Objektorientierte
Softwareentwicklung

- Softwaretechnik -

SoSe 2014

Jähnichen
Duske

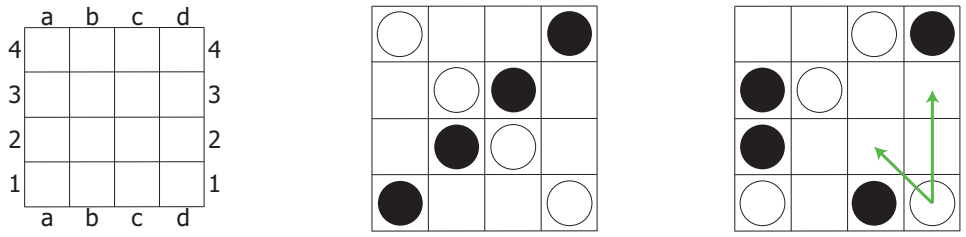
Ausgabe: Mittwoch, 11.06.2014
Abgabe: Freitag, 27.06.2014

3. Übungsblatt: Dao in LOS

Um eine sinnvolle Einsatzmöglichkeit der objektorientierten Programmierung kennen zu lernen, implementieren wir auf Basis der LOS das Brettspiel *Dao*. Dabei handelt es sich um ein Spiel für zwei Personen, das 1999 von Jeff Pickering und Ben van Buskirk entwickelt wurde. [1]
Wir nutzen Klassen und Objekte für die Spiellogik, während Aufzählungstypen für Informationsausgabe und Zugrichtungen verwendet werden.

Spielregeln

Das Dao-Spielbrett besteht aus 4x4 Feldern. Felder können leer oder von einem Stein besetzt sein. Die Benennung der Felder ähnelt derjenigen im Schachspiel, sie ist unten dargestellt.
An einem Dao-Spiel nehmen zwei Spieler teil, einer mit weißen Steinen, einer mit schwarzen. Jeder der Spieler erhält vier Steine, die zu Beginn in einer X-Form liegen, wie hier gezeigt.



Dao-Spielbrett | Startposition | Erlaubte Spielzüge für *d1*

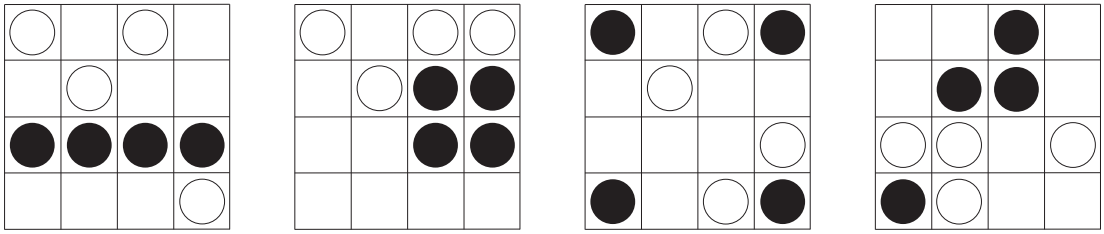
Abbildung 1: Spielbrett- und züge

[1] Dao ist auch in der Theorie der zufallsfreien Zwei-Personen-Spiele mit vollständiger Information interessant, s. z.B. `http://www.fdg.unimaas.nl/educ/donkers/games/dao/`.

Der Spieler mit den weißen Steinen beginnt, die Spieler ziehen abwechselnd. Um einen Zug durchzuführen, wählt der Spieler einen seiner Steine und verschiebt ihn in eine beliebige Richtung (horizontal, vertikal oder diagonal) so weit wie möglich, d.h. bis das nächste Feld von einem anderen Stein belegt ist oder das Spielfeld kein weiteres Feld in dieser Richtung enthält. Der Stein muss um mindestens ein Feld bewegt werden. Das Ändern der Richtung während des Ziehens ist nicht erlaubt, ebenso ist das Überspringen oder Schlagen von Steinen verboten. Nach einem solchen Zug ist der jeweils andere Spieler an der Reihe.

Ein Spieler gewinnt, sobald eine der folgenden Situationen eintritt:

- Die vier Steine des Spielers liegen in einer horizontalen oder vertikalen Linie.
- Die vier Steine des Spielers liegen in einem 2x2 Quadrat.
- Die vier Steine des Spielers liegen in den Ecken des Spielfelds.
- Einer der Steine des Spielers liegt in einer Ecke des Spielfeldes und ist von drei gegnerischen Steinen umgeben.



Horizontale Linie, Schwarz gewinnt | 2x2 Quadrat, Schwarz gewinnt | Ecken besetzt, Schwarz gewinnt | Stein gefangen, Schwarz gewinnt

Abbildung 2: Gewinnsituationen

Die letzte der Gewinnsituationen unterscheidet sich von den anderen dadurch, dass sie durch einen gegnerischen Zug entsteht. Ein Dao-Spiel endet also mit einem Gewinn für einen der Spieler, ein Unentschieden ist nicht möglich.

Implementierung

Die Vorgabe `Game.lua` enthält eine grundlegende Implementierung für Zwei-Personen-Spiele. Eine neue Klasse `Dao` wird nun von `Game` erben. Bereits vorhandene Features der `Game`-Klasse sollen sinnvoll genutzt werden.

Methoden der Klasse `Dao`

Für unsere Parameter und Rückgaben benötigen wir zwei Aufzählungstypen:

- `Direction` mit Werten für alle möglichen Zugrichtungen. Die Werte werden mit Abkürzungen für entsprechende Himmelsrichtungen benannt, `N` (*North*) für einen Zug nach „oben“ (z.B. von b1 auf b3), `NE` (*Northeast*) für einen Zug nach „oben rechts“ usw.
- `Result` als Rückgabewerte für die Methode `makeMove` mit Informationen über den Erfolg des versuchten Zuges. Die Werte sind unten beschrieben.

Der Konstruktor `create` erzeugt ein neues Dao-Spiel, die Funktionalität aus der `Game`-Klasse bleibt erhalten. Zusätzlich erhält der Konstruktor ein drittes Argument, das angibt ob der interaktive Modus ausgeführt wird (s. unten).

Die Methode `getColorAt` gibt die Farbe des Steins zurück, der auf dem übergebenen Feld liegt. Das Feld wird hierbei als String der Form `'c2'` übergeben. Liegt kein Stein auf dem Feld oder existiert das Feld nicht, ist die Rückgabe `NONE` - jeweils als Wert des Aufzählungstyps `PlayerColor`.

Durch Aufruf der Methode `makeMove` wird ein Spielzug ausgeführt. Ein Teil der Funktionalität ist bereits vorgegeben, diese soll insbesondere durch das Überschreiben der Methode `handleMove` erweitert werden. Die Methode erhält als Eingabe ein Feld als String der Form `'c2'` sowie eine *Direction*.
Falls das Spiel bereits beendet ist, wird `gameOver` zurückgegeben. Ist das erste Argument des Aufrufs kein gültiges Feld oder ist das zweite Argument keine *Direction*, ist die Rückgabe `illegalInput`. Liegt auf dem Feld kein Stein oder ein Stein des Spielers, der nicht an der Reihe ist, so wird `noMarble` bzw. `wrongColor` ausgegeben. Liegt dort ein Stein der richtigen Farbe, aber ein Zug in die angegebene Richtung ist nicht möglich, so ist die Rückgabe `noMovement`. In all diesen Fällen, wird keine Änderung des Spiels vorgenommen. Falls keiner dieser Fehler auftritt, wird der Zug gemäß den oben genannten Regeln durchgeführt und `ok` zurückgegeben.

Die Methode `getWinner` gibt den Gewinner des Spiels zurück, falls einer existiert, und `NONE`, falls das Spiel noch nicht beendet ist. Die Rückgaben sind jeweils Werte des Aufzählungstyps `PlayerColor`.

Die Methode `getNextPlayer` informiert über denjenigen Spieler, der den nächsten Zug ausführen muss. Sie gibt `NONE` zurück, falls das Spiel bereits beendet ist.

Ein Aufruf der Methode `printGame` bewirkt eine Ausgabe des aktuellen Spielfelds auf der Konsole in einer geeigneten String-Repräsentation.

Interaktiver Modus

Wird als dritter Parameter im Konstruktor `true` übergeben, so startet das Spiel im interaktiven Modus. Wird ein anderer Wert übergeben, bleibt es bei oben beschriebenem Verhalten.

Im interaktiven Modus wird das Spiel über direkte Eingaben in der Konsole gesteuert:

1. Das aktuelle Spielfeld wird ausgegeben und der Spieler, der den nächsten Zug ausführen muss, wird mit Namen und Farbe benannt.
2. Der Spieler gibt seinen Zug ein, er nutzt dazu die Form 'c2 W', also einen String mit einem Leerzeichen, wobei das erste Wort wie in der Methode `makeMove` das Feld angibt und das zweite Wort der Richtung entspricht.
3. Falls möglich, wird der Zug ausgeführt, andernfalls eine aussagekräftige Fehlermeldung ausgegeben und der Spieler wiederholt seine Eingabe. Bei erfolgreicher Ausführung des gewünschten Zuges wird dieser Ablauf für den anderen Spieler wiederholt.
4. Hat ein Spieler gewonnen, so wird das Spielfeld ausgegeben, der Gewinner über eine Konsolenausgabe bekannt gegeben und der interaktive Modus beendet. Zudem kann der Modus durch Eingabe von `exit` jederzeit beendet werden.

Nach dem Beenden des interaktiven Modus kann mit der Methode `getWinner` der Gewinner abgefragt werden.

Hinweise

- Falls beim Aufruf von `makeMove` mehrere Fehlerfälle vorliegen, wird eine beliebige der entsprechenden Fehlerrückgaben ausgewählt. Diese Wahl soll deterministisch erfolgen.
- Die Methode `handleMove` liefert zwei Rückgabewerte. Falls dies in der LOS-Implementierung noch nicht möglich ist, so soll es jetzt ergänzt werden.
- Für eine geeignete String-Darstellung des Spielfelds ist ein Monospace-Font für die Konsolenausgabe hilfreich.
- Mit der Lua-Funktion `io.read()` sind Konsoleneingaben möglich.
- Ein Beispiel für die Ein- und Ausgaben des interaktiven Modus ist in der Datei `uebung3_test.lua` gegeben.

Bearbeitung und Abgabe

Die Aufgabe soll in Gruppen von maximal 3 Personen bearbeitet werden. Mit der Aufgabe wird ein kleines Testprogramm (`uebung3_test.lua`) bereitgestellt, das natürlich mit der zu erstellenden Dao-Implementierung ausführbar sein muss. *Wir empfehlen, die Dao-Implementierung darüber hinaus mit eigenen Tests zu prüfen. Wir werden das tun!*
Die Abgabe erfolgt im Gruppenforum auf ISIS. Dazu soll ein neues Thema mit dem Betreff *„Abgabe Übung 3“* angelegt werden. Die Dateien werden gepackt in einem einzelnen `zip`-Archiv als Anhang eingefügt. Im Kopf jeder Datei müssen (als Kommentar) Namen und Matrikelnummern aller aktiv mitarbeitenden Gruppenmitglieder stehen.