1 Introduction

Over the course of this semester, you will incrementally build a substantial database-backed application in Java. In each phase of the project you will focus on a particular component, which will have deliverables that you must hand in by a certain date. These deliverables will include self-executing JAR (Java Archive) files, ZIP archive files of your source code, well-written test cases, database schemas and a well-written technical design document.

Each phase builds upon prior phases and may also require updates and modifications to prior phases. It is important that you understand the entire scope of the project. You should read all of the assignments to get a better understanding of where the project will be going.

The iterative nature of this project means that it is important that you do not fall behind. In each phase it is also important that you have a good, well-thought design to make subsequent phases easier to design and implement. Poor designs, bad implementations, bugs and broken code will mean subsequent phases of this project will suffer. Investing your time and resources upfront will minimize your “technical debt” and mitigate the need to update or refactor your design later on. Remember one of the Golden Rules of Coding: only code that which you would not mind having to maintain.

2 Problem Statement

One Basket Finance is a financial services corporation that has hired you to design and implement a simple financial management system to replace the aging AS400 green-screen system that they currently use. It will be your responsibility to design a new system from scratch that is object-oriented, written in Java, and supports OBF’s business model by implementing their business rules and providing the functionality as stated below.
OBF manages portfolios of various assets for their clients. The application that you will build will need to support 3 different kinds of assets.

- Deposit Accounts are FDIC (Federal Deposit Insurance Corporation) insured accounts like Savings Accounts, CDs (Certificates of Deposit), and MMAs (Money Market Accounts). Each Deposit Account also has a balance (its value) and an APR (annual percentage rate).

- Stocks are publicly traded shares of a company, business or corporation such as Google (GOOG) or Berkshire Hathaway (BRK.B). The value of a stock asset in a portfolio is the total number of shares multiplied by the share price.

- Private Investments are investment in non-publicly traded business enterprises such as ownership in a closely held corporation, franchise or an investment property or tangible asset. Instead of shares, a private investment consists of the asset’s total worth (its value) and a stake in the investment (a percentage that is owned by the owner of the portfolio).

Each investment consists of various pieces of data that define their value, annual rate of return, and a measure of risk. Each asset also has a unique alpha-numeric code which identifies it (in the old AS400 system) and a label to describe what the asset is.

2.1 Risk Measures

Depending on the type of asset, there is a certain amount of risk associated with it. Risk measures the relative exposure an investment has—the probability that an investment’s return will be different than expected. Risky investments may not return as much as expected or may even lose value. There are several ways to measure such volatility.

2.1.1 Beta Measure

The beta measure is a measure of the relative risk of stocks and other highly traded commodities. It is a measure of risk relative to a general market (usually the S&P 500). Let $r_a$ be the return of an asset and let $r_b$ be the return of the overall market. Then beta is measured as

$$\beta = \frac{\text{Cov}(r_a, r_b)}{\text{Var}(r_b)},$$

where Cov and Var are the covariance and variance respectively. Equivalently, beta can be measured as

$$\beta = \rho_{a,b} \left( \frac{\sigma_a}{\sigma_b} \right),$$
where $\rho_{a,b}$ is the correlation of the two returns and $\sigma_a, \sigma_b$ are their respectively volatilities.

By definition, the beta measure of the benchmark market itself is always 1. In general, beta is unbounded (that is, $\beta \in (-\infty, \infty)$) but in practice it usually falls in the interval $[-1, 2]$. A low beta value ($\beta < 1$) can either indicate a non-volatile asset (such as a T-Bill) or a volatile investment not highly correlated with the market (commodities such as crude oil or gold). A high beta value ($\beta > 1$) indicates a volatile asset that generally moves with the market. For example, Google stock has a beta value of $\beta = 1.15$. A 1% increase/decrease in the benchmark market would generally correlate to a 1.15% increase/decrease in the value of Google stock.

Negative beta values mean that the asset is negatively correlated with the market. For example, $\beta = -1$ would mean that a 1% decrease in the market would correspond to a 1% increase in the asset’s value.

A higher absolute value of the beta measure indicates a riskier (more volatile) asset but provide an opportunity for higher returns (whether or not this is positive or negative doesn’t matter—you can hedge investments in either direction).

Beta measures are widely available on any financial data website (Yahoo Finance, http://finance.yahoo.com/ for example).

2.1.2 Omega Measure

There are many complex measures of risk for private investments that vary greatly depending on the type of investment. For this project, we’ll introduce a simple measure that we’ll call the omega measure\(^1\).

In particular, the omega measure for a Private Investment has a base risk measure $b \in [0, 1]$ (zero being no risk, 1 being extremely risky) that is determined by a financial expert using proprietary formulas. The omega measure is further augmented by adding an offset $o$ that is determined by the following function.

$$o = e^{-100,000/v}$$

where $v$ is the total value of the private investment. Thus, the final omega measure of a private investment is

$$\omega = b + o$$

---

\(^1\)This is not a real measure, it has been created for this project
2.1.3 Aggregate Risk Measure

For an individual portfolio the aggregate risk measure can be computed as follows. Suppose that the portfolio contains assets $a_1, \ldots, a_n$ with values $v_1, \ldots, v_n$. Let

$$V = \sum_{i=1}^{n} v_i$$

be the total value of the portfolio.

The aggregate risk measure is then the weighted sum of all assets with respect to their risk measure:

$$R = \sum_{i=1}^{n} \varphi_i \left( \frac{v_i}{V} \right),$$

where

$$\varphi_i = \begin{cases} 
\beta_i & \text{if } a_i \text{ is a stock} \\
\omega_i & \text{if } a_i \text{ is a private investment} \\
0 & \text{if } a_i \text{ is a deposit account}
\end{cases}$$

The risk is zero for deposit accounts as they are FDIC insured.

2.2 Rates of Return

Assets also have various rates of return that depend on their type.

For deposit accounts, interest is compounded at some periodic interval (annually, monthly, daily). The usual estimate for an annual rate of return is to use the Annual Percentage Yield (APY) formula,

$$APY = e^{\text{APR}} - 1$$

(where APR is on the scale $[0, 1]$).

Stocks and Private investments have a base rate of return that measures how much the asset’s value is expected to increase on an annual basis. In addition, both may pay quarterly dividends which contribute to the expected annual return on top of the base rate of return. Thus, the expected annual return of a stock or private investment is its base rate of return multiplied by its value plus the value of its 4 quarterly dividends.

For all types of assets, the rate of return is the expected annual return divided by its total value.

2.3 Brokers

Each portfolio consists of any number of assets that are managed by an SEC (Securities and Exchange Commission) certified broker. For tax purposes, one person exclusively owns each
portfolio. The owner may also *optionally* designate a beneficiary of the portfolio who will receive ownership of the portfolio upon the owner’s death. In addition, each portfolio also has per-asset annual fees and commissions associated with it.

There are two types of brokers: Expert brokers and Junior brokers. Portfolios managed by a Junior broker have a $50.00 per-asset annual fee and are assessed a commission equal to 2% of the total annual expected return of the portfolio. Portfolios managed by an expert broker have a $10 per-asset annual fee and a 5% commission rate. Each broker also has a unique alphanumeric SEC identification code. Annual commissions are paid based on the return of the asset, not its value.

Finally, each person in the system, regardless of role (owner, broker, beneficiary) is represented with the same basic data. Every person has an alphanumeric code that uniquely identifies them (in the old system), a last name, first name, and an address (street, city, state, zip, country). In addition, each person can be associated with any number of email addresses.

## 3 Project Outline

Over the course of this semester you will iteratively design an application to support OBF’s business model. Development has been broken down into 5 phases:

- **Phase I: Data Representation & Electronic Data Interchange (EDI)** – in the first phase you will design and implement the objects that will form a basis for the system and create parsers to read data from flat files, generate instances of your objects and export them to an interchange format (XML and/or JSON).

- **Phase II: Summary Report** – In this phase you will further refine your objects and define relationships between them to generate a summary report that aggregates pieces of data together.

- **Phase III: Database Design** – This phase focuses on designing a relational database to model your objects and support your application

- **Phase IV: Database Connectivity** – You will refactor your code to load your objects to your database rather than from flat files

- **Phase V: Database Persistence** – You will implement and use an API to persist (save) data to your database.

- **Phase VI: Sorted List ADT** – In this phase you design and implement a sorted list ADT and integrate it into your application
A Creating Runnable JARs and Archive Files

These instructions are for creating runnable JAR file in Eclipse (Indigo); instructions may differ for other versions or other IDEs.

A.1 Creating a Runnable JAR file

1. Run your program at least once, this creates a “Launch Configuration” in Eclipse
2. Right click on your project, select “Export”
3. Under the Java folder, select “Runnable JAR file”, click Next
4. Select the “Launch Configuration” corresponding to your main method
5. Under “Export destination:” click Browse and select a directory location and file name (ending in .jar) where you want the JAR file to be exported to
6. Under “Library handling:” be sure that “Package required libraries into generated JAR” is selected
7. Click Finish
8. The JAR file should now be in the directory you indicated

A.2 Creating a ZIP archive of your source code

1. Right click your project, select “Export”
2. Under the “General” folder select “Archive File”
3. Click Next
4. Click Browse and select a directory/file name to export to
5. Under “Options” make sure that “Save in zip format” is selected
6. Click Finish; the ZIP file should now be in the directory/file you indicated
B Adding External JAR Libraries to an Eclipse Project

These instructions are for creating runnable JAR file in Eclipse (Kepler); instructions may differ for other versions or other IDEs.

There are many ways to import external JAR libraries (examples: gson-1.7.1.jar, joda-time-2.0.jar). The following instructions will be most compatible with how we expect you to build your runnable JAR file.

1. Create a folder in your project: Right click project → New → Folder

2. Name your folder lib (short for library)

3. Drag and drop your JAR file to this folder, be sure to select “copy files”

4. Right click the new JAR file in your lib folder and select Build Path → Add to Build Path
C Partner Policy

You can always work alone, if you wish, on all aspects of the project. However, if you choose, you may work in pairs for each phase of the project (you can change partners with prior permission from the instructor). If you choose to work in pairs, you must adhere to the following:

- All work must be the result of an equal collaborative effort by each member. You may not simply partition up the work or phases. You should collaborate on each part of the project including design, implementation, testing, debugging, etc.

- Turn in only one copy of the design document with all names on it

- You must turn in only one electronic copy under the first author’s login

- You must follow any additional policies regarding late passes or other items as described in the syllabus

- You are highly encouraged to use some sort of revision control system such as SVN or Git. However, you must ensure that your team’s codebase and artifacts are not publicly accessible. Failure to do so will be considered a violation of the department’s academic integrity policy.

  - For more information on Github, see their bootcamp: https://help.github.com/articles/set-up-git/
  - For more information on setting up an SVN on CSE, see: http://cse.unl.edu/systems-faqs/using-cvs-or-svn-version-control

If you want to work in groups of up to three students, you will need to get special permission from the instructor prior to doing so. You must provide 1. A reason for wanting to work in a group of 3, and 2. a plan of action for dealing with member(s) of the group that end up not putting in their equal share of work.