This project is created as a package **bankapp** which has 5 classes in it. It also has a /lib folder which has some .jar files for sending API requests and catching the JSON responses. accounts.csv contains mock bank data, some of which is purposefully "bad" to test error handling by the app - such as bad formatting or nonnumerical account balances

To compile and run the program add the /lib folder to class path so java can see them, when the working directory is the /bankapp top level folder with all the .java files.

```
javac -d . -cp ".;lib/*" *.java
java -cp ".;lib/*" bankapp.Control
```

The 5 java files are:

Control.java – this is the driver class (executed on the command line) using bankapp.Control

- Main method
    - While user wants to continue (Boolean cont is true…)
    - Create a double finalBalance that will hold the account balance before and after a user interacts with the bankapp; this will be written to the .csv file after a user logs out
    - Create a new Messages() object, which prints out and receives messages on the console (but doesn't handle the logic of whether that input was valid, or which specific message is to be printed)
    - Create a new Authenticator(name) object using the entered user name
        - If user wants to login, see if username has an account
            - If name exists, prompt for pass up to three times
                - If login is successful, set boolean authed to true
                - Else, print login failure
            - If authed is true, create new Accounts(balance) and run account session
        - Else, prompt to create account
            - If name already exists, print error
            - Else, collect user data for new account
                - Create new Accounts(balance) and run account session
        - Write data to csv, and prompt user to continue or to exit

Account.java – this contains the logic for user interaction with an account balance, after being authenticated

- CryptoAPI crypto – object for handling querying historical bitcoin prices
- Double account – money in an account, which is fed at initialization and returned from running a session
- Messages msg – passed during initialization, to send/receive console data
- Account(double account, Messages msg) – initialize a new Account object when called from Control.java
- Double runSession() – driver method for Account() object, to handle user interaction with an account balance
    - While the user wants to continue:
        - Prompt for choice 1, 2, 3, 4)
        - Track session data using change, earnings, enough, pd and crypto
            - Change tracks the change in account balance for a single user interaction
            - Earnings tracks the raw profit from investing
            - Enough tracks whether the account balance is sufficient to invest or withdraw a certain amount
            - Pd is a time duration that tracks the length an investment position was held (for tax purposes)
            - Hashmap crypto is a String, Object hashmap that has labeled data – "NetChange" is the dollar amount change after investing, and "Period" is a time duration for the investment
        - Case switch to the four different user interactions
        1. Prompt to add money, and print resulting balance. Increment Double account
        2. Prompt to withdraw money. Run checksum, and print resulting balance if sufficient. Update Double account
        3. Prompt to invest in a startup. Run checksum, and print result of check

            - If sufficient, then user random number generator (weighted to be +3% on a -50 to +50 percent scale) to basically randomize the startup investment results
                - Print appropriate messages based on positive or negative outcome
                - Increment Double account
        - Prompt to invest in cryptocurrency. Run checksum, and print result of check
            - If sufficient, run the investCrypto Method that returns a hashmap of net change and duration of investment
                - If positive outcome, print appropriate messages
                    - If period > 1 year, assess 20% tax and print appropriate messages. Increment Double account
                    - Else, assess 40% tax, print appropriate messages, and increment Double account
                - Else, if negative outcome, print appropriate messages and increment Double account

- Boolean checksum(change, message) – logic check for the transaction amount against the account balance
    - If less than amount, print appropriate message and return false
    - If equal to 0, print appropriate message and return false
    - Else print success message and return true
- HashMap investCrypto(Double investAmount) – logic to invest in cryptocurrency
    - Create buy and sell hashmaps to hold the buy and sell dates
    - Check that the sell date is after the buy date (do not continue until this is true)
    - Determine purchase price and bitcoin bought, and from that determine the total end balance after selling that bitcoin at the selling price (determined by the sell date)
    - Put the net change and duration (difference between buy and sell date) into data hashmap and return it
- Double investStartup(Double change) – logic to handle investing in startup
    - Create new random and create a percentage change, positively biased by 3%
    - Return that percentage change multiplied by the investment amount

Authenticator.java – logic for creating an authenticator instance and its functionality, for attempting to authenticate with a given username, including login and creating an account with that username

- Int tries: number of failed login attempts; Messages msg: access console input/output; String name: login username being used; HashMap passes and accounts: hashtables associating a username with either a password or an account balance, each simulating a column of the CSV file
- Authenticator(String name, Messages msg) – Authenticator has a name, as well as a messages object to send/receive output
- String getName() – return this.name
- Integer getTries – return this.tries
- Void readFile() – open csv
    - While scanner has more data, read the next line, and parse by splitting by commas
        - Store username, pass and accounts in appropriate hashmaps
    - Handle edge cases such as incorrect number of columns, or non-numerical account balances
        - Print the incorrect data to console to warn user
- Boolean accountExists() - check hashmap for whether a username exists
- Boolean tryAuthenticate() – check if a password is correct for a username
    - If true, return true
    - If false, increment number of tries and return false
- Double getBalance() – return account balance for username
- Void setBalance() – used after a user completes one transaction, update account balance in hashmap
- Void setupAccount(String name, String pass, double account) – add username, password and account to appropriate hashmaps
- Void writeData() – used after a user completes an interactive account session, updating the data on disk by writing the hashmap data to the csv line by line in the same comma format

    - Create printwriter, stringbuilder to write and create strings to be written
    - For each entry in hashmaps, add a line to the stringbuilder of appropriate data
    - At the end, write all the data to the file all at once (a block of text escaped with '\n' characters)

CryptoAPI.java – logic for creating a formatted API request, sending it to a database, and receiving the response as a JSON

- String apiKey – key associated with my cryptocompare account (free)
- String url – server to send API requests to
- Double submitRequest(long ts) – logic to get the price of bitcoin at a specific unix timestamp
    - add API request parameters to NameValuePair params
    - make an API call with the url and params, and catch the response in a JSONObject
        - parse the JSON to get the USD price of BTC, and return this price
    - catch errors – restricted access, invalid resource
- makeAPICall(String url, List<NameValuePair> params) – make an API call using the specific parameters
    - create a http request object using appropriate parameters, headers and credentials
    - execute the request on the http client, and save the response
    - parse the response content as a string and return this string

Messages.java – collection of methods for printing data to console and receiving console input

- console input is tested for syntactic correctness (is this a number or a date?) but not logical correctness (is the withdrawal amount lesser than the account balance?)
- System.console() con for input
- Void greeting() – print greeting
- Void userExists(String name) – notify that the name account exists
- Void userExistsError(String name) – notify error that the name already exists when trying to create an account
- Void userNoExist(String name) – notify error that the name doesn't exist when trying to login
- Void loginSuccess(String name) – notify successful login
- Void loginFailure(int tries) – notify number of tries left after failed login attempt
- Void printAccount(double account) – commonly called method to print account balance
- Void choiceFailure() – commonly called method when a choice prompt has invalid input
- Void investSuccess(Double amount) – notify user of net change after investing (positive outcome)
- Void investFailure(Double amount) – notify user of net change after investing (negative outcome)
- Void operationFailure(String message) – notify user of insufficient balance for the operation (specified by message)
- Void operationSuccess(String message) – notify user of sufficient balance for the operation (specified by the message)
- Void defaultAmount() – notify user of invalid user input for the amount of a transaction
- Void readFailure() – error logging functionality, to notify user of incorrectly formatted data in the csv
- String promptName() – prompt user for name and return it
- String promptPass() – prompt user for password, without showing characters (hidden input), and return as string
- String setPass() – prompt user for password when creating new account
- Boolean promptContinue(String message) – prompt user to continue, either after a single transaction, or an interactive account session. Return the result as a Boolean based on entered input
- Boolean promptLogin() – prompt user to either create a new account or login
- Integer promptChoice() – prompt user to select a transaction choice (1-4), returning either a valid choice or 0 if the entered choice was invalid. The "0" response is handled by the logic in the Account session
- Double promptAccount(String message) – prompt user for amount of a transaction. Parse the input and return either a valid number or 0 (the logic is handled in the Account session)
- HashMap<String, Object> promptDate(String message) – prompt user to enter a date in a specific format, and continue until a valid date is entered and parsed. Return the date as a hashmap of both a LocalDate object, and the unix timestamp (seconds)
- Void printPrice(String descr, double mat, double price, double total) – print a single Bitcoin transaction
- Void printInvestmentPeriod(Period p) – extract and format info for a Period (a time duration), and print it