Do not start coding this project. There will be a series of assignments that lead up to certain parts being developed in a specific order, some in a specific way.

You will be building a bank software system. All input will come in through a series of commands (List of Strings). When you have completed every command in the list, you will output certain data and state of the system (List of Strings). There is no user input or output (no scanner, no sysout).
The input will come in through a List of Strings as a parameter to a public facing method.
The output will return from that public facing method as a List of Strings.

All commands are case **insensitive**. All **invalid** commands should be **stored to be output later**.

The system will support 3 types of bank accounts:
- Checking
- Savings
- Certificate of deposit (CD)

Each account can be created with a unique identifier and an annual percentage rate (APR). For a checking and savings account, APR is calculated **monthly**. For a CD, APR is calculated 4 times per month.

Create Command Rules

The creation command will be in the form of: create <account type> <id> <apr>
- Account type must either be:
  - Checking
  - Savings
  - CD
- Any other type makes the command invalid.
- ID must be a unique 8-digit number. If any other account exists with the ID specified, the command is invalid.
- APR must be in the form of a percentage, e.g. 0.06%, and can range from 0 to 10.
- APR can't be negative.
- Examples of valid APRs: 0, 0.3, 3, 5.7, 7.9, 9.8, 10.

All checking and savings accounts are opened with a balance of $0.
All CDs must specify an additional value in the command: create cd <id> <apr> <amount>
- The minimum amount to create a CD is $1000.
- The maximum amount a CD can be created with is $10000.

The system can have any number of accounts and any number of each type of account.

Example correct commands:
- Create checking 12345678 0.01
- Create savings 98765432 0.6
- Create cd 12345678 1.2 2000

Deposit Command Rules

Money can be deposited into a checking or savings account with the command:

deposit <id> <amount>

Example correct command:

Deposit 12345678 500

Savings accounts have a maximum deposit amount of $2500 per command.
Checking accounts have a maximum deposit amount of $1000 per command.
CDs cannot be deposited into.

You can't deposit a negative amount. You can deposit zero.

Withdrawal Command Rules

Money can be withdrawn from a checking or savings account with the command:

withdraw <id> <amount>

There is no such thing as overdraft in this system. If the $ amount provided is more than the amount in the account, the account goes down to $0. This is valid.
Example correct command:

Withdraw 12345678 300

Savings accounts have a maximum withdrawal amount of $1000 per command. Savings accounts have a maximum of 1 withdrawal per month.
Checking accounts have a maximum withdrawal amount of $400 per command.

CDs have special withdrawal rules. No money can be withdrawn from a CD until 12 months have passed since it was created. When 12 months have passed, only a full withdrawal may happen. The same command is used as above, but if the <amount> specified is less than the total balance in the CD, the command is invalid. If 12 months have not passed, the command is invalid. If 12 months have passed and the <amount> is over the balance in the CD, this is acceptable, and the balance should go to $0.

You can't withdraw a negative amount. You can withdraw zero.

Transfer Command Rules

Money can be transferred between:

- Checking account and checking account
- Checking account and savings account
- Savings account and checking account
- Savings account and savings account
- CD accounts can't be part of a transfer, either from or to.

You can transfer between accounts with the command:

- Transfer <from ID> <to ID> <amount>

Money should be withdrawn from the account with the <from ID> and be deposited into the account with the <to ID>.
Only the amount actually withdrawn should be deposited. For example, if I transfer $300 from a checking account that only has $200 in it, only $200 should be transferred to the destination account.
Example correct command:

- Transfer 12345678 98765432 500

All deposit and withdrawal rules for all accounts still apply.

Pass Time Command Rules

The last command is to pass the time.
Time will only be passed in iterations of months, between 1 and 60.
You can't pass 0 months. You can't pass 61 months.
For each month that passes, there are several system behaviors that must be processed.
The pass time command: pass <months>

Examples of valid pass time command:

- Pass 1
- Pass 12

For every month that passes (in order):

- If an account's balance is $0, close the account (as if it never existed).
- If an account's balance is below $100, deduct $25 from the account (minimum balance fee).
- Calculate and accrue every account's APR.

APR Calculation

APR calculation (monthly):

1. Convert the APR from percentage to decimal by dividing by 100. For example an APR of 0.6% will be 0.60 / 100 = 0.006
2. Now divide that number by 12. So 0.006 / 12 = 0.0005
3. Next, multiply the balance in the account by the number from the previous step. Let's say an account has a balance of $5000. 5000 x 0.0005 = 2.5.
4. Add that number to their balance. 5000 + 2.5 = 5002.5. The new balance is $5002.50.

Example APR for 1 month for a Savings or Checking account:

Account with a balance of $1000 and APR of 3%. Following the steps above:

1. 3 / 100 = 0.03
2. 0.003 / 12 = 0.0025
3. 1000 * 0.0025 = 2.5
4. 1000 + 2.5 = 1002.5. The new balance is $1002.5

For a CD, do step 3 above 4 times per month. While this will look very complicated, remember that this is just a for loop that runs 4 times…

A CD account with a balance of $2000 and APR of 2.1%:

1. 2.1 / 100 = 0.021
2. 0.021 / 12 = 0.00175
3. 2000 * 0.00175 = 3.5
4. 2000 + 3.5 = 2003.5
5. 2.1 / 100 = 0.021
6. 0.021 / 12 = 0.00175
7. 2003.5 * 0.00175 = 3.506125
8. 2003.5 + 3.506125 = 2007.006125
9. 2.1 / 100 = 0.021
10. 0.021 / 12 = 0.00175
11. 2007.006125 * 0.00175 = 3.51226071875
12. 2007.006125 + 3.51226071875 = 2010.51838571875
13. 2.1 / 100 = 0.021
14. 0.021 / 12 = 0.00175
15. 2010.51838571875 * 0.00175 = 3.518407175007813
16. 2010.51838571875 + 3.518407175007813 = the new balance is $2014.036792893758.

Output Requirements

Remember that output is in the form of a List of Strings.

For each **open** bank account:

Add to the output List the current state of the account: <account type> <id> <balance> <apr>
Add to the output List all transactional commands that have ever affected the account (except pass time) in the order that they happened for that account.

Truncate balance to 2 decimal points, ie 100.2345436 should be 100.23, 100.256354 should be 100.25. Truncate APR down to 2 decimal points as well, ie 0.06, or if the APR is a whole number add the decimal points, so APR of 3 should be 3.00.

Helpful code:

```
DecimalFormat decimalFormat = new DecimalFormat("0.00");
decimalFormat.setRoundingMode(RoundingMode.FLOOR);
```

After adding all bank account information:
For each invalid command that was caught, add to the output List the command. Add to the output List all invalid commands in the order they were read. Each invalid command is added separately.

FAQ

Can an amount be input as a decimal instead of a whole number? For example: 100.50.

Yes.

Can you pass time 6.0?

No decimal values allowed. Only positive integers 1+.

Will a transfer command appear under the transaction history of the TO or FROM account?

Both.

Can input commands have numbers not formatted to 2 decimal places, like an APR of 0.6?

Yes.

Can you transfer from an account to itself?

No.

Should all numbers in the *output* be formatted to 2 decimal places?

In the state of the account, yes. For transaction history and invalid commands, all commands should be output *exactly* the way they came in.

Should account state start with a capital letter and then have only lowercase letters?

Yes, please see the example below.

In what order should I output accounts?

In the order they were created.

Is it valid to create a new account with the same ID as one which has been deleted?

Yes. The only record of the old account that should be left is if it was part of a valid transfer (in which case it will be represented in that transfer command in the history of the other account).

Are commands with extra spaces valid or invalid?

At the beginning or in the middle it is invalid. At the end it is valid.

Is scientific/exponential notation valid?

No.

Is it valid to try to withdraw more than the limit if the account has less money than the limit?

No. Validate withdrawal amounts by the value that is in the command.

Can commands have any extra arguments, for example “pass 1 foobar”?

No. For any command, any extra arguments make the whole command invalid.

Example Input/Output Scenario

Keep in mind that your software should work correctly with any input of commands. It is on you to test your code to make sure (confidence) that it will handle anything anyone throws at you.

Input:

```
Create savings 12345678 0.6                 // create a new savings account
Deposit 12345678 700                        // deposit $700 into savings
Deposit 12345678 5000                       // invalid, amount too high
creAte cHecKing 98765432 0.01               // case insensitive!
Deposit 98765432 300                        // deposit $300 into checking
Transfer 98765432 12345678 300              // transfer $300 from checking - savings
Pass 1                                      // Pass 1 month of time
Create cd 23456789 1.2 2000                 // create a new cd with 1.2% APR $2000
```

Output:

```
Savings 12345678 1000.50 0.60               // current state of open account
Deposit 12345678 700                        // transaction history for first account
Transfer 98765432 12345678 300              // transaction history for first account
Cd 23456789 2000.00 1.20                    // current state of open account
Deposit 12345678 5000                       // invalid commands come last
```

Note that the checking account is not listed as it should be closed.
Note that open accounts are listed in the order they were created.