Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

Prof. Dr. Sascha Alda
und Team

6. November 2023

Übungen zur Vorlesung Software Engineering I
WS 2023 / 2024

Übungsblatt Nr. 4
(Abgabe bis: Mittwoch, den 22.11.2023, **10:00 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 ein derartiges Portal 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.

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 können Sie z.B. mit dem freiverfügbaren Tool Freemind entwickeln:

http://freemind.sourceforge.net/wiki/index.php/Download
(Last Access: 06.11.2023)

Alternativ können Sie das Mind-Map mit dem Tool Mermaid spezifizieren:
https://mermaid.live
(Last Access: 06.11.2023)

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.

Leiten Sie für sich ein Definition-of-Ready ab für ihre Sammlung.

Hinweis: eine solche Kollaborationsplattform wird in der Vorlesung SE-2 (für BWI; Prof. Alda, SS 2024) von den Studierenden im Rahmen eines mehrmonatigen Semesterprojekts umgesetzt. Hier leisten sie bereits erste Vorarbeiten dazu.

Aufgabe 2 (Tool zur Verwaltung von User-Stories, 25 Punkte)

Das Unternehmen WeissNix GmbH möchte von ihnen ein Tool zur Verwaltung von User Stories. Dazu gibt sie folgende User Stories (*US*) vor:

US 1: „Als Entwickler möchte ich in einem Eingabedialog die *notwendigen* Daten zur Eingabe einer User Story eingeben. Das soll mir die Verwaltung von User Stories erleichtern."

- Folgende Daten sollen dazu berücksichtigt werden: eine *eindeutige* ID der User Story, eine kurze schriftliche Beschreibung, ein kurzes Akzeptanzkriterium, sowie die vier Kennzahlen zur Berechnung der Priorisierung gemäß der Formel nach Gloger (Kapitel 3) sowie die Zuordnung zu einem Projekt. Details können sie auslassen.
- Auch der resultierende Priorisierungswert sollte für eine User Story vom System berechnet und abgespeichert werden.
- Eine Eingabe von ungültigen Werten aus der Formel von Gloger (z.B. eine negative Zahl für ein Risiko) sollte unterbunden werden.

US 2: „Als Entwickler möchte ich eine Übersicht über die eingegebenen Daten aller User Stories in *tabellarischer Form* als Ausgabe erhalten können, damit ich einen guten Überblick bekomme, welche User Stories ich umzusetzen habe."

- Die Struktur der Tabelle können sie selber wählen.

US 3: „Alle Eingaben sollen *persistent* abgespeichert werden, so dass ich diese nach

einem Neustart des Programms wieder einsehen kann.“

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

- Alle Befehle sollen über eine Kommandozeile („Shell“, „Terminal“, „Console“; keine GUI!) eingegeben werden können. Struktur der Eingabe:

  ```
  > befehl [parameter]
  ```

- Als Befehle sind vorgesehen:
  - `enter` (Eingabe einer User Story, nur Ablage in den RAM-Speicher, also in die Klasse `Container` (siehe Hinweis unten)),
  - `store` (Persistentes Abspeichern von User Stories aus einem Container-Objekt auf einen Datenträger (vgl. Übung Nr. 3-2),
  - `load` (Laden von User Stories von einem Datenträger in ein Container-Objekt.
  - `dump` (eine nach den berechneten Priorisierungen *sortierte* Ausgabe der User Stories inklusive aller eingegeben Angaben).
  - `search` (Suche nach User Stories nach Projekten; Suchwort (= Bezeichnung des Projektes) wird dabei als Parameter übergeben);
  - `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 User Stories 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 einer User Story mittels des Befehls „enter“).

Software-Test:
Entwickeln sie eine repräsentative Anzahl von Äquivalenzklassen und Testfällen, um ihre Anwendung als Ganzes („end-to-end“) hinreichend zu testen. Diese Testfälle sollten manuell durchgeführt werden, eine Test-Automatisierung ist hier *nicht* erforderlich. Zudem sollten Sie einzelne Klassen und Methoden durch JUNIT-Tests hinreichend testen (vgl. dazu auch schon die Lösungen aus der Aufgabe 3-2 (GitHub!).

Finale Hinweise:
Bitte bereiten sie bis zur nächsten Übungsstunde am 15.11.2023 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 22.11.2023.