/**
 * Copyright 2003 Bill Campbell and Ethan Bolker
 */

@version 1

public class BankAccount {
    private int balance; // work only in whole dollars

    /**
     * A constructor for creating a new bank account.
     * @param initialBalance the opening balance.
     */
    public BankAccount( int initialBalance ) {
        this.deposit( initialBalance );
    }

    /**
     * Withdraw the amount requested.
     * @param amount the amount to be withdrawn.
     */
    public void withdraw( int amount ) {
        balance = balance - amount;
    }

    /**
     * Deposit the amount requested.
     * @param amount the amount to be deposited.
     */
    public void deposit( int amount ) {
        balance = balance + amount;
    }

    /**
     * The current account balance.
     * @return the current balance.
     */
    public int getBalance() {
        return balance;
    }

    // other methods
}
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.

To create a Bank and open it for business issue the command <code>java Bank</code>.

@see BankAccount
@version 1

```java
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";

    public Bank( String name ) {
        bankName = name;
        atm = new Terminal();
        account1 = new BankAccount( INITIAL_BALANCE );
        account2 = new BankAccount( INITIAL_BALANCE );
    }

    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 );
    }

    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.
    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 );
                atm.println("transactions: exit, help, deposit, withdraw, balance" );
                continue;
            } else if ( command.equals( "deposit" ) ) {
                atm.println( "Enter deposit amount: ");
                double amount = atm.readNumber();
                account.deposit(amount);
            } else if ( command.equals( "withdraw" ) ) {
                atm.println( "Enter withdrawal amount: ");
                double amount = atm.readNumber();
                account.withdraw(amount);
            } else if ( command.equals( "balance" ) ) {
                atm.println( "Your balance is: " + account.getBalance() );
            } else {
                atm.println( "Unknown command: "+command );
            }
        }
    }
}
```

```java
113}
114} else if (command.equals("deposit")) {
115} int amount = atm.readInt("amount: ");
116} account.deposit(amount);
117} else if (command.equals("withdraw")) {
118} int amount = atm.readInt("amount: ");
119} account.withdraw(amount);
120} else if (command.equals("balance")) {
121} atm.println(account.getBalance());
122} else {
123} atm.println("sorry, unknown transaction");
124} }
125}
126} */
127} /** Program args the command line arguments (handed).
128} Issue the command code.java bank/c ode.
129} The bank simulation program begins here when the user
130} */
131}
132}
133} /**
134} * The Bank simulation program begins here when the user
135} * issues the command code.java bank/c ode.
136} * 
137} * @param args the command line arguments (ignored).
138} */
139}
140} public static void main(String[] args) {
141} Bank javaBank = new Bank("Engulf and Devour");
142} javaBank.open();
143}
144}
145} /**
146} * @param account an account, got from the database.
147} */
148} { } account.open()
149} { { } account.printBalance()
150} if (command == "deposit") {
151} { } account.deposit(amount);
152} else if (command == "withdraw") {
153} { } account.withdraw(amount);
154} else if (command == "balance") {
155} { } account.printBalance();
156} return;