

Quality Assurance

King of Jawa

Isabel Geissmann, Jannik Jaberg

18. Mai 2018



KING OF JAWA

1 Overview

Oberstes Ziel unseres Projekts ist unsere Idee von King of Jawa in ein funktionierendes und flüssig laufendes Spiel umzusetzen. Um dies zu garantieren steht das Merkmal der Funktionalität über allen andere. Desweiteren ist die Idee eines Aufbauspiels bis ins unvorstellbare ausdehnbar, deswegen setzen wir uns als zweites Merkmal die Änderbarkeit. Das Kommunikationsprotokoll ist dem bereits zuvorgekommen und lässt uns neue Packages und Kommunikationsfunktionen mittels relativ einfacher Implementierung dem Spiel hinzufügen ohne dabei den bereits entwickelten Code gross abändern zu müssen.

2 Funktionalität

2.1 Richtigkeit

Um die Richtigkeit von King of Jawa zu garantieren setzen wir uns als Ziel keine ‘ungenauen’ Berechnungen zu machen. Weiter sollen Rechenschritte in ihre eigenen Methoden ausgelagert werden um zu vermeiden dass lange Berechnungen den Aufnahmefähigkeiten eines Developers zu sprengen. Messbar machen wir dies mittels dem Mantis-Bugtracker, welchen Pascal auf seiner Website aufgestellt hat. Uns ist es wichtig dass wir jeden Bug bis zu seinem gesetzten Due-Date fixen oder zumindest bearbeiten. Bei gravierenden Bugs wollen wir höchstens 2 Tage warten bis zum Fix.

2.2 Angemessenheit

Weiter ist die Angemessenheit ein wichtiger Punkt der Funktionalität. Da manchmal eine Abschätzung, ob ein Algorithmus sinnvoll oder nicht ist, ist für uns zu diesem Zeitpunkt noch schwierig, wollen wir dennoch dies als Nebenziel setzen, die Implementierten Algorithmen und/oder Datenstrukturen zu reflektieren und zu entscheiden, ob diese im entsprechenden Konsens angemessen sind. Was jedoch sinnvoll, machbar und konsequent messbar ist, ist Spaghetti-Code zu vermeiden. Deshalb haben wir uns dafür entschieden Methoden mit über 50 Zeilen in weitere Untermethoden abzukapseln. Dies sollte Verständlichkeit und Übersichtlichkeit stark fördern. Die Ausnahme bestätigt die Regel: Klar ist dies nicht zu 100% umsetzbar - es gibt immer Ausnahmen wie beispielsweise `run()` Methoden, welche trotz über 50 Zeilen immer noch übersichtlich sein können. Deshalb setzen wir uns als Mass ungefähr 90% der Methoden 50 Zeilen nicht überschreiten dürfen.

2.3 Sicherheit

Um Spass am Spiel zu garantieren ist die Sicherheit ein weiterer Kernpunkt den QAs. Messbar ist dies nicht wirklich, jedoch probieren wir allfällige Sicherheitslücken im Code, die durch Injection exploited werden können zu minimieren. Beispielsweise wird bei einer `removeRequest` für eine Lobby dir Socket des Owners und der Socket des Clients, der

die Request gesendet hat, verglichen um sicherzugehen, dass nicht ein anderer Client Lobbies schliessen kann.

3 Änderbarkeit

3.1 Modifizierbarkeit

Wie bereits im Overview erwähnt ist die Spielidee von King of Jawa beliebig erweiterbar. Deswegen setzen wir uns als weiteres Kernmerkmal die Modifizierbarkeit. Dem haben wir mit unserem Kommunikationsprotokoll bereits einen fundamentalen Grundbaustein gesetzt. Mit Hilfe eines entity-component-system erreichen wir dies. Messbar ist es ziemlich einfach, nämlich wir haben ein Entity-Component-System oder nicht. Das System wurde bis jetzt schon drei Mal refactored. Mittlerweile ist es ziemlich einfach neue Gebäude zu implementieren und bestehende Gebäude abzuändern. Dafür hat sich Nikolai ein Balancing-System überlegt und implementiert, dies unterstützt das Entity-Component-System extrem.

3.2 Analysierbarkeit

Für die Analyse implementieren wir einen Bugtracker, wie bereits im Punkt 2.1 Richtigkeit beschrieben und einen Debugmode, für Client und Server. Mittlerweile haben wir sogar ein ErrorLog File welches hilft Fehler zu analysieren und auszumertzen.

4 Rahmenbedingungen

4.1 JavaDoc

Der Code wird fortlaufend mit JavaDoc kommentiert. Alle Methoden ausser getter und setter müssen eine JavaDoc Beschreibung haben.

4.2 Style Conventions

Der Code Style orientiert sich am Google Java Style Guide. Wir haben die xml-Datei in IntelliJ und Gradle eingebunden, damit der Code einfach formatieren und überprüfen werden kann.

4.3 Name Conventions

Klassen: CamelCase

Methoden: camelCase

Variablen: camelCase

Paketnamen: klein

Enum: GROSS, Wörter mit __ trennen

4.4 Unit Test

Der Code wird fortlaufend mit Unit Tests überprüft. Im Konzept einigten wir uns darauf, dass jede Klasse die mehr als 100 Zeilen hat, mit Unit Tests überprüft werden muss. Alle Methoden dieser Klassen müssen mit Unit Tests abgedeckt werden. Nach dem Tutorat wurde uns jedoch klar, dass dies nicht umsetzbar ist. Daraufhin überlegten wir uns eine neue Messung. Entschieden haben wir uns dafür, dass wir wichtige Klassen testen. Dazu gehören:

- Chain
- Serialization
- Package
- PackageManager
- User
- Session

4.5 Bugs

Alle Bugs die auftreten werden in unserem Bugtracker festgehalten und gemäss Beschreibung im Punkt 2.1 Richtigkeit bearbeitet.

4.6 Playtesting

King of Jawa wird durch eine Testgruppe von 10 Personen getestet. Dafür wird ein Debug Modus implementiert. Dies gibt uns die Möglichkeit die Fehler einfacher nachzustellen um sie danach zu beheben.

4.7 Log4J

Um Error Logs einheitlich anzuzeigen und abzuspeichern verwenden wir Log4J. Da Log4J die am weitesten verbreitete Logging-Library ist, fiel uns die Entscheidung relativ einfach.

4.8 CI

Auf Grund eines Bonuspunktes hat sich Pascal entschieden das CI Feature auf Git zu benutzen. Mit Hilfe von Jan Schönholz hat er die Pipelines aufgestzt sodass nach jedem Commit der Build, die JavaDocs, die Tests und unser Style-Check direkt getestet wird.