170D Module O Practical 1B

Stock Market

A small community has decided that since there are no commercial stockbrokers serving their needs, they are going to run a small brokerage firm. They bought a computer for the broker, who is going to enroll new customers, create their accounts, and handle all transactions.

Create an object-oriented program in Python to run a stock trading system. The system has customers, their accounts, and the transactions on their accounts.

A customer is a named person who can have a portfolio.

A portfolio has a non-negative amount of cash, a type (Regular or Tax-Free), and a set of holdings.

A holding is a stock symbol, number of shares, and initial price per share.

A transaction is a change in the amount of money in the account. There are four kinds of transactions: cash deposit, cash withdraw, stock purchase and stock sale. Purchases draw from available cash, sales deposit into available cash.

Stock symbols and prices are provided by an exchange. Each stock has a symbol, a low price, a high price, a volatility factor, and an optional initial price. The price of the stock changes each minute, but only within the range of +/- volatility.

The community may be small, but they know better than to use floating point values for currency. They also know better than to accept code without test cases.

The program is to be designed so that the broker, not the end customer, does all of the work on their behalf.

DIcE Rubric

Document	Design Plan	Does the design plan provide a clear general overview of the project?	3%
		Is the design plan easy to understand?	2%
	Test Plan	Are test cases detailed enough to repeat easily?	2%
		Are expected results stated clearly?	2%
		Are requirements adequately covered by test cases?	1%
	Project Writeup	Does the writeup document challenges and successes encountered?	2%
		Does the writeup document any lessons learned?	3%
	Writing	Is the project free of grammatical and spelling errors?	3%
		Is non-code formatting consistent?	1%
	Code Formatting	Does `pycodestyle` produce no warnings or errors?	4%
		Are appropriate names chosen to enable code readability?	2%
		Are comments added where appropriate and aid understanding of the logic?	2%
		Is any outside code cited appropriately?	2%
	Total		**30%**
Implement	Version Control	Does the project have the correct name and default branch?	1%
		Were commits broken down into appropriate scopes?	3%
		Are commit messages simple and informative?	1%
	Architecture	Are effective and efficient data structures used?	5%
		Was the code designed and constructed in a modular fashion?	5%
		Were generally sound decisions made with regard to architecture?	5%
		Does the project pass `mypy` type checking with no warnings?	5%
	Testing	Does the program include robust unit tests?	4%
		Do all automated tests pass?	1%
	Total		**30%**

Execute	Safety	Does the program avoid crashing or infinite loops, even on invalid input?	5%
	Parsing	Does the program pass `python3 compileall .` with no warnings?	5%
	Requirements	Were all inputs parsed correctly and yield the correct output?	5%
		Are all other requirements met?	15%
	Performance	Does the program scale appropriately with input and data?	5%
		Does the program execute in a timely manner?	5%
	Total		**40%**

Requirements

Area	Requirement
Document	All documentation must be in PDF format unless otherwise specified.
Document	All documentation must be located in a `doc/` folder at the top level of the project.
Document	The design document must be located in `doc/design.pdf`
Document	The test plan must be located in `doc/testplan.pdf`
Document	The project writeup must be located in `doc/writeup.pdf`
Document	All code must conform to PEP8 guidelines. `pycodestyle` must report no warnings or errors.
Implement	Project must be stored in the assigned VCS account, under the project name `stock_broker`.
Implement	No third-party header files/libraries may be used unless signed off by the Program Manager or Instructor.
Implement	Project must use appropriate data types or structures.
Implement	All automated tests and test code must be located in a `test/` folder at the top level.
Implement	Project must provide appropriate automated unit tests.
Implement	Project must pass `mypy` type checking with no warnings or errors.
Implement	All monetary amounts must be stored or manipulated as `int`egers or `decimal.Decimal` objects (NEVER `float`s).
Implement	An Account must have a type (regular or tax-free, at least), a unique, automatically generated account ID number, and a starting balance (default of $0.00). An Account may have an optional textual label, such as "Primary", "Retirement", etc.
Implement	A Customer must have a name, a unique, automatically generated customer ID number, and a ZIP code.
Implement	Each Transaction must have a timestamp, amount, and (optional) memo.
Implement	Withdrawals must not be permitted to take an account balance negative.
Execute	Project must have `broker.py` in the top level directory of the repository.
Execute	Upon starting, the program must present a menu to the user that has the options New Customer, Find Customer, and Quit.
Execute	The New Customer option from the main menu must allow the user to create a new customer. Upon successful creation of a new customer, the user must be taken to the Customer Details page to view that new customer.
Execute	The Find Customer option from the main menu must allow the user to search for a customer. It must present a single search prompt. The search must first try to match the input to a customer by ID (exact match); if that fails, then searching by name (sub-string match). Unless there is only one match, the result of this action must present a list of matching customers. This list must display the customer ID number, the customer name, and their current overall balance.

Area	Requirement
Execute	The user must be able to select a customer from the list, and then taken to the Customer Details page. If a Find A Customer search results in only a single choice, the user can be taken directly to the Customer Details page, rather than present a menu with a single choice.
Execute	The Customer Details page for a specific customer must provide the customer’s name, ID number, the date when the customer was enrolled at the bank, and a list of all portfolio accounts belonging to that customer. The listing must show the account number, the type of account, the label (if it exists), its current cash balance, and total value.
Execute	On the Customer Details page, the user must be presented with the options to select an account or create a new account, return to the main menu or quit.
Execute	The Create a New Account page must display the name and ID of the customer, and must allow the user to create a new account for that customer.
Execute	Once a new account has been created, the user must be taken to the Account Details page to view the account.
Execute	The Account Details page must show the name and ID of the account owner, the ID and label of the account, the account type, a list of all current holdings, a list of all transactions for the account, and the current balance of the account.
Execute	The list of transactions must show the timestamp of the transaction, the amount, and the memo for the transaction.
Execute	The Account Details page must give the user the ability to deposit, withdraw, buy, sell, return to the main menu, or quit the program.

Suggested Extra Credit Features

Area	Feature	+
Execution	Save the application’s necessary state on exit, and load it on startup.	+2
Execution	Implement an editable transaction fee for each transaction.	+3