# Softwarequalitätssicherungskonzept JDogs

Im Folgenden werden einige Massnahmen beschrieben, wie wir in Zukunft sicherstellen wollen, dass sich unsere Software unseren Vorstellungen entsprechend verhält und so wenig fehlerhaftes Verhalten wie möglich zeigt.

## Konstruktives Qualitätsmanagement

* Wir treffen uns zusätzlich zur Übungsstunde jede Woche am Dienstag um 15:30 Uhr und am Freitag um 10 Uhr, um erledigte sowie anstehende Arbeiten zu besprechen. Wir erscheinen pünktlich zu den vereinbarten Treffen. Bei kurzfristigen Änderungen informieren wir im Gruppenchat auf WhatsApp.
* Wir verwenden ­– wie verlangt – Javadoc, um unseren Code zu dokumentieren. Dabei möchten wir uns an «Design by Contract» halten.
* Exceptions sollten, wenn möglich, sofort behandelt werden, oder es sollte zumindest vermerkt werden, von wem diese Fehlermeldung behandelt werden sollte.
* Wir verwenden einen WhatsApp-Chat, um uns über den Stand der Dinge auf dem Laufenden zu halten und um Probleme mitzuteilen.
* Wir besprechen die Anforderungen, die wir an unser Spiel stellen, detaillierter als bisher und halten diese in Checklisten fest, welche schliesslich auch zur Überprüfung der Software dienen. Damit es zu so wenig Missverständnissen wie möglich kommt, ist es zentral, dass die unterschiedlichen Vorstellungen kommuniziert werden und wir uns auf eine Version einigen, bevor wir mit der Umsetzung von Details beginnen.

## Analytisches Qualitätsmanagement

* Wenn eine Person eine grössere Arbeit erledigt hat, erklärt sie den anderen ihren Code, sodass danach alle mit diesem Code weiterarbeiten können (Structured Walkthrough). Dazu ist es notwendig, dass zuvor die Aufgaben klar verteilt werden und allen klar ist, wer was macht.
* Zusätzlich führen wir Code Reviews durch, wenn eine Komponente fertig gestellt wurde. Dabei stützen wir uns auf die Checkliste aus der Vorlesung (siehe Anhang). Auch dazu ist es wichtig, dass wir im Vornherein festlegen, wann eine Komponente soweit fertig ist, dass sie überprüft werden kann, bevor sie weiterentwickelt wird.
* Um den Black-Box- und den White-Box-Test durchzuführen, helfen die Listen mit den Anforderungen, die zuvor erstellt wurden. Zudem überlegt sich die zuständige Person entsprechende Äquivalenzklassen und erstellt eine Checkliste, die von den anderen Gruppenmitgliedern ggf. ergänzt werden kann. Der Black-Box- und der White-Box-Test sollte jeweils von einer Person durchgeführt werden, die den zu testenden Code nicht geschrieben hat.
* Bugs möchten wir in Zukunft auf Github mittels des Issue Trackers den anderen mitteilen. Dabei sollte der gefundene Fehler so genau wie möglich beschrieben werden (Voraussetzungen, Schritte zur Reproduktion, Fehlermeldungen, etc.).
* Wir wollen JUnit-Tests als Hilfsmittel benutzen, um korrekt funktionierenden und wartbaren Code zu schreiben.
* Zudem möchten wir Pair Programming (auch zu dritt) bei Gelegenheit einsetzen.
* Die Teammitglieder sollten ihren Code regelmässig einem Quietschentchen o.ä. erklären, um selbst auf allfällige Fehler aufmerksam zu werden.

## Messungen

Wir wollen folgende Messungen durchführen:

* Anzahl Logging-Statements mit Logger Apache Log4j 2
* Code Coverage für Tests mit Jacoco
* Folgende 3 Metrics (min, max, average) mit Metrics Reloaded
  + Lines of Code pro Klasse
  + Lines of Javadoc / Lines of Code pro Klasse
  + Lines of Code pro Methode

# Anhang

## Checkliste Code Review

1. Verstehe ich diesen Code?
2. Hält sich der Code an den vereinbarten Programmierstil?
3. Würde ich diesen Code an dieser Stelle im Projekt erwarten?
4. Hat es bereits Code mit ähnlicher Funktionalität an anderer Stelle?
5. Lässt sich die Lesbarkeit dieses Codes erhöhen?
6. Könne ich diesen Code warten und Änderungen vornehmen?
7. Ist der Code ausreichend dokumentiert?

## Checkliste Milestone3

1. Lobby
   1. Spielstatistik kann aufgerufen werden
   2. Andere Spieler werden angezeigt
   3. Public Chat
   4. WhisperChat
2. Spiel starten
   1. Ein Spiel eröffnen
   2. Einem Spiel beitreten
   3. Ein Spiel beginnen
   4. Public Chat aktiv
3. Spielen (Allgemein)
   1. Reihenfolge der Spieler wird eingehalten
   2. Spieler erhalten entsprechende Anzahl Karten
   3. Spieler erhalten vier Murmeln auf HomeTiles
   4. Wenn die Spieler keine Karten mehr haben, erhalten sie neue
   5. WhisperChat deaktiviert
4. Spielzug
   1. Spieler kann gewünschte Karte aus seiner Hand wählen
   2. Spieler kann nur Aktionen durchführen, die mit der gespielten Karte erlaubt sind
   3. Heimschicken funktioniert
5. Spiel beendet
   1. Der Gewinner wird korrekt ermittelt
   2. Spieler gelangen zurück in die Lobby