MNG

Programmieren eines Spiels in Java

Max Mathys
2. Dezember 2015

1 Einleitung

Für dieses Projekt wurde ein Spiel in Java programmiert, bei dem Aktien gehandelt werden können. Das Ziel des Spiels ist es, dass man möglichst viel Geld generiert.

Die Steuerung des Spiels wird durch die Interaktion im Terminal ermöglicht.

Das Ziel dieses Projektes ist die OOP-Kenntnisse in Java selbstständig anzuwenden und so eine praxisorientierte Kenntnis von OOP zu erlangen.

2 Struktur

Die Applikation wurde so programmiert, dass die Prinzipien der Objektorientierung angewendet werden.

Die wichtigsten Klassen sind Abbildung 2.1 dargestellt.

- Ein gestrichelte Linie bedeutet, dass die Klassen die entsprechende Klasse im Code verwenden.
- Es wurden nicht alle Referenzen (welche anderen Klassen eine Klasse verwendet) eingezeichnet, sondern nur die wichtigsten.
- Eine durchgezogene Linie bedeutet, dass die Klasse von der anderen abstammt. Hierbei ist immer die untere Klasse das Kind.
- Statische Funktionen werden unterstrichen.
- Für eine verbesserte Übersicht werden private Funktionen und alle Attribute nicht dargestellt.

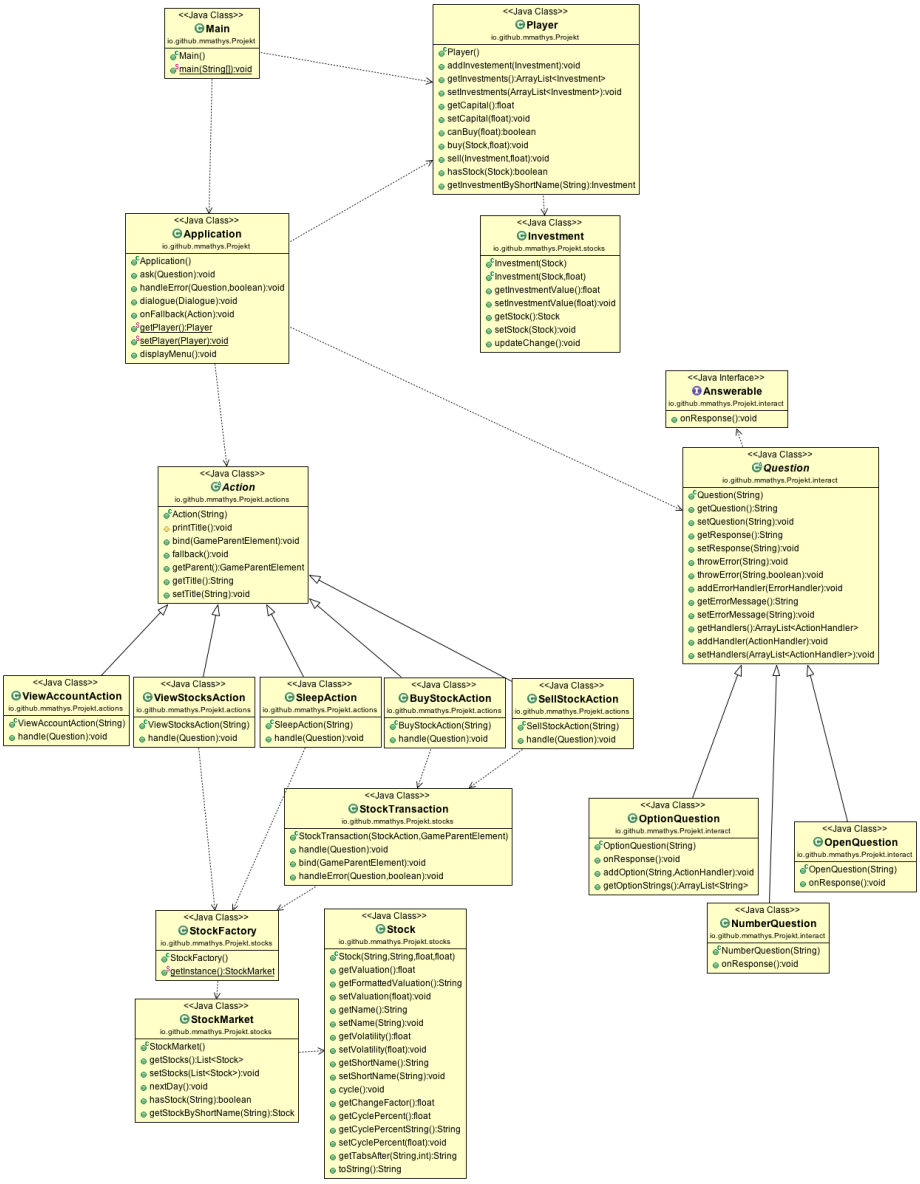

Abbildung 2.1: UML-Diagramm

Main. Hier wird die Applikation gestartet. Es werden Instanzen der Applikation und Spieler erstellt.

Player. Diese Klasse stellt den Spieler dar. Er besitzt Attribute wie das Kapital oder die gekauften Aktien. Zudem verfügt der Spieler über Kauf- und Verkaufsfunktionen für Aktien.

Obwohl es nur eine Instanz des Spielers im Programm gibt, ist er nicht statisch: Es wurde entschieden, dass nur der Spieler-Accessor in `Application` statisch sein sollte.

Application. Diese Klasse ist das Kernstück des Programms: Von dieser Klasse gehen alle Aktionen im Spiel aus. Es kann das Menü anzeigen, Fragen stellen, Dialoge führen und Fehler anzeigen.

Action und Kinder. Wenn das Konto aufgerufen, Aktien betrachtet, gekauft oder verkauft oder geschlafen werden will, so wird über das Menü in `Application` eine bestimmte `Action` aufgerufen.

Das `Action` eine abstrakte Klasse ist, kann sie nicht direkt aufgerufen werden, sondern nur ihre Kinder `ViewAccountAction`, `ViewStocksAction`, `BuyStocksAction`, `SellStocksAction` und `SleepAction`.

`ViewAccountAction`, `ViewStocksAction`, `BuyStocksAction` und `SellStocksAction` haben alle direkt oder indirekt mit Aktien zu tun: Sie benützen alle die Stock-Klassen (`StockFactory`, `StockTransaction`, `StockMarket` und `Stock`).

`ViewAccountAction` zeigt Informationen zum Spieler an.

Stock-Klassen `StockFactory`, `StockTransaction`, `StockMarket` und `Stock` werden als *Stock-Klassen* bezeichnet.

Durch die `StockFactory` kann durch eine statische Funktion die Instanz des `StockMarkets` aus jeder anderen Klasse geholt werden. Die Klasse `StockMarket` beinhaltet die Aktien und aktualisiert deren Kurse.

Die Klasse `Stock` stellt eine Aktie dar. Sie enthält Attribute wie der Gesamtwert der Firma, Name, Kürzel, Schwankungsfreudigkeit oder Änderung des Kurses.

Wenn der Spieler eine Aktie kaufen oder verkaufen will, kommt die Klasse `StockTransaction` ins Spiel. Sie implementiert die Programmlogik für `BuyStockAction` und `SellStockAction`.

Question Die Kinder der `Question`-Klasse implementieren das Fragen im Programm. Es kann nach Zahlen oder nach Strings gefragt werden.

`OptionQuestion` wird beim Menü verwendet. Der Nutzer kann aus Optionen auswählen.

`NumberQuestion` fragt nach Float-Zahlen.

`OpenQuestion` erlaubt eine Antwort beliebiger Zeichenstrings, wie zum Beispiel nach einem Aktienkürzel.

3 Implementierung

Schwankung von Aktien. Der Wert, Name, Kürzel und Schwankung der Aktien wurden in einem Array in einer Java-Klasse als statische finale Variablen gespeichert. Nach jedem Tag im Spiel wird der Wert der Aktien neu berechnet. Dazu wurde zu jeder Aktie die Schwankungsfreudigkeit von 0 bis 1 definiert. Bei 1 ist die maximal mögliche Schwankung 100%, bei 0 0%.

Zugriff von Aktionsklassen auf `Application`. Damit Klassen wie `Question`, `Action` oder `StockTransaction` ebenfalls fragen stellen können, werden sie mit einer `bind()`-Funktion ausgestattet, damit eine Instanz von `Application` übergeben werden kann.

So können diese Klassen so Fragen stellen, Fehler ausgeben und nach Abschliessung ihrer Funktion ein *Callback* geben (damit die Applikation zu ihrem ursprünglichen Zustand, zum Menü, zurückkehrt).

ASCII-Unterstützung. ASCII nennt man die Technik, die in Terminals verwendet wird, falls Farben, Durchstreichungen, Unterstreichungen und ähnliches verwendet werden will.

Falls kein ASCII im Terminal angezeigt werden kann, wird am Anfang des Spiels eine Fehlermeldung ausgegeben. Da man nicht programmatisch prüfen kann, ob der Terminal ASCII-fähig ist, wird folgende Strategie verwendet: Die Textfarbe der Fehlermeldung ist gleich wie sein Hintergrund und wird bei jedem Start der Applikation ausgegeben. Falls kein ASCII unterstützt wird, wird die Fehlermeldung sichtbar.

4 Funktionsweise des Spiels

Im Spiel wird man in die Rolle eines Brokers an der NYSE versetzt. Am Anfang spricht man mit seinem Manager, der einem das Geschäft erklärt.

Anschliessend kann man zwischen folgenden Aktionen auswählen:

- Konto ansehen
- Aktienmarkt betrachten
- Aktie kaufen
- Aktie verkaufen
- Schlafen gehen

Konto ansehen Das Konto zeigt das Kapital und die gekauften Aktien des Spielers an. Je nach Spielstand bekommt er ein Feedback seines Chefs.

Aktienmarkt betrachten Beim Aktienmarkt wird die Aktienmarkt-Tafel angezeigt, wo alle verfügbaren Aktien aufgelistet sind. Zum Namen wird der Kürzel angezeigt, auch die Änderung einer Aktie während einem Tag. Zum Schluss wird noch der Gesamtwert der Aktie aufgelistet.

Aktie kaufen / Aktie verkaufen Wenn der Spieler eine Aktie kaufen oder verkaufen will, muss er zuerst den Aktienkürzel angeben und danach den Wert der Transaktion.

Schlafen gehen Wenn der Spieler schlafen geht, beginnt ein neuer Tag an der Aktienbörse: Die Aktienkurse verändern sich.

5 Schlussfolgerung