Reed McLean

Algorithm for Bank Management System

# Overview

This algorithm outlines the steps required to implement a bank management system in Python. The system includes classes to manage banking operations, such as opening accounts, depositing and withdrawing money, ATM withdrawals, transferring money, and calculating interest. The system also includes utility functions for validating inputs and handling coin changes.

Classes and Their Responsibilities

1. BankUtility

This class provides static utility methods for common banking operations.

- Methods:
    - isNumeric(str): Checks if a given string is numeric.
    - promptUserForString(prompt): Prompts the user for a string input.
    - promptUserForPositiveNumber(prompt): Prompts the user for a positive float input.
    - convertFromDollarsToCents(dollars): Converts an amount in dollars to cents.
    - generateRandomInteger(min, max): Generates a random integer between a specified minimum and maximum.

2. CoinCollector

This class manages the parsing and conversion of various coin types into their respective values in cents.

- Methods:
    - parseChange(coins): Parses a string of coin types and returns the total value in cents.

3. Account

This class represents a bank account and includes functionalities to manage deposits, withdrawals, ATM withdrawals, and account-related operations.

- Attributes:
    - ATM_WITHDRAWAL_FEE: A fixed fee for ATM withdrawals, set to 250 cents.
    - accountNumber: A unique identifier for the account, generated randomly if not provided.
    - ownerFirstName: The first name of the account owner.

    - ownerLastName: The last name of the account owner.
    - socialSecurityNumber: The Social Security Number of the account owner.
    - PIN: A 4-digit personal identification number for the account, generated randomly.
    - balance: The balance of the account, initialized to 0.
- Methods:
    - generate_account_number(): Generates a random 8-digit account number.
    - generate_pin(): Generates a random 4-digit PIN.
    - getOwnerFirstName(), setOwnerFirstName(firstName): Gets/Sets the owner's first name.
    - getOwnerLastName(), setOwnerLastName(lastName): Gets/Sets the owner's last name.
    - getSocialSecurityNumber(), setSocialSecurityNumber(ssn): Gets/Sets the SSN.
    - getPIN(), setPIN(pin): Gets/Sets the PIN.
    - getBalance(), setBalance(balance): Gets/Sets the account balance.
    - deposit(amount): Deposits a specified amount in cents into the account.
    - withdraw(amount): Withdraws a specified amount in cents from the account.
    - atmWithdraw(amount): Withdraws a specified amount from the account via an ATM, including the withdrawal fee.
    - isValidPIN(pin): Checks if a given PIN matches the account's PIN.
    - __str__(): Returns a string representation of the account, masking the SSN except for the last 4 digits.

4. Bank

This class represents a bank that can hold and manage multiple accounts.

- Attributes:
    - NUM_ACCOUNTS: The maximum number of accounts the bank can hold, set to 100.
    - accounts: An array to store the accounts.
- Methods:
    - addAccountToBank(account): Adds an account to the bank.
    - removeAccountFromBank(account): Removes an account from the bank.
    - findAccount(accountNumber): Finds an account in the bank by account number.

- addMonthlyInterest(annualInterestRate): Adds monthly interest to all accounts based on an annual interest rate.

5. BankManager

This class manages the interaction between the user and the bank system, providing a menu-driven interface.

- Attributes:
  - bank: An instance of the Bank class to manage accounts.
- Methods:
  - main(): Provides a menu-driven interface for the user to interact with the bank system.
  - promptForAccountNumberAndPIN(bank): Prompts the user for an account number and PIN and validates them.
  - openAccount(): Manages the process of opening a new account.
  - getAccountInformation(): Retrieves and displays account information.
  - changePIN(): Allows the user to change the PIN for their account.
  - depositMoney(): Manages the process of depositing money into an account.
  - transferMoney(): Manages the process of transferring money from one account to another.
  - withdrawMoney(): Manages the process of withdrawing money from an account.
  - atmWithdrawal(): Manages the process of withdrawing money from an account via an ATM.
  - depositChange(): Manages the process of depositing coin change into an account.
  - closeAccount(): Manages the process of closing an account.
  - addMonthlyInterest(): Adds monthly interest to all accounts in the bank.

# Step-by-Step Algorithm

1. Initialize the Bank Manager

- Instantiate the BankManager class to start the system.
- Call the main() method to display the user menu.

2. User Interaction via Menu

- Display the main menu with options to:

1. Open an account.
2. Get account information and balance.
3. Change PIN.
4. Deposit money in an account.
5. Transfer money between accounts.
6. Withdraw money from an account.
7. ATM withdrawal.
8. Deposit change.
9. Close an account.
10. Add monthly interest to all accounts.
11. End Program.

- Capture the user's choice and invoke the corresponding method in BankManager.

3. Open an Account

- Prompt the user for the account owner's first name, last name, and SSN.
- Create a new Account instance with the provided details.
- Add the new account to the bank using Bank.addAccountToBank(account).
- Display the account details, including the generated account number and PIN.

4. Get Account Information

- Prompt the user for an account number and PIN.
- Validate the account number and PIN using BankManager.promptForAccountNumberAndPIN(bank).
- If valid, display the account information using the Account.__str__() method.

5. Change PIN

- Prompt the user for their current PIN and account number.
- Validate the provided PIN.
- Prompt the user for a new PIN and confirm it.
- Update the account's PIN using Account.setPIN(newPin).

6. Deposit Money

- Prompt the user for their account number and PIN.

- Validate the account number and PIN.
- Prompt the user for the deposit amount in dollars and cents.
- Convert the amount to cents using BankUtility.convertFromDollarsToCents(dollars).
- Deposit the amount into the account using Account.deposit(amount) and display the new balance.

7. Transfer Money

- Prompt the user for their account number and PIN.
- Validate the account number and PIN.
- Prompt the user for the destination account number.
- Validate the destination account.
- Prompt the user for the transfer amount in dollars and cents.
- Convert the amount to cents.
- Withdraw the amount from the source account and deposit it into the destination account.
- Display the new balances for both accounts.

8. Withdraw Money

- Prompt the user for their account number and PIN.
- Validate the account number and PIN.
- Prompt the user for the withdrawal amount in dollars and cents.
- Convert the amount to cents.
- Withdraw the amount from the account using Account.withdraw(amount) and display the new balance.

9. ATM Withdrawal

- Prompt the user for their account number and PIN.
- Validate the account number and PIN.
- Prompt the user for the withdrawal amount in dollars (must be in multiples of $5, limit $1000).
- Convert the amount to cents and add the ATM withdrawal fee.
- Withdraw the amount from the account using Account.atmWithdraw(amount) and display the new balance.
- Calculate and display the number of $20, $10, and $5 bills dispensed.

10. Deposit Change

- Prompt the user for their account number and PIN.
- Validate the account number and PIN.
- Prompt the user for the coin string (e.g., "QPDNNDXHW").
- Parse the coin string and calculate the total value in cents using CoinCollector.parseChange(coins).
- Deposit the value into the account using Account.deposit(cents) and display the new balance.

11. Close an Account
    - Prompt the user for their account number and PIN.
    - Validate the account number and PIN.
    - Remove the account from the bank using Bank.removeAccountFromBank(account).
    - Confirm the account closure.
12. Add Monthly Interest
    - Prompt the user for the annual interest rate.
    - Calculate the monthly interest rate.
    - Iterate over all accounts in the bank.
    - Add monthly interest to each account's balance based on the calculated interest rate.
    - Display the interest added and the new balance for each account.
13. End Program
    - Exit the program when the user selects the "End Program" option.