import java.util.*;

public class Bank
{
private String bankName;           // the name of this Bank
private Terminal atm;              // for talking with the customer
private int balance = 0;           // total cash on hand
private int transactionCount = 0;  // number of Bank transactions
private Month month;               // the current month.

private TreeMap accountList;      // mapping names to accounts.

private static final String BANKER_COMMANDS =
"Banker commands: exit, open, customer, report, help.";

private static final String CUSTOMER_TRANSACTIONS =
"    Customer transactions: deposit, withdraw, transfer, balance, cash check, quit, help.
";

/**
* Construct a Bank with the given name and Terminal.
* @param bankName the name for this Bank.
* @param atm  this Bank's Terminal.
*/
public Bank( String bankName, Terminal atm )
{
this.atm = atm;
this.bankName = bankName;
accountList = new TreeMap();
month = new Month();
}

/**
* Simulates interaction with a Bank.
* Presents the user with an interactive loop, prompting for
* banker transactions and in case of the banker transaction
* "customer", an account id and further customer
* transactions.
*/
public void visit()
{
instructUser();

String command;
while (!(command =
atm.readWord("banker command: ")).equals("exit")) {
  if (command.startsWith("h")) {
    help( BANKER_COMMANDS );
  }
  else if (command.startsWith( "o" ) ) {
    openNewAccount();
  }
  else if (command.startsWith( "r" ) ) {
    report();
  }
  else if (command.startsWith("c")) {
    BankAccount acct = whichAccount();
    if ( acct != null )
      processTransactionsForAccount( acct );
  }
  else {
    // Unrecognized Request
    atm.println( "unknown command: " + command );
  }
}
report();
atm.println( "Goodbye from " + bankName );
}

private void openNewAccount()
{
String accountName = atm.readWord( "Account name: " );
char accountType = atm.readChar( "Checking/Fee/Regular? (c/f/r): " );
int startup = atm.readInt( "Initial deposit: " );
BankAccount newAccount;
switch( accountType ) {
  case 'c':
    newAccount = new CheckingAccount( startup, this );
    break;
  case 'f':
    newAccount = new FeeAccount( startup, this );
    break;
  case 'r':
    newAccount = new RegularAccount( startup, this );
    break;
  default:
    atm.println( "unknown account type: " + accountType );
    atm.printVoidWord( "Transaction failed" );
    break;
}
accountList.put( accountName, newAccount );
accountList.firstEntry().setValue( "Checking Account" );
accountList.lastEntry().setValue( "Regular Account" );
}
```java
break;
case 'f':
    newAccount = new FeeAccount( startup, this );
break;
case 'r':
    newAccount = new RegularAccount( startup, this );
break;
default:
    atm.println( "invalid account type: " + accountType );
    return;
}
accountList.put( accountName, newAccount );
atm.println( "opened new account " + accountName + " with $" + startup );

// Prompt the customer for transaction to process. Then send an appropriate message to the account.
private void processTransactionsForAccount( BankAccount acct ) {
    help( CUSTOMER_TRANSACTIONS );
    String transaction;
    while ( !(transaction = atm.readWord("    transaction: ")).equals("quit")) {
        if ( transaction.startsWith( "h" ) ) {
            help( CUSTOMER_TRANSACTIONS );
        } else if ( transaction.startsWith( "d" ) ) {
            int amount = atm.readInt( "    amount:" );
            atm.println("    deposited " + acct.deposit( amount ));
        } else if ( transaction.startsWith( "w" ) ) {
            int amount = atm.readInt( "    amount:" );
            atm.println("    withdrew " + acct.withdraw( amount ));
        } else if ( transaction.startsWith( "c" ) ) {
            int amount = atm.readInt( "    amount of check: " );
            atm.println("    cashed check for " + ((CheckingAccount)acct).honorCheck( amount ));
        } else if (transaction.startsWith("t")) {
            atm.print( "    to ");
            BankAccount toacct = whichAccount();
            if (toacct != null) {
                int amount = atm.readInt("    amount to transfer: ");
                atm.println("    transfered " + toacct.deposit(acct.withdraw(amount)));
            }
        } else if (transaction.startsWith("b")) {
            atm.println("    current balance " + acct.requestBalance());
        } else {
            atm.println("    sorry, unknown transaction" );
        }
    }
    atm.println();
}

// Prompt for an account name (or number), look it up in the account list. If it's there, return it; otherwise report an error and return null.
private BankAccount whichAccount() {
    String accountName = atm.readWord( "account name: ");
    BankAccount account = (BankAccount) accountList.get(accountName);
    if (account == null) {
        atm.println("not a valid account");
    }
    return account;
}

// Action to take when a new month starts. Update the month field by sending a next message. Loop on all accounts, sending each a newMonth message.
private void newMonth() {
    month.next();
    for (Iterator i = accountList.keySet().iterator(); i.hasNext(); ) {
        String accountName = (String) i.next();
        BankAccount acct = (BankAccount) accountList.get(accountName);
        atm.println(accountName + "	$" + acct.getBalance() + "		" + acct.getTransactionCount());
    }
    atm.println("Bank totals");
    atm.println("open accounts: " + getNumberOfAccounts());
    atm.println("cash on hand: $" + getBalance());
    atm.println("transactions:  " + getTransactionCount());
    atm.println();
}
```
private void instructUser()
{
atm.println( "Welcome to " + bankName );
atm.println( "Open some accounts and work with them." );
help( BANKER_COMMANDS );
}

private void help( String helpString )
{
atm.println( helpString );
atm.println();
}

/*
* Increment bank balance by given amount.
* 
* @param amount the amount increment.
*/

public void incrementBalance(int amount)
{
balance += amount;
}

/*
* Increment by one the count of transactions, 
* for this bank.
*/

public void countTransaction()
{
transactionCount++;;
}

/*
* Get the number of transactions performed by this bank.
* 
* @return number of transactions performed.
*/

public int getTransactionCount()
{
return transactionCount ;
}

/*
* Get the current bank balance.
* 
* @return current bank balance.
*/

public int getBalance()
{
return balance;
}

/*
* Get the current number of open accounts.
* 
* @return number of open accounts.
*/

public int getNumberOfAccounts()
{
return accountList.size();
}

public static void main( String[] args )
{
// parse the command line arguments for the echo
// flag and the name of the bank

boolean echo    = false;         // default does not echo
String bankName = "Faithless Trust"; // default bank name

for (int i = 0; i < args.length; i++ ) {
if (args[i].equals("-e")) {
    echo = true;
}
else {
    bankName = args[i];
}
}
Bank aBank = new Bank( bankName, new Terminal(echo) );
aBank.visit();
}