Verwaltung von Aktiensdaten

Allgemeine Informationen

Mit diesem Programm wollen wir Aktiendaten in einer Hashtabelle speichern. Es ist auch möglich, eine Aktie in der Tabelle hinzuzufügen, zu suchen oder sogar zu löschen. Es ist auch möglich, Kursdaten in eine Aktie zu importieren. Basierend auf den Aktienskursdaten können Sie auch ein ASCII-Diagramm als textuelle Datei drucken. Wenn Sie möchten, können Sie diese Hashtabelle in einer Binärdatei speichern (für Menschen nicht lesbar) oder es ist auch möglich, Binärdateien zu laden.

Aufgrund des Alpha-Faktors von 50% ist Platz für 2003 (Primzahl in Form 4j+3) reserviert, und dann sind es etwa 1000 Stellen, die wir haben wollen.

Dieses Program wurde in Java Programmiersprache geschrieben. Es besteht aus 5 Klassen und eine Enum Klasse. Hauptklasse ist *MainMenu,* wovon alle Benutzeraktien verwaltet werden. Die Enum Klasse wird als Kollektion für Hauptmenü Optionen (ADD, DEL, IMPORT, SEARCH, PLOT, SAVE, LOAD, QUIT) verwendet. Alle Input-Output Funktionen (Ausgabe von ASCII-Grafik, Speicherung und Laden der HashTabelle, Einlesen der Aktienskursdaten) sind in der *IOHandler* Klasse geschrieben.

Falls ein Fehler aufgetreten ist (Dateieingabe-Ausgabestrom) oder aufgrund einer NULL-Ausnahme, wird eine Meldung ausgegeben, dass etwas schief geht.

Beschreibung der Datenstruktur und Klassen

Stock.java

In einer Aktie können Aktiendaten eingelesen werden. Klasse Stock bietet die Möglichkeit, ein Objekt mit Aktiennamen (String), WKN (String), Aktienkürzel (String) und Kursendaten zu bauen. Es ist auch möglich Close Kursdaten zu sortieren, damit ein ASCII-Diagram korrekt ausgedrückt werden kann. Dafür wird ein QuicSort Algorithmus benützt. Um Namen und Kürzeln in einer Hashtable mit einer Aktie besser zu verwalten ist es auch möglich Hash Index von Name und Kürzel zu speichern.

HashTable.java

Eine Hashtabelle hat zwei Arrays vom Typ Stock. Es wurde beschlossen, auf diese Weise vorzugehen, um eine bessere Handhabung von Aktiensnamen und Kürzeln zu ermöglichen und die Möglichkeit zu haben, nach Kürzeln oder Namen zu suchen. Beide Arrays haben Referenz zu gleichem Objekt im Speicher.

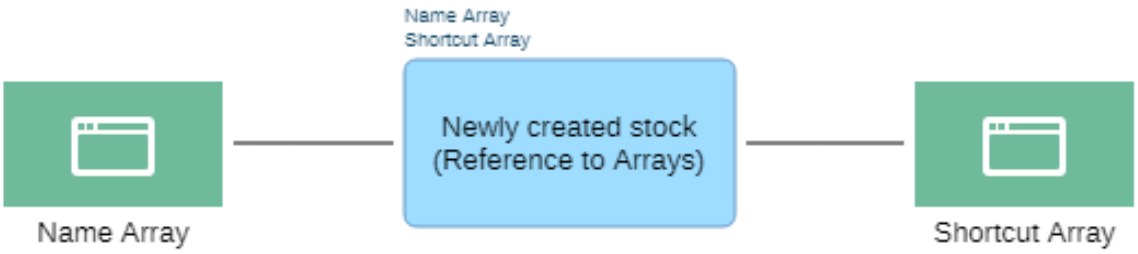
private int generateHashCode (String name)

HashCode Funktion nimmt einen String Wert und macht einen Code, der später als Array Index für die Speicherung-Löschen-Suchen der Aktien verwendet werden. Kriterium aus Vorlesungfolien!

```java
/**
 * Function used to generate hash code from name/shortcut for one stock
 *
 * @param name Name/Shortcut from which will be hash code generated
 * @return hash code
 */
private int generateHashCode(String name) {
    int hash = 0;
    int hashBase = 31;
    for (int i = 0; i < name.length(); i++) {
        hash = (hash * hashBase + name.charAt(i)) % CAPACITY;
    }
    if (hash < 0) {
        hash += CAPACITY; // in case that hash is negative value
    }
    return hash;
}
```

public void add(Stock stock)

Funktion zum Speichern neu erstellte Aktie in beiden Arrays. Einmal wird die Hash-Funktion für den Namen und das zweite Mal für die Kürzel verwendet. Falls ein Objekt bereits in einem Index vorhanden ist, wird dafür eine alternierende quadratische Sondierung verwendet, bis der freie Platz gefunden wird.



public Stock search (String userInput, boolean toPrintList)

Meist verwendete Funktion zum Suchen. Jedes Mal, wenn wir überprüfen müssen, ob ein Objekt existiert, wird diese Funktion aufgerufen. Es funktioniert nach dem gleichen Prinzip wie die ADD-Funktion (quadratisches Sondieren). Es wird zuerst geprüft, ob unser Objekt den ersten Hashwert in beiden Arrays hat. Wenn ja, wird dieses Objekt zurückgegeben. Wenn nicht, wird die quadratische Prüfung verwendet, bis das Objekt gefunden wird. Wird nichts gefunden, wird NULL zurückgegeben. Es gibt auch die Möglichkeit, falls wir Kursdaten ausdrucken sollen, die print Funktion aufzurufen und davon Kursdaten zu drucken.

public void remove (Stock stockToDelete)
Um ein Objekt zu löschen, rufen wir zuerst die Suchfunktion auf, um dieses Objekt zu erhalten. Wenn es existiert, werden wir Objekt-Hash-Werte für Name und Kürzel verwenden und leicht aus Hash-Tabellen entfernen.

CourseData.java
Kursdaten der letzten 30 Tagen werden in einer ArrayListe der Klasse CourseData gespeichert. Eingelesene Daten aus .csv Datei sind: Datum (LocalDate), Open (Double), High (Double), Low (Double), Close (Double), Adjusted Close (Double) und Volume (Integer).

IOHandler.java
Wie bereits erwähnt, ist es möglich, eine Hash-Tabelle zu speichern und zu laden. Dafür wird Java-Serialisierung verwendet und es werden Binärdateien erstellt.

Es ist auch möglich, Kursdaten in eine Aktie zu importieren. Es liest .csv-Dateien und kann maximal 30 oder weniger Tage gelesen werden. Es liest jede Zeile und erstellt jedes Mal ein CourseData-Objekt für jede Zeile. Diese Daten werden in einer ArrayList gespeichert. Falls Daten bereits einmal importiert wurden, werden Sie vom Programm gefragt, ob Sie die Daten überschreiben möchten. Wenn nicht, werden sie normalerweise importiert. Auch nach jeder erfolgreichen Aktion mit IO Handler erhalten Sie die Nachricht, dass alles in Ordnung ist. Die Nachrichten beginnen mit zwei Sternen (**)

Zum Zeichnen eines ASCII-Diagramms müssen Sie zuerst die Datumsangaben sortieren und dann korrekt drucken.

MainMenu.java
Dies ist die Hauptfunktion für Benutzereingaben. Es behandelt jede Aktion vom Hauptmenü aus.

Aufwandsabschätzung
N = 2003 (Unsere Tabellengröße)

Alpha = m / N = 1000 / 2003 = 0.49924 = 49.925 % Füllgrad

Wie viele Operationen sind jeweils bei 1000 Aktien notwendig?

- Durchschnittlich: (1 / (1 – m/N)) * 1000 = 1997

	ADD		DEL		SEARCH	
	Worst Case	Best Case	Worst Case	Best Case	Worst Case	Best Case
Unsere Hashtable	O(N)	O(1)	O(N)	O(1)	O(N)	O(1)
Array	O(N)	O(N)	O(N)	O(N)	O(N)	O(N) oder O(log N))
Linked List	O(N)	O(1)	O(N)	O(1)	O(N)	O(N)