Data Structures and Object Oriented Programming

Portfolio Analyzer

Georges Hawi

Table of Contents

Description

Description and Hierarchies

The project aims to create a simple way to compare the progression of your stock portfolio. It will allow you to see how each stock is performing, with rough projection estimates for dividends as well as for the stock's performance over the next 5 years. The project includes 2 accounts, one that is tax-free and another that is taxable, where there is also an output telling you how much is taxable.

Hierarchies:

Security implements Comparable<Security> (abstract)
- CommonSecurity (abstract)
  - CommonStock
  - CommonEtf
- DividendSecurity (abstract) implements DividendPaying
  - DividendStock
  - DividendEtf

Account (abstract)
- TaxableAccount implements TaxableIncomeCalculable
- NonTaxableAccount

Interfaces and Runtime Polymorphism

Interfaces:

→ TaxableIncomeCalculable

Provides a method to calculate the amount to add to your taxable income.

→ DividendPaying

Provides a method to project dividend earnings over 5 years (year amount is changeable).

RunTime Polymorphism:

→ compareTo(T o1)

Defined in the Security class, it will be overridden in the DividendSecurity subclasses to have it compare based on dividend returned (%) AND the price increase.

TextIO and Data Structures

TextIO:
→ readData(String filePath, Account account)
Implemented in the Main Class, it will read data from a CSV file and will populate the TreeMap<String, Security> map in an account, adding securities to that account.

→ writeReport(String filePath, List<Account> accounts)
Takes a list of accounts and outputs the report, which includes both the taxable and non-taxable accounts, allowing you to see how the account is doing, along with projections, etc.

Data Structures:
→ ArrayList<Account> in the main class to store both accounts and pass them to the export method.

→ TreeMap<String, Security> in the Account super-class that stores the securities and the tickers associated with each security in an account.

Comparable and Comparator

Comparable:
It will be used to compare the stock performances and allow for sorting in the report. It will be implemented in the superclass Security and will sort descendingly based on the % return.
The method will be overridden in the DividendSecurity subclasses to have it compare based on dividend returned (%) AND the price increase.

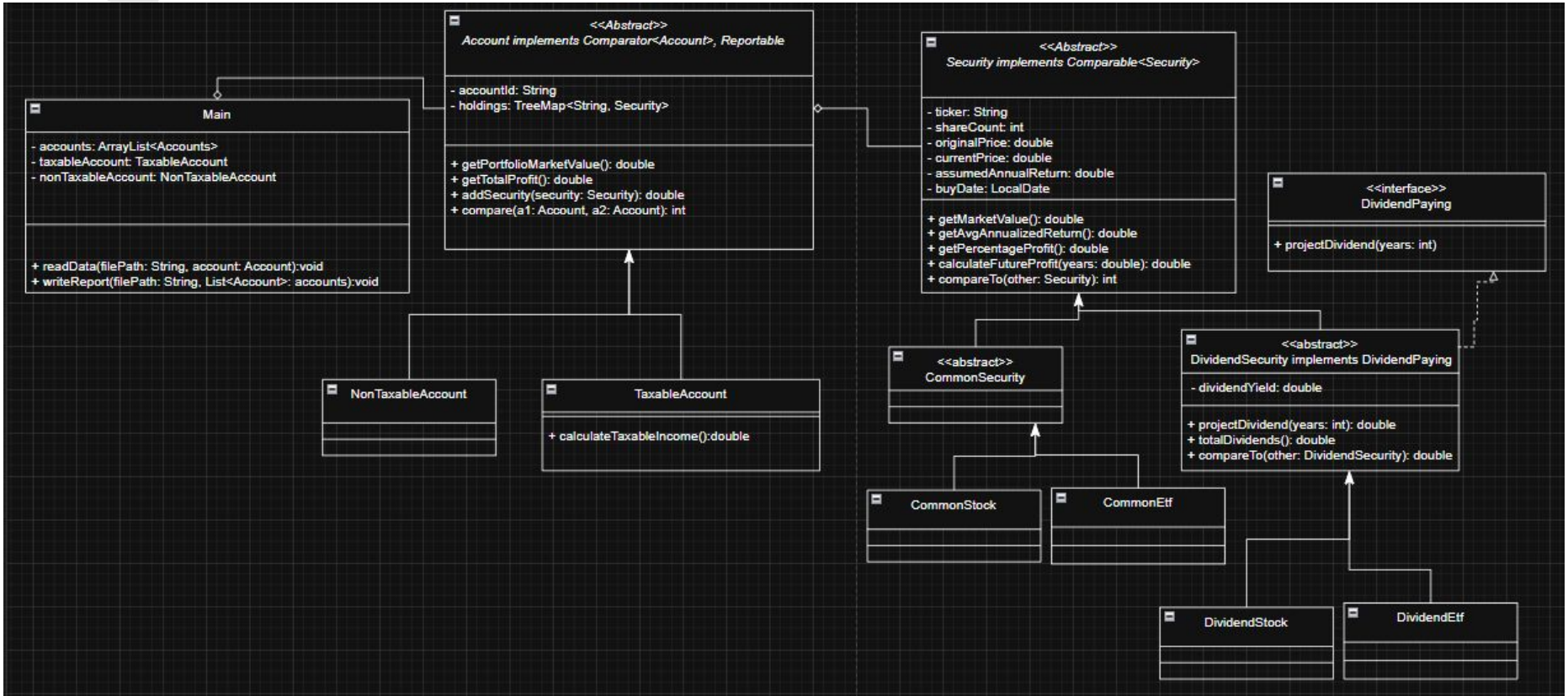
Comparator:
Will be implemented in the Account superclass and will allow the two accounts to be compared based on absolute return. Sorted descendingly.

User Interaction

The user modifies the two CSV files he wants to input.

If need be, the code can be modified to allow for the creation of different accounts by the user, but for my use case and in general, I won’t have more than 2 accounts, and only having to modify the CSV file (which will be done on Excel spreadsheets) is a lot simpler and makes the usage easier (no need to interact with a menu and create an account and then generate a report etc). You just modify the CSV files and then generate the report.

Class Diagram



Features

(From each class)

Account and TaxableAccount

Account (Abstract Super-class):

- Add a security to the Holdings TreeMap
- Get the market value of the entire account
- Get the total profit in $ for the entire account

TaxableAccount (implements TaxableIncomeCalculable)

- Calculate the amount to add to your taxable income at the end of the year from an account's profit (Method Overridden from an interface)

Security and DividendSecurity

Security (Abstract Super-class):

- Calculate the future profit generated from that security over a number of years based on an assumed annual return.
- Calculate the percentage profit of that security since the original buy price
- Get the market value of the total holdings of a security (meaning current price * share count)
- Get the average annual return of a security since buying it (% profit per year)

DividendSecurity (Abstract Super-class):

- Estimate how much you earned from dividends since buying the security
- Project a dollar amount to earn from dividends over a number of years (Method Overridden from an interface).

Main class

Main:

- Populate an account's holdings by reading data from a CSV file
- Generate the report on the account based on the read data

Screenshots

Input for Non-Taxable account

```
ticker,shareCount,originalPrice,currentPrice,assumedAnnualReturn,buyDate,dividend,type
NVDA,5,98,110,7,2025/5/6,0,etf
AAPL,10,150,175,10,2024/11/15,0.8,stock
GOOG,3,2500,2700,12,2023/7/20,0,stock
TSLA,8,800,780,15,2025/1/10,0,stock
VOO,20,350,380,8,2024/3/1,1.5,etf
MSFT,12,280,310,9,2023/12/5,1.1,stock
SPY,15,400,420,7,2024/6/20,1.3,etf
```

Input for Taxable account

```
ticker,shareCount,originalPrice,currentPrice,assumedAnnualReturn,buyDate,dividend,type
NVDA,5,98,110,7,2025/5/6,0,etf
AAPL,10,150,175,10,2024/11/15,0.8,stock
GOOG,3,2500,2700,12,2023/7/20,0,stock
TSLA,8,800,780,15,2025/1/10,0,stock
VOO,20,350,380,8,2024/3/1,1.5,etf
MSFT,12,280,310,9,2023/12/5,1.1,stock
SPY,15,400,420,7,2024/6/20,1.3,etf
AMZN,7,120,135,11,2025/2/18,0,stock
MMA,1000,100,10,5,2024/6/20,0,stock
```

Generated Report; accounts are sorted based on total profit (½):

```
Account Report for ID: 1
Holdings:
-> AAPL: 10 shares @ $175.0 | Profit (%) -> 16.666666666666664 | Avg Annual Return -> 34.369114877589446 | Projected returns over 5 years ($) -> 402
-> GOOG: 3 shares @ $2700.0 | Profit (%) -> 8.0 | Avg Annual Return -> 4.417549167927382 | Projected returns over 5 years ($) -> 1057.4050099200003
-> MSFT: 12 shares @ $310.0 | Profit (%) -> 10.714285714285714 | Avg Annual Return -> 7.477465173449876 | Projected returns over 5 years ($) -> 553.
-> NVDA: 5 shares @ $110.0 | Profit (%) -> 12.244897959183673 | Avg Annual Return -> 893.8775510204082 | Projected returns over 5 years ($) -> 84.15
-> SPY: 15 shares @ $420.0 | Profit (%) -> 5.0 | Avg Annual Return -> 5.615384615384615 | Projected returns over 5 years ($) -> 420.7655192100001 |
-> TSLA: 8 shares @ $780.0 | Profit (%) -> -2.5 | Avg Annual Return -> -7.541322314049586 | Projected returns over 5 years ($) -> -321.81714999999999
-> VOO: 20 shares @ $380.0 | Profit (%) -> 8.571428571428571 | Avg Annual Return -> 7.175622542595019 | Projected returns over 5 years ($) -> 881.59

Total Market Value: $34260.00
Total Profit (without dividends): $2010.00
Estimated earned dividends ($): $148.44

Account Report for ID: 0
Holdings:
-> AAPL: 10 shares @ $175.0 | Profit (%) -> 16.6| Avg Annual Return -> 34.3 | Projected returns over 5 years ($) -> 402.6 | Dividends earned ($ esti
-> AMZN: 7 shares @ $135.0 | Profit (%) -> 12.5 | Avg Annual Return -> 55.640243902439025 | Projected returns over 5 years ($) -> 176.9
-> GOOG: 3 shares @ $2700.0 | Profit (%) -> 8.0 | Avg Annual Return -> 4.417549167927382 | Projected returns over 5 years ($) -> 1057.4
-> MMA: 1000 shares @ $10.0 | Profit (%) -> -90.0 | Avg Annual Return -> -101.07692307692308 | Projected returns over 5 years ($) -> -114865.3406250
-> MSFT: 12 shares @ $310.0 | Profit (%) -> 10.714285714285714 | Avg Annual Return -> 7.477465173449876 | Projected returns over 5 years ($) -> 553.
-> NVDA: 5 shares @ $110.0 | Profit (%) -> 12.244897959183673 | Avg Annual Return -> 893.8775510204082 | Projected returns over 5 years ($) -> 84.15
-> SPY: 15 shares @ $420.0 | Profit (%) -> 5.0 | Avg Annual Return -> 5.615384615384615 | Projected returns over 5 years ($) -> 420.7655192100001 |
-> TSLA: 8 shares @ $780.0 | Profit (%) -> -2.5 | Avg Annual Return -> -7.541322314049586 | Projected returns over 5 years ($) -> -321.81714999999999
-> VOO: 20 shares @ $380.0 | Profit (%) -> 8.571428571428571 | Avg Annual Return -> 7.175622542595019 | Projected returns over 5 years ($) -> 881.59

Total Market Value: $45205.00
Total Profit (without dividends): $-87885.00
Estimated earned dividends ($): $148.44
Taxable Income Estimate: $0.00
```

Generated Report (dividend assets didn't fit in 1 screenshot) (2/2):

```
-> 402.6275000000001 | Dividends earned ($ estimate) -> 0.0 | Projected dividends over 5 years ($) -> 94.01854000000003
00003
> 553.9046237640001 | Dividends earned ($ estimate) -> 38.94 | Projected dividends over 5 years ($) -> 266.9348503957081
 84.15310384200002
001 | Dividends earned ($ estimate) -> 0.0 | Projected dividends over 5 years ($) -> 503.9545116633301
999999
381.5968460800003 | Dividends earned ($ estimate) -> 109.5 | Projected dividends over 5 years ($) -> 722.2959101952001
```

```
$ estimate) -> 0.0 | Projected dividends over 5 years ($) -> 94.01854000000003


4062500003
> 553.9046237640001 | Dividends earned ($ estimate) -> 38.94 | Projected dividends over 5 years ($) -> 266.9348503957081
 84.15310384200002
001 | Dividends earned ($ estimate) -> 0.0 | Projected dividends over 5 years ($) -> 503.9545116633301
999999
381.5968460800003 | Dividends earned ($ estimate) -> 109.5 | Projected dividends over 5 years ($) -> 722.2959101952001
```

Challenges

#1: Unimplemented feature

Sorting the stocks based on performance (percentage profit) in the report. Using the TreeMap led to the stocks being sorted based on the Ticker (in alphabetical order), which ended up not being bad, but now the Comparable for the Security class and sub-classes does not serve much purpose unless I wish to continue adding features to the project, as it might be useful then

#2: Challenge

Deciding where to put the export method. I initially wanted to put it in the Account superclass. But then, since I didn’t want to add a static field to be the file path, and have to override the method itself for the taxable account, I decided to simply put it as a method in the main class and pass a list of accounts instead. This felt easier (in my opinion) and saved some work.

#3: Unimplemented Feature

It would have been great to have a cleaner output that’s easier to read. Learning how to format better in Java is something I should do, as shown by this project. This is something I learned thanks to this project as well as something I should have implemented.

Learning

#1: Using GitHub

I learned how to use GitHub throughout this project, and it was surprisingly easier than anticipated. At the end, I could upload to GitHub in under a minute which was great.

#2: Planning

I learned that thorough planning is essential, as I ended up having to change quite a few parts of my code, even removing an interface, as I did a rough plan (that worked), but I hadn’t thought of how everything would actually be implemented. After completing the project, I now understand that planning is 60% of the work (if not more), and actually coding is 40%. I had it backwards previously.

#3: LocalDate

I learned a few LocalDate methods, mainly the ChronoUnit.DAYS.between method, and how to input a LocalDate from a CSV file.