Due: February 1, 2019, before 11:59 pm.

Edit, January 24: remove requirement to print investors in cryptocurrency report.

Notes

- Please follow the "Programming Standards" for all work you submit. Programming standards 1-22 are in effect, but see also the notes in this assignment.
- Make sure to complete the honesty declaration on umlearn before the assignment submission. You can't submit the assignment without completing the online honesty declaration.
- Hand-in will be via the umlean Assignment facility. Make sure you leave enough time before the deadline to ensure your hand-in works properly. The assignments are submitted using umlearn's time (not the time on your computer). Assignments submitted after the due date will be marked according to the late assignment policy in the ROASS.
- Official test data will be provided several days before the assignment due date. Until then, use your own test data.
- All assignments in this course carry an equal weight.

Assignment 1: Cryptocurrency Management

Cryptocurrencies are virtual, electronic currencies that are not managed by a central authority, like a country's central bank. Because of this lack of a central authority, there are challenges with electronic currencies: how can we buy and sell units with security and confidence if there are no physical copies of the currency? One solution to this problem is blockchains.

Cryptocurrencies use these blockchains to (in our simplification) list of all the transactions that the cryptocurrency has been involved in. Whenever a unit of the cryptocurrency is traded, the information about that trade is added to the list of transactions. The list is never modified (using cryptography in the real world) but only appended to, and any observer can detect if any part of the blockchain list has been modified.

In this assignment, you will simulate the buying and selling of cryptocurrencies in a market using Java. You will manage the cryptocurrencies for several investors, and coordinate the buying and selling of cryptocurrencies. The system is a major simplification of cryptocurrency markets, including a very simplified blockchain system, minimal mining and no cryptography. Your system will work on text-based input that should be processed line-by-line. The input will have commands that are recognized by the system and the output will be written to standard output.

The description of the assignment is organized as follows:

1. A description of the major components of the system: the investors, the cryptocurrencies and the blockchain system.
2. A description of all the commands that are a part of the system.

3. Implementation details on hashing, data structures, input processing and output.
4. Submission information

INVESTORS

Investors are represented by their name, a unique user ID, the cash in their account, and their portfolio, which is a collection of cryptocurrencies and the number of units of that cryptocurrency that they own.

CRYPTOCURRENCIES

Cryptocurrencies are represented by a name and a symbol, which is a (maximum) four letter abbreviation for the cryptocurrency. In this assignment, an initial quantity of the cryptocurrency is created when a new cryptocurrency is created. These are dispensed to investors in a first-come first-served manner using the "mining" command. This command essentially gives away free cryptocurrencies to early investors.

Unlike real cryptocurrencies, in this assignment investors will only be able to mine or trade whole units of a cryptocurrency. There is no fractional mining or trading.

BLOCKCHAIN

For each cryptocurrency, a blockchain is maintained by your system. The blockchain for this system will be a list of transactions. That is:

1. The blockchain is a list of all transactions for a particular cryptocurrency, including mining actions. Each cryptocurrency has its own blockchain. When a cryptocurrency is created, the list is empty.
2. Each time a transaction occurs, a new element is added to the list of transactions in the blockchain.
3. The blockchain consists of blocks, and each block records one transaction (trade or mine).
4. Each block in the blockchain contains its own hashcode and the hashcode of the list of all previously-created blocks. Hashcodes for blocks and lists are described in the section "Hashcodes" below.

COMMANDS

The input to the system is a series of text commands. There are eight types of commands.

1. **New investor**: This action creates a new individual in the system. Initially every investor has no cryptocurrencies, but has a reserve of classical currency (Canadian dollars).

Format	`NEW [NAME] [USERID] [CASH]`

Example	`NEW Anne Vestor avestor 10000`
Outcomes	Success, Duplicate

An investor is considered a duplicate if another investor with the same user ID is already in the system. The amount of cash an investor has will fit in a 32-bit integer. The name will be a first and last name, separated by a single space, each of which will be at most 80 characters. The userid will be at most eight lower case letters, with no spaces.

2. **New cryptocurrency**: This action creates a new cryptocurrency.

Format	`CRYPTO [NAME] [SYMB] [QUANT]`
Example	`CRYPTO COMP2150-COIN C215 10000`
Outcomes	Success, Duplicate

The cryptocurrency will have a symbol as well as a name. The symbol is a sequence of at most four characters. A cryptocurrency is considered a duplicate if another cryptocurrency with the same symbol is already in the system. The quantity of cryptocurrency created will fit in a 32-bit integer. The name of the cryptocurrency will be at most 80 characters long and will not contain any spaces.

3. **"Mine" new cryptocurrency**: This action allows an investor to acquire a cryptocurrency that has been created. Mining is free, unlike in the real world (where computational power is needed).

Format	`MINE [USERID] [CURR SYMB] [QUANT]`
Example	`MINE avestor C215 10`
Outcomes	Not Found, Insufficient Currency, Success

A result of "not found" should be returned if either the investor or the currency is not found. If there is not sufficient currency to mine (e.g., if all of the currency has been distributed to investors), then the "insufficient currency" result should be returned. Otherwise, the command should report as successful. **The results of this command must be logged in the blockchain for the cryptocurrency, if successful.**

4. **Trade cryptocurrency**: This action allows two investors to trade a cryptocurrency. Two cryptocurrencies may be traded for each other, or a cryptocurrency may be purchased or sold using Canadian dollars (abbreviated CAD).

Format	`TRADE [TRADER1] [TRADER2] [CURRENCY1] [QUANT1] [CURRENCY2] [QUANT2]`
Examples	`TRADE avestor other C215 1 BTC 100` `TRADE personx persony BTC 1 CAD 10000`
Outcomes	Not Found, Insufficient funds, Success, Same trader

Both `TRADER1` and `TRADER2` should be given as userids of investors, and `CURRENCY1` and `CURRENCY2` should be cryptocurrency symbols or Canadian dollars. Canadian dollars are given by the symbol CAD. Both quantities of currency will be integers that are at most 32 bits.

A result of "not found" should be reported if any of the traders or cryptocurrencies are not found. If either of the traders does not have sufficient funds to complete the transaction, "insufficient funds" should be reported. The userids of the traders should be different (otherwise the "same trader" error should be raised). However, two Traders could (in theory) trade the same currency.

If all traders and currencies exist and there are sufficient quantities with each trader, the command should report as successful. In particular, `TRADER1` will give `QUANT1` units of `CURRENCY1` to `TRADER2`, and `TRADER2` will give `QUANT2` units of `CURRENCY2` to `TRADER1`. **The results of this command must be logged in the blockchain(s) for the currencies, if successful.** Note that as there are blockchains for each cryptocurrency, two transactions must be logged if two cryptocurrencies are involved, one transaction for each cryptocurrency.

5. Portfolio report: this command should produce a report of all the cryptocurrencies held by an investor.

Format	`REPORT [USERID]`
Example	`REPORT avestor`
Outcomes	Not Found, report produced

If an investor is not in the system, then the system should report this with "Not Found". Otherwise the command should produce a report that includes the investor's name, their ID, the cash they have on hand and a list of all the cryptocurrencies they own, including the quantity.

6. Currency report: this command should produce a report of the status of a cryptocurrency.

Format	`CRYPORT [SYMB]`
Example	`CRYPORT BTC`
Outcomes	Not found, report produced

If a cryptocurrency is not in the system, then the system should report this with "not found". Otherwise, the command should produce a report for that cryptocurrency which includes the cryptocurrency's name, symbol, ~~a list of which investors hold the cryptocurrency~~, how many units are still available, and all the transactions that have taken place for that cryptocurrency (from the blockchain, from oldest transaction to newest).

Additionally, you should verify the blockchain: you should traverse the blockchain list, and verify that all hashcodes in the blockchain have not been altered, by checking that they contain a hash that is consistent with all previous transactions, as described in the blockchain section above. The verification should appear in the report either as successful or a report that a block has been altered.

7. Comments: Any line that starts with a number sign (**#**) is a comment.

Format	`# [TEXT]`
Example	`# this is a comment`
Outcomes	Echo of comment

Any comment should be written to the output exactly as is (including the number sign).

8. End: This command ends the simulation.

Format	`END`
Example	`END`
Outcomes	Done

After the end command is received, the program should print "`DONE`" and terminate. If the input file ends without an `END` command, an error should be printed.

Hashcodes

When creating a block, it needs to have a hashcode. The exact hashcode of a block will depend on the type of transaction (mine or trade), but the hashcode of a block should combine the hashcode of the transaction (i.e., all the information of the transaction: traders, quantities, etc.) with **the hashcode for the all of the previous transactions in the blockchain**. To do this, you can use the Objects.hash() function in the java API.

Input Error Checking

When the format is specified for a command, you can assume all the stated properties hold (i.e., cryptocurrency symbols will all have the proper format, integers will be valid, etc.). Your code does not need to do error checking for these properties.

Output Format

There is no required output format that you must match. You should use your judgement on the output format -- design output that can be understood by the software user based on the outcomes described for each command type. Minor differences in output will not be penalized by the graders. Here are some additional notes that you may find helpful:

- The Success outcome should be displayed when the command completes successfully, including information on what was successfully implemented.
- The Not Found outcome should be displayed when an investor or cryptocurrency is not found in the system. In these cases, print the information that caused the error.
- The Duplicate outcome should be printed if the data in a command is a duplicate of data already in the system. Print the information that caused the error.

Data Structures

Do not use ANY built-in Java collection classes (e.g., ArrayLists, Hashes, ...) or Java Generics in this assignment: any linked structures must be your own, built from scratch. You should not use arrays other than for temporary operations (e.g., split() for strings). **IF YOU USE GENERICS OR COLLECTION DATA STRUCTURES, YOU WILL LOSE MARKS.**

Final Notes

- This is a large programming problem with many pieces to the puzzle - remember incremental development strategy: get small pieces working rather than trying to tackle the whole thing at once.
- Your program will be tested by the markers under Unix, and must run with a simple javac/java as per the "Running Java" instructions on the course website.
- As yet, we do not know much OO - you should follow the rules of abstraction and object-oriented programming based on what you have learned in previous courses and given the limitations specified in this assignment. Use separate classes for entities like investors, cryptocurrencies and the blockchain. Try to make other entities and classes for your ADTs. Do not be afraid of using many classes - breaking things up usually makes for much simpler pieces.
- Official test data will be provided before the assignment is due.

Hand-in

Submit all your source code for all classes. Each class should be declared as public in its own file. You should also submit a text document giving the output on the official test data.

You **MUST** follow these rules for submission:

- You MUST submit all of your files in a single zip file (no rar files, please).
- All of the files required for your project should be in a single directory.

- You must include a README.TXT file that describes exactly how to compile and run your code from the command line. The markers will be using these instructions exactly and if your code does not compile directly, you will lose marks.

The easier it is for your assignment to mark, the more marks you are likely to get. Do yourself a favour.

Submit all files on umlearn. Remember to submit your electronic honesty declaration on umlearn before the assignment deadline.