Institut für Informatik
Prof. Dr.-Ing. Elke Pulvermüller
Nils Baumgartner, M.Sc.

Universität Osnabrück, 08.01.2020
Testat von 19.12.2022 bis 22.12.2022, 14:00 Uhr

Übungen zu Software Engineering

Wintersemester 2022/23

Blatt 8

Auf diesem Aufgabenblatt sollen Sie diverse Diagramme von Software-Modellierungen zeichnen. Dies können Sie entweder handschriftlich oder mit einem geeigneten Programm erledigen. Laden Sie Ihr Diagramm in einem geeigneten Export-Format (PDF, JPG, PNG) hoch.

Aufgabe 8.1: Programmablaufplan, Struktogramm (25 Punkte)

Das Heron-Verfahren ist eine aus dem Newton-Verfahren abgeleitete Iterationsvorschrift, mit der eine Näherung der Quadratwurzel einer Zahl $a > 0$ berechnet werden kann. Die Iterationsvorschrift ist wie folgt definiert:

$$x_{n+1} = \frac{x_n + \frac{a}{x_n}}{2} \tag{1}$$

Zeichnen Sie sowohl einen Programmablaufplan (PAP) als auch einen Struktogramm für eine Implementation des Heron-Verfahrens. Als Eingabe soll es die positive Zahl a und eine Zahl $\varepsilon \geq 0$ erhalten. Das Verfahren bricht ab, wenn $|x_{n+1} - x_n| \leq \varepsilon$. Zusätzlich dazu soll das Verfahren nach maximal 99000 Iterationsschritten abbrechen. In jedem Fall soll das Programm eine verständliche Ausgabe haben.

Aufgabe 8.2: Strukturierte Analyse (25 Punkte)

Führen Sie eine *Strukturierte Analyse* nach Tom DeMarco ausgehend von einem Datenflussdiagramm durch. Das zu beschreibende System soll die Anmeldung zur Bachelorarbeit wie folgt realisieren:

Ein Student gibt sein Wunschthemengebiet an. Das System sucht aus einer von den Dozenten gefüllten Themen-Datei an offenen Themen auf Basis des angegebenen Themengebietes ein passendes Thema heraus. Auf Basis der persönlichen Daten des Studenten wird die Zulassungsberechtigung geprüft. Bei erfolgreicher Prüfung wird ein Anmeldebogen mit dem ausgewählten Thema und den persönlichen Daten des Studenten erstellt und dem betreuenden Dozenten übermittelt. Sobald dieser den Bogen unterschrieben und ans Prüfungsamt geleitet hat, wird von diesem über das Anmeldesystem der Opium-Eintrag zum Studenten aktualisiert. Sollte die Zulassungsberechtigung nicht erteilt werden, teilt das System dem Studenten den Grund der Ablehnung mit. Bei erfolgreicher Anmeldung teilt das System dem Studenten die Abgabe-Informationen mit.

a) Zeichen Sie ein Datenflussdiagramm (DFD) für das oben beschriebene Anmeldesystem. Achten Sie auf eine geeignete Zusammenfassung von Prozessen und entsprechenden Verfeinerungs-Diagrammen. Gibt es Sachverhalte, bzw. Vorgänge die sich nicht darstellen lassen, wenn ja, welche?

b) Erstellen Sie ein Datenlexikon für das obige DFD.

c) Wählen Sie einen beliebigen Knoten und beschreiben Sie eine Minispec für ihn.

Aufgabe 8.3: Klassendiagramm (20 Punkte)

Auf Aufgabenblatt 1 haben Sie bereits ein Klassendiagramm zum Spiel *Sudoku* entworfen, das die grundsätzliche Struktur der Geschäftslogik widerspiegelt. In dieser Aufgabe sollen Sie darauf aufbauend eine einfache Software mit Benutzeroberfläche (GUI) modellieren, mit der man Sudoku spielen kann. Gehen Sie zunächst davon aus, dass es immer nur von einem Spieler gespielt wird. Als Basis können Sie das untenstehende UML-Diagramm der Geschäftslogik verwenden.

Die Benutzeroberfläche soll aus einem Fenster mit 9x9 Text-Feldern bestehen, in die man Zahlen eintragen kann. Den Eintrag von Feldern, die zu Beginn des Spiels gesetzt wurden, kann man nicht verändern. Wenn eine Zahl eingetragen wird, die die Sudoku-Regeln verletzt, soll das gesamte Fenster rot umrahmt werden. Wenn das letzte freie Feld korrekt belegt wurde, soll das Fenster grün umrahmt werden und keine weitere Eingabe mehr möglich sein. Mit einer Tastenkombination (beispielsweise strg-z) kann man den letzten Zug wieder rückgängig machen.

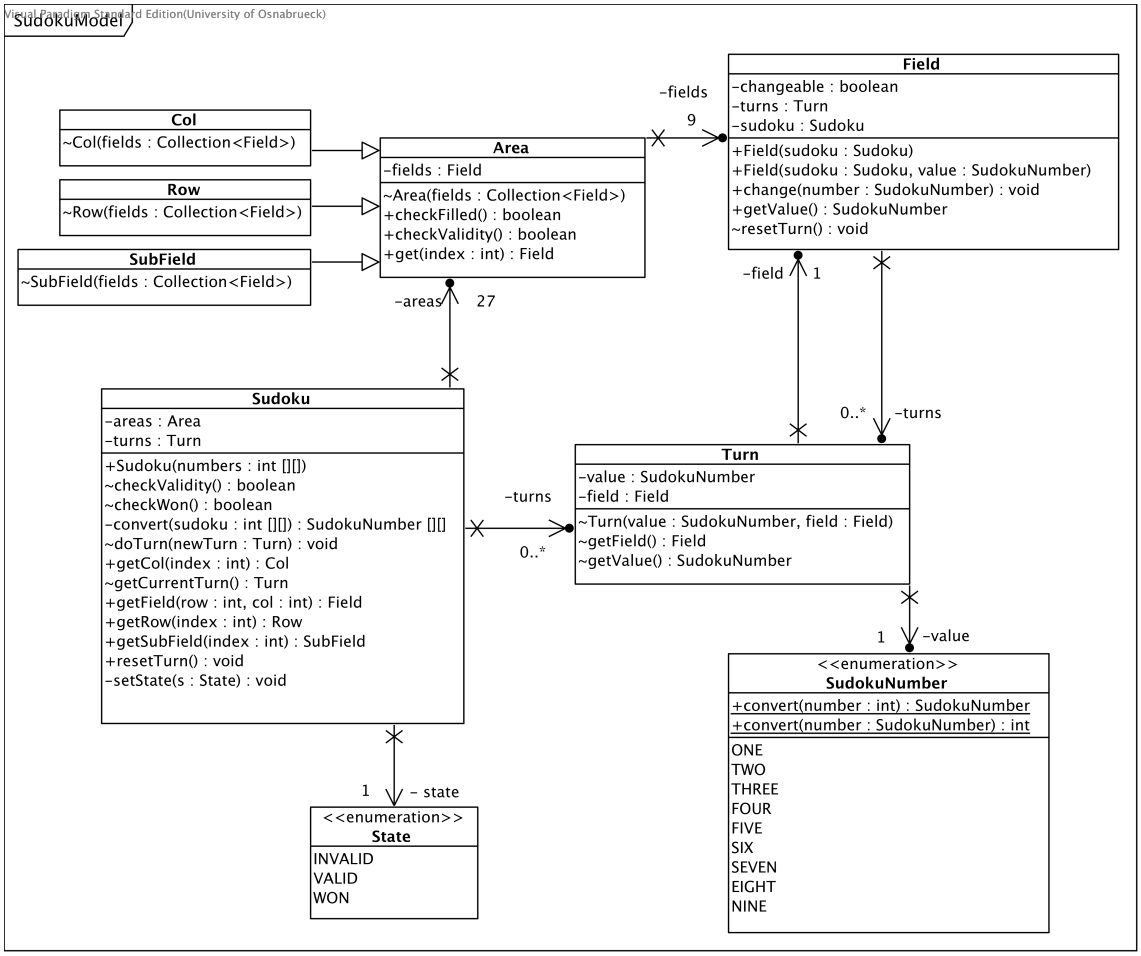

Abbildung 1: Klassendiagramm des Sudoku-Spiels

Die Software soll strikt nach dem Prinzip *Model-View-Controller* unter Verwendung des *Observer*-Design-Patterns aufgebaut werden.

Betrachten Sie zuerst das gegebene *Model* und überlegen Sie, wie Sie *View*- und *Controller*-Elemente

einbinden können. Passen Sie dazu ggf. das bestehende *Model* an, bzw. erweitern Sie es.

Erstellen Sie anschließend die *View* und *Controller*-Elemente und assoziieren Sie diese ggf. untereinander oder mit dem *Model*, wie von den Design-Patterns verlangt. Da Sie Ihre Software in der Programmiersprache Java entwickeln sollen, brauchen Sie nur die Elemente zu spezifizieren, die nicht Teil der Java `swing` und `awt`-Pakete sind und können die Anbindung der Applikation an diese Pakete durch geeignete Kommentare beschreiben. Es soll außerdem deutlich werden, an welchen Stellen das Java-Event-Model genutzt wird.

Hinweis: Wenn Sie Ihre Lösung mit einem Modellierungwerkzeug erzeugen, brauchen Sie das vorgegebene Modell nicht vollständig darin neu zeichnen, sondern können nur die von Ihnen benutzen oder veränderten Teile übernehmen. Machen Sie aber in jedem Fall, beispielsweise durch Kommentare, deutlich, ob Sie die Zeichnung verwenden und ob Teile davon ausgeschlossen werden sollen.

Aufgabe 8.4: Sequenzdiagramm (30 Punkte)

Stellen Sie nun mit Hilfe eines Sequenzdiagrammes dar, wie Ihre Sudoku-Applikation vorgeht, wenn ein neuer Spielzug getätigt wird. Nehmen Sie an, dass der Benutzer auf einem zu Anfang nicht belegten Feld die Zahl 7 einträgt. Im Spiel selber können schon beliebig viele Züge, mindestens jedoch einer, getätigt worden sein.

Stellen Sie dar, wie diese Information über *View* und *Controller* ins *Model* gelangt, wie dort damit umgegangen wird und wie eventuelle Änderungen wieder an die *View* geleitet werden.

Es ist weder sicher, ob der Zug gültig, also das gewählte Feld auf 7 gesetzt werden darf, ohne eine der Sudoku-Regeln zu verletzen, noch, ob der Zug zu einem Sieg und zur Beendigung des Spieles führt. Die Überprüfung, ob der Spielzug korrekt war und ob das Spiel gewonnen wurde, brauchen Sie nicht detailliert zu modellieren. Es reicht aus, wenn Sie die Stellen deutlich machen, an denen die Überprüfungen stattfinden und beispielsweise in einem Kommentar beschreiben, wie diese durchgeführt wird. Wenn die *View* auf spezielle Weise, beispielsweise mit einer Dialog-Nachricht, auf Änderungen des *Models* reagieren soll, reichen dafür auch Kommentare an den entsprechenden Stellen im Sequenzdiagramm.

Achten Sie darauf anzugeben, an welchen Stellen die Attribute der benutzten Instanzen geändert werden. Sie brauchen nur die Instanzen der in Ihrem Klassendiagramm angegebenen Klassen zu modellieren.