



HTW Chur
Hochschule für Technik und Wirtschaft

Fachhochschule Ostschweiz
University of Applied Sciences

Entwurfsdokument

Projekt: *JQuotes*

Version 1.2

TETBörsianer

Fabian Anderegg
David Angleitner
Matthias Kohler
Norman Süsstrunk
Ralph Potztal

Änderungsgeschichte

<u>Datum</u>	<u>Version</u>	<u>Autor</u>	<u>Beschreibung</u>
2008-04-28	1.0	D. Angleitner	Dokument erstellt
2008-05-10	1.1	D. Angleitner N. Süsstrunk	Dokument erweitert (Screenshots etc.) und überarbeitet
2008-05-29	1.2	D. Angleitner	XML, Portfolio hinzugefügt. Letzte Anpassungen und Änderungen.

Inhaltsverzeichnis

1	EINFÜHRUNG	5
1.1	Zweck	5
1.2	Inhalt	5
1.3	Definitionen und Abkürzungen	5
1.4	Referenzen	5
2	HINWEISE ZUM DESIGN	6
2.1	Einschränkungen	6
2.2	Systemumgebung	6
3	ARCHITEKTUR	7
3.1	Überblick	7
3.2	Packages	8
3.3	Klassen	10
3.3.1	dataAccess	10
3.3.2	exceptions	13
3.3.3	portfolio	14
3.3.4	portfolio.popUps	17
3.3.5	views	18
3.3.6	visuels	22
3.3.7	visuels.calculators	26
3.3.8	visuels.graphs	27
3.4	XML	28
A	REFERENZEN	29

Kapitel 1 – Einführung

Abbildungsverzeichnis

Abbildung 1 Packages.....	8
Abbildung 3 AddPortfolioDialog.....	14
Abbildung 4 AddTitleDialog	14
Abbildung 5 EditTitleDialog.....	15
Abbildung 6 PopUps.....	17
Abbildung 7 AboutDialog.....	18
Abbildung 8 AddTitleSimple.....	19
Abbildung 9 ChartOptions Tab 1.....	20
Abbildung 10 ChartOptions Tab 2.....	20
Abbildung 11 Main Window	21
Abbildung 12 ChartPanel.....	23
Abbildung 13 FetchingDataDialog	23
Abbildung 14 GridSelector	25
Abbildung 15 Calculators	26
Abbildung 16 DisplayType.....	27
Abbildung 17 XML.....	28

Kapitel 1 – Einführung

1 Einführung

1.1 Zweck

Dieses Dokument beschreibt das Design der Börsensoftware JQuotes.

JQuotes ermöglicht die Darstellung von Börsenkursen. Es soll Privatanlegern erlauben, die eigenen Aktienkurse mit einer kurzen Verzögerung zum aktuellen Kurs abzurufen. Dem Benutzer stehen verschiedene Optionen zur Darstellung der Kurse zur Verfügung. Weiter besteht die Möglichkeit, ein Portfolio zu erstellen und unterschiedliche Kurse miteinander zu vergleichen.

Als Datenlieferant wird in erster Linie eine Schnittstelle von Yahoo verwendet, das Programm soll jedoch so ausgelegt werden, dass Erweiterungen und alternative Datenquellen einfach eingebunden werden können.

1.2 Inhalt

In erster Linie besteht das Design aus den Klassendiagrammen in UML. Sie zeigen die Eigenschaften, Methoden und Verknüpfungen mit anderen Klassen. Zudem wird die Ordner- (und damit Package-) Struktur aufgezeigt und erklärt. In diesem Dokument wird jede Klasse nur grob erklärt – für Detailinformationen sei auf die API verwiesen [1].

1.3 Definitionen und Abkürzungen

API

Application Programming Interface; Schnittstellen Definition des Programms.

GUI

Graphical User Interface; Die grafischen Bedienelemente, über welche der Benutzer das Programm bedient.

JavaDoc

Eine bestimmte Syntax von Kommentaren, über welche die Dokumentation bzw. API direkt aus dem Sourcecode generiert werden kann.

JRE

Java Runtime Environment; Laufzeitumgebung zur Ausführung von Java Programmen

1.4 Referenzen

Siehe Anhang A auf Seite 29 dieses Dokuments.

Kapitel 2 – Hinweise zum Design

2 Hinweise zum Design

2.1 *Einschränkungen*

Die Software ist auf einen Datenlieferanten für den Bezug der Börsendaten angewiesen. In erster Linie wird dafür die Yahoo-Schnittstelle verwendet. Das Programm ist jedoch so auszulegen, dass der Datenlieferant einfach gewechselt werden kann. Sollte jedoch kein Datenlieferant zur Verfügung stehen, sind praktisch alle Funktionen nicht mehr lauffähig.

2.2 *Systemumgebung*

Um JQuotes auszuführen wird die Java Runtime Environment (JRE) 1.6 benötigt.

Derzeit steht diese JRE für folgende Plattformen zur Verfügung [2]:

- Windows 2000 (SP4+)
- Windows XP (SP1, SP2)
- Windows Vista
- Windows 2003

Zusätzlich werden folgende Libraries benötigt, die jedoch direkt mit der Software ausgeliefert werden:

- JGoodies Forms 1.2.0
- JUnit
- log4J

Kapitel 3 – Architektur

3 Architektur

3.1 *Überblick*

Das Programm wird wie in Java üblich in verschiedene Packages aufgeteilt, siehe dazu Kapitel 3.2.

Bei dem Design wird auf einfache Erweiterbarkeit geachtet. Insbesondere betrifft dies die Schnittstelle des Datenlieferanten, verschiedene Darstellungsvarianten der Graphen sowie Berechnungsoptionen.

Das Gesamtdiagramm mit allen vorhandenen Klassen befindet sich aufgrund der Grösse in einer separaten Datei [3].

Kapitel 3 – Architektur

3.2 Packages

Diese Übersicht zeigt alle Packages und deren enthaltenen Klassen, die Beschreibung der einzelnen Packages folgt auf der nächsten Seite.

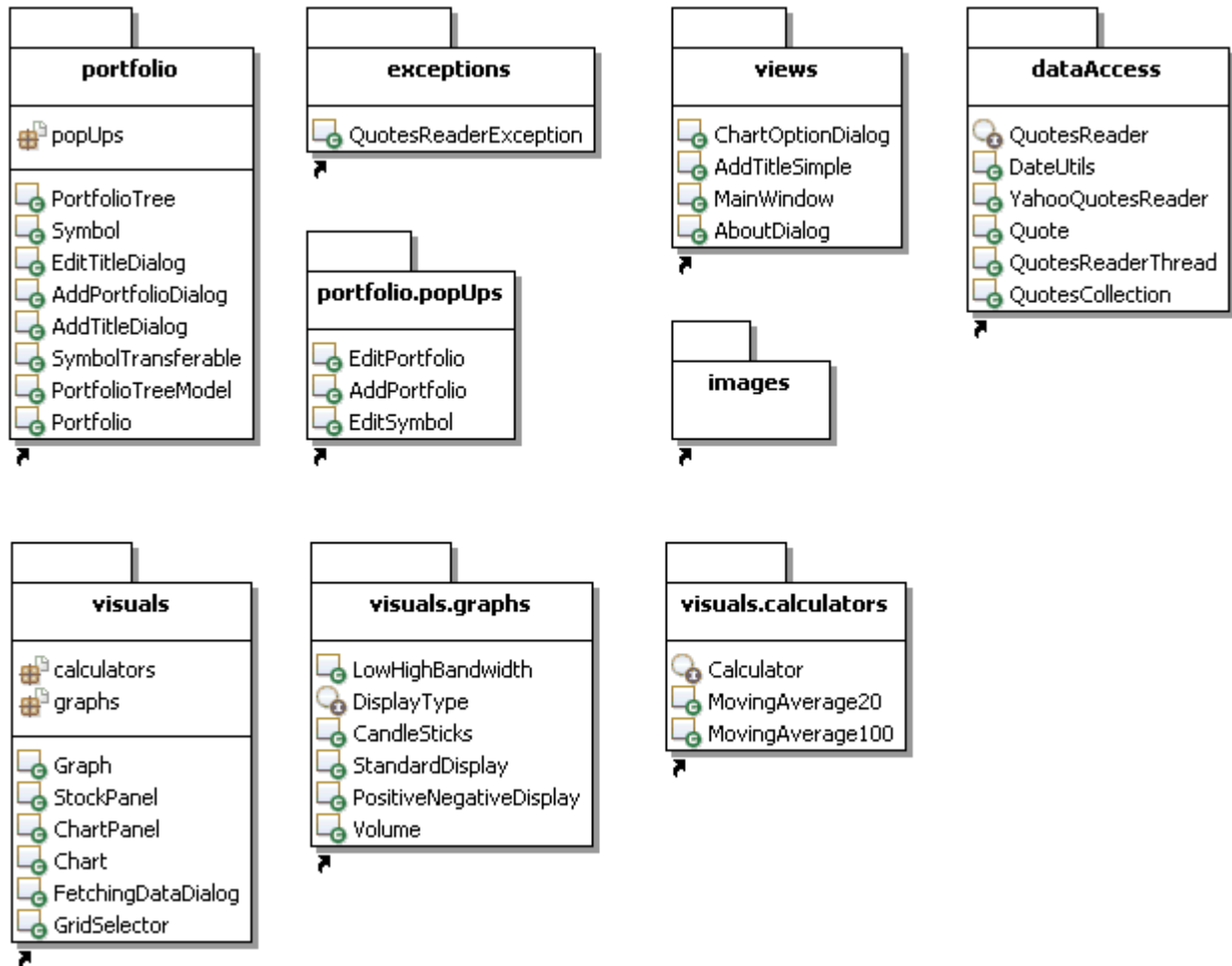


Abbildung 1 Packages

Kapitel 3 – Architektur

dataAccess

In diesem Package befindet sich das Interface "QuotesReader" sowie die Klasse "YahooQuotesReader", welche dieses Interface implementiert. Im restlichen Programmcode wird nur mit diesem Interface gearbeitet, sodass der spezifische Teil für die Yahooschnittstelle völlig verborgen bleibt.

exceptions

In diesem Package befinden sich anwendungsspezifische Ausnahmeklassen.

images

In dieses Package werden alle Bilder abgelegt, bspw. Logo, Buttons etc.

portfolio

In diesem Package befindet sich die ganze Portfolio Verwaltung, dazu gehören beispielsweise eine spezifische Implementierung des JTrees und des TreeModels.

portfolio.popUps

In diesem Package befinden sich die PopUps, welche im Portfolio Manager verwendet werden.

views

In diesem Package befinden sich alle unabhängigen User Interfaces.

visualse

In diesem Package befinden sich alle Klassen, welche für die grafische Darstellung von Graphen benötigt werden.

visualse.calculators

In diesem Package befindet sich das Interface „Calculator“, sowie alle Klassen welche dieses Interface implementieren. Diese werden benötigt um gegebene Daten umzurechnen.

visualse.graphs

In diesem Package befindet sich das Interface „DisplayType“, sowie alle Klassen welche dieses Interface implementieren. Diese werden benötigt, um gegebene Daten grafisch darzustellen.

Kapitel 3 – Architektur

3.3 Klassen

Hier folgt eine grobe Beschreibung des Aufbaus und der Verantwortlichkeiten aller Klassen. Für detailliertere Informationen wird auf die API [1] verwiesen.

3.3.1 dataAccess

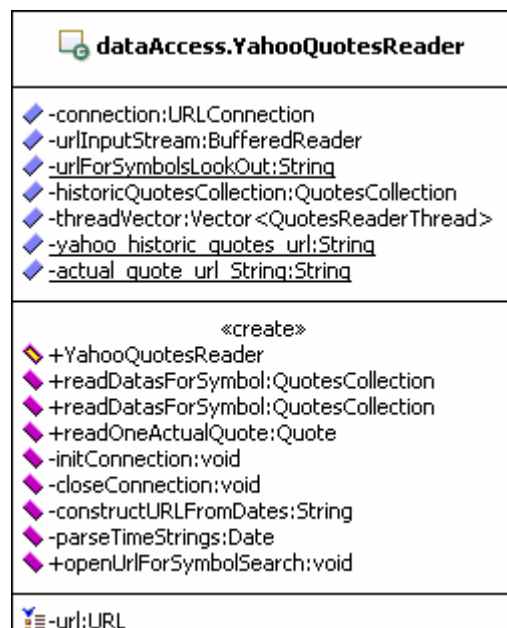
Interface: *QuotesReader*

Beschreibung: Dieses Interface definiert die Schnittstelle, über welche historische und aktuelle Kurse abgefragt werden können.



Klasse *YahooQuotesReader*

Beschreibung Diese Klasse Implementiert das QuotesReader-Interface. Intern wird eine Schnittstelle von Yahoo angesteuert.



Kapitel 3 – Architektur

Klasse *Quote*

Beschreibung Diese Klasse kapselt die abgerufenen Daten für einen Aktienkurs.

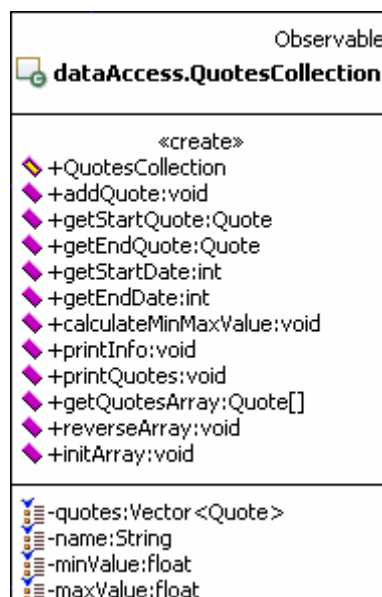
Abhängigkeiten QuotesReader, QuotesCollection



Klasse *QuotesCollection*

Beschreibung Diese Klasse kapselt historische, über einen Zeitraum abgerufenen Kurse (Quotes). Ausserdem stellt sie einfache Methoden für Berechnungen auf den Daten bereit.

Abhängigkeiten QuotesReader

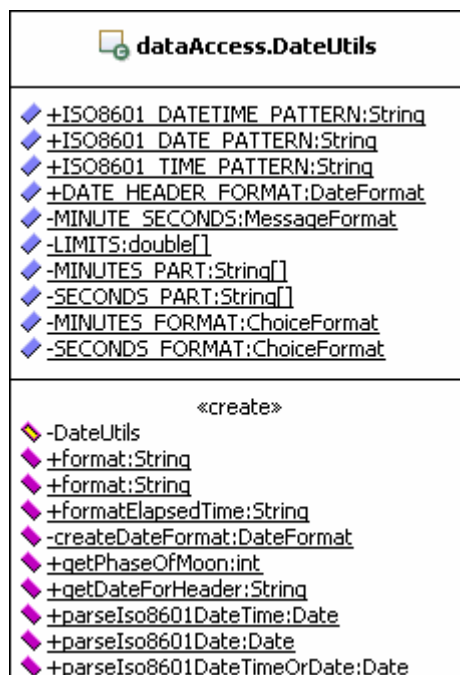


Kapitel 3 – Architektur

Klasse	<i>QuotesReaderThread</i>
Beschreibung	Ein Thread der periodisch die Datenschnittstelle nach neuen Daten befragt und diese an die Collection anhängt.
Abhängigkeiten	QuotesReader, QuotesCollection



Klasse	<i>DateUtils</i>
Beschreibung	Hilfsklasse, um die Arbeit mit verschiedenen Datumsformaten zu vereinfachen. Wird benötigt, um die abgerufenen Datumswerte in geeignete Date-Objekte zu konvertieren.
Abhängigkeiten	Keine



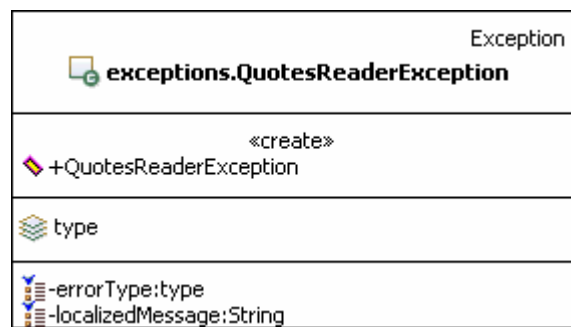
Kapitel 3 – Architektur

3.3.2 exceptions

Klasse *QuotesReaderException*

Beschreibung Eine Exception, welche ausgelöst wird, wenn keine Daten empfangen werden konnten.

Abhängigkeiten Keine



Kapitel 3 – Architektur

3.3.3 *portfolio*

Klasse *AddPortfolioDialog*

Beschreibung Ein GUI, welches es dem Benutzer erlaubt, ein neues Portfolio zu erstellen.

Abhängigkeiten PortfolioTree

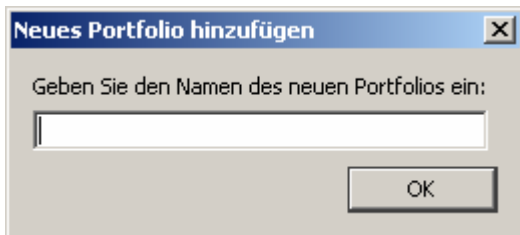
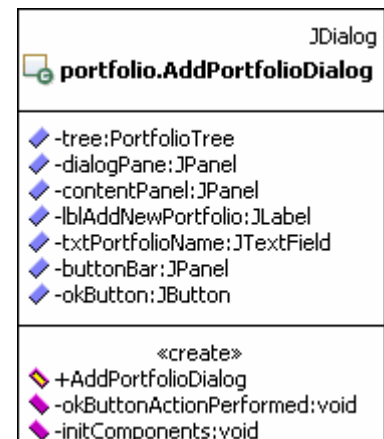


Abbildung 2 AddPortfolioDialog



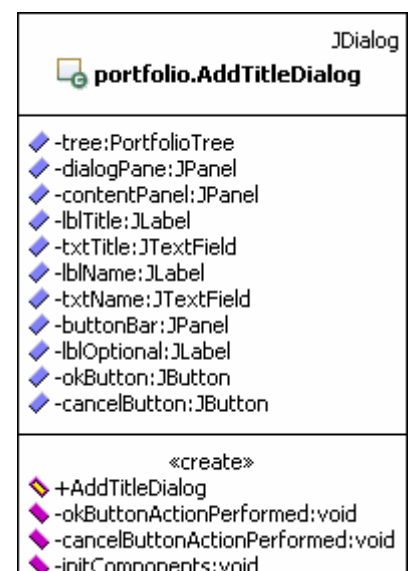
Klasse *AddTitleDialog*

Beschreibung Ein GUI, welches es dem Benutzer erlaubt, einen neuen Titel in ein Portfolio aufzunehmen.

Abhängigkeiten PortfolioTree



Abbildung 3 AddTitleDialog



Kapitel 3 – Architektur

Klasse *EditTitleDialog*

Beschreibung Ein GUI, welches es dem Benutzer erlaubt, einen bestehenden Titel in einem Portfolio zu bearbeiten.

Abhängigkeiten PortfolioTree, Symbol

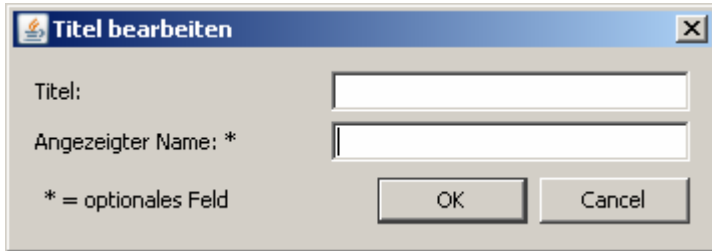
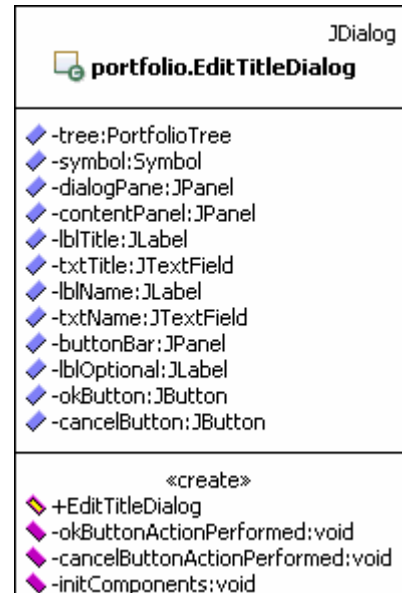


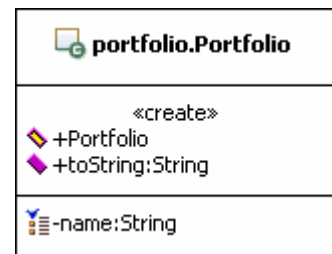
Abbildung 4 EditTitleDialog



Klasse *Portfolio*

Beschreibung Repräsentiert ein Portfolio innerhalb eines Trees

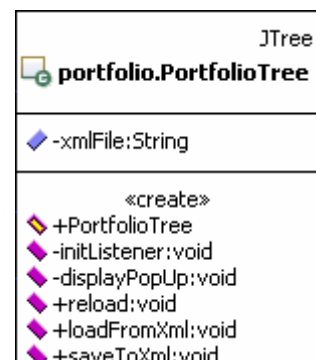
Abhängigkeiten keine



Klasse *PortfolioTree*

Beschreibung Eine Implementierung eines JTrees welche das Laden und Speichern in einem XML File übernimmt, sowie Drag und Drop Funktionalität bereitstellt.

Abhängigkeiten MainWindow, Alle Portfolio.PopUps

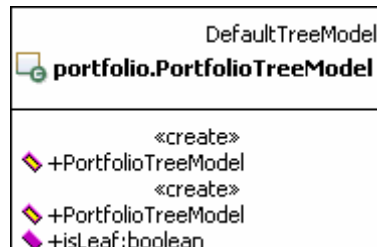


Kapitel 3 – Architektur

Klasse *PortfolioTreeModel*

Beschreibung Eine einfache Implementierung des DefaultTreeModels. Entscheidet anhand des Objekttyps ob ein Knoten ein „Ordner“ oder eine „Datei“ ist.

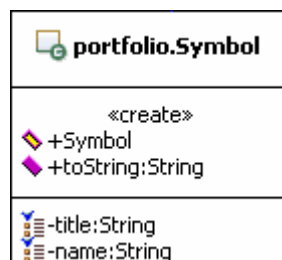
Abhängigkeiten PortfolioTree



Klasse *Symbol*

Beschreibung Repräsentiert ein Symbol innerhalb eines Trees

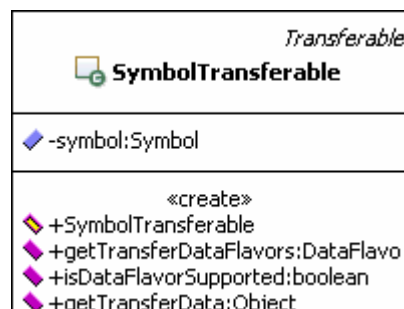
Abhängigkeiten SymbolTransferable



Klasse *SymbolTransferable*

Beschreibung Packt ein Symbol in ein Transferable, um per Drag and Drop übergeben zu werden

Abhängigkeiten Symbol



Kapitel 3 – Architektur

3.3.4 *portfolio.popUps*

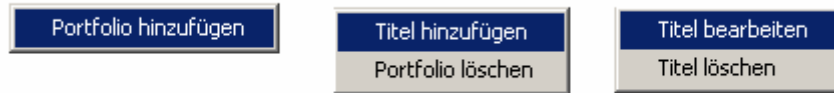
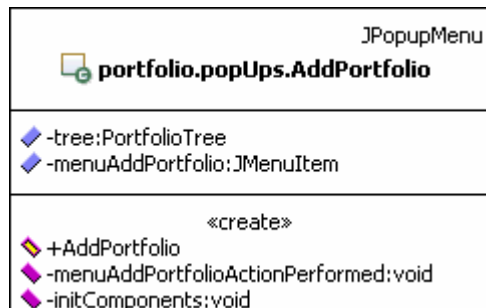
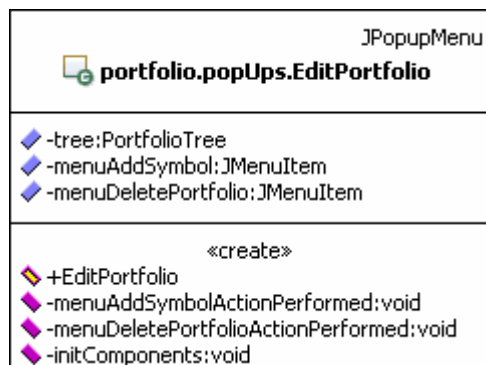


Abbildung 5 PopUps

Klasse	<i>AddPortfolio</i>
Beschreibung	Ein PopUp um ein Portfolio innerhalb des Trees anzulegen
Abhängigkeiten	PortfolioTree



Klasse	<i>EditPortfolio</i>
Beschreibung	Ein PopUp um ein Portfolio innerhalb des Trees zu bearbeiten
Abhängigkeiten	PortfolioTree

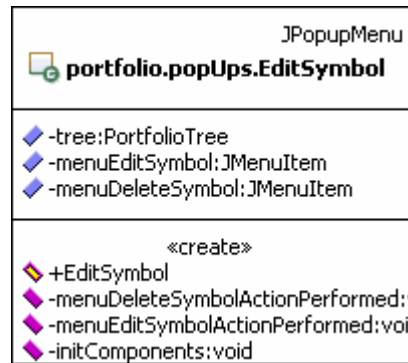


Kapitel 3 – Architektur

Klasse *EditSymbol*

Beschreibung Ein PopUp um ein Symbol innerhalb des Trees zu bearbeiten

Abhängigkeiten PortfolioTree



3.3.5 views

Klasse: *AboutDialog*

Beschreibung: Ein GUI, welches Informationen über das Programm anzeigt, siehe Bild.

Abhängigkeiten: Keine

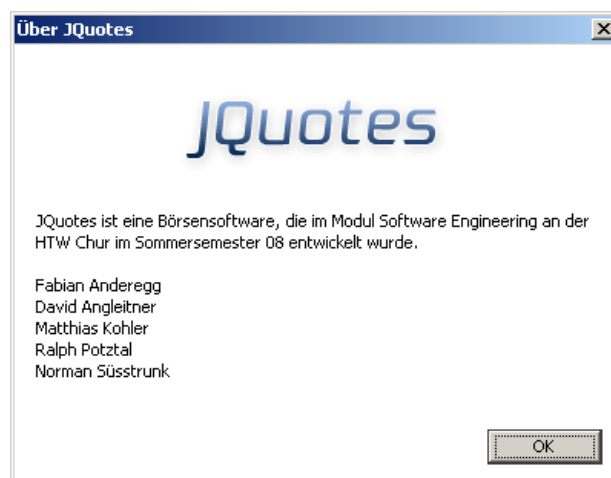


Abbildung 6 AboutDialog



Kapitel 3 – Architektur

Klasse: *AddTitleSimple*

Beschreibung: Ein GUI, welches es dem Benutzer erlaubt unkompliziert einen neuen Titel zu einem Chart hinzuzufügen, ohne diesen in ein Portfolio aufzunehmen.

Abhängigkeiten: Chart

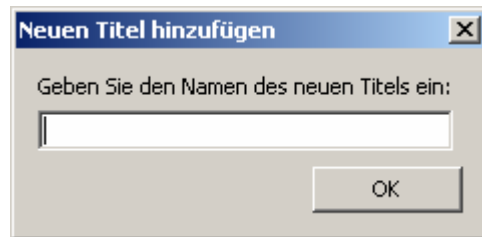
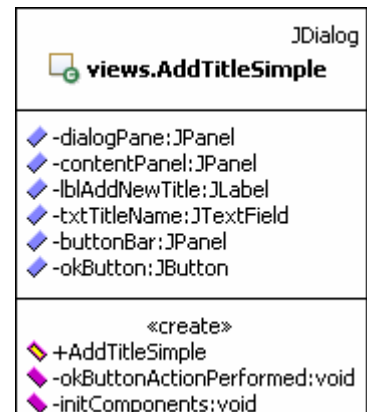


Abbildung 7 AddTitleSimple



Kapitel 3 – Architektur

Klasse: *ChartOptionDialog*

Beschreibung: Ein GUI, welches es dem Benutzer erlaubt Optionen eines Charts, bzw. der darin enthaltenen Graphen zu verändern.

Abhängigkeiten: Chart, Wird von ChartPanel aufgerufen

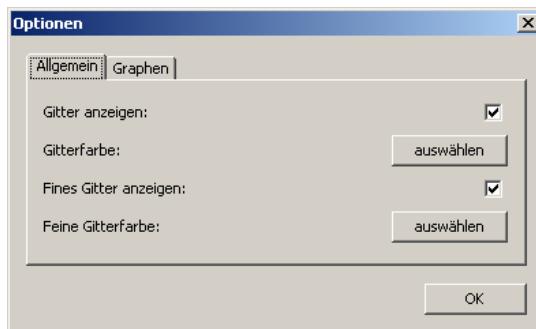


Abbildung 8 ChartOptions Tab 1

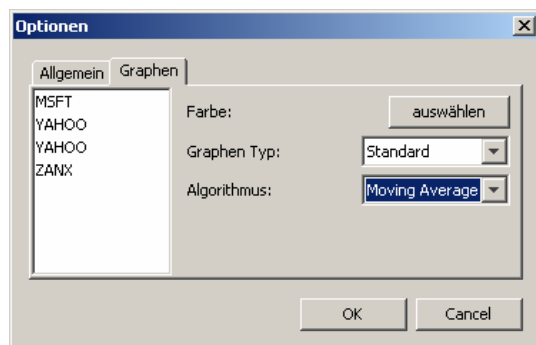


Abbildung 9 ChartOptions Tab 2



Kapitel 3 – Architektur

Klasse: *MainWindow*

Beschreibung: Das Hauptfenster, welches der Benutzer nach dem Starten des Programms zu sehen bekommt. Von hier werden alle weiteren Aktion ausgeführt.

Abhängigkeiten: Keine

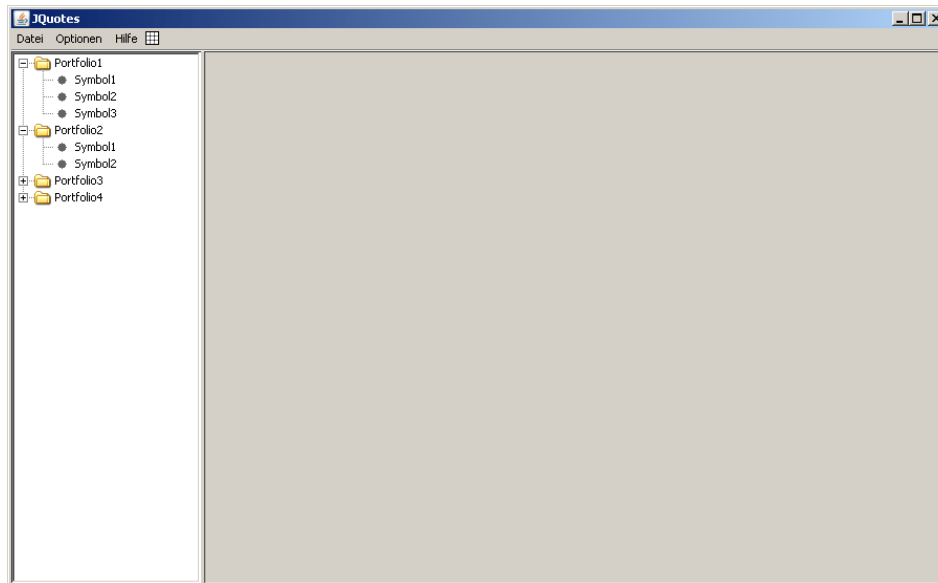
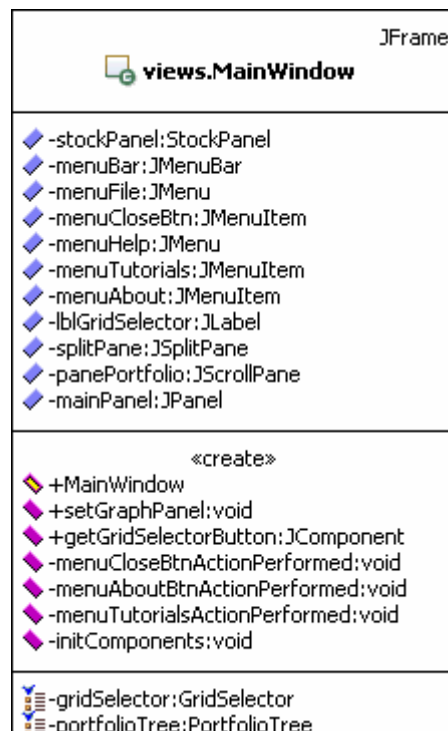


Abbildung 10 Main Window



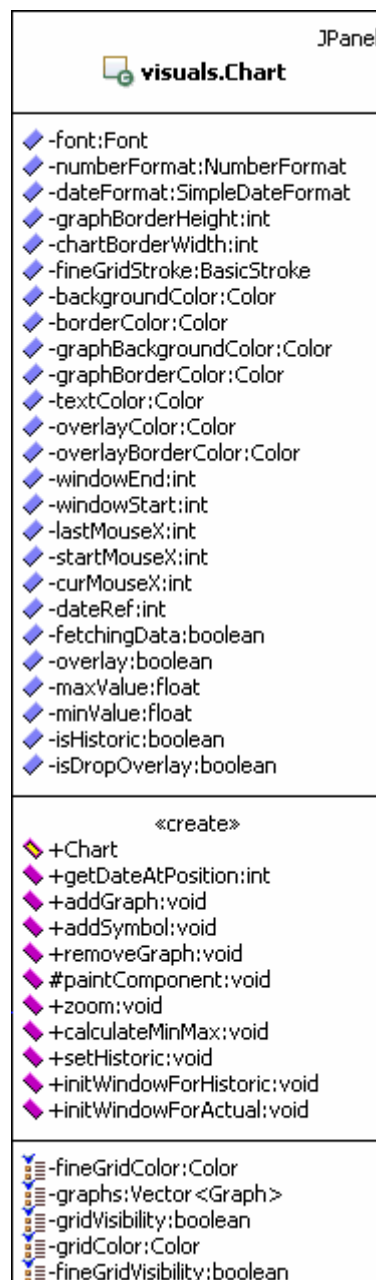
Kapitel 3 – Architektur

3.3.6 visuals

Klasse: *Chart*

Beschreibung: Ein Chart enthält einen oder mehrere Graphen. Ausserdem zeichnet er die Legende, das Gitter und die Beschriftungen der x- und y-Achse. Ebenfalls von dieser Klasse werden Benutzeraktionen wie Zoomen und verschieben des Graphen übernommen.

Abhängigkeiten: ChartPanel, Graph



Kapitel 3 – Architektur

Klasse: *ChartPanel*

Beschreibung: Ein ChartPanel enthält den Rahmen für einen Chart mit den entsprechenden Buttons.

Abhängigkeiten: Chart, StockPanel

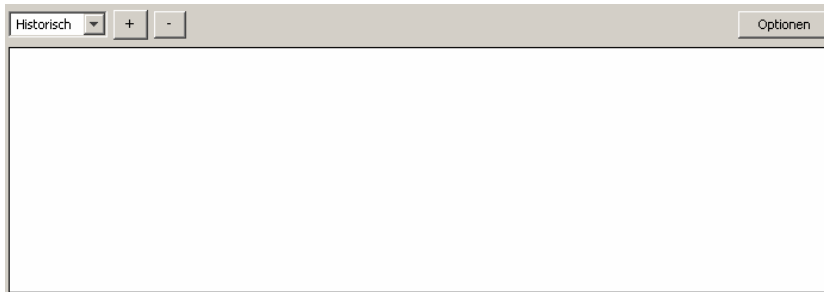
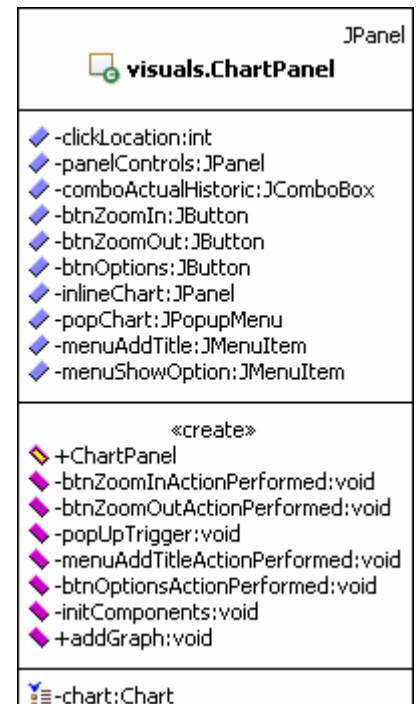


Abbildung 11 ChartPanel



Klasse: *FetchingDataDialog*

Beschreibung: Ein kleines PopUp, welches eine Warteanimation zeigt, während Daten heruntergeladen werden.

Abhängigkeiten: keine

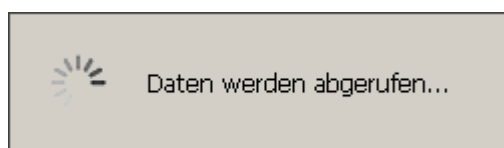
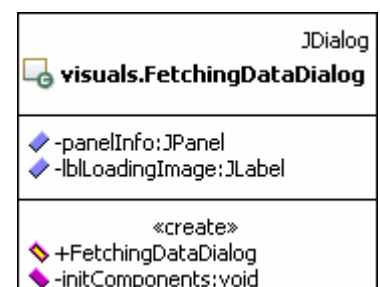
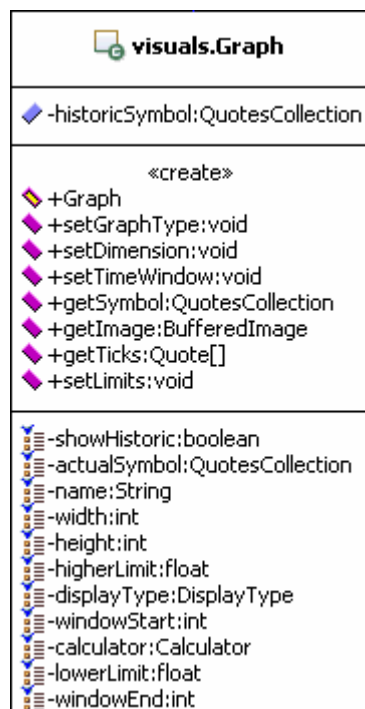


Abbildung 12 FetchingDataDialog



Kapitel 3 – Architektur

Klasse: *Graph***Beschreibung:** Die Klasse Graph verwaltet das Zeichnen jeweils eines Graphen. Über die Klasse DisplayType wird das Aussehen, über die Klasse Symbol die Daten und über die Klasse Calculator allfällige Berechnungen definiert.**Abhängigkeiten:** Symbol, DisplayType, Calculator

Kapitel 3 – Architektur

Klasse: *GridSelector*

Beschreibung: Der GridSelector ist eine kleine Komponent zur Auswahl der Anzahl ChartPanels in einem StockPanel.

Abhängigkeiten: MainWindow, StockPanel

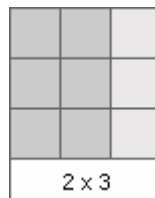
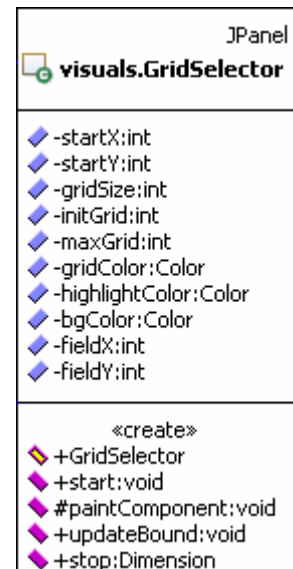


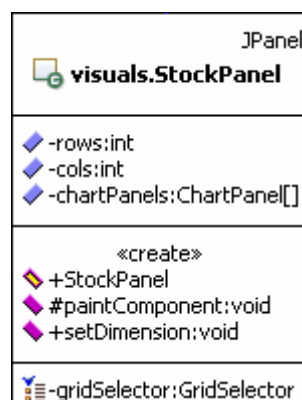
Abbildung 13 GridSelector



Klasse: *StockPanel*

Beschreibung: Das StockPanel wird in das MainWindow eingebunden und stellt eine gewisse Anzahl an ChartPanels gleichzeitig dar. Über den GridSelector kann diese Anzahl verändert werden.

Abhängigkeiten: MainWindow, ChartPanel, GridSelector



Kapitel 3 – Architektur

3.3.7 *visuals.calculators*

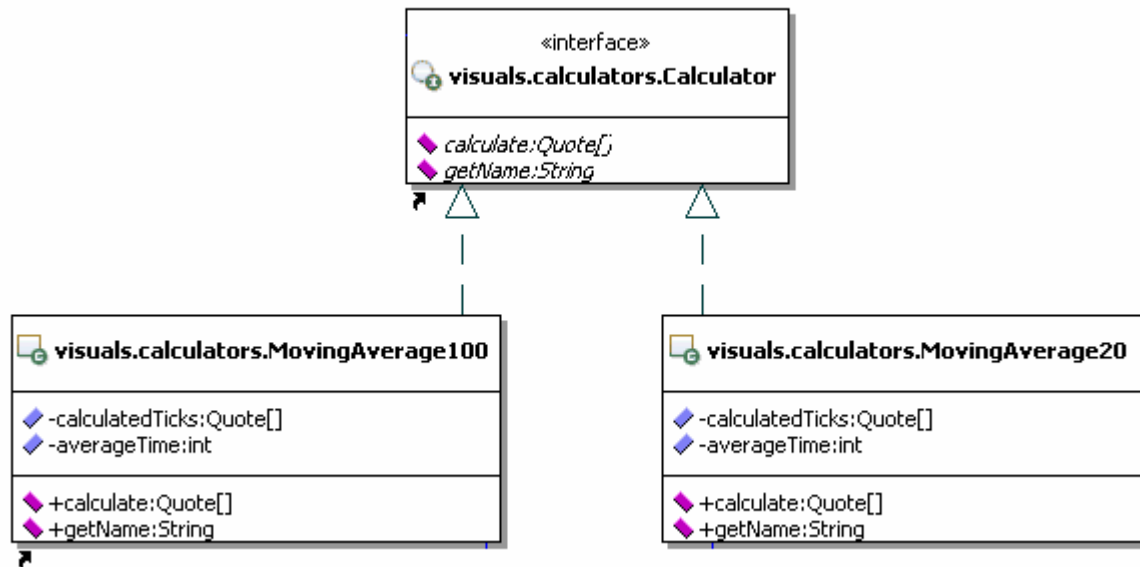


Abbildung 14 Calculators

Interface: *Calculator*

Beschreibung: Dieses Interface definiert die Schnittstellen, welche für einen Calculator benötigt werden. Diese Klassen übernehmen eine Anzahl Quotes und verändern diese nach einem bestimmten Algorithmus.

Abhängigkeiten: Graph, Ticks

Klasse: *MovingAverage20, MovingAverage 100*

Beschreibung: Diese Klassen zeigen Beispiele für Implementierungen eines Calculators. Hierbei werden die Daten über eine bestimmte Zeit gemittelt.

Implementiert: Calculator

Kapitel 3 – Architektur

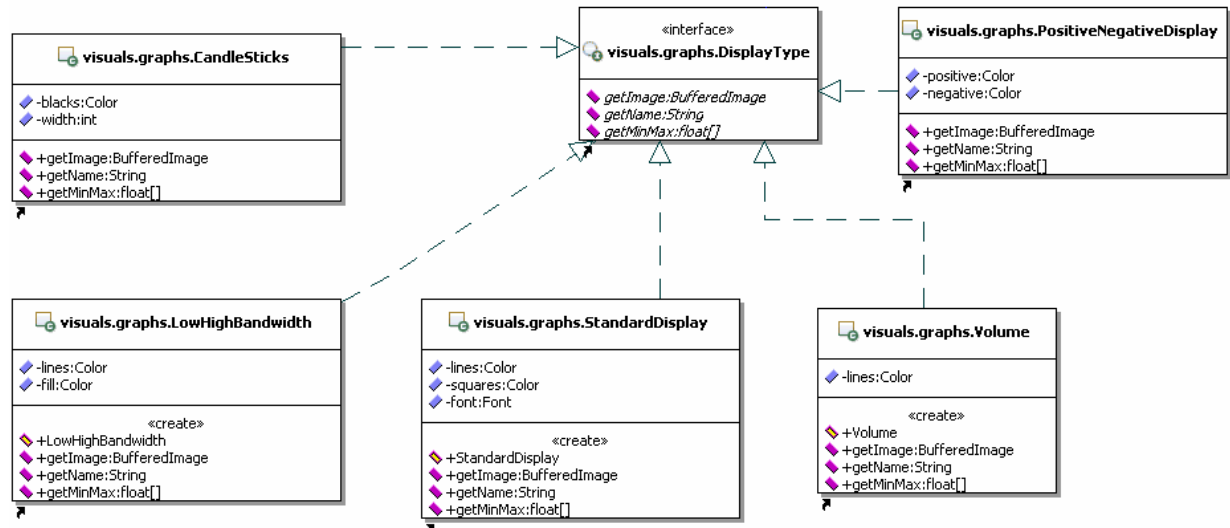
3.3.8 *visuals.graphs*

Abbildung 15 DisplayType

Interface: *DisplayType*

Beschreibung: Dieses Interface definiert die Schnittstellen, welche für einen DisplayType benötigt werden. Diese Klassen übernehmen das Zeichnen der übergebenen Quotes.

Abhängigkeiten: Graph, Ticks

Klasse: *CandleSticks, LowHighBandwidth, PositiveNegativeDisplay, StandardDisplay, Volume*

Beschreibung: Diese Klassen zeigen Beispiele für Implementierungen des DisplayTypes. Je nach Algorithmus und verwendeten Daten werden die Daten unterschiedlich dargestellt..

Implementiert: DisplayType

Kapitel 3 – Architektur

3.4 XML

In der XML Datei „portfolios.xml“ werden die Portfolios des Nutzers gespeichert, damit diese auch nach einem Neustart des Programms wieder zur Verfügung stehen. Die Struktur ist dabei sehr einfach und soll an folgendem Beispiel demonstriert werden:

```
1  <?xml version="1.0" encoding="UTF-8" standalone="no" ?>
2  <portfolios>
3  <portfolio name="Indices">
4      <symbol name="Nasdaq" title="NDAQ" />
5      <symbol name="Swiss Market Index" title="SMI" />
6  </portfolio>
7  <portfolio name="Anlagen">
8      <symbol name="Google" title="GOOG" />
9      <symbol name="Microsoft" title="MSFT" />
10     <symbol name="IBM" title="IBM" />
11 </portfolio>
12 <portfolio name="Beobachten">
13     <symbol name="UBS Namensaktien" title="UBSN" />
14     <symbol name="Yahoo" title="YHOO" />
15 </portfolio>
16 <portfolio name="UBS">
17     <symbol name="Namensaktien" title="UBSN" />
18     <symbol name="Swiss Exchange" title="UBS.SYX" />
19     <symbol name="Frankfurt" title="UBS.FR" />
20 </portfolio>
21 </portfolios>
22
```

Abbildung 16 XML

Anhang A – Referenzen

A Referenzen

- [1] /API/
- [2] <http://www.java.com/de/download/help/6000010300.xml>
- [3] Overview.png
- [4] Anforderungsdokument.pdf