & POSA patterns enhance the framework-based solution



<<Java Class>>
Filter
- Filter()
- Filter(String)
- *applyFilter(ImageEntity):ImageEntity*
- filter(ImageEntity):ImageEntity
- setName(String):void
- getName():String

<<Java Class>>
ImageTaskGang
- ImageTaskGang(Filter[],Iterator<List<URL>>,Runnable)
- getNextInput():List<URL>
- initiateTaskGang(int):void
- processInput(URL):boolean
- taskDone(int):void
- awaitTasksDone():void
- concurrentlyProcessFilteredResults(int):void

<<Java Class>>
ExecutorCompletionService<V>
- ExecutorCompletionService(Executor)
- submit(Callable<V>):Future<V>
- submit(Runnable,V):Future<V>
- take():Future<V>
- poll():Future<V>
- poll(long,TimeUnit):Future<V>

<<Java Class>>
FilterDecorator
- FilterDecorator(Filter)
- *decorate(ImageEntity):ImageEntity*
- applyFilter(ImageEntity):ImageEntity

<<Java Class>>
TaskGang<E>
- TaskGang()
- getInput():List<E>
- setInput(List<E>):List<E>
- setExecutor(Executor):void
- getExecutor():Executor
- incrementCycle():long
- currentCycle():long
- *getNextInput():List<E>*
- initiateHook(int):void
- *initiateTaskGang(int):void*
- advanceTaskToNextCycle():boolean
- *awaitTasksDone():void*
- taskDone(int):void
- *processInput(E):boolean*
- run():void
- makeTask(int):Runnable

<<Java Class>>
NullFilter
- NullFilter()
- NullFilter(String)
- applyFilter(ImageEntity):ImageEntity

<<Java Class>>
GrayScaleFilter
- GrayScaleFilter()
- GrayScaleFilter(String)
- applyFilter(ImageEntity):ImageEntity

<<Java Class>>
OutputFilterDecorator
- OutputFilterDecorator(Filter)
- decorate(ImageEntity):ImageEntity

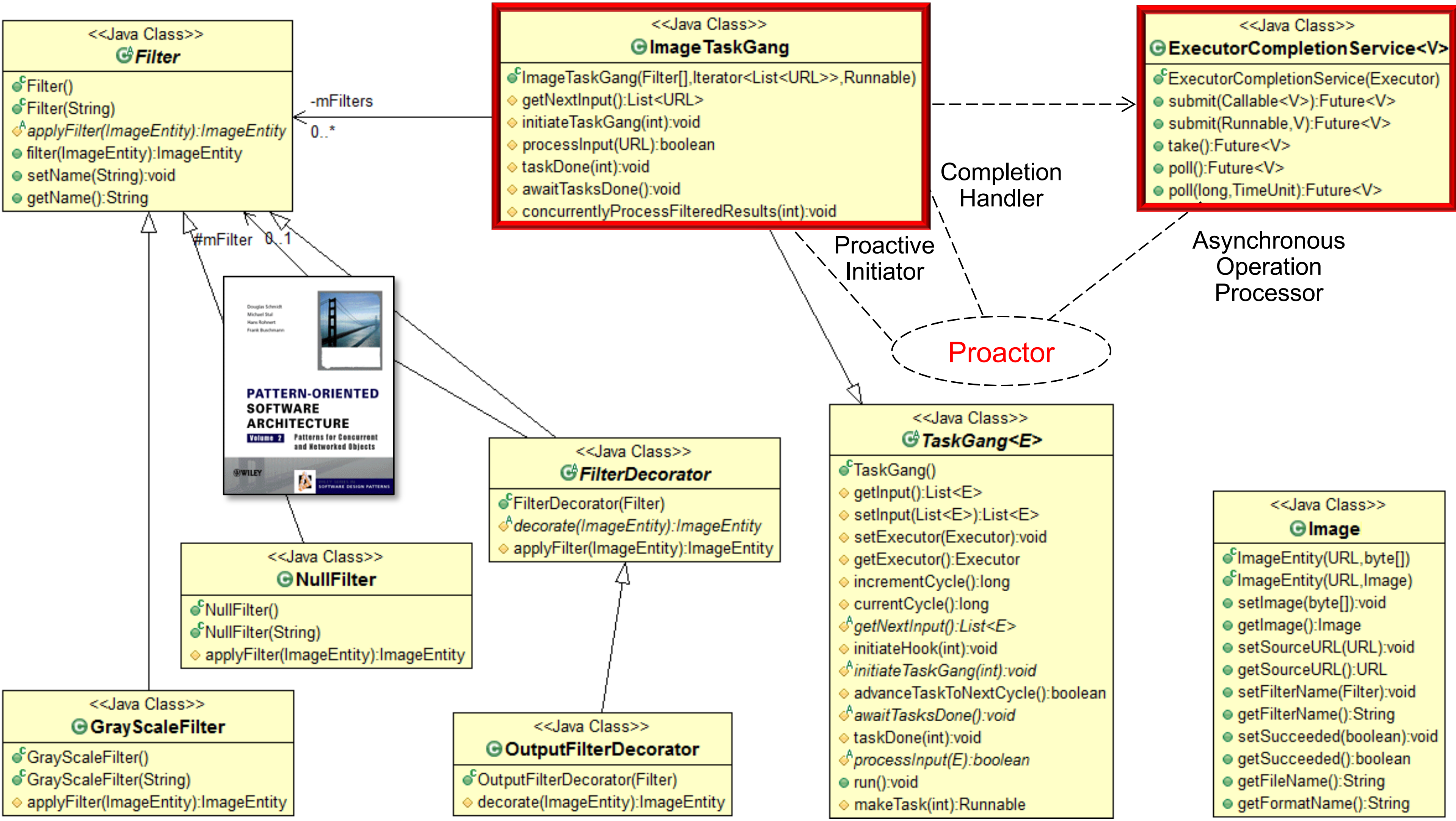
<<Java Class>>
Image
- ImageEntity(URL,byte[])
- ImageEntity(URL,Image)
- setImage(byte[]):void
- getImage():Image
- setSourceURL(URL):void
- getSourceURL():URL
- setFilterName(Filter):void
- getFilterName():String
- setSucceeded(boolean):void
- getSucceeded():boolean
- getFileName():String
- getFormatName():String

-mFilters
0..*
#mFilter
0..1
Design Patterns
Elements of Reusable Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides
Foreword by Grady Booch
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES
PATTERN-ORIENTED SOFTWARE ARCHITECTURE
Volume 2
Patterns for Concurrent and Networked Objects
Douglas Schmidt
Michael Stal
Hans Rohnert
Frank Buschmann
WILEY
PATTERN-ORIENTED SOFTWARE ARCHITECTURE
Volume 3
Patterns for Resource Management
Michael Kircher
Prashant Jain
PATTERN-ORIENTED SOFTWARE ARCHITECTURE
A Pattern Language for Distributed Object Computing
Volume 4
Frank Buschmann
Kevlin Henney
Douglas C. Schmidt
WILEY SERIES IN SOFTWARE DESIGN PATTERNS

Patterns enhance application & framework reusability, flexibility, portability, & performance

Overview of Pattern-Oriented Framework Solution

- "Gang-of-Four" & POSA patterns enhance the framework-based solution



<<Java Class>>
Filter

- Filter()
- Filter(String)
- *applyFilter(ImageEntity):ImageEntity*
- filter(ImageEntity):ImageEntity
- setName(String):void
- getName():String

<<Java Class>>
ImageTaskGang

- ImageTaskGang(Filter[],Iterator<List<URL>>,Runnable)
- getNextInput():List<URL>
- initiateTaskGang(int):void
- processInput(URL):boolean
- taskDone(int):void
- awaitTasksDone():void
- concurrentlyProcessFilteredResults(int):void

<<Java Class>>
ExecutorCompletionService<V>

- ExecutorCompletionService(Executor)
- submit(Callable<V>):Future<V>
- submit(Runnable,V):Future<V>
- take():Future<V>
- poll():Future<V>
- poll(long,TimeUnit):Future<V>

<<Java Class>>
FilterDecorator

- FilterDecorator(Filter)
- *decorate(ImageEntity):ImageEntity*
- applyFilter(ImageEntity):ImageEntity

<<Java Class>>
NullFilter

- NullFilter()
- NullFilter(String)
- applyFilter(ImageEntity):ImageEntity

<<Java Class>>
GrayScaleFilter

- GrayScaleFilter()
- GrayScaleFilter(String)
- applyFilter(ImageEntity):ImageEntity

<<Java Class>>
OutputFilterDecorator

- OutputFilterDecorator(Filter)
- decorate(ImageEntity):ImageEntity

<<Java Class>>
TaskGang<E>

- TaskGang()
- getInput():List<E>
- setInput(List<E>):List<E>
- setExecutor(Executor):void
- getExecutor():Executor
- incrementCycle():long
- currentCycle():long
- *getNextInput():List<E>*
- initiateHook(int):void
- *initiateTaskGang(int):void*
- advanceTaskToNextCycle():boolean
- *awaitTasksDone():void*
- taskDone(int):void
- *processInput(E):boolean*
- run():void
- makeTask(int):Runnable

<<Java Class>>
Image

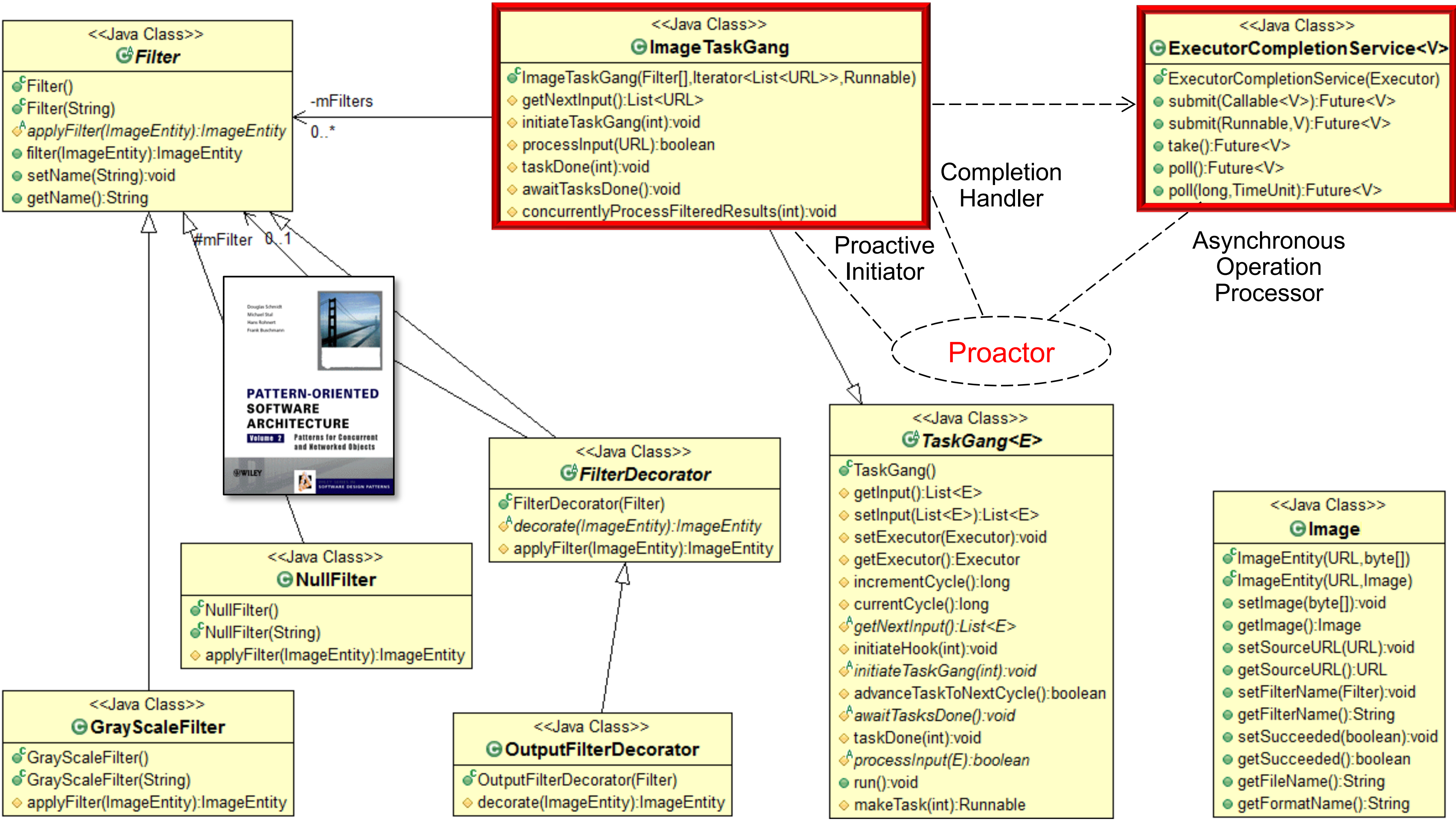
- ImageEntity(URL,byte[])
- ImageEntity(URL,Image)
- setImage(byte[]):void
- getImage():Image
- setSourceURL(URL):void
- getSourceURL():URL
- setFilterName(Filter):void
- getFilterName():String
- setSucceeded(boolean):void
- getSucceeded():boolean
- getFileName():String
- getFormatName():String

-mFilters
0..*
#mFilter
0..1
Completion Handler
Proactive Initiator
Asynchronous Operation Processor
Proactor
Douglas Schmidt
Michael Stal
Hans Rohnert
Frank Buschmann
PATTERN-ORIENTED SOFTWARE ARCHITECTURE
Volume 2
Patterns for Concurrent and Networked Objects
WILEY
WILEY SERIES IN SOFTWARE DESIGN PATTERNS

Proactor allows concurrent applications to efficiently process service requests & responses asynchronously

Overview of Pattern-Oriented Framework Solution

- "Gang-of-Four" & POSA patterns enhance the framework-based solution



<<Java Class>>
Filter
- Filter()
- Filter(String)
- *applyFilter(ImageEntity):ImageEntity*
- filter(ImageEntity):ImageEntity
- setName(String):void
- getName():String

-mFilters
0..*

<<Java Class>>
ImageTaskGang
- ImageTaskGang(Filter[],Iterator<List<URL>>,Runnable)
- getNextInput():List<URL>
- initiateTaskGang(int):void
- processInput(URL):boolean
- taskDone(int):void
- awaitTasksDone():void
- concurrentlyProcessFilteredResults(int):void

<<Java Class>>
ExecutorCompletionService<V>
- ExecutorCompletionService(Executor)
- submit(Callable<V>):Future<V>
- submit(Runnable,V):Future<V>
- take():Future<V>
- poll():Future<V>
- poll(long,TimeUnit):Future<V>

Completion
Handler

Proactive
Initiator

Asynchronous
Operation
Processor

Proactor

#mFilter 0..1

Douglas Schmidt
Michael Stal
Hans Rohnert
Frank Buschmann
PATTERN-ORIENTED SOFTWARE ARCHITECTURE
Volume 2
Patterns for Concurrent and Networked Objects
WILEY

<<Java Class>>
FilterDecorator
- FilterDecorator(Filter)
- *decorate(ImageEntity):ImageEntity*
- applyFilter(ImageEntity):ImageEntity

<<Java Class>>
TaskGang<E>
- TaskGang()
- getInput():List<E>
- setInput(List<E>):List<E>
- setExecutor(Executor):void
- getExecutor():Executor
- incrementCycle():long
- currentCycle():long
- *getNextInput():List<E>*
- initiateHook(int):void
- *initiateTaskGang(int):void*
- advanceTaskToNextCycle():boolean
- *awaitTasksDone():void*
- taskDone(int):void
- *processInput(E):boolean*
- run():void
- makeTask(int):Runnable

<<Java Class>>
NullFilter
- NullFilter()
- NullFilter(String)
- applyFilter(ImageEntity):ImageEntity

<<Java Class>>
Image
- ImageEntity(URL,byte[])
- ImageEntity(URL,Image)
- setImage(byte[]):void
- getImage():Image
- setSourceURL(URL):void
- getSourceURL():URL
- setFilterName(Filter):void
- getFilterName():String
- setSucceeded(boolean):void
- getSucceeded():boolean
- getFileName():String
- getFormatName():String

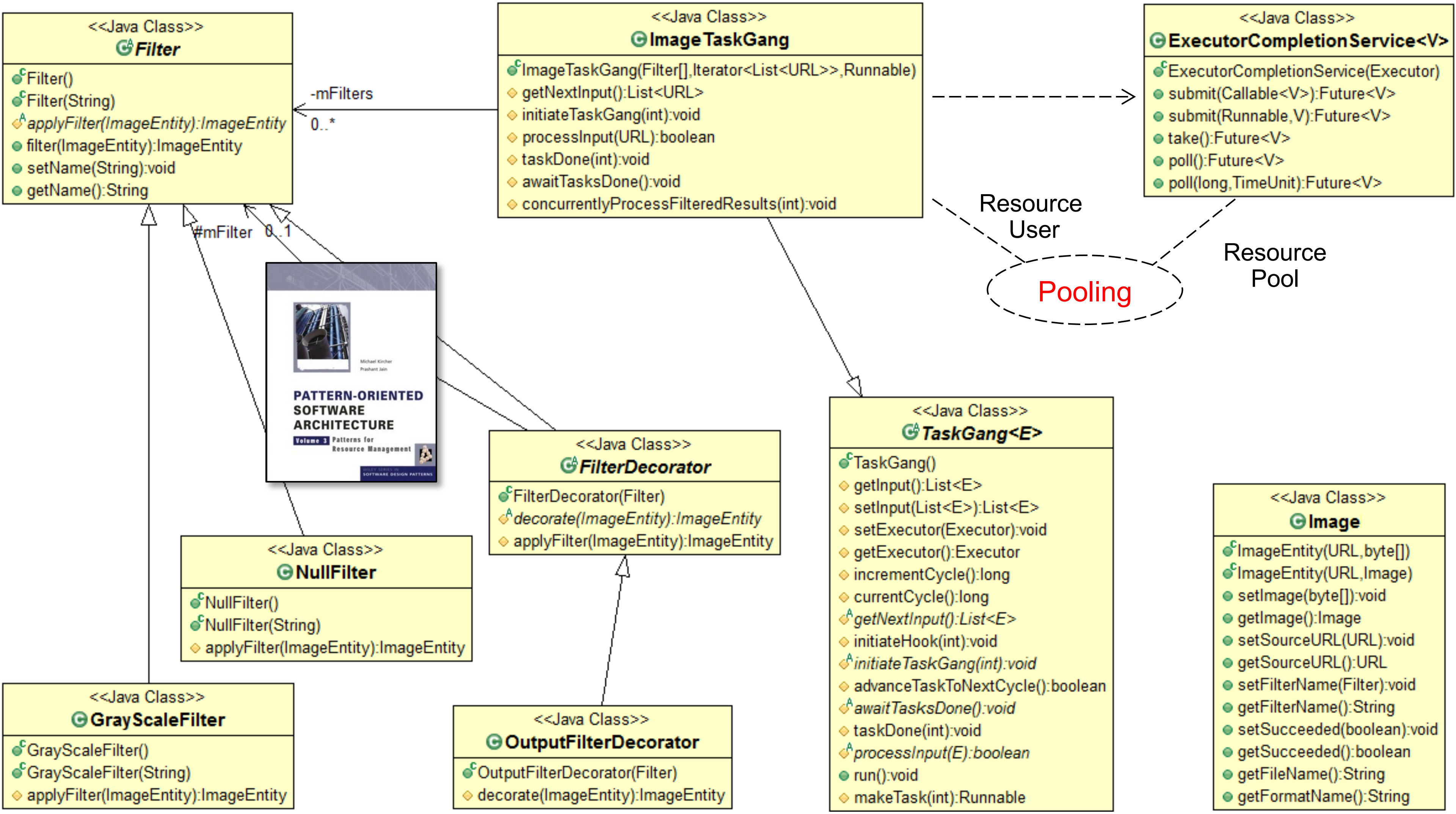
<<Java Class>>
GrayScaleFilter
- GrayScaleFilter()
- GrayScaleFilter(String)
- applyFilter(ImageEntity):ImageEntity

<<Java Class>>
OutputFilterDecorator
- OutputFilterDecorator(Filter)
- decorate(ImageEntity):ImageEntity

See en.wikipedia.org/wiki/Proactor_pattern

Overview of Pattern-Oriented Framework Solution

- “Gang-of-Four” & POSA patterns enhance the framework-based solution



<<Java Class>>
Filter
- Filter()
- Filter(String)
- *applyFilter(ImageEntity):ImageEntity*
- filter(ImageEntity):ImageEntity
- setName(String):void
- getName():String

<<Java Class>>
ImageTaskGang
- ImageTaskGang(Filter[],Iterator<List<URL>>,Runnable)
- getNextInput():List<URL>
- initiateTaskGang(int):void
- processInput(URL):boolean
- taskDone(int):void
- awaitTasksDone():void
- concurrentlyProcessFilteredResults(int):void

<<Java Class>>
ExecutorCompletionService<V>
- ExecutorCompletionService(Executor)
- submit(Callable<V>):Future<V>
- submit(Runnable,V):Future<V>
- take():Future<V>
- poll():Future<V>
- poll(long,TimeUnit):Future<V>

<<Java Class>>
FilterDecorator
- FilterDecorator(Filter)
- *decorate(ImageEntity):ImageEntity*
- applyFilter(ImageEntity):ImageEntity

<<Java Class>>
TaskGang<E>
- TaskGang()
- getInput():List<E>
- setInput(List<E>):List<E>
- setExecutor(Executor):void
- getExecutor():Executor
- incrementCycle():long
- currentCycle():long
- *getNextInput():List<E>*
- initiateHook(int):void
- *initiateTaskGang(int):void*
- advanceTaskToNextCycle():boolean
- *awaitTasksDone():void*
- taskDone(int):void
- *processInput(E):boolean*
- run():void
- makeTask(int):Runnable

<<Java Class>>
NullFilter
- NullFilter()
- NullFilter(String)
- applyFilter(ImageEntity):ImageEntity

<<Java Class>>
Image
- ImageEntity(URL,byte[])
- ImageEntity(URL,Image)
- setImage(byte[]):void
- getImage():Image
- setSourceURL(URL):void
- getSourceURL():URL
- setFilterName(Filter):void
- getFilterName():String
- setSucceeded(boolean):void
- getSucceeded():boolean
- getFileName():String
- getFormatName():String

<<Java Class>>
GrayScaleFilter
- GrayScaleFilter()
- GrayScaleFilter(String)
- applyFilter(ImageEntity):ImageEntity

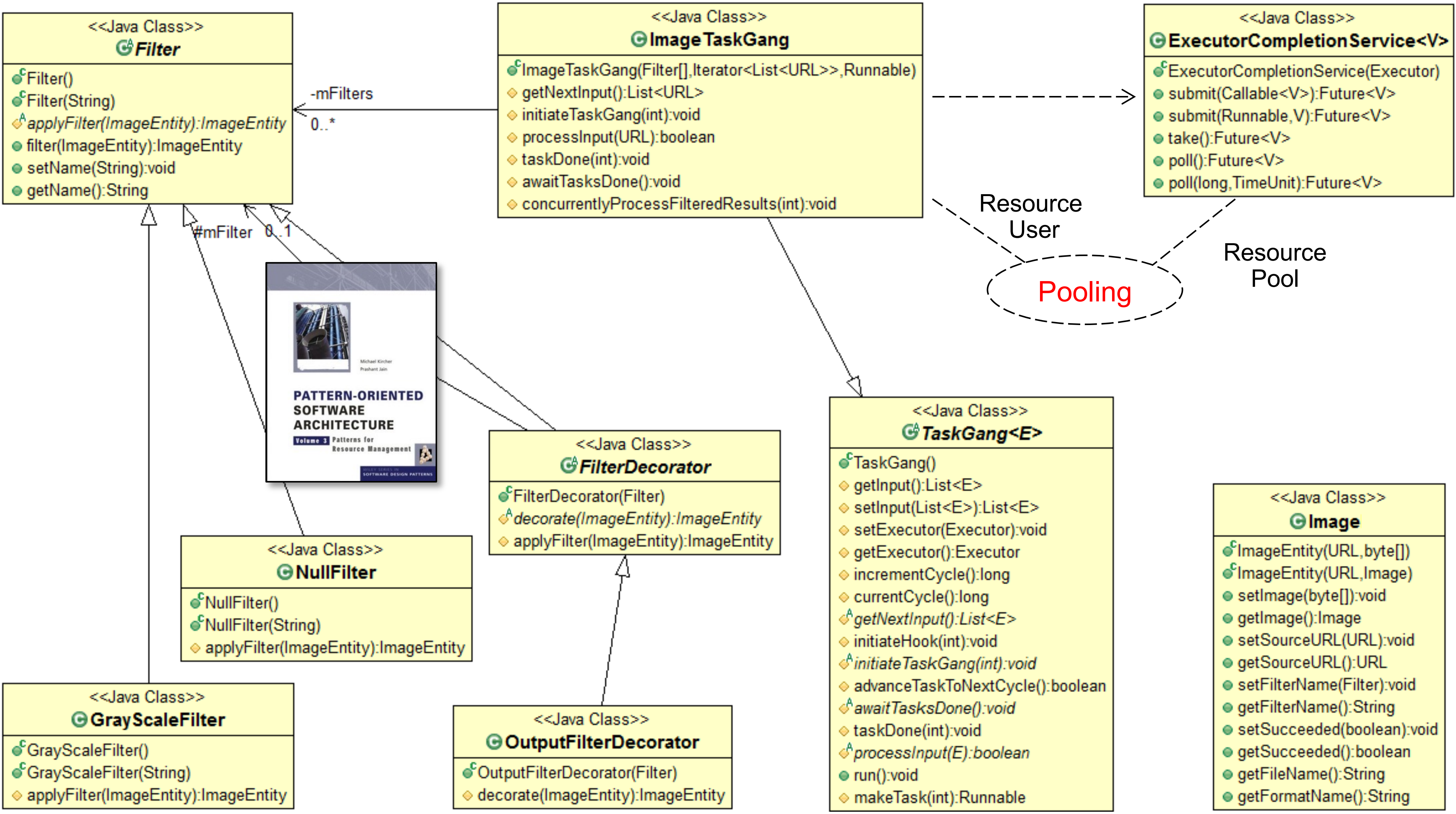
<<Java Class>>
OutputFilterDecorator
- OutputFilterDecorator(Filter)
- decorate(ImageEntity):ImageEntity

-mFilters
0..*
#mFilter
0..1
Resource User
Resource Pool
Pooling
Michael Kircher
Prashant Jain
PATTERN-ORIENTED SOFTWARE ARCHITECTURE
Volume 3 Patterns for Resource Management
WILEY SERIES IN SOFTWARE DESIGN PATTERNS

Pooling manages multiple worker threads & optimizes their acquisition & release

Overview of Pattern-Oriented Framework Solution

- “Gang-of-Four” & POSA patterns enhance the framework-based solution



<<Java Class>>
Filter

- Filter()
- Filter(String)
- *applyFilter(ImageEntity):ImageEntity*
- filter(ImageEntity):ImageEntity
- setName(String):void
- getName():String

<<Java Class>>
ImageTaskGang

- ImageTaskGang(Filter[],Iterator<List<URL>>,Runnable)
- getNextInput():List<URL>
- initiateTaskGang(int):void
- processInput(URL):boolean
- taskDone(int):void
- awaitTasksDone():void
- concurrentlyProcessFilteredResults(int):void

<<Java Class>>
ExecutorCompletionService<V>

- ExecutorCompletionService(Executor)
- submit(Callable<V>):Future<V>
- submit(Runnable,V):Future<V>
- take():Future<V>
- poll():Future<V>
- poll(long,TimeUnit):Future<V>

-mFilters

0..*

#mFilter 0..1

Resource User

Pooling

Resource Pool

<<Java Class>>
FilterDecorator

- FilterDecorator(Filter)
- *decorate(ImageEntity):ImageEntity*
- applyFilter(ImageEntity):ImageEntity

<<Java Class>>
TaskGang<E>

- TaskGang()
- getInput():List<E>
- setInput(List<E>):List<E>
- setExecutor(Executor):void
- getExecutor():Executor
- incrementCycle():long
- currentCycle():long
- *getNextInput():List<E>*
- initiateHook(int):void
- *initiateTaskGang(int):void*
- advanceTaskToNextCycle():boolean
- *awaitTasksDone():void*
- taskDone(int):void
- *processInput(E):boolean*
- run():void
- makeTask(int):Runnable

<<Java Class>>
NullFilter

- NullFilter()
- NullFilter(String)
- applyFilter(ImageEntity):ImageEntity

<<Java Class>>
Image

- ImageEntity(URL,byte[])
- ImageEntity(URL,Image)
- setImage(byte[]):void
- getImage():Image
- setSourceURL(URL):void
- getSourceURL():URL
- setFilterName(Filter):void
- getFilterName():String
- setSucceeded(boolean):void
- getSucceeded():boolean
- getFileName():String
- getFormatName():String

<<Java Class>>
GrayScaleFilter

- GrayScaleFilter()
- GrayScaleFilter(String)
- applyFilter(ImageEntity):ImageEntity

<<Java Class>>
OutputFilterDecorator

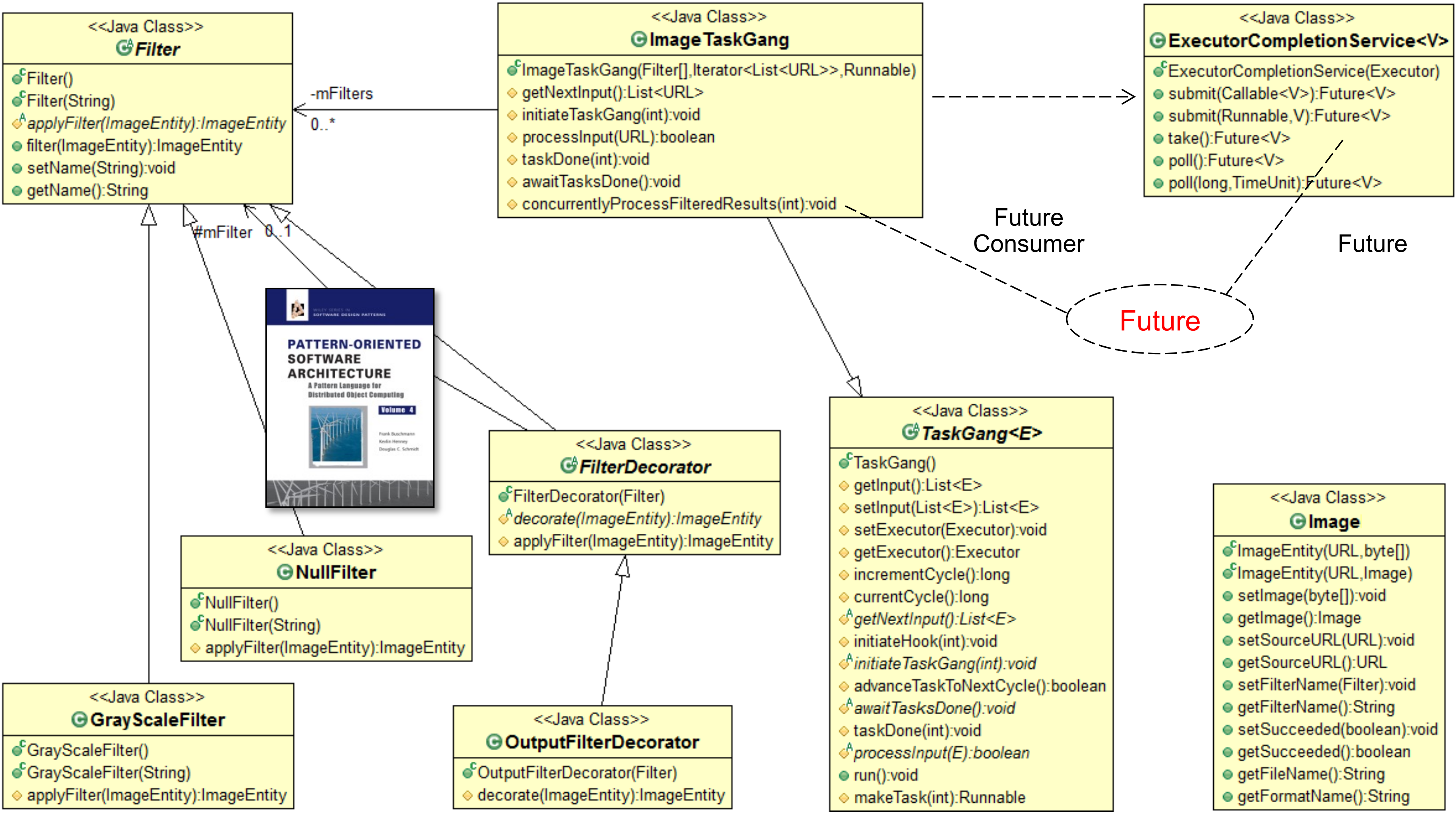
- OutputFilterDecorator(Filter)
- decorate(ImageEntity):ImageEntity

Michael Kircher
Prashant Jain
PATTERN-ORIENTED SOFTWARE ARCHITECTURE
Volume 3 Patterns for Resource Management

See kircher-schwanninger.de/michael/publications/Pooling.pdf

Overview of Pattern-Oriented Framework Solution

- "Gang-of-Four" & POSA patterns enhance the framework-based solution



<<Java Class>>
Filter
- Filter()
- Filter(String)
- *applyFilter(ImageEntity):ImageEntity*
- filter(ImageEntity):ImageEntity
- setName(String):void
- getName():String

<<Java Class>>
ImageTaskGang
- ImageTaskGang(Filter[],Iterator<List<URL>>,Runnable)
- getNextInput():List<URL>
- initiateTaskGang(int):void
- processInput(URL):boolean
- taskDone(int):void
- awaitTasksDone():void
- concurrentlyProcessFilteredResults(int):void

<<Java Class>>
ExecutorCompletionService<V>
- ExecutorCompletionService(Executor)
- submit(Callable<V>):Future<V>
- submit(Runnable,V):Future<V>
- take():Future<V>
- poll():Future<V>
- poll(long,TimeUnit):Future<V>

<<Java Class>>
TaskGang<E>
- TaskGang()
- getInput():List<E>
- setInput(List<E>):List<E>
- setExecutor(Executor):void
- getExecutor():Executor
- incrementCycle():long
- currentCycle():long
- *getNextInput():List<E>*
- initiateHook(int):void
- *initiateTaskGang(int):void*
- advanceTaskToNextCycle():boolean
- *awaitTasksDone():void*
- taskDone(int):void
- *processInput(E):boolean*
- run():void
- makeTask(int):Runnable

<<Java Class>>
FilterDecorator
- FilterDecorator(Filter)
- *decorate(ImageEntity):ImageEntity*
- applyFilter(ImageEntity):ImageEntity

<<Java Class>>
NullFilter
- NullFilter()
- NullFilter(String)
- applyFilter(ImageEntity):ImageEntity

<<Java Class>>
GrayScaleFilter
- GrayScaleFilter()
- GrayScaleFilter(String)
- applyFilter(ImageEntity):ImageEntity

<<Java Class>>
OutputFilterDecorator
- OutputFilterDecorator(Filter)
- decorate(ImageEntity):ImageEntity

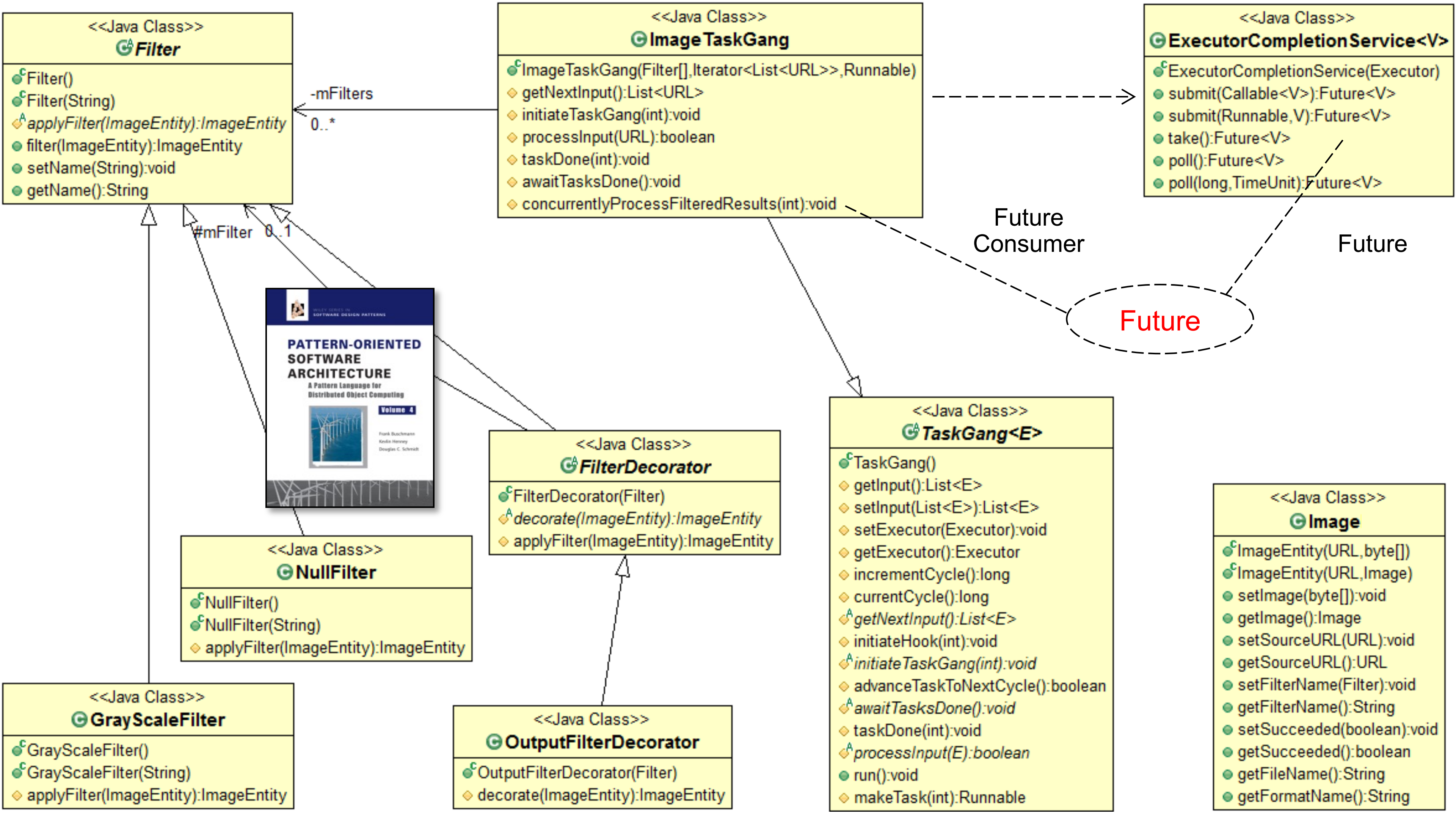
<<Java Class>>
Image
- ImageEntity(URL,byte[])
- ImageEntity(URL,Image)
- setImage(byte[]):void
- getImage():Image
- setSourceURL(URL):void
- getSourceURL():URL
- setFilterName(Filter):void
- getFilterName():String
- setSucceeded(boolean):void
- getSucceeded():boolean
- getFileName():String
- getFormatName():String

-mFilters
0..*
#mFilter
0..1
Future Consumer
Future
Future
WILEY SERIES IN SOFTWARE DESIGN PATTERNS
PATTERN-ORIENTED SOFTWARE ARCHITECTURE
A Pattern Language for Distributed Object Computing
Volume 4
Frank Buschmann
Kevlin Henney
Douglas C. Schmidt

Future provides a proxy for a result that may not be available yet, allowing a program to continue execution without blocking until the result is needed

Overview of Pattern-Oriented Framework Solution

- “Gang-of-Four” & POSA patterns enhance the framework-based solution



<<Java Class>>
Filter
- Filter()
- Filter(String)
- *applyFilter(ImageEntity):ImageEntity*
- filter(ImageEntity):ImageEntity
- setName(String):void
- getName():String

<<Java Class>>
ImageTaskGang
- ImageTaskGang(Filter[],Iterator<List<URL>>,Runnable)
- getNextInput():List<URL>
- initiateTaskGang(int):void
- processInput(URL):boolean
- taskDone(int):void
- awaitTasksDone():void
- concurrentlyProcessFilteredResults(int):void

<<Java Class>>
ExecutorCompletionService<V>
- ExecutorCompletionService(Executor)
- submit(Callable<V>):Future<V>
- submit(Runnable,V):Future<V>
- take():Future<V>
- poll():Future<V>
- poll(long,TimeUnit):Future<V>

<<Java Class>>
FilterDecorator
- FilterDecorator(Filter)
- *decorate(ImageEntity):ImageEntity*
- applyFilter(ImageEntity):ImageEntity

<<Java Class>>
TaskGang<E>
- TaskGang()
- getInput():List<E>
- setInput(List<E>):List<E>
- setExecutor(Executor):void
- getExecutor():Executor
- incrementCycle():long
- currentCycle():long
- *getNextInput():List<E>*
- initiateHook(int):void
- *initiateTaskGang(int):void*
- advanceTaskToNextCycle():boolean
- *awaitTasksDone():void*
- taskDone(int):void
- *processInput(E):boolean*
- run():void
- makeTask(int):Runnable

<<Java Class>>
NullFilter
- NullFilter()
- NullFilter(String)
- applyFilter(ImageEntity):ImageEntity

<<Java Class>>
Image
- ImageEntity(URL,byte[])
- ImageEntity(URL,Image)
- setImage(byte[]):void
- getImage():Image
- setSourceURL(URL):void
- getSourceURL():URL
- setFilterName(Filter):void
- getFilterName():String
- setSucceeded(boolean):void
- getSucceeded():boolean
- getFileName():String
- getFormatName():String

<<Java Class>>
GrayScaleFilter
- GrayScaleFilter()
- GrayScaleFilter(String)
- applyFilter(ImageEntity):ImageEntity

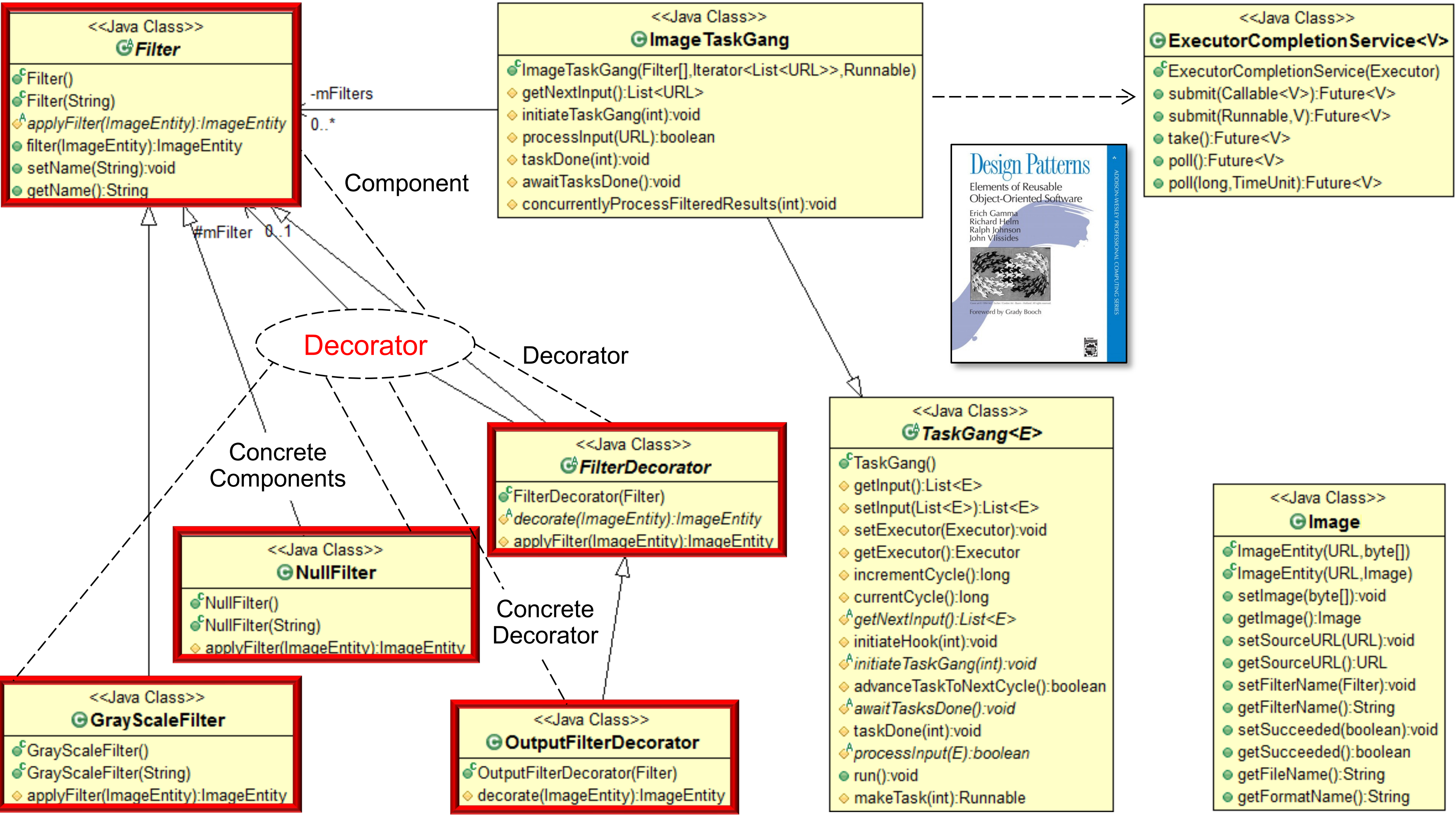
<<Java Class>>
OutputFilterDecorator
- OutputFilterDecorator(Filter)
- decorate(ImageEntity):ImageEntity

-mFilters
0..*
#mFilter
0..1
Future Consumer
Future
Future
WILEY SERIES IN SOFTWARE DESIGN PATTERNS
PATTERN-ORIENTED SOFTWARE ARCHITECTURE
A Pattern Language for Distributed Object Computing
Volume 4
Frank Buschmann
Kevin Henney
Douglas C. Schmidt

See en.wikipedia.org/wiki/Futures_and_promises

Overview of Pattern-Oriented Framework Solution

- "Gang-of-Four" & POSA patterns enhance the framework-based solution



<<Java Class>>
Filter
- Filter()
- Filter(String)
- *applyFilter(ImageEntity):ImageEntity*
- filter(ImageEntity):ImageEntity
- setName(String):void
- getName():String

<<Java Class>>
ImageTaskGang
- ImageTaskGang(Filter[],Iterator<List<URL>>,Runnable)
- getNextInput():List<URL>
- initiateTaskGang(int):void
- processInput(URL):boolean
- taskDone(int):void
- awaitTasksDone():void
- concurrentlyProcessFilteredResults(int):void

<<Java Class>>
ExecutorCompletionService<V>
- ExecutorCompletionService(Executor)
- submit(Callable<V>):Future<V>
- submit(Runnable,V):Future<V>
- take():Future<V>
- poll():Future<V>
- poll(long,TimeUnit):Future<V>

<<Java Class>>
FilterDecorator
- FilterDecorator(Filter)
- *decorate(ImageEntity):ImageEntity*
- applyFilter(ImageEntity):ImageEntity

<<Java Class>>
NullFilter
- NullFilter()
- NullFilter(String)
- applyFilter(ImageEntity):ImageEntity

<<Java Class>>
GrayScaleFilter
- GrayScaleFilter()
- GrayScaleFilter(String)
- applyFilter(ImageEntity):ImageEntity

<<Java Class>>
OutputFilterDecorator
- OutputFilterDecorator(Filter)
- decorate(ImageEntity):ImageEntity

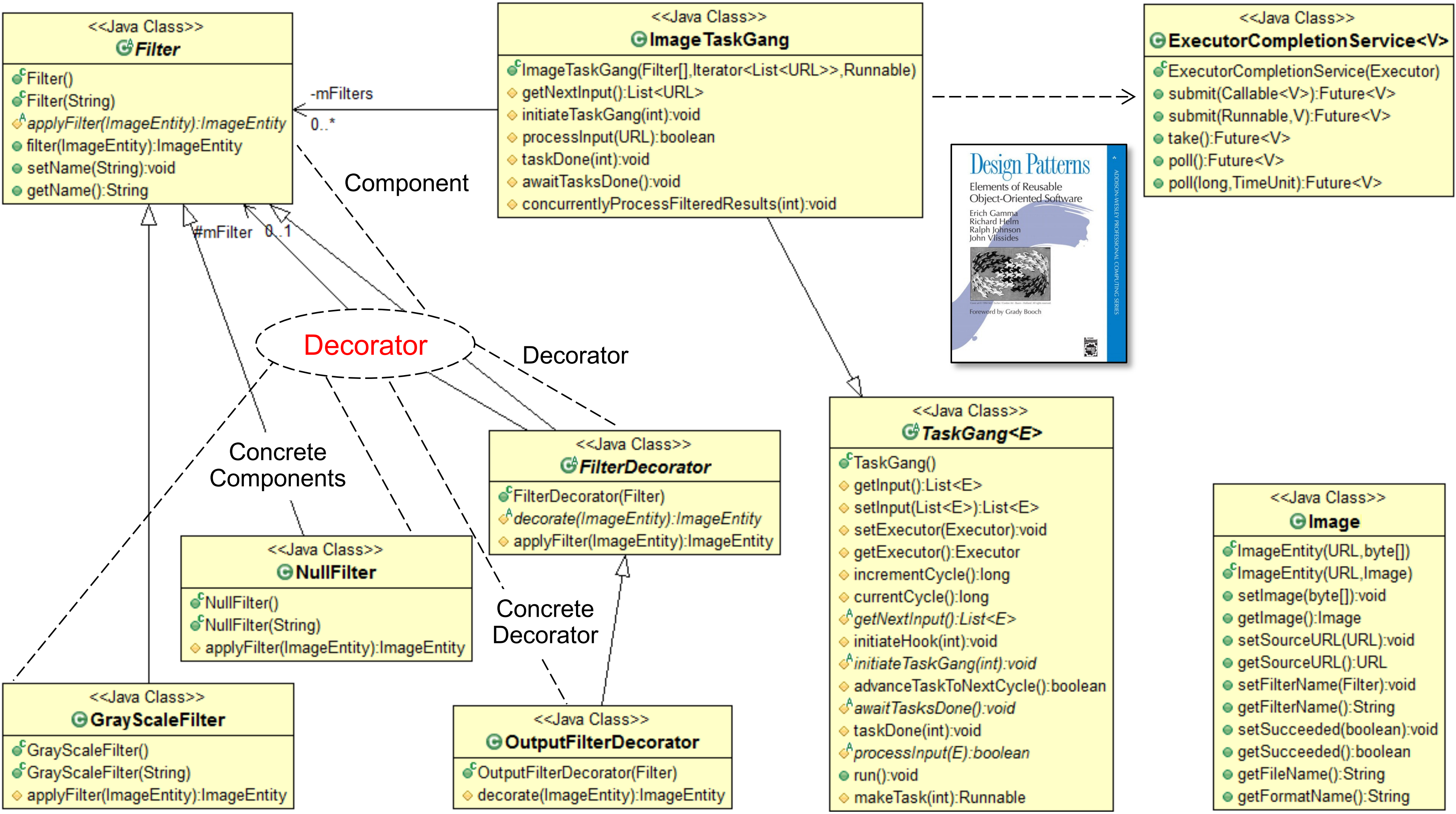
<<Java Class>>
TaskGang<E>
- TaskGang()
- getInput():List<E>
- setInput(List<E>):List<E>
- setExecutor(Executor):void
- getExecutor():Executor
- incrementCycle():long
- currentCycle():long
- *getNextInput():List<E>*
- initiateHook(int):void
- *initiateTaskGang(int):void*
- advanceTaskToNextCycle():boolean
- *awaitTasksDone():void*
- taskDone(int):void
- *processInput(E):boolean*
- run():void
- makeTask(int):Runnable

<<Java Class>>
Image
- ImageEntity(URL,byte[])
- ImageEntity(URL,Image)
- setImage(byte[]):void
- getImage():Image
- setSourceURL(URL):void
- getSourceURL():URL
- setFilterName(Filter):void
- getFilterName():String
- setSucceeded(boolean):void
- getSucceeded():boolean
- getFileName():String
- getFormatName():String

Decorator allows behavior to be added to an individual object transparently, without affecting the behavior of other object's from the same class

Overview of Pattern-Oriented Framework Solution

- “Gang-of-Four” & POSA patterns enhance the framework-based solution



<<Java Class>>
Filter

- Filter()
- Filter(String)
- *applyFilter(ImageEntity):ImageEntity*
- filter(ImageEntity):ImageEntity
- setName(String):void
- getName():String

<<Java Class>>
ImageTaskGang

- ImageTaskGang(Filter[],Iterator<List<URL>>,Runnable)
- getNextInput():List<URL>
- initiateTaskGang(int):void
- processInput(URL):boolean
- taskDone(int):void
- awaitTasksDone():void
- concurrentlyProcessFilteredResults(int):void

<<Java Class>>
ExecutorCompletionService<V>

- ExecutorCompletionService(Executor)
- submit(Callable<V>):Future<V>
- submit(Runnable,V):Future<V>
- take():Future<V>
- poll():Future<V>
- poll(long,TimeUnit):Future<V>

<<Java Class>>
FilterDecorator

- FilterDecorator(Filter)
- *decorate(ImageEntity):ImageEntity*
- applyFilter(ImageEntity):ImageEntity

<<Java Class>>
NullFilter

- NullFilter()
- NullFilter(String)
- applyFilter(ImageEntity):ImageEntity

<<Java Class>>
GrayScaleFilter

- GrayScaleFilter()
- GrayScaleFilter(String)
- applyFilter(ImageEntity):ImageEntity

<<Java Class>>
OutputFilterDecorator

- OutputFilterDecorator(Filter)
- decorate(ImageEntity):ImageEntity

<<Java Class>>
TaskGang<E>

- TaskGang()
- getInput():List<E>
- setInput(List<E>):List<E>
- setExecutor(Executor):void
- getExecutor():Executor
- incrementCycle():long
- currentCycle():long
- *getNextInput():List<E>*
- initiateHook(int):void
- *initiateTaskGang(int):void*
- advanceTaskToNextCycle():boolean
- *awaitTasksDone():void*
- taskDone(int):void
- *processInput(E):boolean*
- run():void
- makeTask(int):Runnable

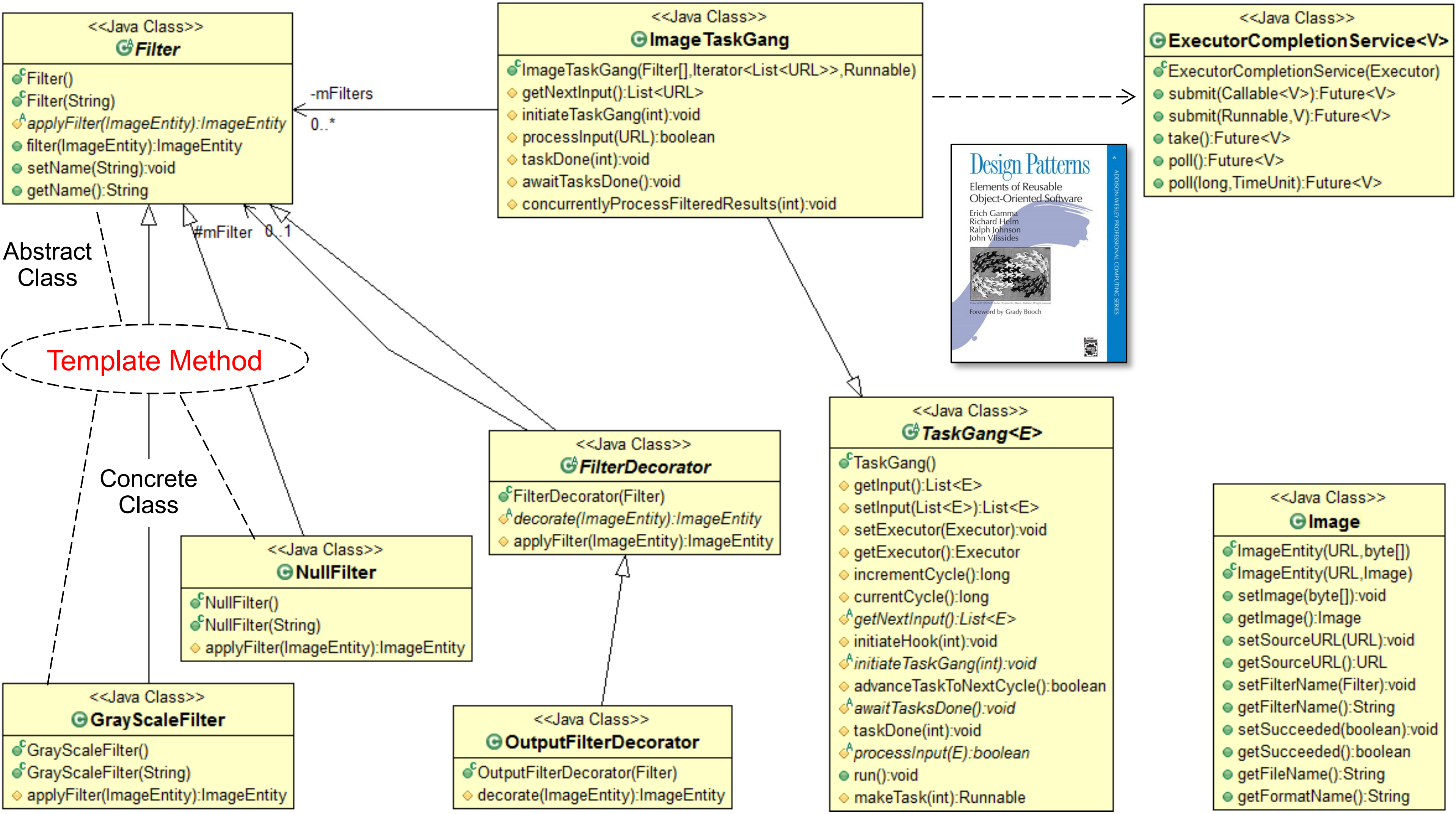
<<Java Class>>
Image

- ImageEntity(URL,byte[])
- ImageEntity(URL,Image)
- setImage(byte[]):void
- getImage():Image
- setSourceURL(URL):void
- getSourceURL():URL
- setFilterName(Filter):void
- getFilterName():String
- setSucceeded(boolean):void
- getSucceeded():boolean
- getFileName():String
- getFormatName():String

See en.wikipedia.org/wiki/Decorator_pattern

Overview of Pattern-Oriented Framework Solution

- "Gang-of-Four" & POSA patterns enhance the framework-based solution



<<Java Class>>
Filter
- Filter()
- Filter(String)
- *applyFilter(ImageEntity):ImageEntity*
- filter(ImageEntity):ImageEntity
- setName(String):void
- getName():String

<<Java Class>>
ImageTaskGang
- ImageTaskGang(Filter[],Iterator<List<URL>>,Runnable)
- getNextInput():List<URL>
- initiateTaskGang(int):void
- processInput(URL):boolean
- taskDone(int):void
- awaitTasksDone():void
- concurrentlyProcessFilteredResults(int):void

<<Java Class>>
ExecutorCompletionService<V>
- ExecutorCompletionService(Executor)
- submit(Callable<V>):Future<V>
- submit(Runnable,V):Future<V>
- take():Future<V>
- poll():Future<V>
- poll(long,TimeUnit):Future<V>

<<Java Class>>
FilterDecorator
- FilterDecorator(Filter)
- *decorate(ImageEntity):ImageEntity*
- applyFilter(ImageEntity):ImageEntity

<<Java Class>>
TaskGang<E>
- TaskGang()
- getInput():List<E>
- setInput(List<E>):List<E>
- setExecutor(Executor):void
- getExecutor():Executor
- incrementCycle():long
- currentCycle():long
- *getNextInput():List<E>*
- initiateHook(int):void
- *initiateTaskGang(int):void*
- advanceTaskToNextCycle():boolean
- *awaitTasksDone():void*
- taskDone(int):void
- *processInput(E):boolean*
- run():void
- makeTask(int):Runnable

<<Java Class>>
NullFilter
- NullFilter()
- NullFilter(String)
- applyFilter(ImageEntity):ImageEntity

<<Java Class>>
GrayScaleFilter
- GrayScaleFilter()
- GrayScaleFilter(String)
- applyFilter(ImageEntity):ImageEntity

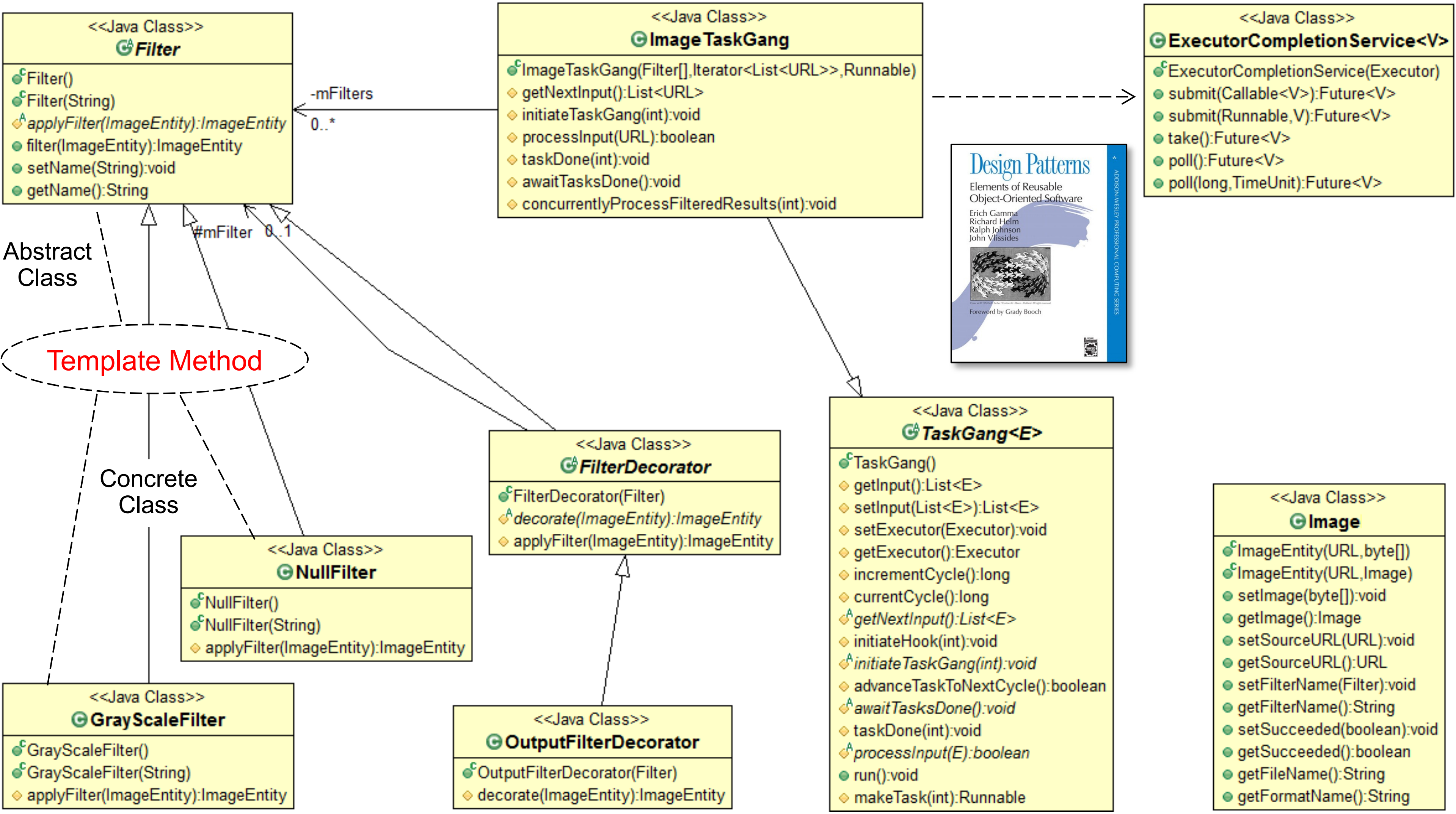
<<Java Class>>
OutputFilterDecorator
- OutputFilterDecorator(Filter)
- decorate(ImageEntity):ImageEntity

<<Java Class>>
Image
- ImageEntity(URL,byte[])
- ImageEntity(URL,Image)
- setImage(byte[]):void
- getImage():Image
- setSourceURL(URL):void
- getSourceURL():URL
- setFilterName(Filter):void
- getFilterName():String
- setSucceeded(boolean):void
- getSucceeded():boolean
- getFileName():String
- getFormatName():String

Template Method defines the skeleton of an algorithm in a method, deferring certain steps to subclass methods

Overview of Pattern-Oriented Framework Solution

- "Gang-of-Four" & POSA patterns enhance the framework-based solution



<<Java Class>>
Filter

Filter()
Filter(String)
applyFilter(ImageEntity):ImageEntity
filter(ImageEntity):ImageEntity
setName(String):void
getName():String

<<Java Class>>
ImageTaskGang

ImageTaskGang(Filter[],Iterator<List<URL>>,Runnable)
getNextInput():List<URL>
initiateTaskGang(int):void
processInput(URL):boolean
taskDone(int):void
awaitTasksDone():void
concurrentlyProcessFilteredResults(int):void

<<Java Class>>
ExecutorCompletionService<V>

ExecutorCompletionService(Executor)
submit(Callable<V>):Future<V>
submit(Runnable,V):Future<V>
take():Future<V>
poll():Future<V>
poll(long,TimeUnit):Future<V>

<<Java Class>>
FilterDecorator

FilterDecorator(Filter)
decorate(ImageEntity):ImageEntity
applyFilter(ImageEntity):ImageEntity

<<Java Class>>
TaskGang<E>

TaskGang()
getInput():List<E>
setInput(List<E>):List<E>
setExecutor(Executor):void
getExecutor():Executor
incrementCycle():long
currentCycle():long
getNextInput():List<E>
initiateHook(int):void
initiateTaskGang(int):void
advanceTaskToNextCycle():boolean
awaitTasksDone():void
taskDone(int):void
processInput(E):boolean
run():void
makeTask(int):Runnable

<<Java Class>>
Image

ImageEntity(URL,byte[])
ImageEntity(URL,Image)
setImage(byte[]):void
getImage():Image
setSourceURL(URL):void
getSourceURL():URL
setFilterName(Filter):void
getFilterName():String
setSucceeded(boolean):void
getSucceeded():boolean
getFileName():String
getFormatName():String

<<Java Class>>
NullFilter

NullFilter()
NullFilter(String)
applyFilter(ImageEntity):ImageEntity

<<Java Class>>
GrayScaleFilter

GrayScaleFilter()
GrayScaleFilter(String)
applyFilter(ImageEntity):ImageEntity

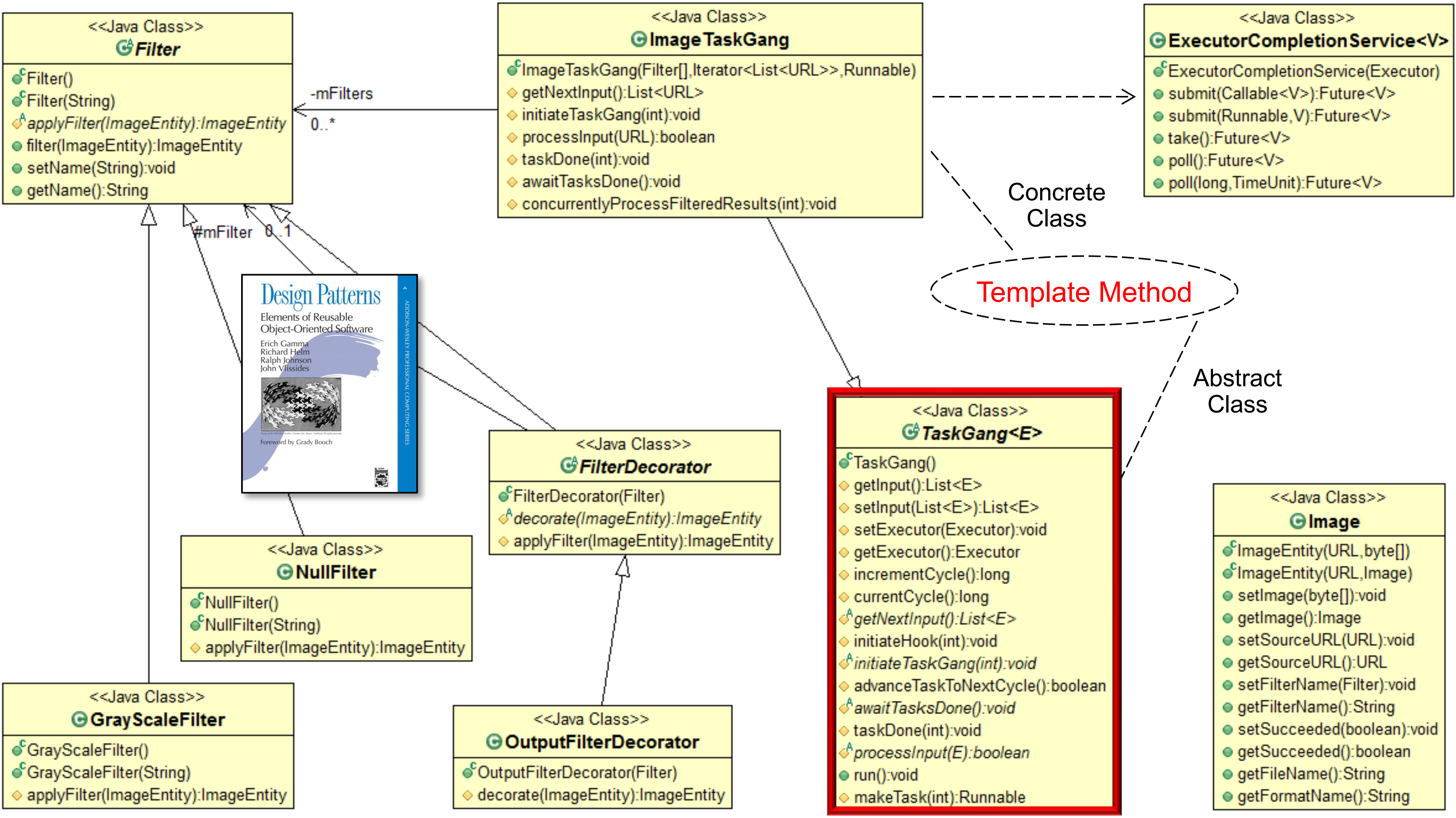
<<Java Class>>
OutputFilterDecorator

OutputFilterDecorator(Filter)
decorate(ImageEntity):ImageEntity

See en.wikipedia.org/wiki/Template_method_pattern

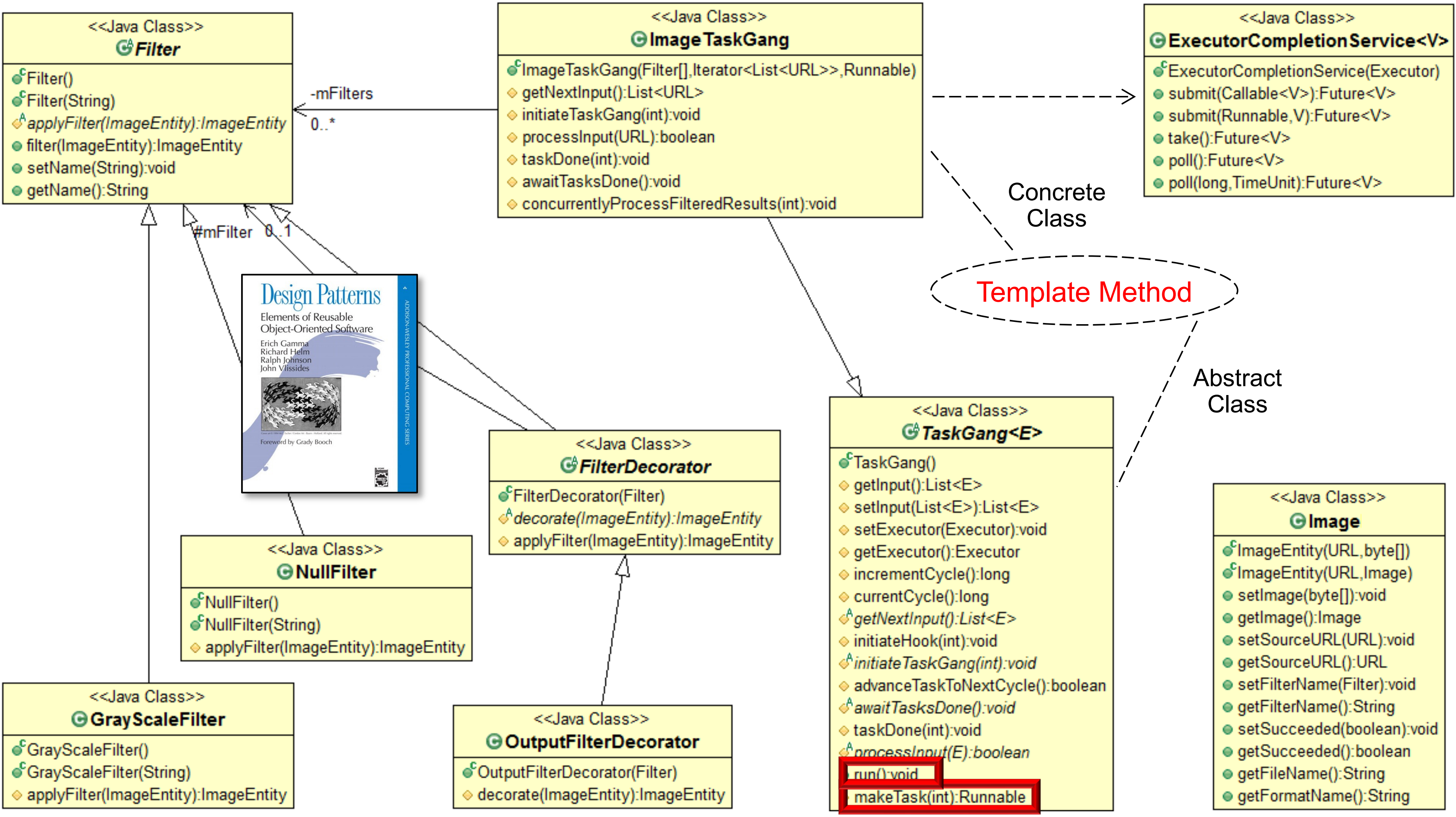
Overview of Pattern-Oriented Framework Solution

- "Gang-of-Four" & POSA patterns enhance the framework-based solution



Overview of Pattern-Oriented Framework Solution

- "Gang-of-Four" & POSA patterns enhance the framework-based solution



<<Java Class>>
Filter
- Filter()
- Filter(String)
- *applyFilter(ImageEntity):ImageEntity*
- filter(ImageEntity):ImageEntity
- setName(String):void
- getName():String

<<Java Class>>
ImageTaskGang
- ImageTaskGang(Filter[],Iterator<List<URL>>,Runnable)
- getNextInput():List<URL>
- initiateTaskGang(int):void
- processInput(URL):boolean
- taskDone(int):void
- awaitTasksDone():void
- concurrentlyProcessFilteredResults(int):void

<<Java Class>>
ExecutorCompletionService<V>
- ExecutorCompletionService(Executor)
- submit(Callable<V>):Future<V>
- submit(Runnable,V):Future<V>
- take():Future<V>
- poll():Future<V>
- poll(long,TimeUnit):Future<V>

-mFilters

0..*

#mFilter

0..1

Concrete Class

Template Method

Abstract Class

<<Java Class>>
FilterDecorator
- FilterDecorator(Filter)
- *decorate(ImageEntity):ImageEntity*
- applyFilter(ImageEntity):ImageEntity

<<Java Class>>
NullFilter
- NullFilter()
- NullFilter(String)
- applyFilter(ImageEntity):ImageEntity

<<Java Class>>
GrayScaleFilter
- GrayScaleFilter()
- GrayScaleFilter(String)
- applyFilter(ImageEntity):ImageEntity

<<Java Class>>
OutputFilterDecorator
- OutputFilterDecorator(Filter)
- decorate(ImageEntity):ImageEntity

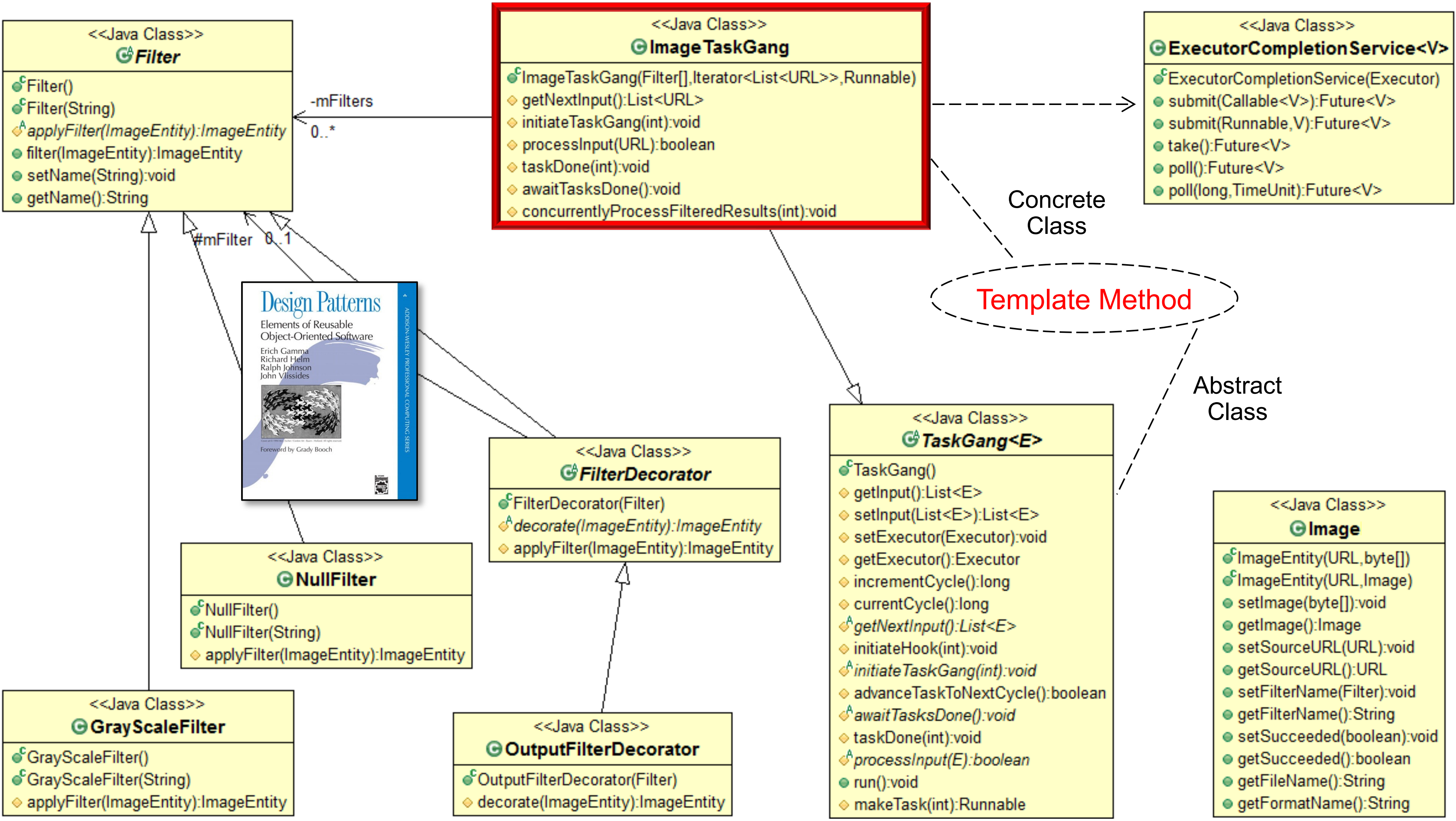
<<Java Class>>
TaskGang<E>
- TaskGang()
- getInput():List<E>
- setInput(List<E>):List<E>
- setExecutor(Executor):void
- getExecutor():Executor
- incrementCycle():long
- currentCycle():long
- *getNextInput():List<E>*
- initiateHook(int):void
- *initiateTaskGang(int):void*
- advanceTaskToNextCycle():boolean
- *awaitTasksDone():void*
- taskDone(int):void
- *processInput(E):boolean*
- run():void
- makeTask(int):Runnable

<<Java Class>>
Image
- ImageEntity(URL,byte[])
- ImageEntity(URL,Image)
- setImage(byte[]):void
- getImage():Image
- setSourceURL(URL):void
- getSourceURL():URL
- setFilterName(Filter):void
- getFilterName():String
- setSucceeded(boolean):void
- getSucceeded():boolean
- getFileName():String
- getFormatName():String

Design Patterns
Elements of Reusable Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides
Foreword by Grady Booch
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

Overview of Pattern-Oriented Framework Solution

- "Gang-of-Four" & POSA patterns enhance the framework-based solution



```
<<Java Class>>
Filter
Filter()
Filter(String)
applyFilter(ImageEntity):ImageEntity
filter(ImageEntity):ImageEntity
setName(String):void
getName():String
```

```
<<Java Class>>
ImageTaskGang
ImageTaskGang(Filter[],Iterator<List<URL>>,Runnable)
getNextInput():List<URL>
initiateTaskGang(int):void
processInput(URL):boolean
taskDone(int):void
awaitTasksDone():void
concurrentlyProcessFilteredResults(int):void
```

```
<<Java Class>>
ExecutorCompletionService<V>
ExecutorCompletionService(Executor)
submit(Callable<V>):Future<V>
submit(Runnable,V):Future<V>
take():Future<V>
poll():Future<V>
poll(long,TimeUnit):Future<V>
```

```
<<Java Class>>
FilterDecorator
FilterDecorator(Filter)
decorate(ImageEntity):ImageEntity
applyFilter(ImageEntity):ImageEntity
```

```
<<Java Class>>
NullFilter
NullFilter()
NullFilter(String)
applyFilter(ImageEntity):ImageEntity
```

```
<<Java Class>>
GrayScaleFilter
GrayScaleFilter()
GrayScaleFilter(String)
applyFilter(ImageEntity):ImageEntity
```

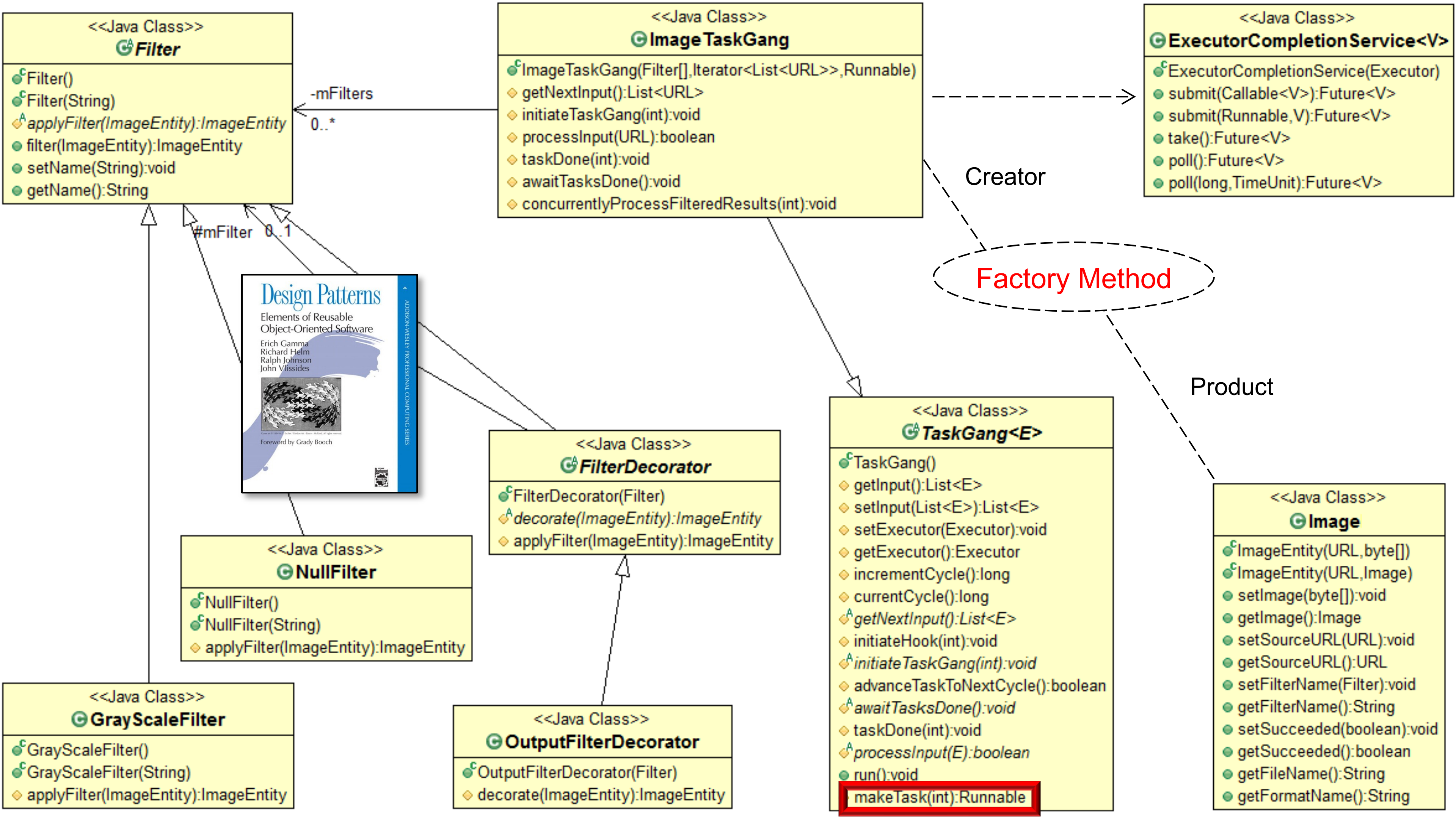
```
<<Java Class>>
OutputFilterDecorator
OutputFilterDecorator(Filter)
decorate(ImageEntity):ImageEntity
```

```
<<Java Class>>
TaskGang<E>
TaskGang()
getInput():List<E>
setInput(List<E>):List<E>
setExecutor(Executor):void
getExecutor():Executor
incrementCycle():long
currentCycle():long
getNextInput():List<E>
initiateHook(int):void
initiateTaskGang(int):void
advanceTaskToNextCycle():boolean
awaitTasksDone():void
taskDone(int):void
processInput(E):boolean
run():void
makeTask(int):Runnable
```

```
<<Java Class>>
Image
ImageEntity(URL,byte[])
ImageEntity(URL,Image)
setImage(byte[]):void
getImage():Image
setSourceURL(URL):void
getSourceURL():URL
setFilterName(Filter):void
getFilterName():String
setSucceeded(boolean):void
getSucceeded():boolean
getFileName():String
getFormatName():String
```

Overview of Pattern-Oriented Framework Solution

- "Gang-of-Four" & POSA patterns enhance the framework-based solution



<<Java Class>>
Filter
- Filter()
- Filter(String)
- *applyFilter(ImageEntity):ImageEntity*
- filter(ImageEntity):ImageEntity
- setName(String):void
- getName():String

<<Java Class>>
ImageTaskGang
- ImageTaskGang(Filter[],Iterator<List<URL>>,Runnable)
- getNextInput():List<URL>
- initiateTaskGang(int):void
- processInput(URL):boolean
- taskDone(int):void
- awaitTasksDone():void
- concurrentlyProcessFilteredResults(int):void

<<Java Class>>
ExecutorCompletionService<V>
- ExecutorCompletionService(Executor)
- submit(Callable<V>):Future<V>
- submit(Runnable,V):Future<V>
- take():Future<V>
- poll():Future<V>
- poll(long,TimeUnit):Future<V>

<<Java Class>>
FilterDecorator
- FilterDecorator(Filter)
- *decorate(ImageEntity):ImageEntity*
- applyFilter(ImageEntity):ImageEntity

<<Java Class>>
NullFilter
- NullFilter()
- NullFilter(String)
- applyFilter(ImageEntity):ImageEntity

<<Java Class>>
GrayScaleFilter
- GrayScaleFilter()
- GrayScaleFilter(String)
- applyFilter(ImageEntity):ImageEntity

<<Java Class>>
OutputFilterDecorator
- OutputFilterDecorator(Filter)
- decorate(ImageEntity):ImageEntity

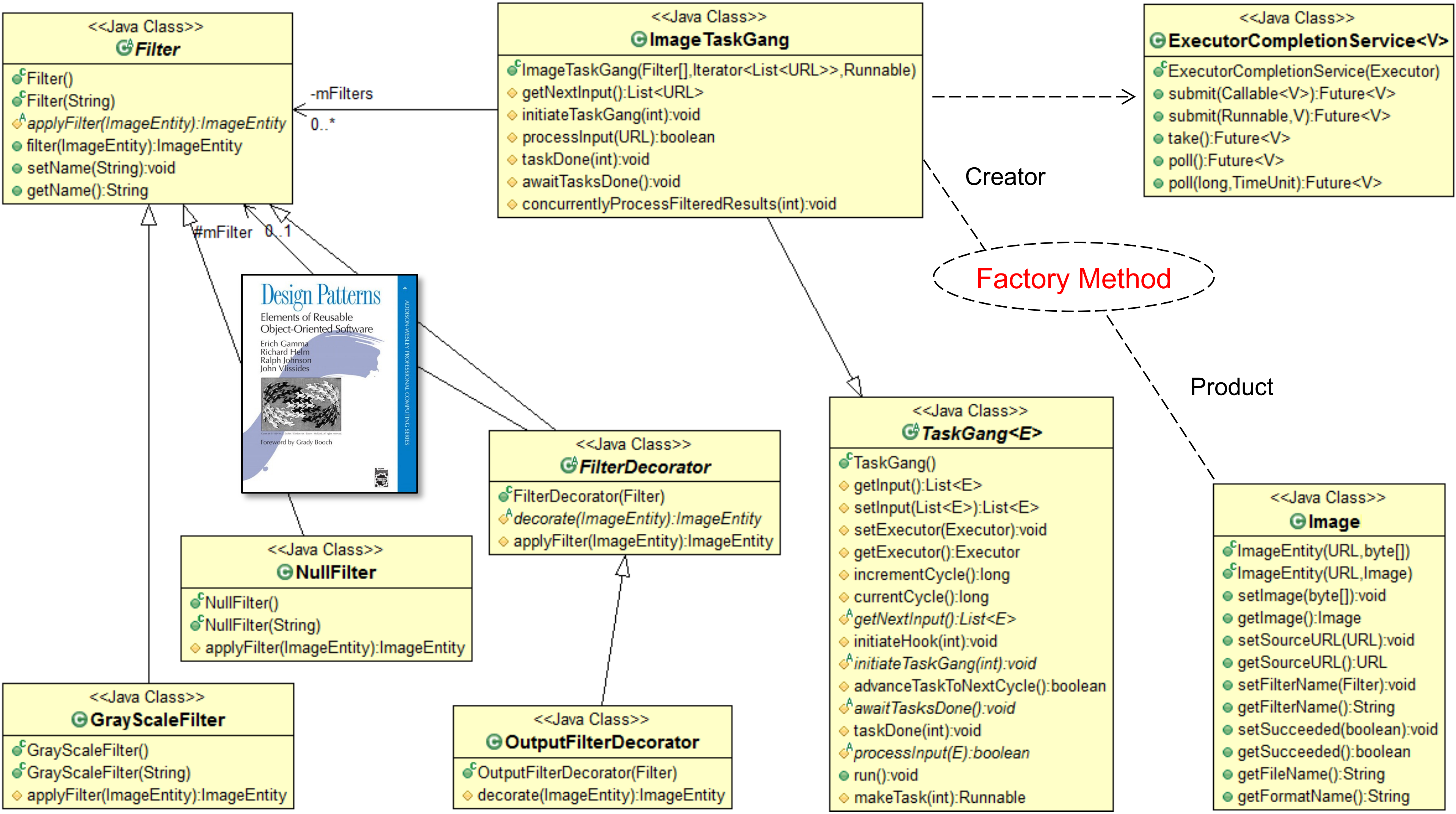
<<Java Class>>
TaskGang<E>
- TaskGang()
- getInput():List<E>
- setInput(List<E>):List<E>
- setExecutor(Executor):void
- getExecutor():Executor
- incrementCycle():long
- currentCycle():long
- *getNextInput():List<E>*
- initiateHook(int):void
- *initiateTaskGang(int):void*
- advanceTaskToNextCycle():boolean
- *awaitTasksDone():void*
- taskDone(int):void
- *processInput(E):boolean*
- run():void
- makeTask(int):Runnable

<<Java Class>>
Image
- ImageEntity(URL,byte[])
- ImageEntity(URL,Image)
- setImage(byte[]):void
- getImage():Image
- setSourceURL(URL):void
- getSourceURL():URL
- setFilterName(Filter):void
- getFilterName():String
- setSucceeded(boolean):void
- getSucceeded():boolean
- getFileName():String
- getFormatName():String

Factory Method defines an interface for creating objects, but let subclasses to decide which class to instantiate

Overview of Pattern-Oriented Framework Solution

- "Gang-of-Four" & POSA patterns enhance the framework-based solution



<<Java Class>>
Filter
- Filter()
- Filter(String)
- *applyFilter(ImageEntity):ImageEntity*
- filter(ImageEntity):ImageEntity
- setName(String):void
- getName():String

<<Java Class>>
ImageTaskGang
- ImageTaskGang(Filter[],Iterator<List<URL>>,Runnable)
- getNextInput():List<URL>
- initiateTaskGang(int):void
- processInput(URL):boolean
- taskDone(int):void
- awaitTasksDone():void
- concurrentlyProcessFilteredResults(int):void

<<Java Class>>
ExecutorCompletionService<V>
- ExecutorCompletionService(Executor)
- submit(Callable<V>):Future<V>
- submit(Runnable,V):Future<V>
- take():Future<V>
- poll():Future<V>
- poll(long,TimeUnit):Future<V>

<<Java Class>>
FilterDecorator
- FilterDecorator(Filter)
- *decorate(ImageEntity):ImageEntity*
- applyFilter(ImageEntity):ImageEntity

<<Java Class>>
NullFilter
- NullFilter()
- NullFilter(String)
- applyFilter(ImageEntity):ImageEntity

<<Java Class>>
GrayScaleFilter
- GrayScaleFilter()
- GrayScaleFilter(String)
- applyFilter(ImageEntity):ImageEntity

<<Java Class>>
OutputFilterDecorator
- OutputFilterDecorator(Filter)
- decorate(ImageEntity):ImageEntity

<<Java Class>>
TaskGang<E>
- TaskGang()
- getInput():List<E>
- setInput(List<E>):List<E>
- setExecutor(Executor):void
- getExecutor():Executor
- incrementCycle():long
- currentCycle():long
- *getNextInput():List<E>*
- initiateHook(int):void
- *initiateTaskGang(int):void*
- advanceTaskToNextCycle():boolean
- *awaitTasksDone():void*
- taskDone(int):void
- *processInput(E):boolean*
- run():void
- makeTask(int):Runnable

<<Java Class>>
Image
- ImageEntity(URL,byte[])
- ImageEntity(URL,Image)
- setImage(byte[]):void
- getImage():Image
- setSourceURL(URL):void
- getSourceURL():URL
- setFilterName(Filter):void
- getFilterName():String
- setSucceeded(boolean):void
- getSucceeded():boolean
- getFileName():String
- getFormatName():String

See en.wikipedia.org/wiki/Factory_method_pattern

End of Pattern-Oriented Software Architecture of the ImageTaskGang Application