// Copyright 2003 Bill Campbell and Ethan Bolker

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.
 *
 * 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]; ///
        accountList[0] = new BankAccount(  0, this);
        accountList[1] = new BankAccount(100, this);
        accountList[2] = new BankAccount(200, this);
    }

    /**
     * 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);
    }

    // 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());
     */
}

// 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("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()
{
    /// 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.
// account balance and the number of transactions.

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 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 NUM_ACCOUNTS; // needs changing ... 
}

/**
* Run the simulation by creating and then visiting a new Bank.
* <p>
* A -e argument causes the input to be echoed. This can be useful for executing the program against
* a test script, e.g.,
* <pre>
*   java Bank -e < Bank.in
* </pre>
* 
* @param args the command line arguments:
*         <pre>
*         -e echo input.
*         bankName any other command line argument.
*         </pre>
*/
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 = "River Bank"; // 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();
}