Klassen entwerfen

Einer der schwierigeren Schritte beim Entwickeln eines klassenbasierten Programms ist das Finden von guten Klassen. Was gute Klassen sind, wird am Ende des Arbeitsblattes zum Projekt Minesweeper (AB226A_09-1) erklärt. Im vorliegenden Arbeitsblatt werden Sie nun eine einfache Technik kennenlernen, wie Sie auf Grund von Anforderungen zu einem ersten klassenbasierten Entwurf kommen können.

Ein Bemerkung vorweg: Das Finden von Klassen gelingt mit zunehmender Erfahrung immer besser. Allerdings setzt dies voraus, dass Sie ihre eigene Arbeit immer wieder kritisch hinterfragen und nicht mit der erstbesten Lösung zufrieden sind, sondern diese bei Bedarf überarbeiten.

Aufgabe 1

Der erste Schritt ist, eine Liste mit möglichen Klassen zu erstellen. Gehen Sie dazu durch die Minimalanforderungen an das Spiel Minesweeper (AB226A_09-1) und markieren Sie im Text Substantive, welche eine Bedeutung für das Spiel Minesweeper haben.

Beispiel:

- Benutzerschnittstelle
- Konsole
- Bombe

Nicht alle Substantive werden später zu einer Klasse. Einige werden stattdessen zu Attributen (Instanzvariablen) einer Klasse oder finden gar keine Entsprechung im Programm (z.B. das erste Substantiv in den Minimalanforderungen, „Fall“). Notieren Sie sich nachfolgend diejenigen Substantive, von denen Sie glauben, dass sie im Rahmen von Minesweeper wichtig sind.

- Spielfeld
- Benutzeroberfläche
- Benutzereingabe
- Zelle
- Bombe
- Nachbarzellen

Aufgabe 2

Der nächste Schritt ist, eine erste Auswahl aus der Liste in Aufgabe 1 zu treffen. Wählen Sie drei bis vier Substantive aus, von denen Sie ziemlich sicher sind, dass diese eine Klasse repräsentieren. Hier kommen vor allem die Bestandteile des Spiels in Frage, also „Dinge“, welche während der Ausführung des Programms (zur Laufzeit) als eigenständige Objekte auftreten werden.

Versuchen Sie nun in zwei, drei prägnanten Sätzen die Aufgabe dieser Klassen, man spricht von der Verantwortlichkeit, zu beschreiben.

Beispiel:
Zelle

- Eine Zelle entspricht einem einzelnen Ort auf dem Spielfeld.
- Sie verwaltet ihren Zustand (verdeckt, markiert, aufgedeckt, Bombe explodiert)

K 1.4

Hier müssen wir nun das Single-Responsibilty-Prinzip (SRP) beachten: Jede Klasse sollte nur eine Verantwortlichkeit haben. Wenn wir feststellen, dass zwei, drei Sätze für die Beschreibung nicht ausreichen, haben wir vermutlich das SRP verletzt und müssen die Klasse aufteilen.

Auch hier gilt, mit der Erfahrung werden wir klüger. Also nicht von Anfang an nach einer perfekten Lösung suchen, sondern später zurückkommen und allenfalls die Lösung überarbeiten. Wichtig ist: Keine Verantwortlichkeiten auf Vorrat („das könnte dann mal später gebraucht werden“) einbauen!

Beschreiben Sie nachfolgend Ihre drei bis vier Klassen.

Spielfeld
- Ein Spielfeld entspricht einer Spiel-Instanz
- Auf einem Spielfeld befinden sich mehrere Zellen.
- Ein Spielfeld kann mit Operationen wie Feld aufdecken umgehen

Benutzeroberfläche
- Die Benutzeroberfläche kann Eingaben einlesen und Ausgaben ausgeben.
- Die Benutzeroberfläche entscheidet, was mit Eingaben geschehen soll.

InputProcessor
- Der InputProcessor verarbeitet Eingaben
- Er entscheidet, was damit geschehen soll.

Zelle
- Eine Zelle definiert ein Feld auf dem Spielfeld.
- Eine Zelle kann vom Typ Bombe sein.
- Eine Zelle kann sich aufdecken.
- Eine Zelle kann die umliegenden Zellen über das Aufdecken informieren.

Aufgabe 3

K 1.1 Teil

Jetzt geht es darum, die Eigenschaften der von Ihnen in der Aufgabe 2 beschriebenen Klassen genauer festzulegen. Darunter verstehen wir vorerst die Attribute / Instanzvariablen. Später kommen die Operationen / Methoden hinzu.

Wir beginnen mit den Attributen. Überlegen Sie sich, welche Informationen die Objekte einer Klasse brauchen, um ihre Verantwortlichkeiten wahrzunehmen. Oft hilft auch ein erneuter Blick auf Ihre Liste aus Aufgabe 1. Wenn Sie mit den dort zu findenden Substantiven Sätze der Form „ein <Subtantiv 1>-Objekt *hat* ein <Substantiv 2>“ bilden können, so ist das ein Hinweis auf ein Attribut.

Beispiel:
„Eine Zelle *hat* eine X- und eine Y-Koordinate“. Daraus folgt, dass die Klasse Zelle zwei Attribute mit den Name xKoordinate und yKoordinate enthält. Den Datentyp müssen wir noch festlegen.

Jetzt ist auch der Moment gekommen, ein Klassendiagramm zu eröffnen. Sie können das entweder von Hand oder in Modelio tun. Im ersten Fall empfiehlt es sich das Diagramm nur skizzenhaft und mit Bleistift zu zeichnen, da Sie im Laufe der Entwicklung noch einiges ergänzen oder ändern werden.
Im zweiten Fall haben Sie den Vorteil, dass Sie jederzeit das Diagramm ohne grossen Aufwand ändern können und auch automatisch ein Code-Gerüst in Java erzeugen können. Dafür müssen Sie sich mit einem Tool herumschlagen.

Erstellen Sie nun ein Klassendiagramm mit Ihren drei, vier Klassen und tragen Sie die nötigen Attribute ein.

Aufgabe 4

Nun wenden wir uns den Operationen / Methoden der einzelnen Klassen zu. Zu diesem Zweck gehen wir nochmals in die Minimalanforderungen. Diese Mal suchen wir nach Verben, welche in Zusammenhang mit den in Aufgabe 2 und 3 beschriebenen Klassen stehen. Diese geben uns Hinweise auf mögliche Operationen / Methoden.

Beispiel:
„Der Spieler kann eine Zelle markieren“. Daraus lesen wir ab, dass die Klasse Zelle eine Operation / Methode markieren() anbietet.

Gehen Sie jetzt durch den Text der Minimalanforderungen und suchen Sie wie oben beschrieben nach Operationen / Methoden. Fügen Sie diese in das Klassendiagramm aus Aufgabe 3 ein. Überlegen Sie sich bei jeder Operation / Methode, welche Parameter benötigt werden und nehmen Sie auch diese ins Klassendiagramm auf.

Aufgabe 5

Zum Schluss suchen wir noch nach Beziehungen zwischen den gefundenen Klassen. Beziehungen bestehen überall da, wo ein Objekt eine dauerhafte Referenz auf ein anderes Objekt hat. Das sind typischerweise Instanzvariablen, mit einer Klasse als Typ (wie Sie das in Lektion 7 auch schon gesehen haben).

Beziehungen im klassenbasierten Entwurf findet man wiederum, indem man Sätze in der Form „ein …-Objekt *kennt* ein /mehrere …-Objekte“

Beispiel:
„Das Spielfeld kennt alle Zellen“. Daraus leiten wir her, dass die Klasse Spielfeld eine Beziehung zur Klasse Zelle hat. Wir können sogar noch weiter gehen und sagen, dass *ein* Spielfeldobjekt eine Beziehung zu *vielen* Zellen-Objekten hat.

Suchen Sie nun nach solchen Beziehungen und zeichnen Sie diese in Ihrem Klassendiagramm ein. Bestimmen Sie auch die Multiplizitäten, Navigierbarkeit und Rollennamen. Überprüfen Sie dabei die schon bestehenden Attribute in den einzelnen Klassen. Sollten Sie solche finden, welche eine Klasse aus Ihrem Diagramm als Typ haben, können Sie diese durch eine Beziehung ersetzen.

Aufgabe 6

Jetzt sind Sie soweit, dass Sie Ihren klassenbasierten Entwurf in Java-Code umsetzen können. Wenn Sie mit Modelio gearbeitet haben können Sie den Code in Modelio automatisch generieren und anschliessend in ein neues Eclipse-Projekt importieren. Mehr Informationen dazu finden Sie in den Ressourcen zu Lektion 7.
Wenn Sie das Klassendiagramm von Hand gezeichnet haben, müssen Sie nun dieses von Hand in Java-Code übersetzen. Tun Sie diese in einem neuen Eclipse-Projekt.

Sie erhalten damit noch kein lauffähiges Programm, aber ein erstes Gerüst. Dieses werden Sie in der nächsten Lektion erweitern und zum fertigen Programm ergänzen.

Schlussbemerkung

Das in den Aufgaben 1 bis 6 beschriebene Vorgehen wird in den wenigsten Fällen im ersten Durchgang zu einem perfekten klassenbasierten Entwurf führen. Vielmehr werden Sie feststellen, dass Sie immer wieder auf Entscheidungen, welche Sie in einem früheren Schritt gemacht haben, zurückkommen müssen. Scheuen Sie sich nicht davor!
Wenn man beim Entwerfen an einem bestimmten Punkt nicht mehr weiter kommt, ist es oft besser einen alternativen Weg zu suchen, als einen einmal eingeschlagenen Weg auf Biegen und Brechen durchzuziehen.