Listing 1.1 BankAccount.java

1// joi/1/bank/BankAccount.java
2//
3// Copyright 2003 Bill Campbell and Ethan Bolker
4
5/**
6* A BankAccount object has a private field to keep track
7* of this account's current balance, and public methods to
8* return and change the balance.
9*
10* @see Bank
11* @version 1
12*/
13
14public class BankAccount
15{
16private int balance;  // work only in whole dollars
17
18/**
19* A constructor for creating a new bank account.
20*
21* @param initialBalance the opening balance.
22*/
23
24public BankAccount( int initialBalance )
25{
26this.deposit( initialBalance );
27}
28
29/**
30* Withdraw the amount requested.
31*
32* @param amount the amount to be withdrawn.
33*/
34
35public void withdraw( int amount )
36{
37balance = balance - amount;
38}
39
40/**
41* Deposit the amount requested.
42*
43* @param amount the amount to be deposited.
44*/
45
46public void deposit( int amount )
47{
48balance = balance + amount;
49}
50
51/**
52* The current account balance.
53*
54* @return the current balance.
55*/
56
57public int getBalance()
58{
59return balance;
60}
61}
62}
// Copyright 2003 Bill Campbell and Ethan Bolker

/**
* A Bank object simulates the behavior of a simple bank/ATM.
* It contains a Terminal object and two BankAccount objects.
*
* Its single public method is open, which 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 1
*/

public class Bank
{
    private String bankName;      // the name of this Bank

    private Terminal atm;         // for talking with the customer

    private BankAccount account1; // two accounts to play with
    private BankAccount account2;

    private static final int INITIAL_BALANCE = 200;
    private static final String HELPSTRING =
        "Transactions: exit, help, deposit, withdraw, balance";

    /**
    * Construct a Bank with the given name.
    * Create two new BankAccounts, each with a starting balance
    * of initialBalance.
    *
    * @param name the name of the Bank.
    */
    public Bank( String name )
    {
        bankName = name;
        atm      = new Terminal();
        account1 = new BankAccount( INITIAL_BALANCE );
        account2 = new BankAccount( INITIAL_BALANCE );
    }

    /**
    * Open the Bank for business.
    *
    * Send a whichAccount message prompting for a BankAccount
    * number, then send an appropriate message to account.
    *
    * Prompt the user for transaction to process.
    */
    public void open()
    {
        atm.println( "Welcome to " + bankName );
        boolean bankIsOpen = true;
        while ( bankIsOpen ) {
            BankAccount account = this.whichAccount();
            if ( account == null ) {
                bankIsOpen = false;
            }
            else {
                this.processTransactionsForAccount(account);
            }
        }
        atm.println( "Goodbye from " + bankName );
    }

    // Prompt the user for an account number and return the
    // corresponding BankAccount object. Return null when
    // the Bank is about to close.

    private BankAccount whichAccount()
    {
        int  accountNumber =
            atm.readInt("Account number (1 or 2), 0 to shut down: ");
        if ( accountNumber == 1 ) {
            return account1;
        }
        else if ( accountNumber == 2 ) {
            return account2;
        }
        else if ( accountNumber == 0 ) {
            return null;
        }
        else {
            atm.println( "No account numbered "+
                accountNumber + "; try again" );
            return this.whichAccount();
        }
    }

    // Prompt the user for transaction to process.
    // Then send an appropriate message to account.

    private void processTransactionsForAccount( BankAccount account)
    {
        atm.println( HELPSTRING );

        boolean moreTransactions = true;
        while ( moreTransactions ) {
            String command = atm.readWord( "transaction: " );
            if ( command.equals( "exit" ) ) {
                moreTransactions = false;
            }
            else if ( command.equals( "help" ) ) {
                atm.println( HELPSTRING );
            }
        }
    }

```java
else if (command.equals( "deposit" ) ) {
    int amount = atm.readInt( "amount: " );
    account.deposit( amount );
}
else if (command.equals( "withdraw" ) ) {
    int amount = atm.readInt( "amount: " );
    account.withdraw( amount );
}
else if (command.equals( "balance" ) ) {
    atm.println( account.getBalance() );
}
else{
    atm.println("sorry, unknown transaction");
}
```