

Dokumentation

Java Projekt

2020

Projektname: Aktien-Depot

Kurs: ITA19

Matrikelnummer: 2533282

Dozent: Dr. Werner Jost

1 Grobe Spezifikation der gewünschten Leistung

1.1 Ist-Zustand

Der Kunde hat in der Vergangenheit Aktien von Firmen aus dem DAX Leitindex gekauft. Die Transaktionen hat der Kunde in einer CSV Datei dokumentiert (Datei: database/KaufHistorie.csv). Ein Eintrag (eine Zeile) in dieser Dokumentation enthält:

- Das Datum des Aktienkaufs bzw. -Verkaufs
- Das offizielle Aktienkürzel der gekauften bzw. verkauften Aktie (z.B. DAI.DE für Daimler-Aktien)
- Den Preis der Aktie, für den die Aktie gekauft bzw. verkauft wurde
- Die Anzahl der gekauften bzw. verkauften Aktien und
- Die durchgeführte Aktion („k“ für kaufen und „v“ für verkaufen)

Die historischen Kursdaten der Firmen im DAX sind in CSV-Dateien gespeichert und werden im Internet bereitgestellt (z.B. <https://finance.yahoo.com>).

1.2 Soll-Zustand

1. Der Kunde möchte alle historischen Kursdaten der 30 DAX Unternehmen aus dem Internet laden und auf seinem lokalen Rechner in einem Ordner „database“ speichern.
2. Der Kunde möchte, dass die CSV-Datei, in der er seine Kaufhistorie dokumentiert hat, vom Programm ausgewertet wird. Der Kunde möchte sehen,
 - a. welche Aktien an einem bestimmten Datum in seinem Depot vorhanden sind,
 - b. wie viele Aktien er an diesem Datum von einer Firma besitzt,
 - c. wie viel er im Durchschnitt für eine Aktie gezahlt hat,
 - d. wie viel er bereits in den DAX investiert hat und
 - e. ob seine Transaktionen gewinnbringend waren oder nicht.

Diese Informationen möchte er nach jeder Transaktion (also jedem Aktienkauf bzw. –verkauf) erhalten, um sich somit einen Überblick der zeitlichen Entwicklung seines Depots zu verschaffen.

Zusätzlich soll der Kunde die Möglichkeit erhalten:

3. die Kaufhistorie durch Käufe bzw. Verkäufe von Aktien zu erweitern und
4. sich alle historischen Kursdaten der im DAX gehandelten Aktien anzeigen zu lassen

- a. für eine ausgewählte Aktie (für alle für diese Aktie gespeicherten Daten der historischen Kursinformationen) und
- b. für ein ausgewähltes Datum (für alle historischen Kursinformationen der im Dax gehandelten Aktien).

1.3 Entwurf

Wer wird das System benutzen?

Es gibt nur einen Benutzer. Der Benutzer verwaltet mit dem System sein Aktiendepot.

Wer gibt Daten in das System ein?

- Das System lädt die Kursdaten der 30 DAX-Unternehmen automatisch aus dem Internet.
- Der Kunde muss Transaktionsdaten (Käufe und Verkäufe von Aktien) in Form einer CSV-Datei dem System zur Verfügung stellen.
- Der Kunde kann über ein Formular Käufe und Verkäufe von Aktien durchführen. Diese Transaktionen werden der CSV-Datei hinzugefügt.

Wer erhält Daten aus dem System?

- Der Kunde erhält anhand der übergebenen Transaktionsdaten Informationen zur Wirtschaftlichkeit seiner Transaktionen vom System sowie einen Überblick über die zeitliche Entwicklung seines Depots.
- Der Kunde erhält die Informationen über die historischen Kursdaten der 30 DAX Unternehmen, je nach Auswahl eines Unternehmens über alle gespeicherten Daten oder über alle Unternehmen für ein konkretes Datum.

Welche Aufgaben werden die Akteure mit dem System durchführen?

Der Kunde kann mit dem System Transaktionen auswerten, die er mit Aktien der DAX-Unternehmen getätigt hat. Des Weiteren kann er Transaktionen simulieren und sie auf ihre Wirtschaftlichkeit überprüfen.

Welches Ergebnis will der Akteur nach der Ausführung eines Anwendungsfalls haben?

Der Kunde erhält nach Ausführung des Programms Informationen über die zeitliche Entwicklung seines Depots. Darin enthalten sind:

- der Investitionsbetrag,
- die von ihm gehaltenen Aktien mit jeweiliger Anzahl,
- der für eine Aktie bezahlte Durchschnittspreis sowie
- der aktuelle Wert Gewinn bzw. Verlust.

Welches Ziel hat er?

Sein Ziel liegt in der Auswertung von Daten.

Was ist der Standard-Fall eines Anwendungsfalls?

Der Kunde wertet eine CSV-Datei seiner Transaktionen (Aktienkäufe und -verkäufe) aus.

Welche Erweiterungen und Sonderfälle sind bei einem Anwendungsfall möglich?

Der Kunde erweitert seine CSV-Datei mit Transaktionen (Aktienkäufe und -verkäufe) über das Programm.

Anwendungsfall 1	Kursdaten der 30 DAX-Unternehmen aus dem Internet laden und auf dem lokalen Rechner speichern.
Ziel	Die Kursdaten der 30 DAX-Unternehmen in Form von CSV-Dateien befinden sich auf dem lokalem Rechner.
Vorbedingung	Internetverbindung
Nachbedingung Erfolg	CSV-Dateien werden im Ordner „database“ gespeichert. Sie werden für die weiteren Anwendungsfälle benötigt
Nachbedingung Fehler	Das Programm terminiert.
Akteure	Kunde
Auslösendes Ereignis	Beim Start des Programmes
Erweiterungen	Depot erstellen

Anwendungsfall 2	Depot erstellen
------------------	-----------------

Ziel	Anhand von Transaktionsdaten wird ein Depot angelegt. Im Depot sind Aktien gespeichert, die gekauft bzw. verkauft wurden. Das Depot beinhaltet Informationen über die Anzahl der gekauften Aktien und über den Durchschnittspreis der gekauften Aktien.
Vorbedingung	Der Kunde hat eine CSV-Datei mit Informationen über seine Transaktionen dem Programm zur Verfügung gestellt. Die Transaktionen beziehen sich auf Aktien im DAX. Das Programm hat die Kursdaten der 30 DAX Unternehmen erfolgreich runtergeladen.
Nachbedingung Erfolg	Ein Depot wurde erstellt durch sukzessive Auswertung der Transaktionen, d.h. Aktien, die gekauft wurden, werden dem Depot hinzugefügt und Aktien, die verkauft wurden, werden aus dem Depot entfernt. Für die Aktien im Depot wird der durchschnittliche Einkaufspreis ermittelt.
Nachbedingung Fehler	Ein Depot konnte nicht erstellt werden, das Programm terminiert.
Auslösendes Ereignis	Dem Programm wurde eine CSV-Datei mit Transaktionen übergeben.
Erweiterung	Die zeitliche Entwicklung des Depots wird ausgegeben.

Anwendungsfall 3	Die zeitliche Entwicklung des Depots wird ausgegeben
Ziel	Es wird eine CSV-Datei erstellt, in der die Zeitliche Entwicklung des Depots abgebildet wird.
Vorbedingung	Es liegt ein Depot vor
Nachbedingung Erfolg	Die Zeitliche Entwicklung des Depot wird in der Konsole ausgegeben.
Nachbedingung Fehler	
Akteure	Ein Kunde
Auslösendes Ereignis	Dem Programm wurde eine CSV Datei mit Transaktionen übergeben.

1.4 Nutzerschnittstelle (Ein- und Ausgabe)

Für das Nutzerkonzept wurde ein Use-Case- Diagramm erstellt (siehe Abbildung 1).

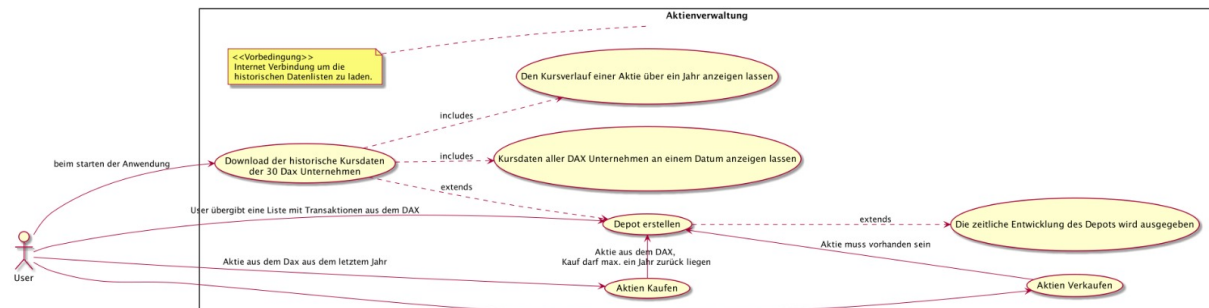


Abbildung 1: Use-Case-Diagramm

Die vier Nutzerschnittstellen sind:

1. Die Ausgabe der zeitlichen Entwicklung des Depots. Dazu muss das Depot unter Verwendung der Kauf- und Verkaufshistorie und der Kursdaten der entsprechenden Aktien zunächst erstellt werden.
2. Der Kauf einer bestimmten Anzahl von Aktien zu einem bestimmten Datum.
3. Der Verkauf einer bestimmten Anzahl von Aktien zu einem bestimmten Datum.
4. Die Anzeige von Kursdaten der Aktien gefiltert nach
 - a. entweder einer Aktie, deren Kursdaten über den gespeicherten Zeitraum, angezeigt werden oder
 - b. einem Datum, um die Kursdaten aller Aktien zu diesem Datum anzuzeigen.

1.4.1 Aktivitätsdiagramm

Das Aktivitätsdiagramm ist in der Abbildung 2 dargestellt und wird im folgendem genauer erleutert.

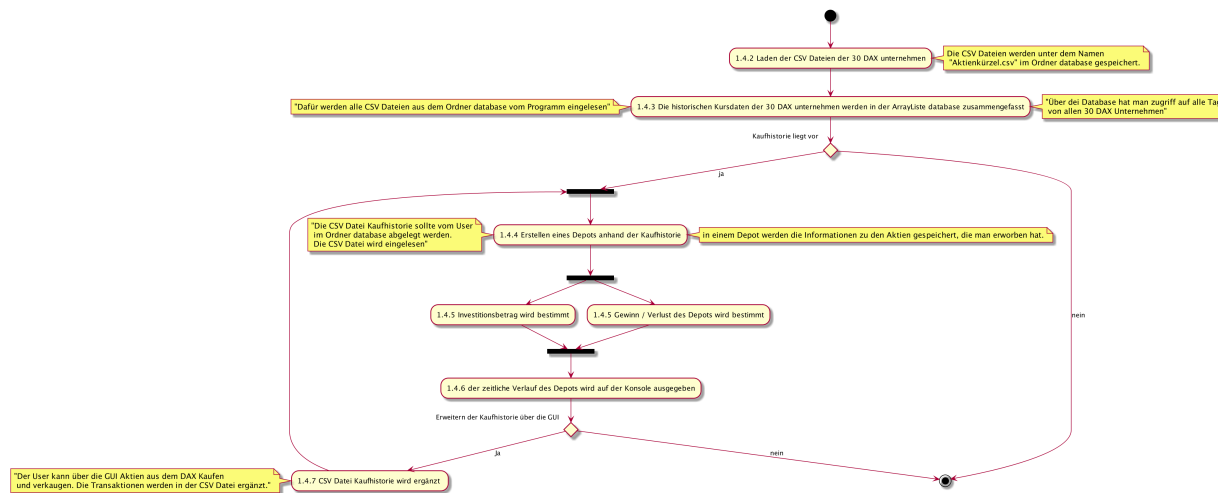


Abbildung 2 Aktivitätsdiagramm

1.4.2 Einlesen der Kursdaten aller DAX-Aktien

Nach dem Start des Programms erscheint zunächst folgende Ausgabe ohne jegliche Vorbedingung:

----- 1. alle csv-Dateien laden -----

```
csv für FRE.DE ... erzeugt (database/FRE.DE.csv)
csv für FME.DE ... erzeugt (database/FME.DE.csv)
csv für ALV.DE ... erzeugt (database/ALV.DE.csv)
csv für BMW.DE ... erzeugt (database/BMW.DE.csv)
csv für BEI.DE ... erzeugt (database/BEI.DE.csv)
csv für LIN.DE ... erzeugt (database/LIN.DE.csv)
csv für HEI.DE ... erzeugt (database/HEI.DE.csv)
csv für BAYN.DE ... erzeugt (database/BAYN.DE.csv)
csv für DBK.DE ... erzeugt (database/DBK.DE.csv)
csv für LHA.DE ... erzeugt (database/LHA.DE.csv)
csv für ADS.DE ... erzeugt (database/ADS.DE.csv)
csv für VOW3.DE ... erzeugt (database/VOW3.DE.csv)
csv für DTE.DE ... erzeugt (database/DTE.DE.csv)
csv für TKA.DE ... erzeugt (database/TKA.DE.csv)
csv für RWE.DE ... erzeugt (database/RWE.DE.csv)
csv für DPW.DE ... erzeugt (database/DPW.DE.csv)
csv für DB1.DE ... erzeugt (database/DB1.DE.csv)
csv für CON.DE ... erzeugt (database/CON.DE.csv)
csv für IFX.DE ... erzeugt (database/IFX.DE.csv)
csv für BAS.DE ... erzeugt (database/BAS.DE.csv)
csv für MRK.DE ... erzeugt (database/MRK.DE.csv)
csv für LXS.DE ... erzeugt (database/LXS.DE.csv)
csv für SIE.DE ... erzeugt (database/SIE.DE.csv)
csv für EOAN.DE ... erzeugt (database/EOAN.DE.csv)
csv für HEN3.DE ... erzeugt (database/HEN3.DE.csv)
csv für MUV2.DE ... erzeugt (database/MUV2.DE.csv)
csv für SAP.DE ... erzeugt (database/SAP.DE.csv)
csv für CBK.DE ... erzeugt (database/CBK.DE.csv)
csv für SDF.DE ... erzeugt (database/SDF.DE.csv)
csv für DAI.DE ... erzeugt (database/DAI.DE.csv)
```

Abbildung 3: Ausgabe des Einlesens der historischen Kursdaten aller DAX-Aktien (Schritt 1)

Die Kursdaten werden aus dem Internet gezogen (finance.yahoo.com) und die CSV-Dateien im „database“-Ordner abgelegt.

1.4.3 Database-Objekt erstellen

Aus diesen Dateien wird ein Objekt vom Typ „Database“ erstellt. Dazu geschieht folgendes:

1. jede Zeile in einer solchen CSV-Datei wird zu einem Objekt vom Typ „TagesInfo“ eingelesen,
2. eine gesamte CSV-Datei (die Menge aller TagesInfo-Objekte einer Aktie) wird zu einem Objekt vom Typ „HistorischeDatenListe“,

3. alle CSV-Dateien (die Menge aller `HistorischeDatenListe`-Objekte) werden zu einem „Database“-Objekt.

Für die Erstellung des Database-Objektes erscheint folgende Ausgabe (siehe Abbildung 4):

----- 2. Datenbasis erstellen -----

Erstelle Database ...

HistorischeDatenListe von FRE.DE zur Database hinzugefuegt.
HistorischeDatenListe von FME.DE zur Database hinzugefuegt.
HistorischeDatenListe von ALV.DE zur Database hinzugefuegt.
HistorischeDatenListe von BMW.DE zur Database hinzugefuegt.
HistorischeDatenListe von BEI.DE zur Database hinzugefuegt.
HistorischeDatenListe von LIN.DE zur Database hinzugefuegt.
HistorischeDatenListe von HEI.DE zur Database hinzugefuegt.
HistorischeDatenListe von BAYN.DE zur Database hinzugefuegt.
HistorischeDatenListe von DBK.DE zur Database hinzugefuegt.
HistorischeDatenListe von LHA.DE zur Database hinzugefuegt.
HistorischeDatenListe von ADS.DE zur Database hinzugefuegt.
HistorischeDatenListe von VOW3.DE zur Database hinzugefuegt.
HistorischeDatenListe von DTE.DE zur Database hinzugefuegt.
HistorischeDatenListe von TKA.DE zur Database hinzugefuegt.
HistorischeDatenListe von RWE.DE zur Database hinzugefuegt.
HistorischeDatenListe von DPW.DE zur Database hinzugefuegt.
HistorischeDatenListe von DB1.DE zur Database hinzugefuegt.
HistorischeDatenListe von CON.DE zur Database hinzugefuegt.
HistorischeDatenListe von IFX.DE zur Database hinzugefuegt.
HistorischeDatenListe von BAS.DE zur Database hinzugefuegt.
HistorischeDatenListe von MRK.DE zur Database hinzugefuegt.
HistorischeDatenListe von LXS.DE zur Database hinzugefuegt.
HistorischeDatenListe von SIE.DE zur Database hinzugefuegt.
HistorischeDatenListe von EOAN.DE zur Database hinzugefuegt.
HistorischeDatenListe von HEN3.DE zur Database hinzugefuegt.
HistorischeDatenListe von MUV2.DE zur Database hinzugefuegt.
HistorischeDatenListe von SAP.DE zur Database hinzugefuegt.
HistorischeDatenListe von CBK.DE zur Database hinzugefuegt.
HistorischeDatenListe von SDF.DE zur Database hinzugefuegt.
HistorischeDatenListe von DAI.DE zur Database hinzugefuegt.

... Database erstellt.

Abbildung 4: Ausgabe zur Erstellung des Database-Objektes

Das Einlesen der Kursdaten aller Aktien sowie die Erstellung des Database-Objektes erfolgt ohne Vorbedingung. Um das Depot zu erstellen, wird eine Kauf- und Verkaufshistorie benötigt. Diese wird in der `KaufHistorie.csv` zur Verfügung gestellt.

1.4.4 Depot-Erstellung

Um das Depot zu erstellen, werden zunächst die Kauf- und Verkaufshistorie eingelesen. Dazu wird die Datei `KaufHistorie.csv` aus dem `database`-Ordner eingelesen. Diese muss als Vorbedingung existieren. Über das Einlesen der Kauf- und Verkaufshistorie wird der Nutzer informiert (siehe Abbildung 5)

----- 3. Kaufhistorie einlesen -----

Abbildung 5: Information über das Einlesen der Kauf- und Verkaufshistorie

Wir nehmen im Folgenden an, die eingelesene `KaufHistorie.csv` sieht wie folgt aus:

Datum	Aktien-Name	Anzahl	Handelsaktion
2019-10-02	SIE.DE	10	k
2019-10-08	SIE.DE	20	k
2019-09-25	SIE.DE	30	k
2019-10-04	IFX.DE	30	k
2019-10-10	IFX.DE	30	k
2019-11-01	DAI.DE	20	k
2019-11-11	DAI.DE	100	k
2019-11-11	SIE.DE	100	k
2019-11-11	SIE.DE	100	k
2019-11-12	SIE.DE	100	v
2019-11-12	DAI.DE	50	v
2020-08-06	DAI.DE	20	v
2020-08-06	ADS.DE	10	k

Mithilfe der Kauf- und Verkaufshistorie wird das Depot erstellt.

1.4.5 Investitionsbetrag und Gewinn beziehungsweise Verlust bestimmen

Das Programm berechnet anhand der Transaktionen, wie viel Geld investiert wurde, um die Aktien zu erwerben. Der investitionsbetrag, wird mit dem aktuellem Wert der Aktien verglichen. Die Differenz zwischen dem Investitionsbetrag und dem aktuellen Wert der Aktien wird als Gewinn oder Verlust angegeben.

1.4.6 Über das Depot wird der Nutzer wie in Abbildung 6 gezeigt, informiert:

Datum 2019-10-02		
Die Aktie SIE.DE wurde für 95,89€ gekauft		
Im Depot befinden sich folgende Aktien:		
Name der Aktie,	Anzahl der Aktien,	Durchschnittskaufpreis
SIE.DE	10	95,89€
Investiert: 958,95€		
Gewinn: 0,00€		
Datum 2019-10-08		
Die Aktie SIE.DE wurde für 95,33€ gekauft		
Im Depot befinden sich folgende Aktien:		
Name der Aktie,	Anzahl der Aktien,	Durchschnittskaufpreis
SIE.DE	30	95,52€
Investiert: 2865,65€		
Gewinn: 5,60€		
Datum 2019-09-25		
Die Aktie SIE.DE wurde für 95,26€ gekauft		
Im Depot befinden sich folgende Aktien:		
Name der Aktie,	Anzahl der Aktien,	Durchschnittskaufpreis
SIE.DE	60	95,39€
Investiert: 5723,30€		
Gewinn: 8,00€		

Datum 2020-08-06		
Die Aktie DAI.DE wurde für 40,65€ verkauft		
Im Depot befinden sich folgende Aktien:		
Name der Aktie, Anzahl der Aktien, Durchschnittspreis		
SIE.DE	160	105,25€
IFX.DE	60	16,03€
DAI.DE	50	57,63€
Investiert: 20683,36€		
Verlust: -1024,64€		
Datum 2020-08-06		
Die Aktie ADS.DE wurde für 245,85€ gekauft		
Im Depot befinden sich folgende Aktien:		
Name der Aktie, Anzahl der Aktien, Durchschnittspreis		
SIE.DE	160	105,25€
IFX.DE	60	16,03€
DAI.DE	50	57,63€
ADS.DE	10	245,85€
Investiert: 23141,86€		
Verlust: -1024,64€		

1.4.7 Das Kaufen- und Verkaufenformular

----- 5. GUI starten -----

und das Fenster öffnet sich:

Abbildung 8: Formulare zum Kauf bzw. Verkauf von Aktien

Sowohl beim Kaufen als auch beim Verkaufen müssen das jeweilige Datum und die Anzahl eingegeben sowie die Aktie ausgewählt werden. Sowohl die Eingabe des Datums als auch die Eingabe der Anzahl wird geprüft. Ist das Datum nicht im korrekten Format bzw. ist die Anzahl keine Zahl erscheint jeweils ein Fenster als Nachricht – die Eingabe muss korrigiert werden.

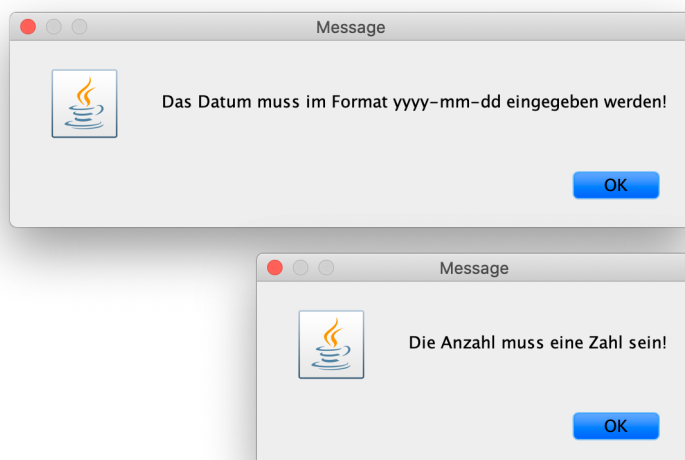


Abbildung 9: jeweilige Warnfenster über fehlerhafte Eingabe

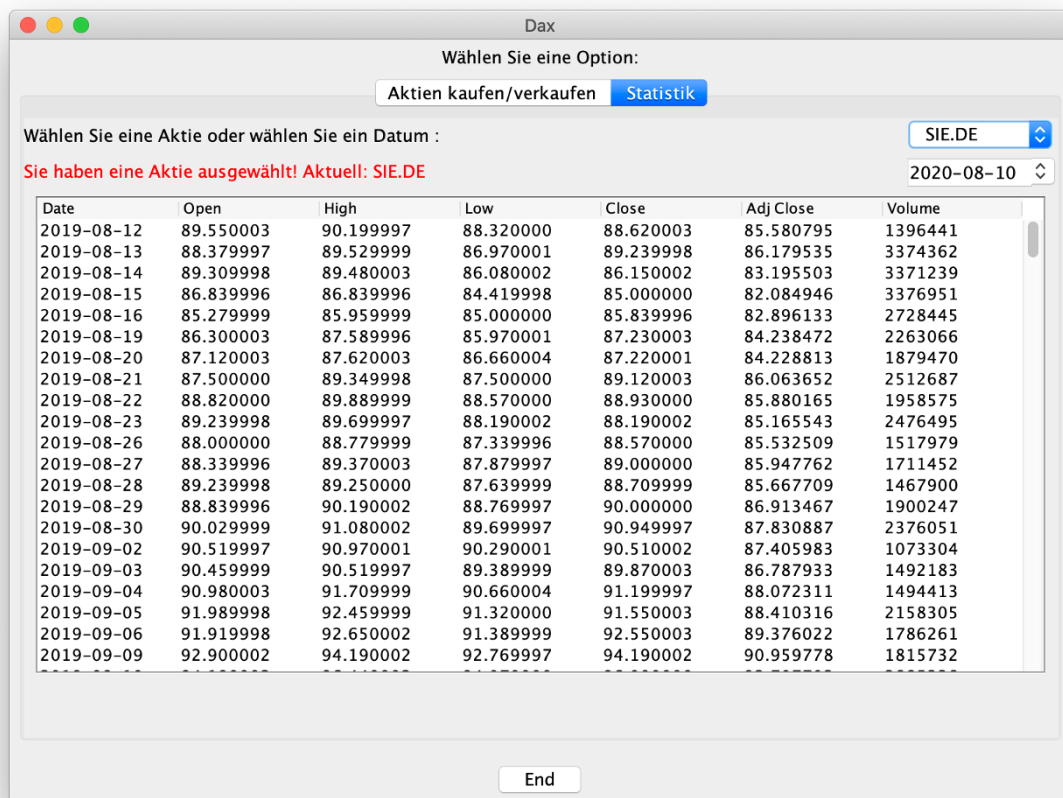
Es wird außerdem geprüft, ob das Datum in den Kursdaten der Aktie überhaupt vorkommt. Dazu wird das Database-Objekt verwendet. Taucht es nicht auf, wird ebenfalls eine Warnung ausgegeben (jedoch auf Konsole – JOptionPane hier in der Ausnahmebehandlung noch nicht implementiert).

`error.TagesInformationenNichtVorhanden: Es existieren keine Tagesinformationen zu dem Datum 2020-08-01`

Existiert das entsprechende Datum in den Kursdaten der Aktie wird daraus der Durchschnittspreis der Aktie an diesem Tag entnommen und die Aktie zu diesem Preis ge- bzw. verkauft. Die neue Transaktion wird in `KaufHistorie.csv` gespeichert und das Depot erneut erstellt und ausgegeben.

1.4.8 Statistik aus den Kursdaten

Der Reiter „Statistik“ in der GUI zeigt den Kursverlauf einer Aktie (am Anfang SIE.DE), siehe Abbildung 10.



Date	Open	High	Low	Close	Adj Close	Volume
2019-08-12	89.550003	90.199997	88.320000	88.620003	85.580795	1396441
2019-08-13	88.379997	89.529999	86.970001	89.239998	86.179535	3374362
2019-08-14	89.309998	89.480003	86.080002	86.150002	83.195503	3371239
2019-08-15	86.839996	86.839996	84.419998	85.000000	82.084946	3376951
2019-08-16	85.279999	85.959999	85.000000	85.839996	82.896133	2728445
2019-08-19	86.300003	87.589996	85.970001	87.230003	84.238472	2263066
2019-08-20	87.120003	87.620003	86.660004	87.220001	84.228813	1879470
2019-08-21	87.500000	89.349998	87.500000	89.120003	86.063652	2512687
2019-08-22	88.820000	89.889999	88.570000	88.930000	85.880165	1958575
2019-08-23	89.239998	89.699997	88.190002	88.190002	85.165543	2476495
2019-08-26	88.000000	88.779999	87.339996	88.570000	85.532509	1517979
2019-08-27	88.339996	89.370003	87.879997	89.000000	85.947762	1711452
2019-08-28	89.239998	89.250000	87.639999	88.709999	85.667709	1467900
2019-08-29	88.839996	90.190002	88.769997	90.000000	86.913467	1900247
2019-08-30	90.029999	91.080002	89.699997	90.949997	87.830887	2376051
2019-09-02	90.519997	90.970001	90.290001	90.510002	87.405983	1073304
2019-09-03	90.459999	90.519997	89.389999	89.870003	86.787933	1492183
2019-09-04	90.980003	91.709999	90.660004	91.199997	88.072311	1494413
2019-09-05	91.989998	92.459999	91.320000	91.550003	88.410316	2158305
2019-09-06	91.919998	92.650002	91.389999	92.550003	89.376022	1786261
2019-09-09	92.900002	94.190002	92.769997	94.190002	90.959778	1815732

Abbildung 10: Darstellung des Kursverlaufes einer Aktie (hier SIE.DE)

Die Daten werden dazu aus der entsprechenden CSV-Datei (z.B. `database/SIE.DE.csv`) eingelesen. Mit den Listen rechts oben hat der Nutzer die Möglichkeit, sich entweder den Kursverlauf einer anderen Aktie oder ein Datum auszuwählen, um sich die Kursdaten aller Aktien zu diesem Datum anzuzeigen (siehe Abbildung 11).

Wählen Sie eine Option:

Aktien kaufen/verkaufen Statistik

Wählen Sie eine Aktie oder wählen Sie ein Datum :

SIE.DE

2020-08-06

Sie haben ein Datum ausgewählt! Aktuell: 2020-08-06

Share	Open	High	Low	Close	Adj Close	Volume
FRE.DE	40.070000	40.599998	38.950001	39.240002	39.240002	2679926
FME.DE	72.300003	74.000000	72.059998	72.699997	72.699997	709459
ALV.DE	179.520004	179.880005	175.839996	177.899994	177.899994	1401103
BMW.DE	55.980000	56.230000	54.750000	55.230000	55.230000	2453738
BEI.DE	100.800003	102.949997	96.940002	97.139999	97.139999	647179
LIN.DE	208.699997	212.100006	206.899994	207.300003	207.300003	730106
HEI.DE	49.830002	50.400002	49.209999	49.639999	49.639999	630870
BAYN.DE	56.009998	57.160000	55.799999	55.930000	55.930000	3403786
DBK.DE	7.861000	7.975000	7.680000	7.753000	7.753000	8434776
LHA.DE	8.580000	8.850000	7.842000	8.098000	8.098000	7127677
ADS.DE	243.000000	250.199997	241.500000	244.300003	244.300003	986147
VOW3.DE	135.500000	137.520004	133.339996	134.000000	134.000000	1434341
DTE.DE	14.570000	14.595000	14.390000	14.455000	14.455000	7819169
TKA.DE	7.138000	7.208000	6.960000	7.024000	7.024000	2602943
RWE.DE	32.930000	33.310001	32.669998	32.930000	32.930000	2377626
DPW.DE	36.369999	37.029999	36.220001	36.400002	36.400002	2940953
DB1.DE	155.899994	157.699997	154.399994	154.550003	154.550003	429877
CON.DE	84.480003	85.220001	82.320000	82.540001	82.540001	615728
IFX.DE	21.900000	22.190001	21.340000	21.514999	21.514999	5895037
BAS.DE	48.915001	49.599998	48.154999	48.605000	48.605000	2910562
MRK.DE	110.250000	113.400002	108.300003	113.300003	113.300003	702440

End

Abbildung 11: Darstellung der Kursdaten aller Aktien zu einem Datum (hier 2020-08-06)

1.5 Design Pattern

Im Folgenden soll noch kurz auf einige Implementierungsdetails eingegangen werden. Insbesondere soll kurz die Verwendung von Design Pattern erläutert werden.

1.5.1 Observer Pattern

Bei allen Ereignisbehandlungen für Nutzereignisse auf der GUI wird (implizit) das Observer Pattern verwendet. Es wurde stets in der folgenden Form (als anonyme Klasse) implementiert:

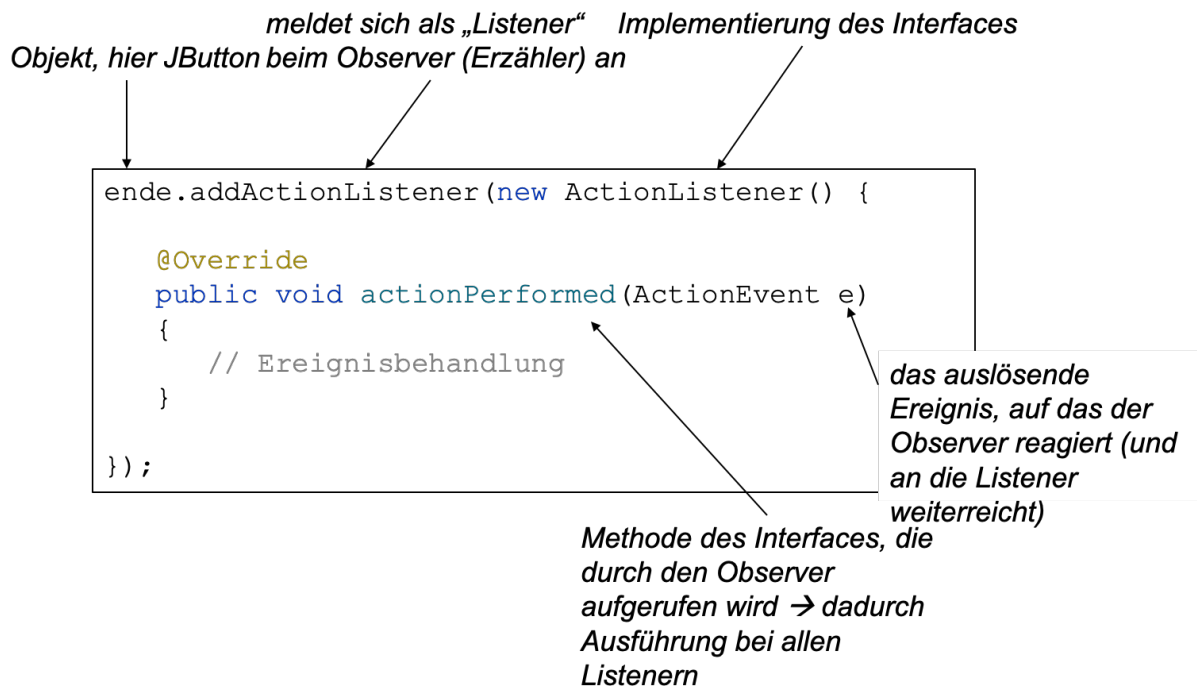


Abbildung 12: aus view/MainWindow.java

Wie hier für den ActionListener eines JButton-Objektes wurden auch ActionListener für JComboBox-Objekte (Datum und Anzahl) und weitere JButton-Objekte (Kaufen und Verkaufen) implementiert. Außerdem wurde in ähnlicher Art auch der ItemListener (`itemStateChanged(ItemEvent e)`) implementiert.

1.5.2 Facade Pattern

Die beiden Formulare für das Kaufen und das Verkaufen von Aktien sehen gleich aus und unterscheiden sich nur im Titel und den Labels. Es wurde deshalb entschieden, eine Klasse `TradingPanelFactory.java` zu erstellen, die die Erzeugung eines solchen Formulars (als `JPanel`) zur Aufgabe hat. Dem Konstruktor dieser Klasse müssen dann nur noch die entsprechenden Werte für die Label-Beschriftungen und den Titel übergeben werden.

Man hätte anstelle des Konstruktors auch eine *Fabrikmethode* verwenden können, die das entsprechende `JPanel`-Objekt erzeugt. Allerdings wird hier mittels *Dependency Injection* ein Objekt von `Depot` übergeben, welches an die jeweiligen Formulare weitergereicht wird (um beim Kaufen oder Verkaufen von Aktien das Depot zu aktualisieren). Aufgrund der dann entstandenen Vermischung von Klassen- und Objekteigenschaften) wurde sich gegen eine statische Fabrikmethode und für den Konstruktor als Fassade entschieden.

1.5.3 Adapter Pattern

In dem Statistik-Tab werden mithilfe einer `JTable` die Kursdaten einer Aktie bzw. aller Aktien zu einem Datum angezeigt. Die Verwendung eines Models (das von der abstrakten Klasse `AbstractTableModel`, die das Interface `TableModel` implementiert, abgeleitet ist) für die Tabelle entspricht einem Adapter. Das Interface `TableModel` enthält einige wenige Methoden, mit denen Tabellen generiert werden können, wie z.B. `getColumnCount()`, `getRowCount()`, `getColumnName()`, `getValueAt(row, col)`. Zwischen der Darstellung der Tabelle und den eigentlichen Daten (Spaltennamen und Daten) bedarf es nun nur noch eines Adapters, der beides zusammenbringt. Das ist in meinem Code die Klasse `MyTableModel`. Einem Objekt dieser Klasse werden dann nur noch die tatsächlichen Daten (`data`) und die Überschriften (`titles`) übergeben und das `MyTableModel`-Objekt wird der Tabelle übergeben:

```
this.model = new MyTableModel(data, titles);  
this.table = new JTable( this.model );
```


2 UML Klassendiagramm

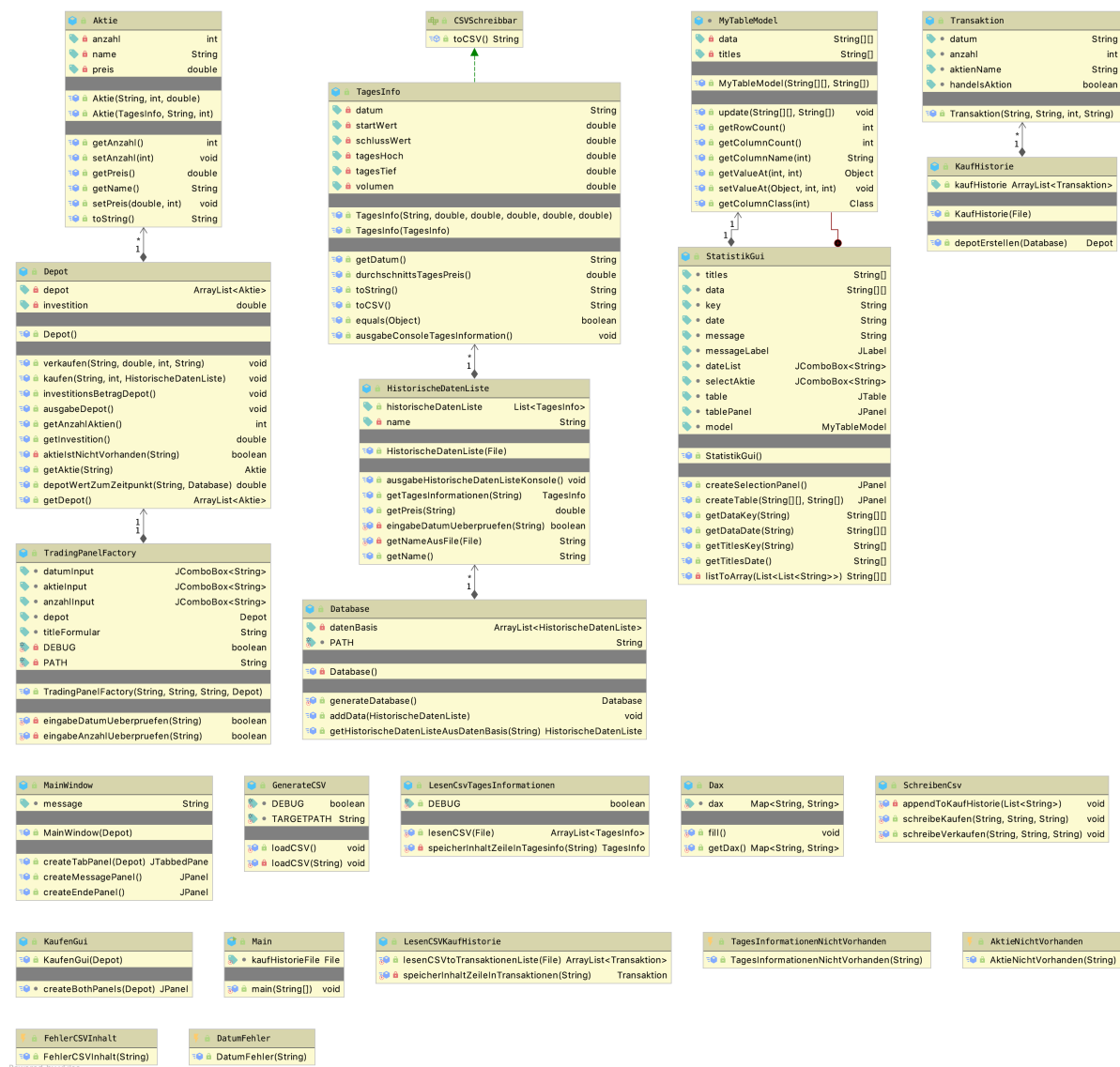


Abbildung 13 UML Klassendiagramm

In Abbildung 13 wird das UML Klassendiagramm zum Aktien-Projekt dargestellt. Die einzelnen Klassen werden mit ihrem Namen, ihren Attributen und ihren Methoden dargestellt. Die Attribute sind, wenn möglich, privat, um das Geheimhaltungsprinzip zu gewährleisten. Aus dem Klassendiagramm lassen sich die Assoziationen ablesen. In diesem Projekt stehen die meisten Klassen in Form einer Komposition in Beziehung zueinander. Das bedeutet zum Beispiel, dass ein Objekt der Klasse `TagesInformation` nur so lange existiert wie ein Objekt der Klasse `HistorischeDatenListe`. Die meisten Assoziationen zwischen den Klassen im package `model` weisen eine Kardinalität von **1 zu n** auf. Zum Beispiel kann ein Objekt der Klasse `HistorischenDatenListe` mit beliebig vielen Objekten von der Klasse `TagesInformationen` in Beziehung stehen.