**TGM**

**S06 - Mustergültige Zusammenarbeit**

**Thomas Taschner, Michael Weinberger 4AHITT 2014/15, am 18.12.2014**

Inhaltsverzeichnis

[Aufgabenstellung 2](#_Toc406702025)

[Beschreibung auf Moodle 2](#_Toc406702026)

[Designüberlegung 2](#_Toc406702027)

[Erster Ansatz 2](#_Toc406702028)

[Konkrete Idee 2](#_Toc406702029)

[Umsetzung 3](#_Toc406702030)

[Detaillierte Arbeitsaufteilung (Aufwandsabschätzung, Endzeitaufteilung) 4](#_Toc406702031)

[Aufgabeneinteilung 4](#_Toc406702032)

[Aufwandabschätzung 4](#_Toc406702033)

[Endzeitaufteilung 4](#_Toc406702034)

[Fazit 4](#_Toc406702035)

[Arbeitsdurchführung (Resultate/Niederlagen) 5](#_Toc406702036)

[Resultate 5](#_Toc406702037)

[Niederlagen 5](#_Toc406702038)

[Testbericht 6](#_Toc406702039)

[Beschreibung 6](#_Toc406702040)

[Quellenangaben 7](#_Toc406702041)

# Aufgabenstellung

## Beschreibung auf Moodle

Geben Sie den entsprechenden Beispiel-Code (Java -> jar) und die gewünschte Dokumentation der Patterns (PDF) ab.

Es sind keine Test-Cases verlangt.

# Designüberlegung

## Erster Ansatz

Nach Studieren des UML-Diagramms aus der Angabe, kamen wir zu allererst auf die Idee, dies in Astah umzusetzen. Dies nahm etwas Zeit in Anspruch, stellte aber kein größeres Problem dar.

Bei der Frage der Implementierung waren wir zunächst ratlos, da dies nach sehr viel Aufwand aussah. Wir versuchten uns daran, Design Pattern zu erkennen und einen ersten Plan zu schmieden.

## Konkrete Idee

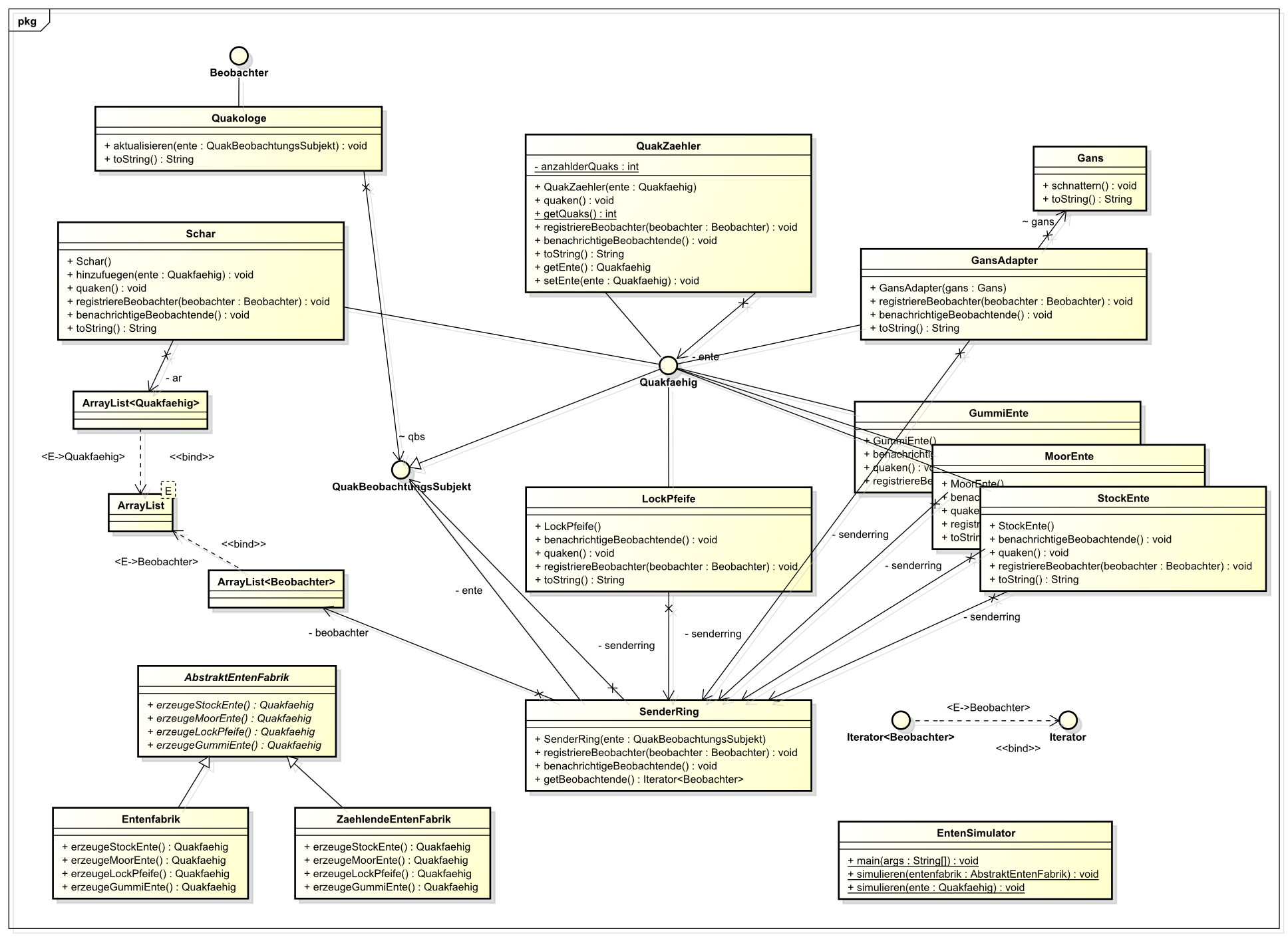
Doch wir bekamen einen entscheidenden Tipp: Uns wurde gesagt, dass es den Code bereits vollständig lauffähig gibt, wenn man nur weiß, wo man suchen müsste. Gesagt, getan. Wie wir später herausfinden würden, hatte uns der Tipp viel wertvolle Zeit erspart.

Beim O’Reilly-Verlag [1], der u.a. das vielzitierte Buch ‚Head First Design Patterns ‘ vertreibt wurden wir fündig. Der Verlag hat das bereits umgesetzte Projekt aus der Angabe gratis zur Verfügung gestellt.

Nach einer kurzen Suche konnte auch das richtige Verzeichnis mit allen 17 Klassen aufgefunden werden.

So reichte es in die Files reinzuschauen, um die Aufgabe lösen zu können.

## Umsetzung



Dieses UML-Diagramm wurde per Hand erstellt, sowie wurden den Klassen Dokumentationskommentare eingefügt.

# Detaillierte Arbeitsaufteilung (Aufwandsabschätzung, Endzeitaufteilung)

## Aufgabeneinteilung

|  |  |
| --- | --- |
| UML nachmalen | Michael Weinberger |
| Implementierung | Thomas Taschner, Michael Weinberger |
| Deployment | Thomas Taschner, Michael Weinberger |
| Testen | Thomas Taschner |

## Aufwandabschätzung

Auf den ersten Blick: einige Arbeitstage

## Endzeitaufteilung

|  |  |  |
| --- | --- | --- |
| Task | Voraussichtlich benötigt | Tatsächlich benötigt |
| UML nachmalen | 30 Min | 2 h |
| AbstraktEntenFabrik | 30 Min | 1 min |
| Entenfabrik | 30 Min |
| Enten | 30 Min |
| Gans | 30 Min |
| GansAdapter | 30 Min |
| GummiEnte | 30 Min |
| LockPfeife | 30 Min |
| MoorEnte | 30 Min |
| Schar | 30 Min |
| SenderRing | 30 Min |
| StockEnte | 30 Min |
| Quakologe | 30 Min |
| QuakZaehler | 30 Min |
| ZaehlendeEntenFabrik | 30 Min |
| Dokumentation | 1 h | 1 h |
| Testen | 1 h | 2 h |
| Gesamt | 9h 30 Min | 5h 1 Min |

## Fazit

Mit der richtigen Herangehensweise konnte das Beispiel noch am selben Tag komplett fertiggestellt werden. Die Testcases folgten dann am anderen…

# Arbeitsdurchführung (Resultate/Niederlagen)

## Resultate

*Erkennen Sie die verwendeten Muster!*

* Observer Pattern
* Factory Pattern
* Adapter Pattern
* Strategy Pattern
* Abstract Factory Pattern
* Decorator Pattern (?)

*Wann und wo wurden die Muster eingesetzt?*

* Factory Pattern, Abstract Factory: Entenfabrik
* Observer Pattern: Wurde ausgiebig verwendet, zum Registrieren wenn Enten gequakt haben
* Adapter Pattern: Gans, die sich als Ente ausgibt
* Strategy Pattern: Verschiedene Entenarten

*Erkläre die verwendeten Muster (kleiner Tipp, es sind deren sechs!)*

Siehe: S05 – Design Principles

## Niederlagen

\n verursachte Assert Fails, warum auch immer, wurde durch *System.lineSeparator()* behoben.

# Testbericht

## Beschreibung

|  |
| --- |
| Entensimulator: mit Observer  Quak  Quakologe: Moorente hat gerade gequakt.  Kwaak  Quakologe: Lockpfeife hat gerade gequakt.  Quietsch  Quakologe: Gummiente hat gerade gequakt.  Schnatter  Quakologe: sich als Ente ausgebende Gans hat gerade gequakt.  Quak  Quakologe: Stockente hat gerade gequakt.  Quak  Quakologe: Stockente hat gerade gequakt.  Quak  Quakologe: Stockente hat gerade gequakt.  Quak  Quakologe: Stockente hat gerade gequakt.  Die Enten haben 7-mal gequakt. |

Programm lässt sich einwandfrei ausführen, alle Ausgaben korrekt.

Die Testcoverage beträgt 100%. Das Testen von quaken() in Schar gestaltete sich ursprünglich etwas schwierig.

# Quellenangaben

[1]: http://examples.oreilly.de/german\_examples/hfdesignpatger/, O’Reilly-Verlag, oreilly.de zuletzt aufgerufen am 11.12.2014