Institut für Informatik
Prof. Dr. Elke Pulvermüller
Dipl.-Systemwiss. Mathias Menninghaus

Universität Osnabrück, 11.12.2014
http://www-lehre.inf.uos.de/~swe
Testat bis 18.12.2014, 14:00 Uhr

Übungen zu Software Engineering

Wintersemester 2014/15

Blatt 8

Aufgabe 8.1: 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.

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 8.2: 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.

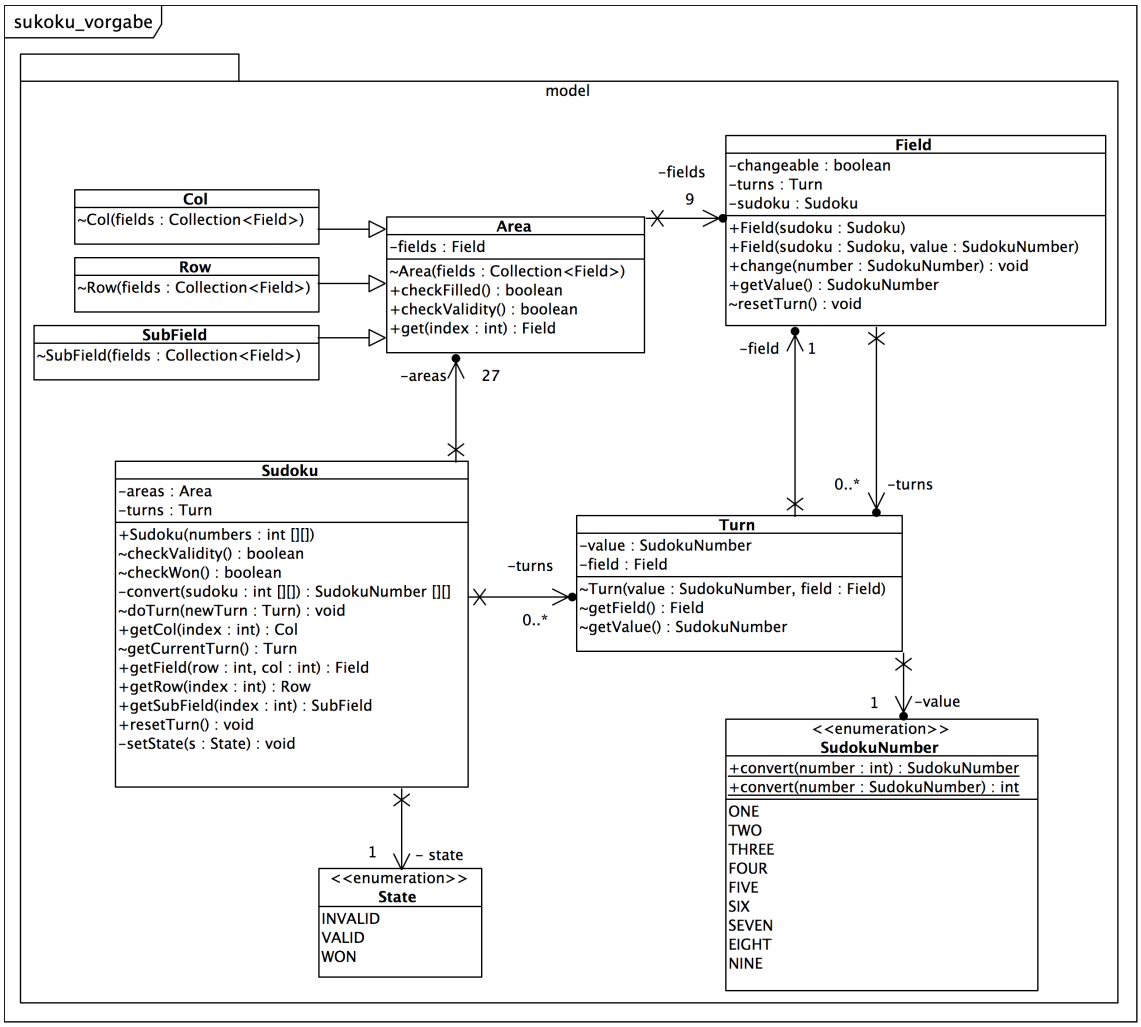

Abbildung 1: Klassendiagramm des Sudoku-Spiels

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.

Aufgabe 8.3: Kommunikationsdiagramm (15 Punkte)

Stellen Sie denselben Ablauf, den Sie in der vorherigen Aufgabe in einem Sequenzdiagramm modelliert haben, als UML-Kommunikationsdiagramm dar.

Aufgabe 8.4: Objektdiagramm (10 Punkte)

Erstellen Sie nach der Modellierung des Sudoku-Spiels aus den vorherigen Aufgaben ein Objektdiagramm, das den Zustand eines beliebigen Sudoku-Spieles nach einem (dem ersten) gültigen Zug widerspiegelt.

Aufgabe 8.5: Zustandsdiagramm (25 Punkte)

Gegeben sei das Anwendungsszenario der Autovermietung vom letzten Aufgabenblatt. Nun stellt sich die Situation wie folgt dar: alle Fahrzeuge des Fuhrparks müssen regelmäßig gewartet werden (Inspektion/Reinigung/Reparatur u.ä.). und können für den Zeitraum der Wartung natürlich nicht vermietet werden. Ruft ein Kunde an oder kommt er vorbei und möchte für den Tag oder sofort ein Fahrzeug mieten, so stehen ihm nur alle sich nicht in Wartung befindenden Fahrzeuge, die aktuell nicht vermietet und für diesen Tag auch nicht reserviert sind, zur Verfügung. Reservierte Fahrzeuge, die bis 1 Stunde nach vereinbartem Termin nicht abgeholt wurden, gelten wieder als nicht reserviert. Fahrzeuge, bei deren Wartung sich herausstellt, dass eine Reparatur zu teuer ist oder sie mehr als 90.000 km gefahren sind, werden abgestoßen. Erstellen Sie ein Zustandsdiagramm für die verschiedenen Zustände bzgl. der Verfügbarkeit von Fahrzeugen. Geben Sie soweit sinnvoll auch Bedingungen für die Zustandsübergänge mit an.