import java.util.*;

/**
* A Bank object simulates the behavior of a simple bank/ATM.
* It contains a Terminal object and a collection of BankAccount objects.
* The visit method opens this Bank for business, prompting the customer for input.
* To create a Bank and open it for business issue the command <code>java Bank</code>.
*
* @see BankAccount
* @version 5
*/

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.

    // what the banker can ask of the bank
    private static final String BANKER_COMMANDS =
        "Banker commands: exit, open, customer, report, help."
        "  Customer transactions: deposit, withdraw, transfer, balance, cash check, quit, help."
    ;

    // Open a new bank account,
    // prompting the user for information.
    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( "invalid choice: " )
                atm.println( "c = checking, f = fee, r = regular" );
                break;
        }
        accountList.put( accountName, newAccount );
        atm.println( "Acct " + newAccount.asString() );
    }

    public Bank( String bankName, Terminal atm )
    {
        this.atm = atm;
        this.bankName = bankName;
        accountList = new TreeMap();
    }

    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 );
    }

    // what the customer can ask of the bank
    private static final String CUSTOMER_TRANSACTIONS =
        "    Customer transactions: deposit, withdraw, transfer, balance, cash check, quit, help."
        "  Account commands: open, close, report, help."
    ;

    // Construct a bank with the given name and Terminal.
    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( "invalid choice: " )
                atm.println( "c = checking, f = fee, r = regular" );
                break;
        }
        accountList.put( accountName, newAccount );
        atm.println( "Acct " + newAccount.asString() );
    }

    // what the customer can ask of the bank
    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( "invalid choice: " )
                atm.println( "c = checking, f = fee, r = regular" );
                break;
        }
        accountList.put( accountName, newAccount );
        atm.println( "Acct " + newAccount.asString() );
    }
```java
113 break;
114 case 'f':
115     newAccount = new FeeAccount(startup, this);
116     break;
117 case 'r':
118     newAccount = new RegularAccount(startup, this);
119     break;
120 default:
121     atm.println("invalid account type: " + accountType);
122     return;
123 }
124 accountList.put(accountName, newAccount);
125     atm.println("opened new account " + accountName + " with $" + startup);
126 }
127
128 // Prompt the customer for transaction to process. Then send an appropriate message to the account.
129
130 private void processTransactionsForAccount(BankAccount acct) {
131     help(CUSTOMER_TRANSACTIONS);
132     
133     String transaction;
134     while (!(transaction =
135             atm.readWord("    transaction: ")).equals("quit")) {
136         if (transaction.startsWith("h")) {
137             help(CUSTOMER_TRANSACTIONS);
138         } else if (transaction.startsWith("d")) {
139             int amount = atm.readInt("    amount: ");
140             atm.println("    deposited " + acct.deposit(amount));
141         } else if (transaction.startsWith("w")) {
142             int amount = atm.readInt("    amount: ");
143             atm.println("    withdrew " + acct.withdraw(amount));
144         } else if (transaction.startsWith("c")) {
145             int amount = atm.readInt("    amount of check: ");
146             atm.println("    cashed check for " + ((CheckingAccount)acct).honorCheck(amount));
147         } else if (transaction.startsWith("t")) {
148             atm.print("    to ");
149             BankAccount toacct = whichAccount();
150             if (toacct != null) {
151                 int amount = atm.readInt("    amount to transfer: ");
152                 atm.println("    transfered " + toacct.deposit(acct.withdraw(amount)));
153             }
154         } else if (transaction.startsWith("b")) {
155             atm.println("    current balance " + acct.requestBalance());
156         } else {
157             atm.println("    sorry, unknown transaction");
158         }
159     }
160     atm.println();
161 }
162
163 // 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.
164
165 private BankAccount whichAccount() {
166     String accountName = atm.readWord("account name: ");
167     BankAccount account = (BankAccount) accountList.get(accountName);
168     if (account == null) {
169         atm.println("not a valid account");
170     }
171     return account;
172 }
173
174 // 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.
175
176 private void newMonth() {
177     month.next();
178     // for each account
179     //    account.newMonth();
180 }
181
182 // Report bank activity. For each BankAccount, print the customer id (name or number), account balance and the number of transactions. Then print Bank totals.
183
184 private void report() {
185     atm.println(bankName + " report for " + month);
186     atm.println("Summaries of individual accounts:");
187     atm.println("account  balance   transaction count");
188     for (Iterator i = accountList.keySet().iterator();
189             i.hasNext(); ) {
190         String accountName = (String) i.next();
191         BankAccount acct = (BankAccount) accountList.get(accountName);
192         atm.println(accountName + "	" + "$" + acct.getBalance() + 
193             "		" + acct.getTransactionCount());
194     }
195     atm.println("Bank totals");
196     atm.println("open accounts: " + getNumberOfAccounts());
197     atm.println("cash on hand: 
198             "$" + getBalance());
199     atm.println("transactions:  " + getTransactionCount());
200     atm.println();
201 }
202 }
// Welcome the user to the bank and instruct her on
// her options.

private void instructUser()
{
atm.println( "Welcome to " + bankName );
atm.println( "Open some accounts and work with them." );
help( BANKER_COMMANDS );
}

// Display a help string.

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 )
{
    int balance = 0;
    int transactionCount = 0;
    boolean echo = false;         // default does not echo
    String bankName = "Faithless Trust"; // default bank name

    for (int i = 0; i < args.length; i++ )    {
        String arg = args[i];
        if (arg.equals("-e")) {
            echo = true;
        } else {
            bankName = arg;
        }
    }

    Bank aBank = new Bank( bankName, new Terminal(echo) );
aBank.visit();
}