Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

Prof. Dr. Sascha Alda
Robert Hartmann (Dipl.-Inform.)

27. Oktober 2021

Übungen zur Vorlesung Software Engineering I
WS 2021 / 2022

Übungsblatt Nr. 4
(**Achtung:** Abgabe bis: Mittwoch, den 10.11.2021, **08:00 Uhr)**

Aufgabe 1 (Entwicklung von User Stories, Mind Map 5 Punkte):

Der Präsident der Hochschule ist an uns herangetreten und hat uns (ihnen!) einen Projektauftrag überreicht. Das Ziel soll es sein, ein Kollaborationsportal für Unternehmen der Region sowie Studierenden der HBRS zu entwickeln (Projekt Coll@HBRS).

Aufgrund des Fachkräftemangels in der Region soll so ein Portal interessierten Unternehmen unterstützen, Studierende für anstehende IT-Projekte *effektiv* zu gewinnen. Studierende, auf der anderen Seite, lernen *in vielerlei Hinsicht* Unternehmen kennen und können durch die gewonnenen Einblicke ihre Kompetenzen sowie ihre Chancen auf dem Arbeitsmarkt verbessern. Das Kollaborationsportal soll auch die aktuellen Rahmenbedingungen rund um die Corona-Pandemie adressieren und für mögliche Probleme bei der Kollaboration zwischen Studierenden und Unternehmen konstruktive Lösungen formulieren.

Ihre Aufgaben:

a.)
Der Präsident wünscht sich zunächst ein Mind Map mit potentiellen Ideen für ein solches Portal aus Studenten- sowie aus Unternehmersicht. Dieses Mind Map sollten sie mit dem freiverfügbaren Tool Freemind entwickeln:

http://freemind.sourceforge.net/wiki/index.php/Download
(Last Access: 24.10.2021)

Ihre Ideen sollten sie mit Hilfe des Mind Map generell strukturieren. Exportieren oder speichern sie ihr Mind Map aus dem Tool und laden sie die Datei als Ergebnis via LEA hoch. Sie sollten auf erster Ebene *mindestens* 5 Ideen generieren und diese, auf zweiter Ebene, noch mal mit je *mindestens* 2-3 Unterideen verfeinern.

b.)
Es soll dann ihre nächste Aufgabe sein, mögliche Anforderungen an dieses Portal durch acht *aussagekräftige* User Stories niederzuschreiben, um die Ziele des Präsidenten zu konkretisieren. Gehen sie wie folgt vor:

a.) Jede User Story sollte die in der Vorlesung eingeführten INVEST-Eigenschaften erfüllen. Jede Story sollte zu einem Epic zugeordnet sein.

b.) Geben sie für jede User Story eine aus ihrer Sicht plausible Mehrwertschätzung sowie eine aus ihrer Sicht denkbare Aufwandsschätzung basierend auf einer Fibonnacci-Zahl (hier ggf. vorarbeiten).

c.) Jede User Story sollte mit verifizierbaren Akzeptanzkriterien versehen sein. Formulieren sie pro User Story mindestens zwei Akzeptanzkriterien.

Hinweis: eine solche Kollaborationsplattform wird in der Vorlesung SE-2 (für BIS, BWI; Prof. Alda, SS 2022) von den Studierenden im Rahmen eines mehrmonatigen Semesterprojekts umgesetzt. Hier leisten sie bereits erste Vorarbeiten dazu.

Aufgabe 2 (Tool zur Verwaltung von Mitarbeitern im Rahmen einer Sprint-Planung, 25 Punkte)

Das Unternehmen SoGutWieK*ein*Plan GmbH beauftragt sie mit der Entwicklung eines Tools zur Planung von Sprints innerhalb von Scrum-Projekten. Ziel soll es sein, die internen und freien Mitarbeiter des Unternehmens u.a. gemäß ihrer Verfügbarkeiten und auf Basis ihrer individuellen Expertisen effektiv auf Sprints zu verplanen. In dieser Aufgabe soll dazu zunächst die für eine Planung notwendige Erfassung und Verwaltung von Mitarbeiter realisiert werden. Die eigentliche Planung erfolgt zu einem späteren Zeitpunkt (u.a. im Mid-Term-Projekt). Als Akteur soll ein Projektleiter fungieren, der Scrum-Projekte übergreifend plant und organisiert.

Für die Verwaltung von Mitarbeitern gibt das Unternehmen folgende User Stories (*US*) vor, die nach einem Interview mit einem Projektleiter entstanden sind:

US 1: „Als Projektleiter möchte ich in einem Eingabedialog die *notwendigen* Daten eines Mitarbeiters eingeben, um dann später eine effektive Planung der Sprints durchführen zu können."

- Folgende Daten sollen dazu berücksichtigt werden: eine *eindeutige* ID des Mitarbeiters, Vor- und Nachname, Rolle im Unternehmen, Abteilung (falls vorhanden), Expertisen (Plural!).
- Eine Eingabe von ungültigen Werten sollte am Beispiel des Nachnamens exemplarisch unterbunden werden (Beispiel: Eingabe einer Zahl)
- Bei der Eingabe von Expertisen können sie sich zunächst auf maximal drei Expertise beschränken, die man pro Mitarbeiter eingeben kann. Pro Expertise soll zudem auch das Expertise-Level (1: Beginner, 2: Experte, 3: Top-Performer) eingegeben werden.

US 2: „Als Projektleiter möchte ich eine Übersicht über die eingegebenen Daten aller Mitarbeiter in *tabellarischer Form* als Ausgabe erhalten können, damit ich einen guten Überblick bekomme, welche Mitarbeiter aktuell für eine Sprint-Planung bereitstehen."

- Die Struktur der Tabelle können sie selber wählen.

US 3: „Als Projektleiter möchte ich, dass die eingegebenen Mitarbeiter *persistent* abgespeichert werden können, so dass ich diese nach einem Neustart des Programms wieder einsehen kann."

US 4: „Als Projektleiter möchte ich nach Expertisen suchen können. Nach der Eingabe einer Expertise als Suchwort möchte ich eine *einfache Übersicht* über die Mitarbeiter erhalten, welche diese Expertise besitzen. Damit ich feststellen, welche Mitarbeiter für spezifische Aufgaben in einem Projekt geeignet sind.

- Die Struktur der „einfachen Übersicht" können sie selber wählen.

Anmerkungen zur Entwicklung:
Weitere Details zu den User Stories, die vom Unternehmen noch genannt wurden:

- Alle Befehle sollen über eine Kommandozeile (Synonyme: „Shell", „Terminal", „Command Line Interface („CLI") „Console"; keine GUI!)) eingegeben werden können. Struktur der Eingabe:

  ```
  > befehl [parameter]
  ```

- Als Befehle sind vorgesehen:
  - `enter` (Eingabe eines Mitarbeiters, nur Ablage in den RAM-Speicher, also in die Klasse `Container` (siehe Hinweis unten)),
  - `store` (Persistentes Abspeichern von Mitarbeiter-Objekte aus einem Container-Objekt auf einen Datenträger (vgl. Übung Nr. 3-2),
  - `load` (Laden von Mitarbeiter-Objekten von einem Datenträger in ein Container-Objekt. Dieser Befehl kann zwei Parameter aufnehmen:
    - `merge` (die geladenen Mitarbeiter-Objekte werden mit den Mitarbeiter-Objekten im Speicher vereinigt)
    - `force` (die geladenen Mitarbeiter-Objekte überschreiben die Mitarbeiter-Objekte im Speicher (siehe dazu die Übung Nr. 3-2))
  - `dump` (eine nach den eingegeben IDs *sortierte* Ausgabe der Mitarbeiter-Objekte inklusive aller eingegeben Angaben (ohne Expertisen)).
  - `search` (Suche nach Expertisen; Suchwort wird dabei als Parameter übergeben; Ausgabe der Mitarbeiter erfolgt in einer einfachen Übersicht)
  - `exit` (Verlassen der Anwendung)
  - `help` (Anzeige aller möglichen Befehle)

- Es soll der Source Code der Klasse Container aus der Übung 2 bzw. 3 zur internen Abspeicherung der Mitarbeiter in dem RAM-Speicher *wiederverwendet* werden. Passen sie dazu die Klasse entsprechend an (hier: Anpassung der internen Liste!). Alternativ können sie eine generische Variante eines Containers

mit Java Generics implementieren. Auch die Klassen zur persistenten Speicherung von Objekten gemäß Strategy Pattern sollten verwendet werden.

- Die Implementierung der Ausgabe („Dump“) der Liste soll *stream*-basiert unter Anwendung des Patterns „Filter Map Reduce“ erfolgen. Siehe dazu auch Beispiele und Erläuterungen in der Musterlösung der Klasse Container in der Aufgabe 2-2 sowie die Folien zu Kapitel 9 (siehe LEA). Hinweis: es müssen nicht alle Verarbeitungsschritte „Filter“, „Map“ oder „Reduce“ zum Einsatz kommen.

Weitere Anforderungen hat die Firma *zunächst* nicht, ist aber gespannt auf einen ersten Java-Prototypen! Insgesamt haben sie viele Freiheitsgrade bei der Entwicklung, worüber sie selber entscheiden (Beispiel: Eingabe eines Mitarbeiters mittels des Befehls „enter“).

Software-Test:
Entwickeln sie eine repräsentative Anzahl von Äquivalenzklassen und Testfällen, um ihre Anwendung als Ganzes hinreichend zu testen. Diese Testfälle sollten manuell durchgeführt werden, eine Test-Automatisierung ist *nicht* erforderlich. Betrachten sie bei der Entwicklung der Testfälle auch die verschiedenen Load-Parameter – insbesondere bei der Merge-Option kommt es auf einen soliden Test an!

Finale Hinweise:
Bitte bereiten sie bis zur nächsten Übungsstunde am 3.11.2021 einen ersten Prototyp vor, der aber nicht auf LEA hochgeladen werden muss. Zur *interaktiven* Klärung von möglichen Rückfragen stehe ich als Product Owner (PO), stellvertretend für den Kunden, gerne zur Verfügung. In der nächsten Woche kommen weitere Anforderungen in Form von User Stories dazu, die sie agil in der Übungsstunde (oder später) implementieren sollten. Die Gesamtabgabe erfolgt somit für diese Übung erst zum 10.11.2021.

Ressourcen und Quellen:

Eine Einführung zu dem Muster Filter-Map-Reduce finden sie hier:
https://www.torsten-horn.de/techdocs/java-lambdas.htm#Filter-Map-Reduce

Eine weitere gute Quelle für das Muster mit sehr guten Beispielen ;-)
http://javatricks.de/tricks/java-8-streams-grundlagen-und-motivation

(Alle Links: Last Access: 26.10.2021)