DOKUMENTATION
UMSETZUNG CASE ARBEIT OOA

Aktienportfolio

Ersteller: Fabio Mandanici, Timi Osoko, Michael Rossi, Reto Hegi
Dozent: Stefan Berger

Inhalt

Ausgangslage

Auftrag

In der Lehrveranstaltung Object Oriented Analysis haben wir den Auftrag erhalten, ein Depotverwaltungssystem zu erstellen. Die Grundidee lag darin, es einem bestehenden Kunden zu ermöglichen, gemäss seinen individuellen Präferenzen Wertschriften kaufen und verkaufen zu können. Mit dem Wissen, welches wir uns mit der Unified Modeling Language und dem objektorientierten Programmieren im Allgemeinen angeeignet haben, haben wir ein Konzept auf die Beine gestellt. In der Lehrveranstaltung Programming Tool 1 geht es nun darum, dieses Konzept effektiv umzusetzen und mit Java zu programmieren.

Beschreibung des Projekts

Mit dem in OOA erarbeiteten Konzept soll in Programming Tool 1 das Depotverwaltungssystem programmiert werden. Die Grundidee des Konzepts haben wir in der Gruppe anhand von 3 Use Cases entworfen und festgehalten. Mit den Use Cases als Basis, haben wir ein Klassendiagramm entworfen, welches uns für die Programmierung als Grundlage dienen sollte. Damit sollte sichergestellt werden, dass beim Programmieren keine Fragen und Unklarheiten auftauchen. Ziel des Projektes war es ein Verwaltungssystem für Aktien zu gestalten, bei welchem der Kunde die Anlageentscheidung aufgrund des Anfangsbuchstabens der Aktien wählen, als auch den Betrag, den er investieren möchte dementsprechend anpassen kann. Zudem sollten vor der Investition in die Wertschriften die Kundenangaben überprüft sowie ein Depot für den Kunden eröffnet werden.

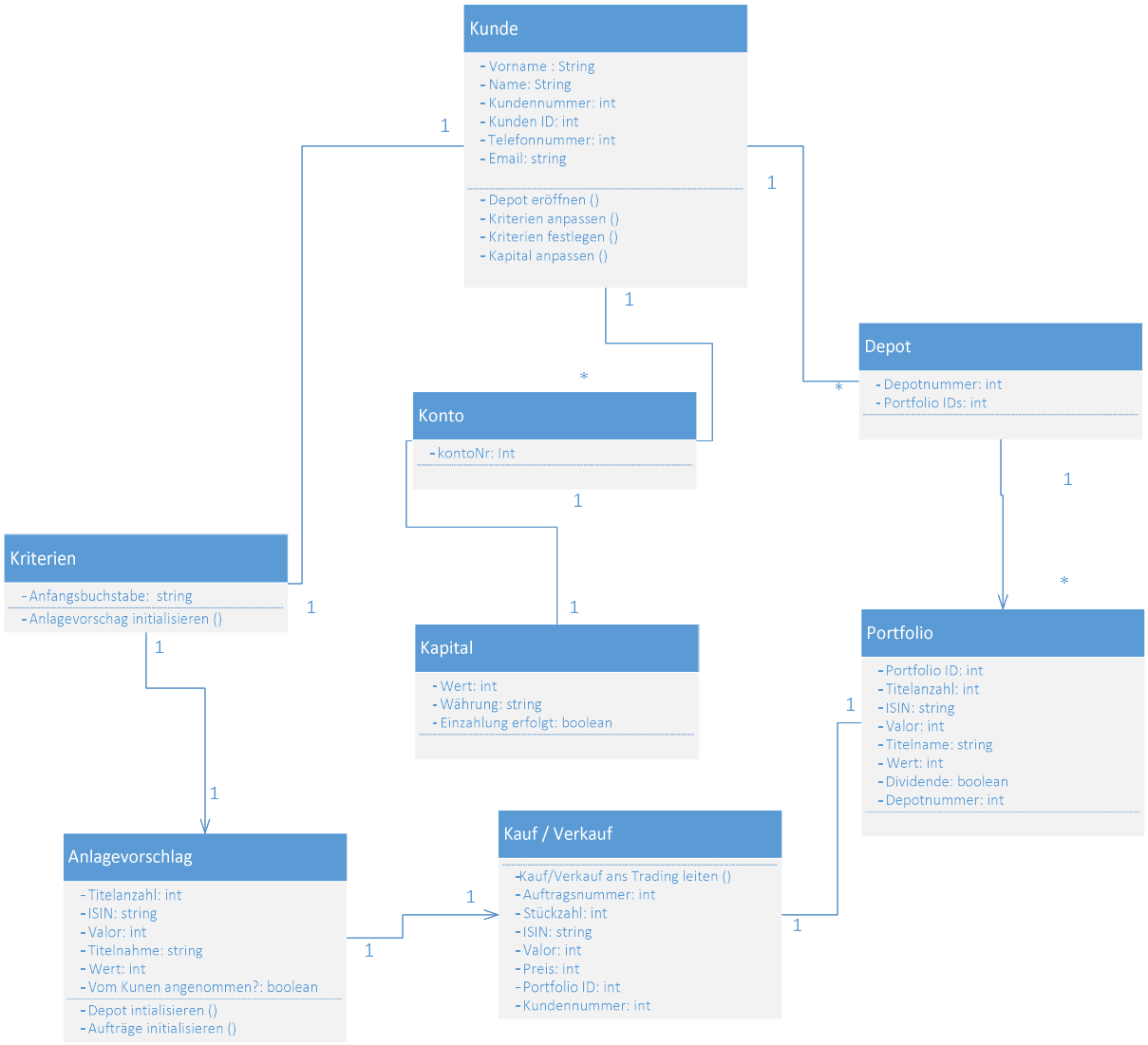
Tools & More

Für die Umsetzung des Konzeptes haben wir die folgenden Tools benutzt:

- Visio, um das Klassendiagramm darzustellen und bearbeiten zu können
- Eclipse, eine integrated development environment (IDE), das heisst eine Entwicklungsumgebung für Java welche für die Entwicklung von Software benutzt wird
- Github, ein webbasierter Online-Dienst, auf dessen Servern man die Quellcodes speichern und somit der Gruppe zugänglich machen kann
- Die Kommunikation verlief über WhatsApp
- Word um die finale Dokumentation zu erstellen.
- OneNote als zentrale Austauschstelle für die Dokumente sowie Notizen.

Erstes Klassendiagramm

Dies ist das Klassendiagramm, wie es aufgrund unserer Konzeptarbeite aus OOA entworfen wurde. Im Zentrum soll die Klasse **Kunde** stehen, unter welcher jegliche Informationen unserer Kunden gespeichert. Die Klassen **Konto**, **Depot** und **Kriterien** sind direkt der Klasse Kunde angegliedert. Das Konto wird noch von der Klasse **Kapital** ergänzt. In der Klasse **Kriterien** wird der Anfangsbuchstabe festgelegt, welche den **Anlagevorschlag** initialisiert. Die Klasse **Kauf/Verkauf** setzt den Anlagevorschlage um welcher danach in der Klasse **Portfolio** dargestellt wird. Das Portfolio wiederum ist dem Depot angehängt.



Umsetzung des Auftrages und aufgetretene Schwierigkeiten

Vorgehen in der Gruppe

Unsere Gruppe setzt sich zusammen aus vier Mitgliedern, welche bisher alle keine Vorkenntnisse oder Erfahrungen im Bereich des Programmierens aufweisen können. In den Lehrveranstaltungen hat sich jedoch bald einmal gezeigt, dass es zweien der Gruppe leichter fällt die Kunst des Programmierens zu verstehen. Dementsprechend hat sich auch die Umsetzung des Programms gestaltet. Das Programmieren des Depotverwaltungssystems lag im Fokus von Timi Osoko und Michael Rossi. Michael Rossi hat sich dabei aufgrund seines Fachwissens als Bankmitarbeiter mehr mit der Einbindung der Aktieninformationen beschäftigt, während Timi Osoko sich mit dem ganzen administrativen und grösseren Teil des Programms auseinandergesetzt hat. Fabio Mandanici und Reto Hegi haben sich mehrheitlich den ganzen administrativen Arbeiten und der Dokumentation verschrieben.

Aufgetretene Schwierigkeiten beim Programmieren

Während dem Programmieren des Depotverwaltungssystems sind verschiedene Schwierigkeiten aufgetreten. Wir haben versucht, diese Herausforderungen und Probleme in drei Themenbereiche zu unterteilen.

Probedurchlauf

Beim Probedurchlauf haben wir uns verschätzt. Wir haben uns erst mit dem Probedurchlauf befasst, als wir das Grundgerüst des Programms schon programmiert hatten. Beim anschliessenden Programmieren des Probedurchlaufs sind uns dann die Fehler im Grundgerüst aufgefallen, welche anschliessend wieder beseitigt werden mussten. Hätten wir von Anfang an mit dem Probedurchlauf gestartet, wäre es uns stets klar gewesen wie das Programm aufgebaut sein muss. Denn wie können wir sicher sein, dass alles nach Kundenwunsch durchgeführt wird und dass alles richtig funktioniert, wenn wir es nicht testen? Ein Programm kann auch Denkfehler beinhalten, die man nicht auf den ersten Blick erkennt.

Klassen

Die Problematik beim Arbeiten mit den Klassen haben sich bei der Übersicht in den einzelnen Klassen sowie der Verbindung von verschiedenen Klassen ergeben. Wird viel in einer Klasse programmiert, kann diese schnell zu einer Code-Überflutung führen. Mit vielen Methoden, Instanzvariabeln und Konstruktoren, wird der Code trotz Ordnungsfunktion unübersichtlich. Wenn versucht wird ein Fehler zu beheben, ohne denn Überblick über jegliche Informationen der Klasse zu haben, wird das Debugging problematisch und komplizierter als nötig.
Wir gehen davon aus, dass dies viel mit der Übung und Erfahrung beim Programmieren zu tun hat. Denn wenn sich das Auge solche Code-Mengen gewöhnt ist, wird es auch einfacher sich auf die Probleme des Programms zu konzentrieren.

Allgemeine Herausforderungen

Nachdem der OOA Auftrag fertig war und es ans Programmieren gehen konnte, mussten wir uns zuerst reorganisieren. Als Erstes mussten wir eine Lösung finden, wie wir den Code untereinander austauschen können. Wir haben uns für Github entschieden, da uns dies empfohlen wurde und auch eine grosse Online-Community dahintersteckt. Das Team hatte Anfangsschwierigkeiten mit der Synchronisation auf das Medium jedes einzelnen Teammitgliedes. Natürlich haben wir an dieser Lösung herumexperimentiert. Wir wollten nicht, dass unser Programm plötzlich gelöscht war. Leider hatten wir immer wieder Probleme, bei der gleichzeitigen Bearbeitung des Codes. Darum haben wir uns entschieden, jeweils zu kommunizieren, wenn jemand am Programmieren ist. Diese Lösung war

jedoch suboptimal, da dies uns in der Zeitplanung einschränkte. Dennoch hielten wir dies für die beste Lösung um unser Projekt fertig zu stellen.
Als wir uns der Aufgabe annahmen, stellten wir schnell fest, dass uns die richtige Vorgehensweise fehlte. Uns war bewusst wie man die einzelnen Klassen programmieren und bearbeiten soll, jedoch fehlte uns das Knowhow für das Gesamtbild. Wo fängt man an? Wie teilt man die Arbeit am besten auf?
Für die Teammoral war dies ein Tiefpunkt. Somit mussten wir uns genau überlegen wie wir vorgehen wollten und was unsere Milestones sind. Es war uns klar, dass die Zeit läuft und da dies unser erstes Java Projekt war, wussten wir nicht wieviel Zeit wir benötigen um die Milestones zu erreichen. Unser Hauptziel war es, dass unser Programm funktioniert und wir alle notwendigen Klassen im Programm haben. Unser grosses Ziel war das Programm soweit erstellen zu können, dass bei Zeitknappheit nur noch die «Fleissarbeiten», wie z.B weitere Arrays hinzuzufügen, fehlen würden. Die Qualität sowie die Funktionen wollten wir vorweisen können.
Im Laufe des Projekts ergab sich immer wieder ein Muster. Bei jeder gefundenen Lösung ergaben sich wieder 2 neue Fehler. Auch die Verbindung mit mehreren Klassen und der Austausch von Informationen der verschiedenen Array Lists, hat uns vor ein grosses Problem gestellt. Dennoch konnten wie uns nach mehreren Tiefs wieder motivieren und erreichten unseren Milestone. An dieser Stelle ein grosses Dankeschön für die hilfreichen Tipps und Hilfe von unseren Mitstudenten.

Kurzgefasst, durch den zeitlichen Druck haben wir uns auf das Wesentliche konzertiert und die Qualität abgeliefert, die mit unserem Fachwissen möglich war. Viele IT-Projekte fallen in ihrem Zeitplan zurück. Wir denken aber, dass wir das Beste herausgeholt haben aus dem was möglich war und wir können hinter unserem Produkt stehen und dieses repräsentieren.

Umsetzung der Software

Teil 1 – Vom Kunden zum Kapital

Im ersten Teil liegt der Fokus auf der Auswahl, Bearbeitung und Erstellung von Kundenbezogenen Informationen. Dieser werden in 3 Klassen generiert und bearbeitet: Kunde, Konto, Kapital.

Klasse Kunde

Die Klasse Kunde verfügt über jegliche für die Unternehmung wichtige Kundeninformationen: Kundenummer, Vor- und Nachname, Adresse, E-Mail und Telefonnummer. Um einen Neukunden zu erfassen werden sämtliche Informationen benötigt, wobei die Kundennummer vom System selbst generiert wird. Mithilfe dieser Informationen wird ein Kunde instanziiert und in einer Array Liste namens kundenListe gespeichert.

Methoden

addKunde(): Fügt den instanziierten Kunden der Array Liste kundenListe hinzu.

addKonto(): Fügt ein instanziiertes Konto dem Kunden hinzu. Dies wird über die Array Liste kontoListe getätigt.

createDepot(): Nach Eingabe einer validen Kundennummer, wird diesem Kunden ein neues Depot hinzugefügt, welches anschliessend die einzelnen Portfolios tragen wird. Was hierbei noch nicht implementiert wurde ist das Kontrollieren der einzelnen Kundeninformationen, welches vor der Depoterstellung durchgeführt werden sollte.

addDepot(): Fügt ein instanziiertes Depot der Array Liste depotList hinzu.

kundenEingabe(): Ist das Herzstück der Klasse Kunden. Es ermöglicht die Auswahl eines bestehenden Kundenkontos mittels deren Kundennummer oder das erstellen und instanziieren eines neuen Kunden mittels den oben beschriebenen Kundeninformationen.

frageKonto(): Diese Methode kommt zum Einsatz, wenn ein neuer Kunde erstellt wird. In dieser Methode kann der User bestimmen ob beim Neukunden direkt auch ein neues Konto angelegt werden soll oder ob schon ein Konto besteht, dass dem Neukunden zugewiesen werden kann. Mittels ‘Ja’ & ‘Nein’ Input kann diese Methode gesteuert werden.

Klasse Konto

Die Klasse Konto verläuft ganz im Beispiel der Klasse Kunde. Einzige Ausnahme ist hierbei, dass keine Informationen vom Kunden eingegeben werden, sondern dass die Informationen dieses Kontos, welches lediglich die Kontonummer betrifft, automatisch durch das Programm erstellt wird.

Methoden

addKapital(): Fügt das instanziierte Kapital der Array Liste kapitalListe hinzu.

kontoEingabe(): Folgt dem Beispiel der kundenEingabe() aus der Klasse Kunde. Auch hier kann ein bestehendes Konto ausgewählt oder ein neues erstellt und instanziiert werden. Die Kundennummer wird hierbei, wie oben erwähnt, vom System selbst erstellt.

frageKapital(): Folgt dem Beispiel des frageKonto(). Diese Methode wird nur bei der Erstellung eines neuen Kontos ausgeführt. Es fragt den User ob dieser dem Konto direkt eine bestehende Kapital Instanziierung anfügen möchte oder ob ein neues Kapital instanziiert werden soll. Mittels ‘Ja’ & ‘Nein’ Input kann diese Methode gesteuert werden.

Klasse Kapital

In der Klasse Kapital wird die Währung und das Kapital definiert. Zusätzlich muss mit einem boolean bestätigt werden, ob dieses Kapital schon eingezahlt wurde oder nicht.
Diese Klasse verfügt über keine eigenen Methoden, da sämtliche Kapital bezogenen Funktionen durch andere Klassen ausgeführt werden.

Teil 2 – vom Anlagevorschlag bis zum endgültigen Depot

Klasse abstractPortfolio

Dies ist eine abstrakte Klasse, welche den Klassen Anlageuniversum und Portfolio mitgibt, welche Array Liste sie verwenden sollen. Dies ist die Array Liste gewaehlteListe.

Klasse Aktie

Diese Klasse übernimmt die aktuellen Aktien Kurse aus einem externen Yahoo API. Es übernimmt aus dem API der Name der Aktien und dessen Preis.
Dementsprechend ist es für das Programm wichtig, dass diese API installiert ist. Dies sollte grundsätzlich keine Problematik sein, da wir es in das Programm miteingefügt haben und es deshalb nicht vom User heruntergeladen werden muss.

Klasse Anlageuniversum

In dieser Klasse werden die einzelnen Anlagestrategien definiert. Unsere Anlagestrategien unterscheiden sich durch den jeweiligen Anfangsbuchstaben der Aktie. Das Anlageuniversum definiert nun welcher Buchstabe (welche Strategie) welche Aktien beinhaltet.

Enumeration Anlage

Die Klasse verfügt über eine Enumeration welche sämtliche Buchstaben von A bis Z beinhaltet. Diese Enumeration wird anschliessend in der Methode selection() benötigt.

Methoden

selection(): In dieser Methode wird die Auswahl der Anlagestrategie durchgeführt. Die Auswahl der Strategie wird mittels cases gemacht. Aus diesem Grund habe wir 26 verschiedene cases angelegt, für jeden Enumeration (für jede Strategie) einen. Durch die Eingabe des Buchstaben wird hiermit der dazugehörige case und somit das Aktienpakt ausgewählt.
Mit einer neuen Instanzvariabel ‘summe’ wird anschliessend der summierte Preis aller Aktien in diesem Aktienpaket für den späteren Gebrauch gespeichert.

einverstanden(): Es wird der Kunde gefragt, ob er mit diesem Aktienkauf einverstanden ist. Bei Einverständnis wird die Methode annahme() durchgeführt.

annahme(): Bei der Annahme der einverstanden() wird in dieser Methode das Portfolio auf unsere Anlagestrategie, welche in der Methode selection() definiert wurde, angepasst.

Klasse Portfolio

Diese Klasse übernimmt die Werte des Anlageuniversums, wenn diese definitiv sind. In einem späteren verfahren wird implementiert, dass im gleichen Schritt wie der Übergabe der Informationen an diese Klasse, der Gesamtbetrag der Aktienpakete dem Kapital des Kunden abgezogen werden.
Diese Klasse verfügt über eine Array Liste portfolioList in welche das Portfolio gespeichert wird.

Klasse Depot

Der letzte Schritt eines kompletten Durchlaufes findet im Zusammenhang mit der Klasse Depot statt. In dieser Klasse wird mittels der Methode addPortfolio das definitive Portfolio hinzugefügt und somit durch die Klasse Depot dem Kunden zugewiesen.

Teil 3 – Das Testing

Das Testing unseres Projektes setzt sich aus zwei Programmteilen zusammen. Der Programmteil Display ist im Prinzip das Herzstück des Testdurchganges. Der zweite Programmteil Probedurchlauf dient zur Testung der reinen Code-Bausteine.

Programmteil Probedurchlauf

Im Programmteil Probedurchlauf wurden die Grundbausteine des Depotverwaltungssystems getestet. Diese Klasse dient dazu einen neuen Kunden zu erfassen, ein neues Konto anzulegen und einem Kunden zuzuweisen, dem Konto ein Kapital anzugeben und ein Depot zu eröffnen. Weiter kann in diesem Programm ein Aktienpaket anhand eines Buchstaben ausgewählt werden, das gewählte Aktienpaket wird in ein Portfolio umgewandelt und einem Depot weitergegeben. Mit Hilfe eines mathematisch berechneten Integers werden dabei so viele Aktienpakete gekauft bis keine weiteren ganze Pakete mehr gekauft werden können.

Hier ist zu notieren, dass keine Inputs von aussen eingebbar sind und auch einige Methoden umgangen wurden um die reinen Bausteine des Systems zu überprüfen.

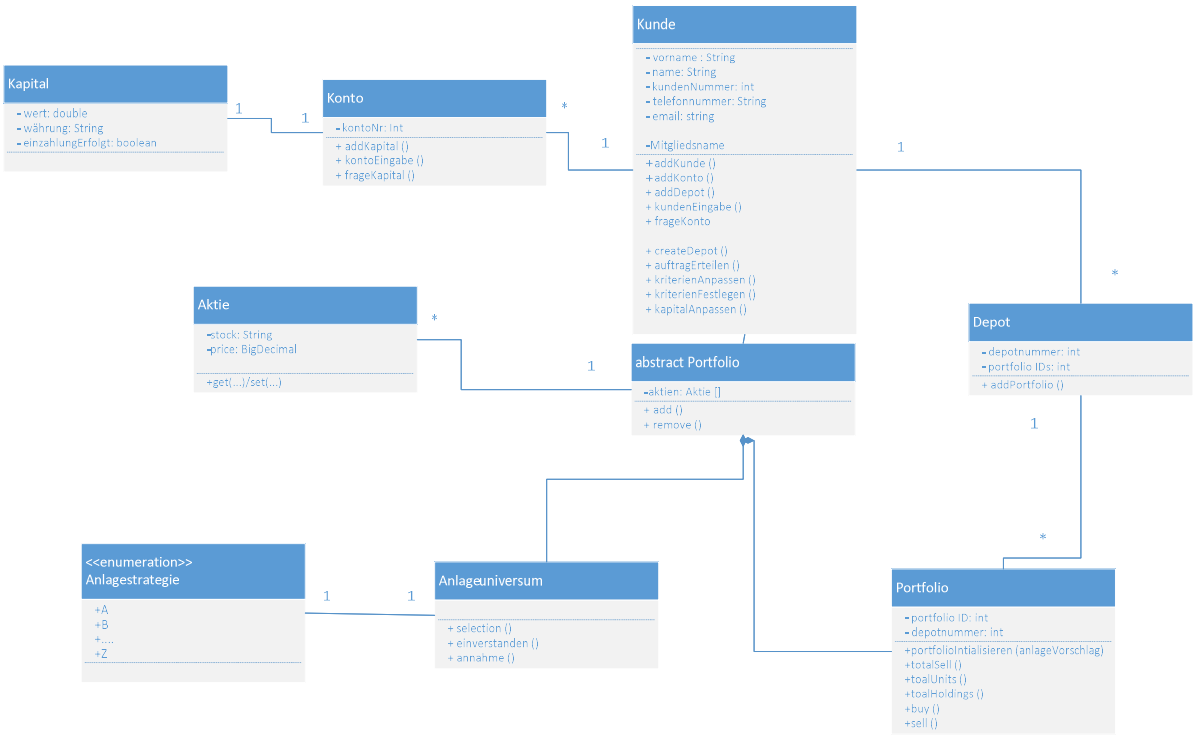
Programmteil Display

Beim zweiten Programmteil Display werden in einem Durchlauf bei korrekter Eingabe diverse Informationen und Schritte abgefragt. Display soll der effektive Ablauf des Programmes simulieren und deswegen auch Inputs von aussen über einen Scanner annehmen.
In einem ersten Schritt wird gefragt, ob man ein bestehender Kunde oder ein neuer Kunde sei. Ein bestehender Kunde kann seine Kundennummer eingeben. Bei der Auswahl neuer Kunde wird eine neue Kundenummer generiert und der Kunde wird nach Vorname, Name, E-Mail-Adresse und Adresse gefragt. In einem nächsten Schritt wird man gefragt, ob ein neues Konto eröffnet werden möchte. Wenn dies bejaht wird, wird gleich eine neue Kontonummer generiert. Zu guter Letzt kann auf das neue Konto je nach Wunsch auch gleich ein Betrag einbezahlt werden.

Dieser Probedurchlauf ist noch nicht zu Ende Programmiert. Es fehlt noch der komplette Teil 2, Anlagevorschlag bis Depot. Dieser wird in einer späteren Instanz noch programmiert und implementiert werden.

Neues Klassendiagramm

Während der Programmierung unseres Depotverwaltungssystems mussten wir zudem immer wieder Anpassungen am Klassendiagramm vornehmen, da die ursprünglichen Ideen nicht wie gewünscht umgesetzt werden konnten. Nachfolgend ist das neue, aktuelle Klassendiagramm ersichtlich.



Noch nicht implementierte Features

Es gab mehrere Instanzen die zu einer Verzögerung unseres Zeitplanes führten. Zum einten ist es die Problematik des «First Times». Da es das erste Projekt unseres Teams in einem Informatikbereich ist, war der Ablauf nicht von Anfang an gegeben. Wir mussten uns an den Ablauf und an das Projekt herantasten, was uns Zeit gekostet hat. Ein anderer Grund ist die Unterschätzung des Aufwandes. Bei der Berechnung des Zeitplans haben wir nicht mit grösseren Programmierproblematiken gerechnet, die uns Stunden und Tage unserer Zeit beraubten.

Aus Gründen dieser Zeitverzögerung war es uns bis zum heutigen Tag nicht möglich das Projekt zu Ende zu Programmieren. Für die Komplettierung des Programmes rechnet das Team einen zeitlichen Aufwand von ca. 2-4 Wochen ein.

Dies hat als Konsequenz, dass einige Teile des Programmes noch nicht getestet oder implementiert sind. Betroffen davon sind die Features: Anpassung des Kapitals, Anpassung des Anlagevorschlags, sowie Kontrollen bei der Mutation von Kundendaten.

Zeitliche Verzögerungen sind im IT-Sektoren keine Einzelheit, da der effektive Aufwand sehr schwer im Voraus zu berechnen ist. Das Team ist aber zuversichtlich mit der zusätzlichen Zeit das Projekt beenden zu können.

Nächste Schritte

Der aktuelle Stand unseres Programmes wurde im neuen Klassendiagramm beschrieben. Der nächste Schritt wäre die Weiterführung der Klasse Display. Display soll uns eine Input Maske in der Konsole geben, in welche die einzelnen Daten eingegeben werden. Bis jetzt können wir Kunden, Konten und Kapital so auswählen und neu erfassen.
In einem zweiten Teil soll jetzt die ganze Depotangelegenheit über Display bearbeitet werden können.

Display

Zuerst sollen bei einer Depoteröffnung die Kundeninformationen kontrolliert und falls nötig, aktualisiert werden. Als nächsten Schritt wird das zu investierende Kapital eingegeben und mit dem Kontobestand verglichen. Anschliessend wird eine vordefinierte Anlagestrategie ausgewählt, welche seinerseits wieder das zu kaufende Aktienpaket definiert. Es wird berechnet, wie viel Mal solch ein Aktienpaket mit dem vorgegebenen Betrag gekauft werden kann und diese Information wird dann über die Konsole vermittelt.
Wird der Anlagevorschlag bestätigt, so wird dieser zu einem tatsächlichen Portfolio umgewandelt und dem Depot hinzugefügt. Das Geld wird dem Konto abgezogen und dem Kunden wird vermittelt, dass sein Portfolio gekauft wurde.