# General notes

* Account balances are sometimes negative internally, but the UI converts them to positive numbers when needed.
* Throughout the program, all interest rates are annual.
* BigDecimal is consistently used for monetary amounts throughout the program, so that fractions of a cent are never discarded, which is important when dealing with money.

# UML diagram notes

* Enumerations have no fields indicated because of limitations of the UML program and language, but in general, their fields correspond to their accessor methods. An exception is on the Bank class, where the getUser(String) method refers to a field of type Map<String, User>.
* Abstract members are italicized.
* Visibility is indicated with markers adjacent to each member. + is public, # is protected, ~ is package-private (default), and – is private.
  + Access to some methods is limited based on the user’s role or other factors, but this kind of access control is implemented at the UI level rather than at the programmatic/visibility level.

# Implementation notes

* The Bank class is a singleton enumeration type. It is the only directly persisted type, and it stores all the bank’s User instances, as well as the global time offset. (An offset is stored against the system clock to model the passage of time; it can only be increased, and only by the operations manager.)
  + Not shown on the diagram are methods which calculate and return global statistics, for the use of the accountant user role.
  + When persisted, Bank also makes note of the current date and time. This is checked when the program loads to see if any interest/fee periods have passed, and posts interest and fees if needed, using the Account.doPayments() method.
* User.role is null if the user is not an employee. Customers and employees are not mutually exclusive, so both are represented by a User instance.
* Transactions can be marked with their FraudStatus as REVERSED, but marking them as such does not directly alter the account state. Instead, the auditor must manually add transactions, dated at the same time as the reversal, which reverse the fraudulent transaction’s effects on the account. This decision was made because subsequent transactions often depend on the previous state, making the appropriate action difficult to determine without manual intervention.
* Account.doPayments() is intended to be called for every account, every month, most likely by the Bank class (which also handles time changes). It processes any interest owed and monthly fees charged, as well as automatic payments set up for the account. Interest payments and fees generate Transaction objects of special TransactionType, but they are otherwise handled identically to ordinary transactions.
  + For automatic payments, the Transaction object has a null timestamp and fraud status, because it hasn’t actually occurred yet. A complete object is created when the transaction is made.
  + If the account is closed, no payments of any kind are made.
* Teller service fees are implemented with the Account.applyServiceFee(BigDecimal) method, which will be called by the UI as needed. If the teller indicates that the fee is waived, the method will not be called, but this is handled at the UI level.
* PaymentSchedule is essentially a convenient way to store the many global parameters which specify interest, fees, and the like. Each value is set by the operations manager. Implementation code accesses the current payment schedule using Bank.INSTANCE.getPaymentSchedule().
* Line of credit fees, interest, minimum payments, etc.:
  + At the end of the month, the required minimum payment is calculated using whichever is greater of the fixed or percentage minimum payments. For the percentage, the balance at the end of the month is used, before interest or fees are applied.
  + Throughout the month, the LineOfCredit keeps track of the (gross, not net) deposits made in the depositsToDate instance variable. This is checked against the calculated minimum payment to determine if it has been made. If it has not, the penalty applies, again before interest is accrued.
  + Finally, interest is calculated at the per-account interest rate and added to the outstanding balance.