1. **Document the public interface and data model for the Client, Checking Account, Savings Account, Name, Address, Phone Number, Password, Bank Account, and Transaction classes.**

| **Public Interface and Data Model for the Client Class** | | | |
| --- | --- | --- | --- |
| **Task** | **Method** | **Data Needed** | **Return Value** |
| Initialize a new client | \_\_init\_\_(Name name, PhoneNumber phoneNumber, Address address, String accountType, Password password) | Name (Name object)  Phone number (Phone number object)  Address (Address object)  Account type (String)  Next available client number (int)  Next account number (int)  Client password (Password object) | Client object |
| Display the details of a Client | \_\_repr\_\_() | Name (Name object)  Phone number (Phone number object)  Address (Address object)  List of client’s bank accounts (List)  Client number (int) | String (Contains all client information with string formatting) |
| Changes the client password to a new password that meet the requirements | changePassword() | Client password (Password object)  New client password | N/A (changes the password in the file where the client’s password is stored) |
| Open a Bank Account | openBankAccount(String accountType, float balanceIn) | Account type of the bank account to be added (String)  Starting balance (float)  List of client’s bank accounts (List) | BankAccount object created |
| Close a Bank Account | closeBankAccount(BankAccount accountIn) | The account that the client wants to remove (BankAccount)  List of client’s bank accounts (List) | N/A (withdraws all funds and removes specific Bank Account from list of bank accounts, if applicable) |
| Accessor/getter method for the client’s bank accounts | getClientAccounts() | Client account list (List) | Client account list (List) |
| Accessor/getter method for the client number | getClientNumber() | Client number (int) | Client number (int) |
| Accessor/getter method for the next available client number | getNextClientNumber() | Next available client number (int) | Next available client number (int) |
| Accessor/getter method for the next available bank account number | getNextAccountNumber() | Next available account number (int) | Next available account number (int) |
| Accessor/getter method for the client’s first name | getFirstName() | First name (String from Name object) | First name (String) |
| Accessor/getter method for the client’s last name | getLastName() | Last name (String from Name object) | Last name (String) |
| Accessor/getter method for the client’s street | getStreet() | Street (String from Address object) | Street (String) |
| Accessor/getter method for the client’s city | getCity() | City (String from Address object) | City (String) |
| Accessor/getter method for the client’s state abbreviation | getState() | State (String from Address object) | State (String) |
| Accessor/getter method for the client’s phone number | getPhoneNumber() | Phone number (String from PhoneNumber object) | Phone number (String) |
| Mutator/setter method for the client’s first name | setFirstName(String firstIn) | First name (String from Name object) | N/A |
| Mutator/setter method for the client’s last name | setLastName(String lastIn) | Last name (String from Name object) | N/A |
| Mutator/setter method for the client’s street | setStreet(String streetIn) | Street (String from Address object) | N/A |
| Mutator/setter method for the client’s city | setCity(String cityIn) | City (String from Address object) | N/A |
| Mutator/setter method for the client’s state abbreviation | setState(String stateIn) | State (String from Address object) | N/A |
| Mutator/setter method for the client’s phone number | setPhoneNumber(String numberIn) | Phone number (String from PhoneNumber object) | N/A |
| Determines if two Client objects are equal | \_\_eq\_\_(self, otherClient) | Second client that is being compared (Client object)  Assigned client number of the first Client object being compared (int)  Assigned client number of the second Client object being compared (int) | Boolean (True if the Client objects are equal, False if not) |

| **Public Interface and Data Model for the Bank Account Class** | | | |
| --- | --- | --- | --- |
| **Task** | **Method** | **Data Needed** | **Return Value** |
| Initialize a new bank account | \_\_init\_\_( int accountNum, int clientNum, float balanceIn = 0.0, String accountType = ‘checking’) | Account number (int)  Client number of the account (int)  Initial balance (float)  Account type (String)  List of all account transactions (list)  Next available transaction number (int) | BankAccount object |
| Accessor/getter method for the next available transaction number | getNextTransactionNum() | Next available transaction number (int) | Next available transaction number (int) |
| Accessor/getter method for the account number | getAccountNumber() | Assigned account number (int) | Assigned account number (int) |
| Accessor/getter method for the account type | getAccountType() | Account type (String) | Account type (String) |
| Accessor/getter method for the account balance | getBalance() | Current account balance (float) | Current account balance (float) |
| Accessor/getter method for the interest rate | getInterestRate() | Interest rate (float) | Interest rate (float) |
| \*\*Abstract  Display the account details | \_\_repr\_\_() | Defined in subclasses | String (Contains all account information with string formatting) |
| Print the list of transactions for an account (the accessor/getter method for the transaction list) | printTransactionList() | List of transactions for the account (List or Set) | String (Contains all transactions with string formatting) |
| \*\*Abstract  Transfer an amount to one account from another account | transfer(float amount, BankAccount otherAccount) | Defined in subclasses | Boolean (True if transfer is possible, False if not) |
| \*\*Abstract  Deposit an amount into the account | deposit(float amount) | Defined in subclasses | Boolean (True if the amount passed in was valid and thus the deposit is valid, False if not) |
| \*\*Abstract  Withdraw an amount from the account (if possible) | withdraw(float amount) | Defined in subclasses | Boolean (True if withdrawal is possible, False if not) |
| \*\*Abstract  Calculates and adds interest to the account (if applicable) | calcInterest() | Defined in subclasses | Boolean (True if interest has been added) |
| Determines if two Bank Account objects are equal | \_\_eq\_\_(self, otherAccount) | Second account that is being compared (BankAccount object)  Assigned account number of the first BankAccount object being compared (int)  Assigned account number of the second BankAccount object being compared (int) | Boolean (True if the BankAccount objects are equal, False if not) |

| **Public Interface and Data Model for the Checking Account Class** | | | |
| --- | --- | --- | --- |
| **Task** | **Method** | **Data Needed** | **Return Value** |
| Initialize a new checking account | \_\_init\_\_(float balanceIn = 0.0) | Starting balance of the account (float)  Account type (String)  Overdrawn counter (int) | Checking Account object |
| Deposit an amount into the checking account | deposit(float amount) | Amount being deposited (float)  Account balance (float)  List of checking account transactions (list) | Boolean (True if deposit has been completed) |
| Withdraw an amount from the checking account | withdraw(float amount) | Amount being withdrawn (float)  Account balance (float)  List of checking account transactions (list) | Boolean (True if withdraw has been completed) |
| Transfer an amount to ***this account*** from the ***account passed as a parameter*** | transfer(float amount, BankAccount otherAccount) | Amount being transferred (float)  Account being transferred the money (BankAccount)  Account balance of the account transferring (float)  List of checking account transactions for the account transferring (list)  Account balance for the account being transferred (float)  List of checking account transactions for the account being transferred (list) | Boolean (True if transfer has been completed) |
| Calculates and adds interest to the account (if applicable) | calcInterest() | Current balance (float)  Interest rate (float)  List of transactions for the account (List) | Boolean (True if interest has been added) |
| Print all checking account transactions | printTransactionList() | List of transactions for the account (List) | String detailing transactions on a checking account |
| Display the details of a checking account | \_\_repr\_\_() | Account number (int)  Account balance (float)  Account type (String)  List of transactions for the account (List) | String (Formatted details of the checking account) |

| **Public Interface and Data Model for the Savings Account Class** | | | |
| --- | --- | --- | --- |
| **Task** | **Method** | **Data Needed** | **Return Value** |
| Initialize a new savings account | def \_\_init\_\_(float balanceIn = 0.0) |  | Savings Account object |
| Accessor/getter method for the overdraft penalty value for when the account is overdrawn | getOverdraft() | Overdrawn counter value (int) | Overdraft penalty value (float) |
| Accessor/getter method for counter for when the account is overdrawn | getOverdrawnCount() | Overdrawn counter value (int) | Overdrawn counter value (int) |
| Deposit an amount into the savings account | deposit(float amount) | Amount to deposit (float)  List of account transactions (List)  Overdrawn counter (int)  Account balance (float) | Boolean (True if the amount is able to be deposited) |
| Withdraw an amount from the savings account | withdraw(float amount) | Amount to withdraw (float)  Account balance (float)  List of account transactions (List)  Overdrawn counter (int) | Boolean (True if the amount is able to be withdrawn) |
| Transfer an amount to ***this account*** from the ***account passed as a parameter*** | transfer(float amount, BankAccount otherAccount) | Amount to transfer (float)  Other account’s balance (float)  Account to transfer’s balance (float)  List of account transactions for the other account (List)  List of account transactions for the account doing the transfer (List)  Overdrawn counter for the other account (int)  Overdrawn counter for the account doing the transfer (int) | Boolean (True if the amount is able to be transferred) |
| Calculates and adds interest to the account (if applicable) | calcInterest() | Current balance (float)  Interest rate (float)  List of transactions for the account (List) | Boolean (True if interest has been added, False if not) |
| Print all savings account transactions | printTransactionList() | List of transactions for the account (List) | String detailing transactions on a savings account |
| Display the details of a savings account | \_\_repr\_\_() | Account number (int)  Account balance (float)  Account type (String)  Overdrawn count (int)  List of transactions for the account (List) | String (Formatted details of the savings account) |

| **Public Interface and Data Model for the Transaction Class** | | | |
| --- | --- | --- | --- |
| **Task** | **Method** | **Data Needed** | **Return Value** |
| Initialize a new transaction | \_\_init\_\_(str tType, int tNumber, float amount = 0.0) | Results from \_setDate(), which requires access to the current day (int), current month (int), and current year (int)  Transaction amount (float)  Transaction type (String)  Transaction number (int) | N/A |
| Determines if two Transactions are equal | \_\_eq\_\_(Transaction other) | Second transaction that is being compared (Transaction object)  Transaction amount of the first transaction object (float)  Transaction amount of the second transaction object (float)  Date of the first transaction object (String)  Date of the second transaction object (String)  Transaction type of the first transaction object (String)  Transaction type of the second transaction object (String) | Boolean (True if the Transaction objects are equal, False if not) |
| Determines if two Transactions are not equal | \_\_ne\_\_(Transaction other) | Second transaction that is being compared (Transaction object)  Transaction amount of the first transaction object (float)  Transaction amount of the second transaction object (float)  Date of the first transaction object (String)  Date of the second transaction object (String)  Transaction type of the first transaction object (String)  Transaction type of the second transaction object (String) | Boolean (True if the Transaction objects are not equal, False if they are equal) |
| Adds two transaction amounts together | \_\_add\_\_(Transaction other) | Second transaction that is being added (Transaction object)  Transaction amount of the first transaction object (float)  Transaction amount of the second transaction object (float) | Float |
| Subtracts one transaction amount from another | \_\_sub\_\_(Transaction other) | Second transaction that is being subtracted (Transaction object)  Transaction amount of the first transaction object (float)  Transaction amount of the second transaction object (float) | Float |
| Implements the sum() function that will sum a list of the Transactions | \_\_radd\_\_(Transaction other) | Amount that has been summed so far (float)  Transaction amount of the transaction object (float) | Float |
| Accessor/getter method for the transaction day | getDay() | Current day (int) | Int |
| Accessor/getter method for the transaction month | getMonth() | Current month (int) | Int |
| Accessor/getter method for the transaction year | getYear() | Current year (int) | Int |
| Accessor/getter method for the transaction amount | getAmount() | Transaction amount (float) | Float |
| Accessor/getter method for the transaction date | getDate() | Current date (contains the current day (int), current month (int), and current year (int)) | String |
| Accessor/getter method for the account number | getAccountNumber() | Assigned account number (int) | Assigned account number (int) |
| Accessor/getter method for a specific transaction number | getTNumber() | Transaction number (int) | Int |
| Accessor/getter method for the transaction type | getTType() | Transaction type (String) | String |
| Accessor/getter method for the next available transaction number | getNextTNumber() | Next available transaction number (int) | Int |
| Display the details of a transaction | printTransaction() | Date of the transaction (String)  Transaction amount (float)  Transaction type (String)  Transaction number (int) | String (Contains transaction information with string formatting) |
| Prints all of the transaction instance variables in a human-readable form | \_\_str\_\_() | Transaction number (int)  Transaction amount (float)  Date of the transaction (String)  Transaction type (String) | String (Contains the formatted, human-readable transaction information) |
| Prints all of the transaction instance variables in a machine-readable form | \_\_repr\_\_() | Transaction number (int)  Transaction amount (float)  Date of the transaction (String)  Transaction type (String) | String (Contains the formatted, machine-readable transaction information) |

| **Public Interface and Data Model for the Name Class** | | | |
| --- | --- | --- | --- |
| **Task** | **Method** | **Data Needed** | **Return Value** |
| Initialize a new name of a Client | \_\_init\_\_(str first\_name, str last\_name) | First name (String)  Last name (String) | Name object |
| Accessor/getter method for the first name | getFirstName() | First name (String) | String |
| Accessor/getter method for the last name | getLastName() | Last name (String) | String |
| Mutator/setter for the first name | setFirstName(str first) | First name (String) | N/A |
| Mutator/setter for the last name | setLastName(str last) | Last name (String) | N/A |
| Display the name details | \_\_repr\_\_() | First name (String)  Last name (String) | String (formatted name) |

| **Public Interface and Data Model for the Address Class** | | | |
| --- | --- | --- | --- |
| **Task** | **Method** | **Data Needed** | **Return Value** |
| Initialize a new address of a Client | \_\_init\_\_(str street, str city, str state) | Street name (String)  City name (String)  State abbreviation (String) | Address object |
| Accessor/getter method for the client’s street name | getStreet() | Street (String from Address object) | String |
| Accessor/getter method for the client’s city name | getCity() | City (String from Address object) | String |
| Accessor/getter method for the client’s state abbreviation | getState() | State (String from Address object) | String |
| Mutator/setter for the street | setStreet() | Street (String) | N/A |
| Mutator/setter for the city | setCity() | City (String) | N/A |
| Mutator/setter for the state abbreviation | setState() | State (String) | N/A |
| Validation for street names | validStreet(str street) | Street (String) | Boolean (True if valid, False otherwise) |
| Validation for city names | validCity(str city) | City (String) | Boolean (True if valid, False otherwise) |
| Displays the address details | \_\_repr\_\_() | Street (String)  City (String)  State (String) | String (formatted address) |

| **Public Interface and Data Model for the Phone Number Class** | | | |
| --- | --- | --- | --- |
| **Task** | **Method** | **Data Needed** | **Return Value** |
| Initialize a new phone number of a Client | \_\_init\_\_(str phoneNum) | Phone number (String) | Phone Number object |
| Accessor/getter method for the phone number | getPhoneNumber() | Phone number (String) | String |
| Mutator/setter for the phone number | setPhoneNumber(str number) | Phone number (String) | N/A |
| Displays the phone number | \_\_repr\_\_() | Phone number (String) | String (formatted phone number) |

| **Public Interface and Data Model for the Password Class** | | | |
| --- | --- | --- | --- |
| **Task** | **Method** | **Data Needed** | **Return Value** |
| Initialize a new password of a Client | \_\_init\_\_(str password) | Password (String) | Password object |

1. **Document the responsibilities for each of the classes.**

The Client class needs to know…

* A Client’s name (Name)
* A Client’s address (Address)
* A Client’s phone number (PhoneNumber)
* A Client’s password (Password)
* The client number (int)
* The next available client number (int)
* The next available account number (int)
* The client’s list of accounts (list)
* The account type of each client’s accounts (String)

The Client class needs to have implemented (do)...

* Accessor (getter) methods for all instance variables
* Mutator (setter) methods for all mutable instance variables
* Open a new bank account for a client
* Close a pre-existing bank account for a client
* Change a Client’s password
* Display the details of a client
* Compare two Clients and determine if they are equal

The Bank Account class needs to know…

* The client number where the Bank Account will be stored (int)
* The account number of the Bank Account (int)
* The initial balance being held within the Bank Account (float)
* The next available transaction number for the Bank Account (int)
* The interest rate for all Bank Accounts (float)
* The account types possible for a Bank Account (String)
* Each transaction being stored within a Bank Account (List)

The Bank Account class needs to have implemented (do)…

* Accessor (getter) methods for all instance variables
* Mutator (setter) methods for all mutable instance variables
* Calculate and add interest to the account (if applicable)
* Print the list of transactions for an account
* Determine if two accounts are equal/not equal to one another
* Abstractly declare that it is possible to…
  + Deposit an amount into an account
  + Withdraw an amount from an account (if possible)
  + Transfer an amount from one account to another account
  + Calculate and adds interest to the account (if applicable)
  + Display the details of an account

The Checking Account class needs to know…

* Client number (int)
* Account number (int)
* Initial balance (float)
* Account balance (float)
* Account type (String)
* Next available transaction number (int)
* List of account transactions (list)
* Interest rate (float)

The Checking Account class needs to have implemented (do)...

* Accessor (getter) methods for all instance variables
* Mutator (setter) methods for all mutable instance variables
* Deposit an amount into an account
* Withdraw an amount from an account (if possible)
* Transfer an amount from one account to another account
* Calculate interest for an account (if possible)
* Display details about the Checking Account with String formatting
* Determine equality between two Checking Accounts

The Savings Account class needs to know…

* Overdraft fee (float)
* Overdrawn counter (int)
* Client number (int)
* Account number (int)
* Initial balance (float)
* Account balance (float)
* Account type (String)
* Next available transaction number (int)
* List of account transactions (list)
* Interest rate (float)

The Savings Account class needs to have implemented (do)...

* Accessor (getter) methods for all instance variables
* Mutator (setter) methods for all mutable instance variables
* Deposit an amount into an account
* Withdraw an amount from an account (if possible)
* Transfer an amount from one account to another account
* Calculate interest for an account (if possible)
* Display details about the Savings Account with String formatting
* Determine equality between two Savings Accounts

The Transaction class needs to know…

* The current day (int)
* The current month (int)
* The current year (int)
* The transaction amount (float)
* The transaction type (String)
* The current transaction number (int)

The Transaction class needs to have implemented (do)…

* Accessor (getter) methods for all instance variables
* Mutator (setter) methods for all mutable instance variables
* Get the transaction date (current date) from the system
* Display the details of a transaction
* Determines if two Transactions are equal/not equal
* Add/subtract transactions together
* Sum all transactions within a transaction list
* Set the date for a transaction

The Name class needs to know…

* First name (String)
* Last name (String)

The Name class needs to have implemented (do)...

* Accessor/getter methods for all instance variables
* Mutator/setter methods for all mutable instance variables
* Display the name with String formatting

The Address class needs to know…

* Street (String)
* City (String)
* State (String)

The Address class needs to have implemented (do)...

* Accessor/getter methods for all instance variables
* Mutator/setter methods for all mutable instance variables
* Validity checks to ensure that Street and City meet all of their constraints
* Display the address with String formatting

The PhoneNumber class needs to know…

* Phone number (String)

The PhoneNumber class needs to have implemented (do)...

* Accessor/getter methods for all instance variables
* Mutator/setter methods for all mutable instance variables
* Display the phone number with String formatting

The Password class needs to know…

* Password (String)

The Password class needs to have implemented (do)...

* Constructor that checks the validity of the password based on the requirements

1. **Document the definitions for each of the classes.**

The class definitions for the Client class are as follows:

* Data
  + Client’s name: Created class Name to represent a client’s first and last name as a String with constraints
  + Client’s address: Created class Address to represent a client’s street, city, and state as a String with constraints
  + Client’s phone number: Created class PhoneNumber to represent a client’s phone number as a String with constraints
  + Client’s password: Created class Password to represent a client’s password as a String with constraints
  + Client number: Int that represents the client number for the account
  + Next available client number: Int that represents the next available client number to be taken by a newly initialized client
  + Next available account number: Int that represents the next available account number to be taken by a newly initialized account within the client
  + Client’s list of accounts: List of all the client’s BankAccounts stored in one place
  + Account type of each client’s accounts: String that represents the account type of the client’s BankAccounts stored in their account list
* Queries
  + getClientAccounts(): Accessor/getter method for the client’s bank accounts
  + getClientNumber(): Accessor/getter method for the client number
  + getNextClientNumber(): Accessor/getter method for the next available client number
  + getNextAccountNumber(): Accessor/getter method for the next available account number
  + getFirstName(): Accessor/getter method for the client’s first name
  + getLastName(): Accessor/getter method for the client’s last name
  + getStreet(): Accessor/getter method for the client’s street name
  + getCity(): Accessor/getter method for the client’s city
  + getState(): Accessor/getter method for the client’s state abbreviation
  + getPhoneNumber(): Accessor/getter method for the client’s phone number
* Commands
  + setFirstName(String firstIn): Mutator/setter method for the client’s first name
  + setLastName(String lastIn): Mutator/setter method for the client’s last name
  + setStreet(String streetIn): Mutator/setter method for the client’s street
  + setCity(String cityIn): Mutator/setter method for the client’s city
  + setState(String stateIn): Mutator/setter method for the client’s state abbreviation
  + setPhoneNumber(String numberIn): Mutator/setter method for the client’s phone number
  + openBankAccount(String accountType, float balanceIn): Open a new Bank Account for a Client
  + closeBankAccount(int accountNum): Close a pre-existing Bank Account for a Client
  + changePassword(): Changes the password of a Client after the Client proves they know their original password and inputs a new password that meets all password requirements
  + \_\_repr\_\_(): Display details of a Client
  + \_\_eq\_\_(self, otherClient): Determines if two Client objects are equal

The class definitions for the Checking Account class are as follows:

* Data
  + Initial balance: Float that represents the starting balance of the Checking Account
  + Account balance: Float that represents the current balance of the Checking Account
  + Account type: String that represents the account type, always ‘checking’, of the Checking Account
  + Account number: Int that represents the account number of the Checking Account
  + Next available transaction number: Int that represents the next available transaction number to be used within the account
  + List of account transactions: List of all account transactions for the Checking Account
  + Interest rate: Float that represents the interest rate to use when calculating interest for a Checking Account
* Queries
  + N/A
* Commands
  + deposit(float amount): Deposit an amount into the checking account
  + withdraw(float amount): Withdraw an amount from the checking account
  + transfer(float amount, BankAccount otherAccount): Transfer an amount to this account from the account passed as a parameter
  + calcInterest(): Calculates and adds interest to the account (if applicable)
  + printTransactionList(): Print all checking account transactions
  + \_\_repr\_\_(): Display the details of a checking account

The class definitions for the Savings Account class are as follows:

* Data
  + Overdraft fee: Float that represents the fee that must be paid when a Savings Account is overdrawn
  + Overdrawn count: Int that represents how many times the Savings Account has been overdrawn
  + Initial balance: Float that represents the starting balance of the Checking Account
  + Account balance: Float that represents the current balance of the Checking Account
  + Account type: String that represents the account type, always ‘checking’, of the Checking Account
  + Account number: Int that represents the account number of the Checking Account
  + Next available account number: Int that represents the next available account number to be initialized to the newest Checking Account
  + List of account transactions: List of all account transactions for the Checking Account
  + Interest rate: Float that represents the interest rate to use when calculating interest for a Checking Account
* Queries
  + getOverdraft(): Accessor/getter method for the overdraft penalty value for when the account is overdrawn
  + getOverdrawnCount(): Accessor/getter method for counter for when the account is overdrawn
* Commands
  + deposit(float amount): Deposit an amount into the savings account
  + withdraw(float amount): Withdraw an amount from the savings account
  + transfer(float amount, BankAccount otherAccount): Transfer an amount to this account from the account passed as a parameter
  + calcInterest(): Calculates and adds interest to the account (if applicable)
  + printTransactionList(): Print all savings account transactions
  + \_\_repr\_\_(): Display the details of a savings account

The class definitions for the Bank Account class are as follows:

* Data
  + Next available transaction number: Int that represents the next available transaction number within a Bank Account
  + Current account number: Int that represents the account number for a Bank Account
  + Initial balance: Float that represents the initial deposit/withdrawal value of a new Bank Account
  + Current balance: Float that represents the overall value within a Bank Account
  + Interest rate: Float that represents the interest rate for Bank Accounts
  + Transaction list: List or set that holds all transactions for a Bank Account; always starts with an initial deposit/withdrawal depending on how the Bank Account is initialized
* Queries
  + getAccountType(): Returns the account type as a String
  + getAccountNumber(): Returns the account number of the Bank Account as an int
  + getInterestRate(): Returns the interest rate of the Bank Account as a float
  + getBalance(): Returns the total held balance of the Bank Account as a float
  + getNextTransactionNum(): Accessor/getter for the next transaction number within a Checking Account
* Commands
  + \_\_repr\_\_(): Displays the account details with String formatting
  + printTransactionList(): Displays the list of transactions associated with the Bank Account with String formatting
  + deposit(float amount): Deposits an amount into the account
  + withdraw(float amount): Withdraws an amount from the account (if possible)
  + calcInterest(): Calculates and adds interest to the account (if applicable)
  + transfer(float amount, BankAccount otherAccount): Transfers an amount to one account from another account (if applicable)
  + \_\_eq\_\_(BankAccount otherAccount): Determines if two BankAccount objects are equal to one another

The class definitions for the Transaction class are as follows:

* Data
  + Day: Int that represents the day of the month; must be between 1 – 31
  + Month: Int that represents the month of the year; must be between 1 – 12
  + Year: Int that represents the year; must be greater than or equal to 2022
  + Amount: Float that represents the amount of the transaction
  + transactionNumber: Int that represents the transaction number
* Queries
  + getDay(): Returns the day of a transaction as an int
  + getMonth(): Returns the month of a transaction as an int
  + getYear(): Returns the year of a transaction as an int
  + getAmount(): Returns the amount of the transaction as a float
  + getDate(): Returns the day, month, and year of a transaction with String formatting
  + getTNumber(): Returns the transaction number of the transaction as an int
  + getTType(): Returns the transaction type of the transaction as a String
  + getNextTNumber(): Returns the next transaction number to be assigned to a new transaction as an int
* Commands
  + \_\_eq\_\_(Transaction other): Determines if two Transactions are equal
  + \_\_ne\_\_(Transaction other): Determines if two Transactions are not equal
  + \_\_add\_\_(Transaction other): Adds two transaction amounts together
  + \_\_sub\_\_(Transaction other): Subtracts one transaction amount from another
  + \_\_radd\_\_(Transaction other): Implements the sum() function that will sum a list of the Transactions
  + printTransaction(): Prints the day, month, year, amount, transaction number, and transaction type with String formatting
  + \_\_str\_\_(): Prints all of the transaction instance variables in a human-readable form
  + \_\_repr\_\_(): Prints all of the transaction instance variables in a machine-readable form

The class definitions for the Name class are as follows:

* Data
  + first\_name: String that represents a person’s first name; should be valid (potentially check for not empty string; 1 – 25 alphabetical characters with no special characters)
  + last\_name: String that represents a person’s last name; should be valid (potentially check for not empty string; 1 – 40 alphabetical characters with no special characters)
* Queries
  + getFirst(): Returns the first name of the account holder as a String
  + getLast(): Returns the last name of the account holder as a String
* Commands
  + setFirst(str first): Sets a new first name as a String
  + setLast(str last): Sets a new last name as a String
  + \_\_repr\_\_(): Displays the name with String formatting

The class definitions for the Address class are as follows:

* Data
  + Street: The street the client lives on; length <= 30 characters, cannot contain special characters, cannot be empty
  + City: The city the client lives in; length <= 30 characters, cannot contain special characters or numbers, cannot be empty
  + State: The state the client lives in; length must be 2 characters, cannot contain special characters, acceptable values are VA, MD, NJ, PA, DE, NC, WV, or DC
* Queries
  + getStreet(): Returns the street of the account holder as a String
  + getCity(): Returns the city name of the account holder as a String
  + getState(): Returns the state name of the account holder as a String
* Commands
  + setStreet(): Sets a new street for the account holder as a String
  + setCity(): Sets a new city name for the account holder as a String
  + setState(): Sets a new state name for the account holder as a String
  + validStreet(String street): Determines if a street name is valid (if the street is comprised of characters and numbers and doesn’t contain any special characters outside of spaces)
  + validCity(String city): Determines if a city name is valid (if the city is comprised of characters and doesn’t contain any special characters outside of spaces)
  + \_\_repr\_\_(): Displays the address with String formatting

The class definitions for the PhoneNumber class are as follows:

* Data
  + phoneNumber: The client’s phone number; made up of a String; must be 10 digits long, cannot start with 0, must be entirely numeric
* Queries
  + getPhoneNum(): Returns the phone number of the account holder as a String
* Commands
  + setPhoneNum(): Sets a new phone number for the account holder as a String
  + \_\_repr\_\_(): Displays the phone number with String formatting

The class definitions for the Password class are as follows:

* Data
  + password: The client’s password; made up of a String; must be between 8 and 16 characters inclusive and must not contain any of the special characters /, \, <, >, |, or a space (“ “)
* Queries
  + N/A
* Commands
  + N/A