Source Code (SC)

Source Code Fachcoach ist Herr Robert von Burg. Bitte kontaktieren Sie ihm nur per Mail: robert.vonburg@fhnw.ch

Eine weitere Ansprechperson für die Programmierung ist Frank del Porte. (Präsentation Microteaching: JavaOnRaspberryPi_FHNW.pdf)

Die Files zum Starten der Entwicklung finden Sie unter Informationen zum technischen Setup

Verwenden Sie ein Template Projekt mit dem MVC-Pattern: https://github.com/Pi4J/pi4j-template-javafx

Source Code Qualität

Um die Qualität von Source Code zu erhöhen, sollten alle Personen das gleiche Verständnis haben, wie Code Qualität gemessen und verbessert werden kann. Nachfolgend ist eine Auflistung von Kriterien die im Projekt als Richtlinien erfasst werden sollten. Diese können abgeändert werden, aber jedes Team soll die Kriterien definieren, dokumentieren und entsprechend prüfen dass sie eingehalten werden.

- Einhaltung branchenüblicher Code Conventions:
  - Namenskonventionen: Packagenamen sind immer klein, Klassen sind CamelCase, Variablen sind drinkingCamelCase, Konstanten und Enums sind in UPPER_CASE.
  - 1 Tabulator/4 Leerschläge Einrückung pro Block, Leerzeichen wie hier beschrieben.
  - Keine Mehrfachdeklarationen pro Zeile.
  - Öffnende Klammer auf der vorherigen Zeile, schliessende auf eine neue Zeile.
  - Zeilen sind max. 120 Zeichen lang
  - Verkettungen sind umgebrochen auf neue Zeilen mit 8 Leerzeichen Einrückung
  - In der IDE können viele dieser Punkte konfiguriert werden (z.B.: https://www.jetbrains.com/help/idea/code-style.html), und exportiert werden. Diese können dann jedem Teamitglied verteilt werden.
  - Weiterführende Links:
    - https://javabeginners.de/Grundlagen/Code-Konventionen.php
    - https://www.oracle.com/java/technologies/javase/codeconventions-namingconventions.html
- Vokabular: Alle Personen die Code schreiben, sollten die gleichen Wörter für die Benennung von Variablen, Klassen, etc. verwenden. So kann Code besser gelesen und entsprechend verstanden werden.
- Code Komplexität, hierin enthalten sind z.B. folgende Metriken:
  - Klassengrösse, z.B. nicht grösser als 300 Zeilen
  - Methodengrösse z.B. kleiner als 25 Zeilen
  - Verschachtelungstiefe (zyklomatische Komplexität) z.B. max. 3
  - Diese Metriken können mittels Tools im GitLab automatisiert geprüft werden. (https://docs.gitlab.com/ee/ci/testing/code_quality.html)
- Dokumentation: JavaDoc und etwaige Kommentare in den Codeblöcken. Generell sollte gelten:
  - Jede Klasse hat min. einen Klassenheader, welcher die Zuständigkeit der Klasse beschreibt
- Organisation: Die Software ist in fachlichen Packages organisiert
- Trennung von Zuständigkeiten (Separation of concerns)
  - Jede Klasse/Methode hat eine Aufgabe

Source Code Verwaltung

Übersicht

Damit Teams an Source Code arbeiten können, braucht es eine Dateiverwaltung. Diese Dateiverwaltung hat folgende Hauptanforderungen:

- Datensicherheit Kein gegenseitiges Überschreiben, aber auch Sicherheit bei der Übertragung über Datenträger und Netzwerk
- Nachvollziehbarkeit Wer hat was wann geändert, und vor allem: Warum?
- Versionierbarkeit Welche Version ist aktuell, Welche Funktionen sind in welcher Version
- Zusammenführbarkeit von Versionen Strukturiertes Vorgehen zum Verschmelzen von verschiedene Versionen von Source Code

Es gibt zwei Arten von Source Code Verwaltung, oder VCS, Version Control System:

- zentrales VCS ein Server als SSOT, single source of truth, oder einziger Punkt der Wahrheit
  - Nur Server hat alle Daten wie Historie, etc.
  - Es muss immer online gearbeitet werden
  - Beispiel: SVN / Subversion
- dezentrale VCS jeder Klon beinhaltet alle Daten und kann zum SSOT werden
  - Somit kann lokal die Historie betrachtet werden
  - es kann offline gearbeitet werden
  - Beispiel: GIT

Terminologie

- VCS Version Control System Versionierungs Kontrollsystem
- SCM Source Code Management Quellcode Verwaltung
- repository Ablage für Quellcode
- clone Das Klonen eines Repositories auf das lokale Gerät
- commit Das Speichern von Änderungen mit einer Nachricht und somit erstellen einer neuen Version des Source Codes

- branch Eine Verzweigung in der Versionierungshistorie des Source Codes. Wird verwendet für das Arbeiten an neuen Funktionen ohne bestehende zu stören
- merge Das zusammenführen von Branches. Ein Branch mit einer neuen Funktion wird z.B. wieder auf den Hauptbranch zusammengeführt.
- rebase Ein Rebase ist auich ein Merge, jedoch sieht die Versionierungshistorie anders aus.
- Pull Requst Ein Pull Request ist die Aufforderung ein Branch in ein anderer Branch zusammenzuführen mittels Merge

GIT

Was ist Git

- garantiert dass jedes gespeicherte Byte später genauso wieder gelesen werden kann
- gewährleistet dass Daten nicht überschrieben werden können (Merging, Rebasing)
- hat eine dezentrale Struktur, so sind alle Daten immer lokal

Wozu Git

- kann zurückverfolgt werden, wann, was und von wem geändert wurde
- kann lokal gearbeitet werden und später wieder die Daten zentral zur Verfügung gestellt werden

Wie mit Git

- Auf Branches/Äste arbeiten
- `main` Branch wird benutzt zum Freigeben der Daten für die Verwendung
- Arbeitsweise: Neuer Branch Commits Pull Request Merge auf `main` / Branch wieder entfernen
- Commits haben aussagekräftige Nachrichten
- Commits sind eher klein und in sich geschlossen
- Mit Pull Requests wird Feedback eingeholt
- **Für aussenstehende ist nur der `main` Branch interessant, andere Branches werden ignoriert**

Links:

- https://git-scm.com/docs/
- https://nvie.com/posts/a-successful-git-branching-model/
- https://www.jetbrains.com/help/idea/configuring-code-style.html (Coding conventions in IntelliJ hinterlegen)

LinkedIn Learning

Unter dem folgenden Link sehen Sie wie sie zu den LinkedIn-Tutorials (ehemals Lynda, ehemals Video2Brain) kommen: https://www.linkedin.com/learning

- GIT Grundkurse Deutsch: https://www.linkedin.com/learning/git-grundkurs/willkommen-zu-git-grundkurs?u=50817177
- GIT Basics English: https://www.linkedin.com/learning/git-essential-training-the-basics/use-git-version-control-software-to-manage-project-code?u=50817177
- GitLab Basics English: https://www.linkedin.com/learning/learning-gitlab-2/version-control-and-more?u=50817177
- Code Qualität: https://www.linkedin.com/learning/grundlagen-der-programmierung-clean-code-und-solid/willkommen-im-projektalltag-eines-software-architekten?contextUrn=urn%3Ali%3AlyndaLearningPath%3A60d198db498ebb58b4f92db1&u=50817177
- Code Conventions:
    - https://javabeginners.de/Grundlagen/Code-Konventionen.php
    - https://www.oracle.com/java/technologies/javase/codeconventions-namingconventions.html