TECHNISCHE UNIVERSITÄT DRESDEN

Faculty of Computer Science Institute of Software and Multimedia Technology, Software Technology Group

WS2019/20 – Design Patterns and Frameworks

Frameworks

Professor: Prof. Dr. Uwe Aßmann
Lectuer: Dr.-Ing. Sebastian Götz
Tutor: Dr. rer. nat. Marvin Triebel

Task 1 Framework Hook Fundamentals

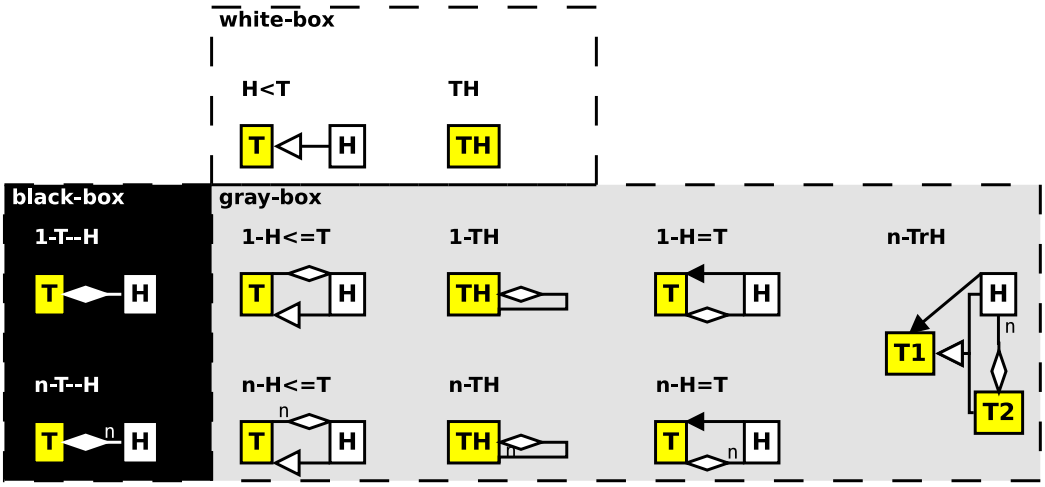
This exercise focuses on the *framework hook patterns* introduced by Pree [1] and extended in the lecture.

a) Enumerate the framework hook patterns introduced in the lecture?

Solution: The following figure (cf. Task 1b) enumerates the different framework hook patterns discussed in the lecture.

b) Classify these framework hook patterns with respect to whether they foster *black-box*, *gray-box*, or *white-box* reuse?

Solution: In general, all design patterns, where hooks are only bound by means of *inheritance*, only permit *white-box* reuse. In contrast, design patterns, which bind hooks solely by means of *delegation*, fully facilitate *black-box* reuse. Besides that, design patterns that employ both *inheritance* and *delegation* to bind hooks, are denoted to permit *gray-box* reuse. Applying this simple classification to the framework hook patterns yields the following figure.

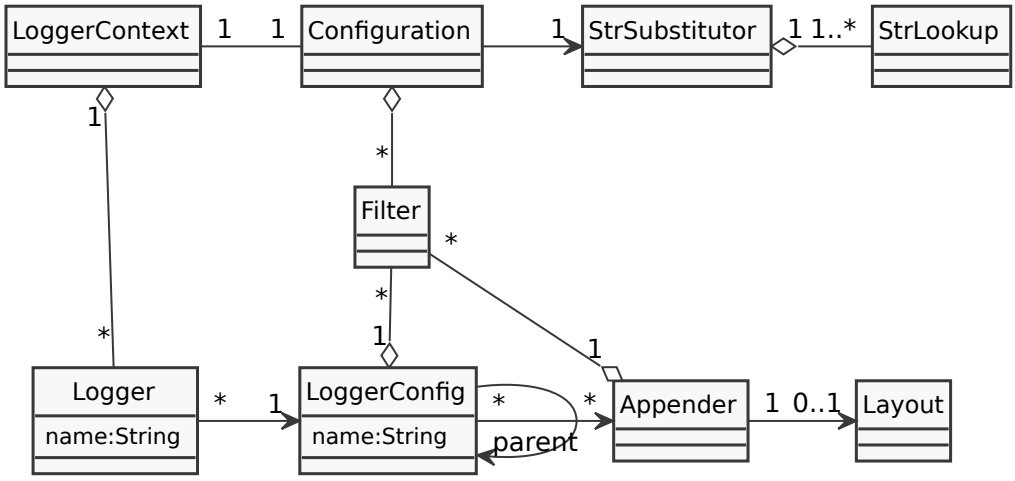


Task 2 The Log4J Framework

Log4J is a *Java*-based logging framework supporting powerful logging statements.[1] The framework's architecture[2] revolves around different kinds of *Loggers* that use multiple *Appenders* to *Layout* logging messages.[3]

a) Investigate the relation between the interfaces `Appender` and `Layout`. Which framework hook pattern can be identified?

Solution: The following simplified class diagram can be retrieved from the Log4J documentation:



Apparently, there is a *T–H* pattern with `Appender` as the template and `Layout` as the hook.

b) Investigate the relation between the interfaces `Logger` and `Appender`. Which framework hook pattern can be identified?

Solution: There is a reference from `Logger` to `LoggerConfig`, whereas there is another *many-to-many* reference from `LoggerConfig` to `Appender`. Consequently, we can identify a *n-T–H* pattern from `Logger` as template to `Appender` as hook, which is established via the `LoggerConfig`.

c) Following the identified hooks: Is *Log4J* rather a *black-box* or a *white-box* framework?

Solution: As the framework hook patterns in *Log4J* are either *T–H* or *n-T–H*, it can be classified as a *black-box* framework.

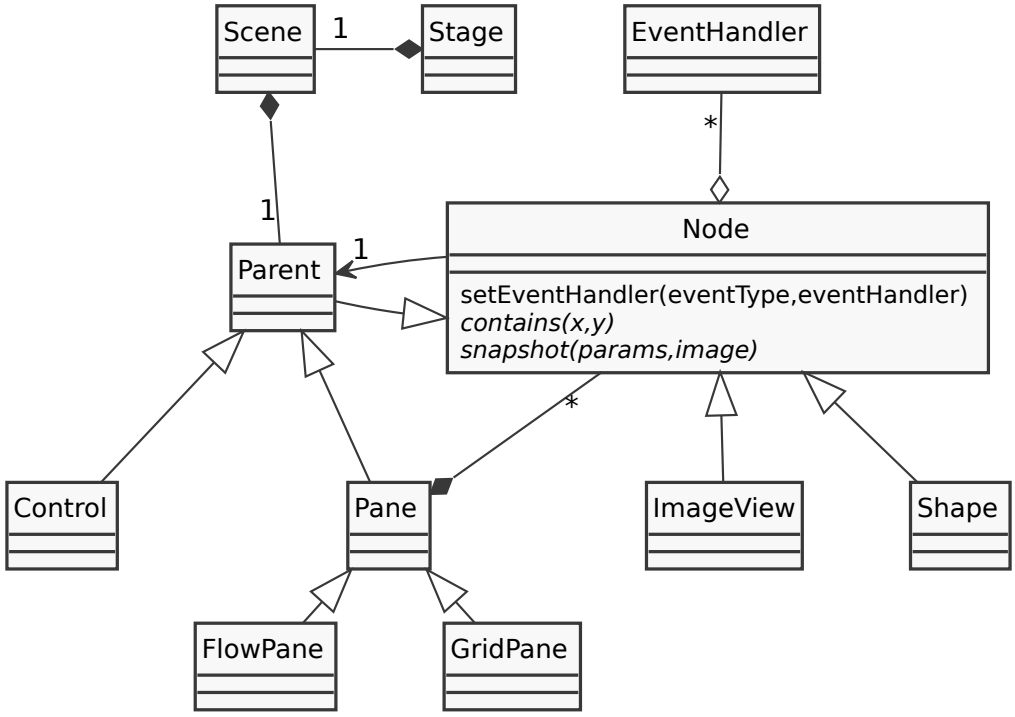
[1] http://logging.apache.org/log4j
[2] https://logging.apache.org/log4j/2.0/manual/architecture.html
[3] https://logging.apache.org/log4j/2.0/log4j-core/apidocs/index.html

Task 3 The JavaFX Framework

The *JavaFX* framework[4] [2] is a state-of-the-art framework for the development of interactive user interfaces supporting both classic 2D applications as well as 3D applications. The core concept of this framework revolves around scenes modeling the user interface, events describing user interactions, and effects transforming elements in a scene.

a) Look at the core classes in `javafx.scene` and their relationships. Identify at least one framework hook pattern?

Solution: The following class diagram, sketches a portion of the `javafx.scene` package, which highlights the `Composite` design pattern employed to create complex scenes.

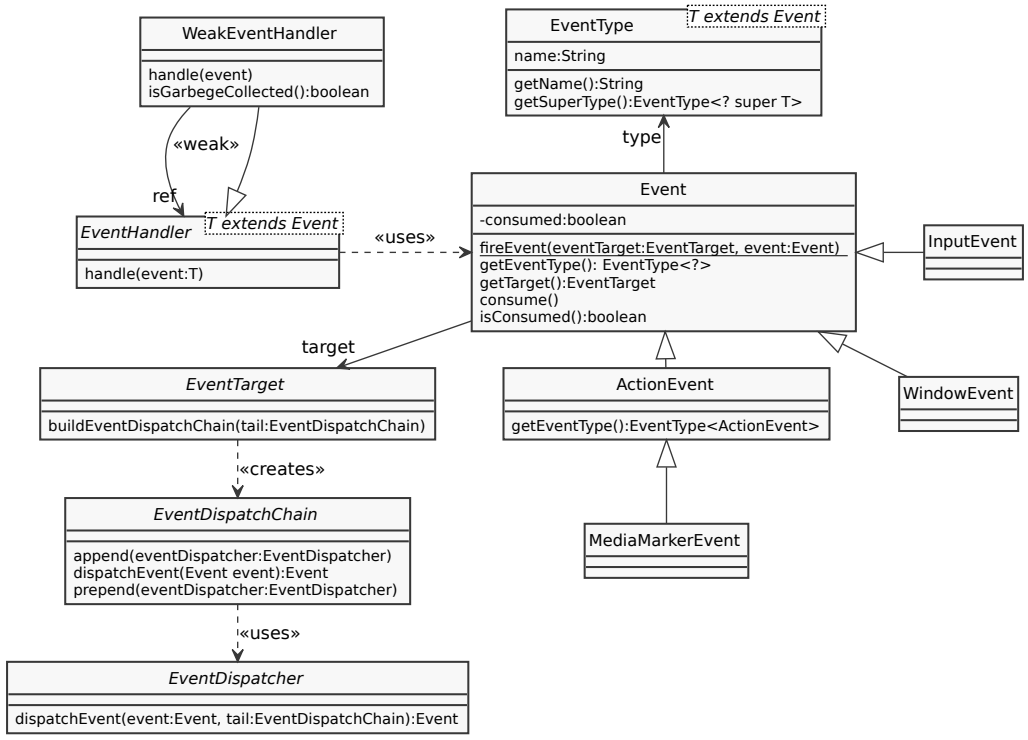


Conversely, the prevalent framework hook pattern is the n-H<=T pattern, whereas the `Pane` is the template and the `Node` is the hook. Besides that, there is a relationship between `Node` and `EventHandler`, which is a n-T–H pattern.

b) Look at the core classes in `javafx.event` and their relationships. Identify at least one framework hook pattern?

Solution: In contrast to `javafx.scenes`, the following class diagram highlights the core classes and interfaces of the `javafx.event` package.

[4] https://docs.oracle.com/javase/8/javafx/api/overview-summary.html



First of all, the `Proxy` pattern between the `EventHandler` and the `WeakEventHandler` yields the obvious $H<=T$ framework hook pattern. Additionally, the `EventDispatchChain` can be considered as an $n\text{-}TH$ framework hook pattern.

c) Following the identified hooks: Is *JavaFX* rather a *black-box* or a *white-box* framework?

Solution: As the framework hook patterns in *JavaFX* are mostly $H<=T$ or $n\text{-}H<=T$, it should be classified as a *gray-box* framework.

References

[1] Wolfgang Pree. Essential framework design patterns. *Framework*, 2:1–7. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.28.5510&rank=1`.

[2] Kim Topley. *JavaFX Developer's Guide*. Pearson Education, 2010.