/**
 * A Bank object simulates the behavior of a simple bank/ATM.
 * It contains a Terminal object and a collection of BankAccount objects.
 *
 * Its public method visit opens the 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 4
 */

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 done

    private BankAccount[] accountList; /// collection of BankAccounts
    /// omit next line when accountList is dynamic
    private final static int NUM_ACCOUNTS = 3;

    // what the banker can ask of the bank

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

    // what the customer can ask of the bank

    private static final String CUSTOMER_TRANSACTIONS =
    "    Customer transactions: 
    deposit, withdraw, transfer, balance, 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;
        // initialize collection:
        accountList   = new BankAccount[NUM_ACCOUNTS]; ///

        /// When accountList is an array, fill it here.
        /// When it's an ArrayList or a TreeMap, delete these lines.
        accountList[0] = new BankAccount(  0, this);
        accountList[1] = new BankAccount(100, this);
        accountList[2] = new BankAccount(200, this);
    }

    /**
     * Simulates interaction with a Bank.
     * It opens the Bank for business and prompts the user for input.
     */

    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 );
    }

    // Open a new bank account,
    // prompting the user for information.

    /* * // Open a new bank account, prompting the user for information.
    */
private void openNewAccount()
{
    // when accountList is a dynamic collection
    // remove the next two lines, uncomment and complete
    atm.println(bankName + " is accepting no new customers
    return;

    /*
     // prompt for initial deposit
     int startup = atm.readInt("Initial deposit: ");
     // create newAccount
     BankAccount newAccount = new BankAccount(startup, this);
     // and add it to accountList
     //
     // inform user
     atm.println("opened new account " + ??? /// name or number
     + " with ") + newAccount.getBalance());
    */
    }

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("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();
}

private BankAccount whichAccount()
{
    /// prompt for account name or account number
    /// (whichever is appropriate)
    int accountNumber = atm.readInt("account number: ");

    /// look up account in accountList
    /// if it's there, return it
    /// else the following two lines should execute
    if ( accountNumber >= 0 && accountNumber < NUM_ACCOUNTS ) {
        return accountList[accountNumber];
    }
    else {
        atm.println("not a valid account" );
        return null;
    }
}

// Report bank activity.
// For each BankAccount, print the customer id (name or number),
// account balance and the number of transactions.
// Then print bank totals, print the number of transactions.
// For each bank account, print the current bank activity.
// Then print bank totals.

private void report()
{
    atm.println( "Summaries of individual accounts:" );
    atm.println( "account  balance   transaction count" );
    for (int i = 0; i < NUM_ACCOUNTS; i++ ) {              
        atm.println(i + "  " + accountList[i].getBalance() + "  " +
        accountList[i].getTransactionCount()); 
    }

    atm.println( "Bank totalsprefix report (account=Bank)"
    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 NUM_ACCOUNTS;    /// needs changing ...
}
/**
* A BankAccount object has private fields to keep track
* of its current balance, the number of transactions
* performed and the Bank in which it is an account, and
* and public methods to access those fields appropriately.
* 
* @see Bank
* @version 4
*/

public class BankAccount
{
private int balance = 0;          // Account balance (whole dollars)
private int transactionCount = 0; // Number of transactions performed.
private Bank issuingBank;          // Bank issuing this account

/**
* Construct a BankAccount with the given initial balance and
* issuing Bank. Construction counts as this BankAccount's
* first transaction.
*
* @param initialBalance the opening balance.
* @param issuingBank the bank that issued this account.
*/

public BankAccount( int initialBalance, Bank issuingBank )
{
this.issuingBank = issuingBank;
deposit( initialBalance );
}

/**
* Withdraw the given amount, decreasing this BankAccount's
* balance and the issuing Bank's balance. Counts as a transaction.
*
* @param amount the amount to be withdrawn
* @return amount withdrawn
*/

public int withdraw( int amount )
{
incrementBalance( -amount );
countTransaction();
return amount;
}

/**
* Deposit the given amount, increasing this BankAccount's
* balance and the issuing Bank's balance. Counts as a transaction.
*
* @param amount the amount to be deposited
* @return amount deposited
*/

public int deposit(int amount)
{
incrementBalance( amount);
countTransaction();
return amount;
}

/**
* Request for balance. Counts as a transaction.
*
* @return current account balance
*/

public int requestBalance()
{
countTransaction();
return getBalance();
}

/**
* Get the current balance. Does NOT count as a transaction.
*
* @return current account balance
*/

public int getBalance()
{
return balance;
}

/**
* Increment account balance by given amount. Also increment
* issuing Bank's balance. Does NOT count as a transaction.
*
* @param amount the amount increment.
*/

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

/**
* Get the number of transactions performed by this
* account. Does NOT count as a transaction.
*
* @return number of transactions performed.
*/

public int getTransactionCount()
{
return transactionCount;
}
}
public int getTransactionCount()
{
  return transactionCount;
}

/**
 * Increment by 1 the count of transactions, for this account
 * and for the issuing Bank.
 * Does NOT count as a transaction.
 * Get the Bank that issued this account.
 */

public void countTransaction()
{
  transactionCount++;
  this.getIssuingBank().countTransaction();
}

/**
 * Get the bank that issued this account.
 * Does NOT count as a transaction.
 *
 * @return issuing bank.
 */

public Bank getIssuingBank()
{
  return issuingBank;
}
<table>
<thead>
<tr>
<th>Action</th>
<th>Amount</th>
<th>Customer Balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>open</td>
<td>1000</td>
<td>9999</td>
</tr>
<tr>
<td>open</td>
<td>2000</td>
<td>9999</td>
</tr>
<tr>
<td>help</td>
<td></td>
<td></td>
</tr>
<tr>
<td>report</td>
<td></td>
<td></td>
</tr>
<tr>
<td>open</td>
<td>3000</td>
<td>9999</td>
</tr>
<tr>
<td>transfer</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>transfer</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>quit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>exit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>customer</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>deposit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>balance</td>
<td></td>
<td></td>
</tr>
<tr>
<td>customer</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>transfer</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>transfer</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>45</td>
<td></td>
<td></td>
</tr>
<tr>
<td>quit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>exit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>customer</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>deposit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>balance</td>
<td></td>
<td></td>
</tr>
<tr>
<td>customer</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>deposit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>balance</td>
<td></td>
<td></td>
</tr>
<tr>
<td>customer</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>deposit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>balance</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Welcome to River Bank
Open some accounts and work with them.
Banker commands: exit, open, customer, report, help.

banker command: open
Initial deposit: 1000
opened new account 0 with $1000

banker command: open
Initial deposit: 2000
opened new account 1 with $2000

banker command: help
Banker commands: exit, open, customer, report, help.

banker command: report
Summaries of individual accounts:
account  balance   transaction count
0          $1000       1
1          $2000       1

Bank totals
open accounts: 2
cash on hand: $3000
transactions: 2

banker command: open
Initial deposit: 3000
opened new account 2 with $3000

banker command: customer
account number: 0
Customer transactions: deposit, withdraw, transfer, balance, quit
transaction: balance
current balance 1000
transaction: deposit
amount: 9999
deposited 9999
transaction: balance
current balance 10999
transaction: quit

banker command: customer
account number: 1
Customer transactions: deposit, withdraw, transfer, balance, quit
transaction: transfer
to account number: 9
not a valid account
transaction: transfer
to account number: 2
amount to transfer: 45
transfered 45
transaction: quit

Goodbye from River Bank

1 open
2 groucho
3 10000
4 customer
5 harpo
6 open
7 harpo
8 20000
9 help
10 report
11 balance
12 deposit
13 balance
14 quit
15 customer
16 chico
17 3000
18 customer
19 balance
20 deposit
21 balance
22 quit
23 exit
Welcome to River Bank
Open some accounts and work with them.
Banker commands: exit, open, customer, report, help.

banker command: open
Account name: groucho
Initial deposit: 1000
opened new account groucho with $1000

banker command: open
Account name: harpo
Initial deposit: 2000
opened new account harpo with $2000

banker command: help
Banker commands: exit, open, customer, report, help.

banker command: report
Summaries of individual accounts:

<table>
<thead>
<tr>
<th>account</th>
<th>balance</th>
<th>transaction count</th>
</tr>
</thead>
<tbody>
<tr>
<td>groucho</td>
<td>$1000</td>
<td>1</td>
</tr>
<tr>
<td>harpo</td>
<td>$2000</td>
<td>1</td>
</tr>
</tbody>
</table>

Bank totals
open accounts: 2
cash on hand: $3000
transactions: 2

banker command: open
Account name: chico
Initial deposit: 3000
opened new account chico with $3000

banker command: customer
account name: groucho
Customer transactions: deposit, withdraw, transfer, balance, quit, help.

transaction: balance
current balance 1000

transaction: deposit
amount: 9999
deposited 9999

transaction: balance
current balance 10999

transaction: quit

banker command: customer
account name: harpo
Customer transactions: deposit, withdraw, transfer, balance, quit, help.

transaction: transfer
to account name: chico
amount to transfer: 45
transfered 45

transaction: quit

banker command: exit
Summaries of individual accounts:

<table>
<thead>
<tr>
<th>account</th>
<th>balance</th>
<th>transaction count</th>
</tr>
</thead>
<tbody>
<tr>
<td>chico</td>
<td>$3045</td>
<td>2</td>
</tr>
<tr>
<td>groucho</td>
<td>$10999</td>
<td>4</td>
</tr>
<tr>
<td>harpo</td>
<td>$19552</td>
<td>2</td>
</tr>
</tbody>
</table>

Bank totals
open accounts: 3
cash on hand: $15999
transactions: 8

Goodbye from River Bank