Lucerne University of
Applied Sciences and Arts

HOCHSCHULE
LUZERN

[*Team* 1]

[*Christopher Christensen*] [*Valentin Bürgler*] [*Lukas Arnold*] [*Melvin Werthmüller*]

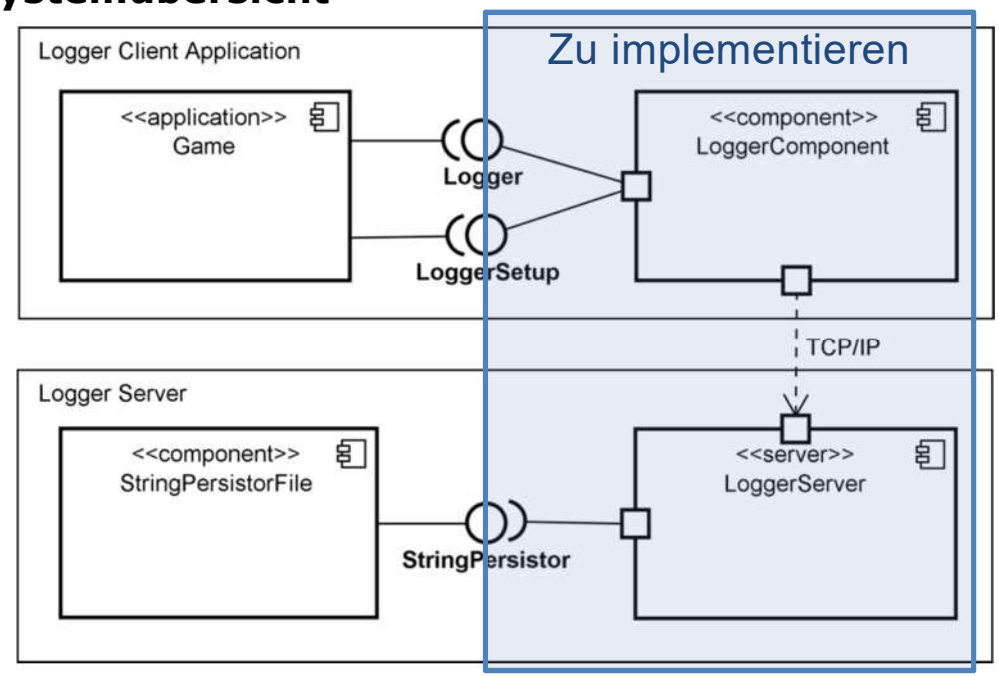
Message Logger

System-Spezifikation

Versionen:

Rev.	Datum	Autor	Bemerkungen	Status
0.1	24.10.2017	Valentin Bürgler	1. Entwurf	Fertig
0.2	31.10.2017	Christopher Christensen	2. Erweiterung der Systemübersicht 3. Architektur und Designentscheide erweitert	Fertig
0.3	01.11.2017	Valentin Bürgler	Kapitel 2 überarbeitet und erweitert Kapitel 4 erzeugt	Fertig
0.4	03.11.2017	Valentin Bürgler	Kapitel 1 erweitert Diagramme hinzugefügt Kapitel 3 überarbietet 2.3.3 Konfigurationsdatei hinzugefügt	Fertig
0.5	05.11.2017	Christopher Christensen	Überarbeitung für Zwischenabgabe	Fertig

1. Systemübersicht



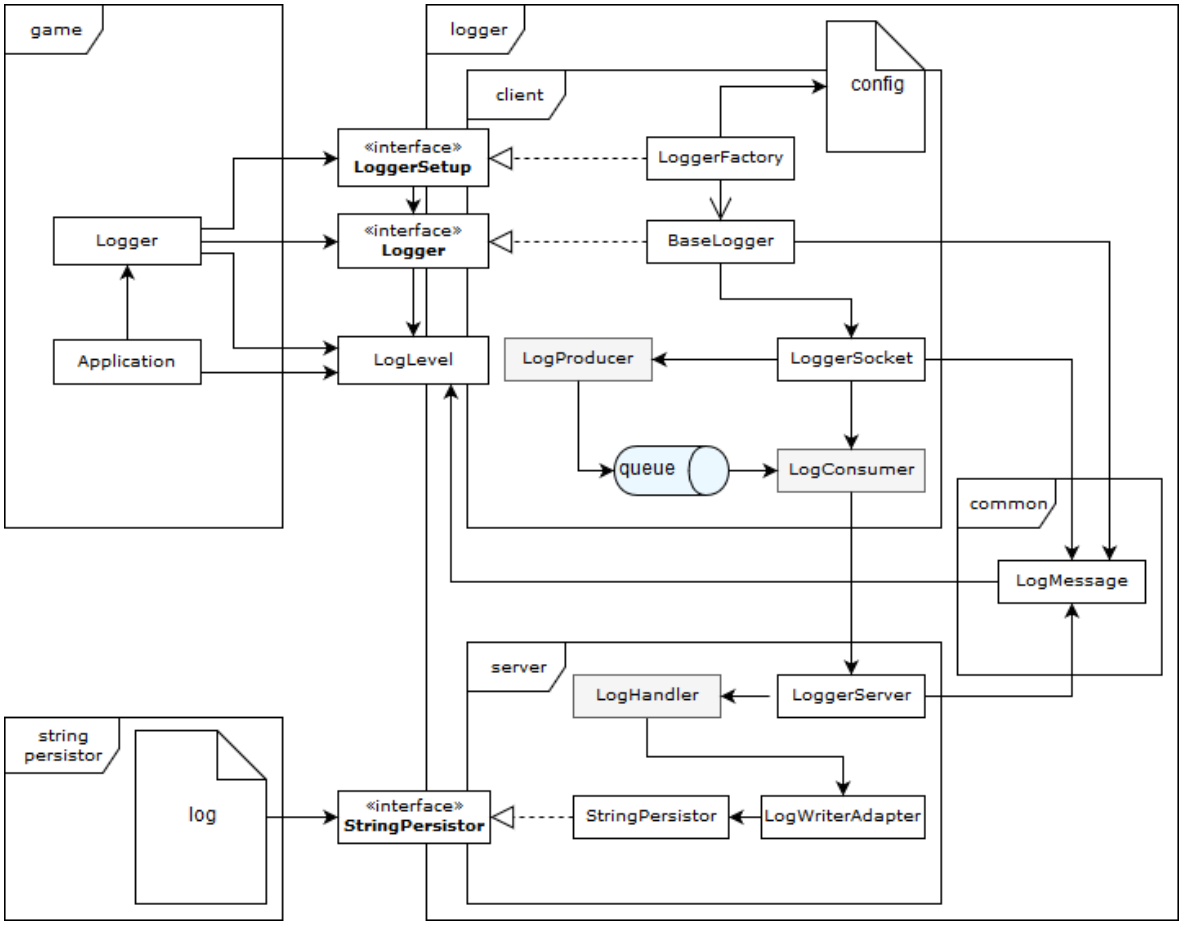
Es soll eine Logger-Komponente implementiert werden, die eingebunden in eine bestehende Java-Applikation über Methodenaufrufe Meldungen aufzeichnet, welche dann per TCP/IP an einen Logger-Server gesendet werden, wo sie in einem wohldefinierten Format gespeichert werden.

Sinnvolle Ereignisse und Situationen, die geloggt werden müssen, sind zu definieren und die entsprechenden Aufrufe in der Java-Applikation zu integrieren.

Die durch ein Interface-Team definierten LogLevels sind sinnvoll und konsistent zu nutzen. Weiter sind die vorgegebenen Schnittstellen Logger, LoggerSetup und StringPersistor einzuhalten. Es müssen sich mehrere Clients mit einem Server verbinden können.

Im späteren Verlauf des Projektes kommen weitere Anforderungen hinzu.

Das folgende UML soll eine Übersicht über das implementierte System schaffen:

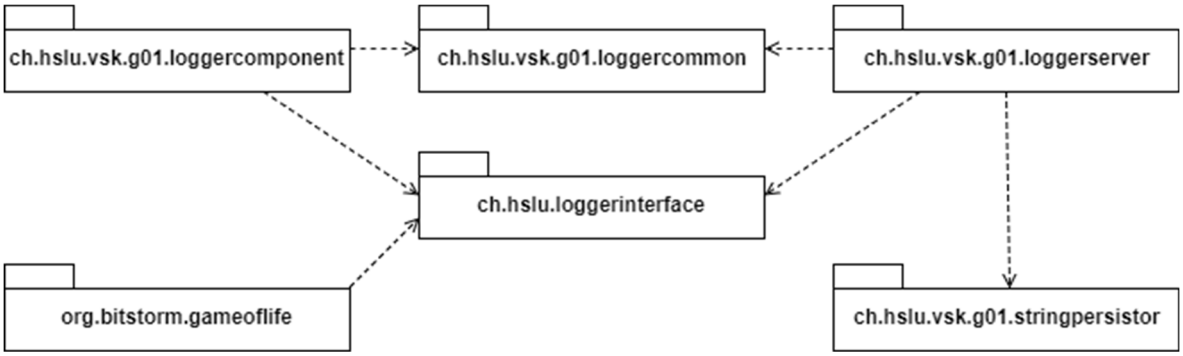


Im beiliegenden Dokument **DokumentationMessageLogger.pdf** werden die einzelnen Komponenten detaillierter beschrieben. Auch die Relationen untereinander werden ausführlich aufgezeigt.

3. Architektur und Designentscheide

3.1. Modell(e) und Sichten

3.1.1. Paketdiagramm



3.2. Entwurfsentscheide

Wir haben generell über das Projekt hinweg versucht uns an den Clean-Code-Prinzipien zu orientieren. Wir versuchten Vererbung zu vermeiden und das «Favour Composition over Inheritance»-Prinzip zu verfolgen. Dazu strebten wir an die Wiederverwendbarkeit zu erhöhen indem wir das DRY-Prinzip vor Augen hielten und die einzelnen Komponenten so zu gestalten, dass sie nur jeweils eine Aufgabe erfüllen (Seperation of Concerns).

3.2.1. Strategie-Pattern, Singleton-Pattern und Fabrikmethode

Die Verbindung zwischen dem Spiel und der Logger-Komponente wurde nach dem Strategie-Pattern umgesetzt. Der Klient ist das Spiel. Den Kontext bildet die zusätzlich eingefügte Klasse Logger im Spiel-Package. Die Strategie wird von der Instanz-Variabel «instance» im Kontext in Form eines Singletons vom Interface-Typ Logger gehalten. Beim Ausführen der start-Methode wird die Strategie mit einer Fabrikmethode instanziiert.

3.2.2. Adapter-Pattern

Für die Übertragung der LogMessage vom LogHandler zum StringPersistor, welcher danach die LogMessage in ein File schreibt, verwenden wir das Adapter-Modell. Damit verletzen wir die Wiederverwendbarkeit des StringPersistor nicht und können eine angepasste Implementation für den LogHandler erstellen. Damit erhalten wir die effektiv gewünschte Zielschnittstelle.

3.2.3. Konfigurationsdatei

Um die Logger-Komponente austauschbar zu implementieren wurde eine Konfigurationsdatei config.properties eingeführt. Diese enthält den Fully Qualified Class Name der LoggerFactory, die IP Adresse des Logger Servers und die Portnummer über den kommuniziert werden soll. Die Datei wird während der start-Methode der Klasse Logger im Spiel-Package eingelesen.

4. Schnittstellen

4.1. Externe Schnittstellen

Folgende Schnittstellen wurden vorgeschrieben:

- Logger
- LoggerSetup
- LogLevel
- LogMessage
- StringPersistor

4.2. Wichtige interne Schnittstellen

Folgende Schnittstellen wurden von uns vorgeschrieben:

- LogWriterAdapter
- config.properties
- TCP/IP Schnittstelle

Die Schnittstellen werden alle im beiliegenden Dokument **DokumentationMessageLogger.pdf** ausführlich beschrieben.

5. Environment-Anforderungen

- Die Logger-Komponente ist mit Java 1.8.0 realisiert. Es gelten die entsprechenden System-Anforderungen für Java 1.8.0.
- Der Fully-Qualified Class Name der LoggerFactory, die IP Adresse und die Portnummer des Servers müssen in einer Konfigurationsdatei «config.properties» vorliegen, um eine beliebige Logger-Komponente eines anderen Teams ohne Anpassungen im Code an das Spiel zu koppeln.
- Eine Internetverbindung wird benötigt, um die Nachrichten an den Server zu senden.