ЗАТВЕРДЖЕНО

КІТ26В.15395 - 01 12 01

РОЗРОБКА МОДЕЛЮВАННЯ БАНКОМАТУ «MYBANKATM»

Текст програми

КІТ26В.15395-01 12 01

Аркушів 17

Харків 2018

ЗМІСТ

atmTopComponent.java 3

Account.java 11

AccumulateSavingBatch.java 12

Bank.java 13

CheckingAccount.java 14

Customer.java 15

OverdraftException.java 16

SavingAccount.java 18

1. Файл fragments atmTopComponent.java

package com.mybank.atm;

import com.mybank.domain.\*;

import java.awt.Component;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.JButton;

import org.netbeans.api.settings.ConvertAsProperties;

import org.openide.awt.ActionID;

import org.openide.awt.ActionReference;

import org.openide.windows.TopComponent;

import org.openide.util.NbBundle.Messages;

/\*\*

\* Top component which displays something.

\*/

@ConvertAsProperties(

dtd = "-//com.mybank.atm//atm//EN",

autostore = false

)

@TopComponent.Description(

preferredID = "atmTopComponent",

//iconBase="SET/PATH/TO/ICON/HERE",

persistenceType = TopComponent.PERSISTENCE\_ALWAYS

)

@TopComponent.Registration(mode = "editor", openAtStartup = true)

@ActionID(category = "Window", id = "com.mybank.atm.atmTopComponent")

@ActionReference(path = "Menu/Window" /\*, position = 333 \*/)

@TopComponent.OpenActionRegistration(

displayName = "#CTL\_atmAction",

preferredID = "atmTopComponent"

)

@Messages({

"CTL\_atmAction=atm",

"CTL\_atmTopComponent=atm Window",

"HINT\_atmTopComponent=This is a atm window"

})

public final class atmTopComponent extends TopComponent {

Bank bank;

Customer currentCustomer;

Account currentAccount;

public atmTopComponent() {

bank = Bank.getBank();

Customer firstCustomer = new Customer("John", "Doe");

Customer secondCustomer = new Customer("Jane", "Doe");

SavingsAccount johnSavings = new SavingsAccount(1000, 5);

CheckingAccount johnAccount = new CheckingAccount(5000, 1000);

CheckingAccount janeAccount = new CheckingAccount(500, 100);

firstCustomer.addAccount(johnSavings);

firstCustomer.addAccount(johnAccount);

secondCustomer.addAccount(janeAccount);

bank.addCustomer(firstCustomer);

bank.addCustomer(secondCustomer);

initComponents();

setName(Bundle.CTL\_atmTopComponent());

setToolTipText(Bundle.HINT\_atmTopComponent());

for (Component c : jPanel3.getComponents()) {

if ((c.getClass() == JButton.class) && (!"ENTER".equals(((JButton) c).getText()))) {

JButton currentButton = (JButton) c;

currentButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

addDigit(e);

}

});

}

}

}

private void addDigit(ActionEvent evt) {

// TODO add your handling code here:

amountField.setText(amountField.getText() + ((JButton) evt.getSource()).getText());

}

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

jPanel1 = new javax.swing.JPanel();

jPanel2 = new javax.swing.JPanel();

balanceButton = new javax.swing.JButton();

depositButton = new javax.swing.JButton();

withdrawButton = new javax.swing.JButton();

amountField = new javax.swing.JTextField();

jPanel3 = new javax.swing.JPanel();

oneButton = new javax.swing.JButton();

twoButton = new javax.swing.JButton();

threeButton = new javax.swing.JButton();

fourButton = new javax.swing.JButton();

fiveButton = new javax.swing.JButton();

sixButton = new javax.swing.JButton();

sevenButton = new javax.swing.JButton();

eightButton = new javax.swing.JButton();

nineButton = new javax.swing.JButton();

zeroButton = new javax.swing.JButton();

pointButton = new javax.swing.JButton();

enterButton = new javax.swing.JButton();

jScrollPane1 = new javax.swing.JScrollPane();

historyArea = new javax.swing.JTextArea();

statusField = new javax.swing.JTextField();

setLayout(new java.awt.BorderLayout());

jPanel1.setLayout(new java.awt.GridLayout(2, 1));

jPanel2.setLayout(new java.awt.GridLayout(4, 1));

org.openide.awt.Mnemonics.setLocalizedText(balanceButton, org.openide.util.NbBundle.getMessage(atmTopComponent.class, "atmTopComponent.balanceButton.text")); // NOI18N

balanceButton.setEnabled(false);

balanceButton.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

balanceButtonActionPerformed(evt);

}

});

jPanel2.add(balanceButton);

org.openide.awt.Mnemonics.setLocalizedText(depositButton, org.openide.util.NbBundle.getMessage(atmTopComponent.class, "atmTopComponent.depositButton.text")); // NOI18N

depositButton.setEnabled(false);

depositButton.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

depositButtonActionPerformed(evt);

}

});

jPanel2.add(depositButton);

org.openide.awt.Mnemonics.setLocalizedText(withdrawButton, org.openide.util.NbBundle.getMessage(atmTopComponent.class, "atmTopComponent.withdrawButton.text")); // NOI18N

withdrawButton.setEnabled(false);

withdrawButton.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

withdrawButtonActionPerformed(evt);

}

});

jPanel2.add(withdrawButton);

amountField.setToolTipText(org.openide.util.NbBundle.getMessage(atmTopComponent.class, "atmTopComponent.amountField.toolTipText")); // NOI18N

jPanel2.add(amountField);

jPanel1.add(jPanel2);

jPanel3.setLayout(new java.awt.GridLayout(4, 3));

org.openide.awt.Mnemonics.setLocalizedText(oneButton, org.openide.util.NbBundle.getMessage(atmTopComponent.class, "atmTopComponent.oneButton.text")); // NOI18N

jPanel3.add(oneButton);

org.openide.awt.Mnemonics.setLocalizedText(twoButton, org.openide.util.NbBundle.getMessage(atmTopComponent.class, "atmTopComponent.twoButton.text")); // NOI18N

jPanel3.add(twoButton);

org.openide.awt.Mnemonics.setLocalizedText(threeButton, org.openide.util.NbBundle.getMessage(atmTopComponent.class, "atmTopComponent.threeButton.text")); // NOI18N

jPanel3.add(threeButton);

org.openide.awt.Mnemonics.setLocalizedText(fourButton, org.openide.util.NbBundle.getMessage(atmTopComponent.class, "atmTopComponent.fourButton.text")); // NOI18N

jPanel3.add(fourButton);

org.openide.awt.Mnemonics.setLocalizedText(fiveButton, org.openide.util.NbBundle.getMessage(atmTopComponent.class, "atmTopComponent.fiveButton.text")); // NOI18N

jPanel3.add(fiveButton);

org.openide.awt.Mnemonics.setLocalizedText(sixButton, org.openide.util.NbBundle.getMessage(atmTopComponent.class, "atmTopComponent.sixButton.text")); // NOI18N

jPanel3.add(sixButton);

org.openide.awt.Mnemonics.setLocalizedText(sevenButton, org.openide.util.NbBundle.getMessage(atmTopComponent.class, "atmTopComponent.sevenButton.text")); // NOI18N

jPanel3.add(sevenButton);

org.openide.awt.Mnemonics.setLocalizedText(eightButton, org.openide.util.NbBundle.getMessage(atmTopComponent.class, "atmTopComponent.eightButton.text")); // NOI18N

jPanel3.add(eightButton);

org.openide.awt.Mnemonics.setLocalizedText(nineButton, org.openide.util.NbBundle.getMessage(atmTopComponent.class, "atmTopComponent.nineButton.text")); // NOI18N

jPanel3.add(nineButton);

org.openide.awt.Mnemonics.setLocalizedText(zeroButton, org.openide.util.NbBundle.getMessage(atmTopComponent.class, "atmTopComponent.zeroButton.text")); // NOI18N

jPanel3.add(zeroButton);

org.openide.awt.Mnemonics.setLocalizedText(pointButton, org.openide.util.NbBundle.getMessage(atmTopComponent.class, "atmTopComponent.pointButton.text")); // NOI18N

jPanel3.add(pointButton);

org.openide.awt.Mnemonics.setLocalizedText(enterButton, org.openide.util.NbBundle.getMessage(atmTopComponent.class, "atmTopComponent.enterButton.text")); // NOI18N

enterButton.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

enterButtonActionPerformed(evt);

}

});

jPanel3.add(enterButton);

jPanel1.add(jPanel3);

add(jPanel1, java.awt.BorderLayout.LINE\_START);

historyArea.setEditable(false);

historyArea.setColumns(20);

historyArea.setRows(5);

jScrollPane1.setViewportView(historyArea);

add(jScrollPane1, java.awt.BorderLayout.CENTER);

statusField.setEditable(false);

statusField.setText(org.openide.util.NbBundle.getMessage(atmTopComponent.class, "atmTopComponent.statusField.text")); // NOI18N

add(statusField, java.awt.BorderLayout.PAGE\_END);

}// </editor-fold>

private void balanceButtonActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

historyArea.append("Balance of " + currentCustomer.getFirstName() + "'s first account is $" + currentAccount.getBalance());

if (currentAccount instanceof CheckingAccount) {

historyArea.append(". This is a Checking Account with overdraft protection $" + ((CheckingAccount) currentAccount).getOverdraftAmount() + "\n");

} else {

historyArea.append(". This is a Savings Account with interest rate " + ((SavingsAccount) currentAccount).getInterestRate() + "%\n");

}

statusField.setText("READY");

}

private void depositButtonActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

try {

double amt = Double.parseDouble(amountField.getText());

currentAccount.deposit(amt);

historyArea.append("Deposit: $" + amt + ", new balance is $" + currentAccount.getBalance() + "\n");

statusField.setText("READY");

} catch (Exception e) {

historyArea.append("ERROR: can't complete deposit oiperation!\n");

statusField.setText("ERROR");

}

amountField.setText("");

}

private void withdrawButtonActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

try {

double amt = Double.parseDouble(amountField.getText());

if (currentAccount.withdraw(amt)) {

historyArea.append("Deposit: $" + amt + ", new balance is $" + currentAccount.getBalance() + "\n");

statusField.setText("READY");

}

} catch (OverdraftException ex) {

historyArea.append("ERROR: Insufficient funds!\n");

} catch (Exception e) {

historyArea.append("ERROR: can't complete deposit oiperation!\n");

statusField.setText("ERROR");

}

amountField.setText("");

}

private void enterButtonActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

int customerID = 0;

try {

customerID = Integer.parseInt(amountField.getText());

currentCustomer = bank.getCustomer(customerID);

currentAccount = currentCustomer.getAccount(0);

historyArea.append("Customer with ID = " + customerID + " is " + currentCustomer.getLastName() + ", " + currentCustomer.getFirstName() + "\n");

balanceButton.setEnabled(true);

depositButton.setEnabled(true);

withdrawButton.setEnabled(true);

enterButton.setEnabled(false);

} catch (Exception ex) {

historyArea.append("ERROR: Customer not found or wrong Customer ID!\n");

}

amountField.setText("");

statusField.setText("Customer: "+currentCustomer.getLastName() + ", " + currentCustomer.getFirstName());

}

// Variables declaration - do not modify

private javax.swing.JTextField amountField;

private javax.swing.JButton balanceButton;

private javax.swing.JButton depositButton;

private javax.swing.JButton eightButton;

private javax.swing.JButton enterButton;

private javax.swing.JButton fiveButton;

private javax.swing.JButton fourButton;

private javax.swing.JTextArea historyArea;

private javax.swing.JPanel jPanel1;

private javax.swing.JPanel jPanel2;

private javax.swing.JPanel jPanel3;

private javax.swing.JScrollPane jScrollPane1;

private javax.swing.JButton nineButton;

private javax.swing.JButton oneButton;

private javax.swing.JButton pointButton;

private javax.swing.JButton sevenButton;

private javax.swing.JButton sixButton;

private javax.swing.JTextField statusField;

private javax.swing.JButton threeButton;

private javax.swing.JButton twoButton;

private javax.swing.JButton withdrawButton;

private javax.swing.JButton zeroButton;

// End of variables declaration

@Override

public void componentOpened() {

// TODO add custom code on component opening

}

@Override

public void componentClosed() {

// TODO add custom code on component closing

}

void writeProperties(java.util.Properties p) {

// better to version settings since initial version as advocated at

// http://wiki.apidesign.org/wiki/PropertyFiles

p.setProperty("version", "1.0");

// TODO store your settings

}

void readProperties(java.util.Properties p) {

String version = p.getProperty("version");

// TODO read your settings according to their version

}

}

2. Файл Account.java

package com.mybank.domain;

public class Account {

protected double balance;

/\*\*

\* Constructor that provides initial balance

\*

\* @param balance a positive account balance

\*/

protected Account(double balance) {

if (balance >= 0) {

this.balance = balance;

} else {

this.balance = 0;

}

}

public Account() {

this.balance = 0;

}

/\*\*

\* Method to add money to account

\*

\* @param amt a positive amount of money

\* @return

\*/

public boolean deposit(double amt) {

if (amt > 0) {

balance += amt;

return true;

}

return false;

}

/\*\*

\* Method to withdraw money

\*

\* @param amt a positive amount of money

\*/

public boolean withdraw(double amt) throws Exception{

if (amt <= balance) {

balance -= amt;

return true;

}

return false;

}

/\*\*

\* Method to check account balance

\*

\* @return the balance

\*/

public double getBalance() {

return balance;

}}

3. файл AccumulateSavingBatch.java

package com.mybank.domain;

import com.mybank.domain.\*;

public class AccumulateSavingsBatch {

public AccumulateSavingsBatch() {

}

public void doBatch() {

Bank bank= Bank.getBank();

// For each customer...

for ( int cust\_idx = 0;

cust\_idx < bank.getNumOfClients();

cust\_idx++ ) {

Customer customer = bank.getCustomer(cust\_idx);

// For each account for this customer...

for ( int acct\_idx = 0;

acct\_idx < customer.getNumOfAccounts();

acct\_idx++ ) {

Account account = customer.getAccount(acct\_idx);

String account\_type = "";

// Determine the account type

if ( account instanceof SavingsAccount ) {

SavingsAccount savings = (SavingsAccount) account;

savings.addInterestRate();

} else {

// ignore all other account types

}

}

}

}

}

4 файл Bank.java

package com.mybank.domain;

import java.util.ArrayList;

public class Bank {

private ArrayList<Customer> customers = new ArrayList<Customer>();

private int numOfClients=0;

private static Bank myBank = new Bank();

public static Bank getBank() {

return myBank;

}

private Bank() {

}

public Customer getCustomer(int custNo) {

if (custNo < customers.size()) {

return customers.get(custNo);

}

return null;

}

public void addCustomer(Customer newCustomer) {

customers.add(newCustomer);

numOfClients++;

}

public int getNumOfClients() {

return numOfClients;

}

}

5 файл CheckingAccount.java

package com.mybank.domain;

public class CheckingAccount extends Account {

private double overdraftAmount;

public CheckingAccount(double initBalance, double overdraftAmount) {

this.balance=initBalance;

this.overdraftAmount=overdraftAmount;

}

public CheckingAccount(double initBalance) {

this(initBalance, 0);

}

public boolean withdraw(double amt) throws OverdraftException{

if(amt<=balance+overdraftAmount){

balance=balance-amt;

return true;

}

throw new OverdraftException(amt-balance-overdraftAmount, "Error! Insufficient funds");

}

public double getOverdraftAmount() {

return overdraftAmount;

}}

6.Файл Customer.java

package com.mybank.domain;

import java.util.ArrayList;

public class Customer {

private ArrayList<Account> accounts;

private String firstName;

private String lastName;

public int getNumOfAccounts() {

return numOfAccounts;

}

public String getFirstName() {

return firstName;

}

public String getLastName() {

return lastName;

}

private int customerNumber;

private static int customerNumberBase = 1000;

private int numOfAccounts;

public Customer(String firstName, String lastName) {

accounts = new ArrayList<Account>();

this.firstName = firstName;

this.lastName = lastName;

this.customerNumber = customerNumberBase++;

this.numOfAccounts = 0;

}

public Account getAccount(int accNo) {

if (accNo < accounts.size() && numOfAccounts != 0) {

return accounts.get(accNo);

}

return null;

}

@Override

public String toString() {

String s = "Customer :" + "fullName=" + lastName + ", "+firstName+", customerNumber=" + customerNumber + ", numOfAccounts=" + numOfAccounts;

for (int i = 0; i < this.numOfAccounts; i++) {

if (getAccount(i) instanceof SavingsAccount) {

s = s + "\n Savings Account with interest rate " + ((SavingsAccount) getAccount(i)).getInterestRate() + "%";

} else {

s = s + "\n Checking Account with overdraft $" + ((CheckingAccount) getAccount(i)).getOverdraftAmount();

}

s = s + ", balance $" + getAccount(i).getBalance();

}

return s;

}

public void addAccount(Account acc) {

accounts.add(acc);

numOfAccounts++;

}

}

7.Файл OverdraftException.java

package com.mybank.domain;

public class OverdraftException extends Exception{

private double deficit;

public double getDeficit() {

return deficit;

}

public OverdraftException(double deficit, String message) {

super(message);

this.deficit = deficit;

}

}

8.Файл SavingsAccount.java

package com.mybank.domain;

public class SavingsAccount extends Account {

private double interestRate;

public SavingsAccount(double initBalance, double interestRate) {

this.balance =initBalance;

this.interestRate=interestRate;

}

public void addInterestRate()

{

this.balance=this.balance\*(1+interestRate);

}

public double getInterestRate() {

return interestRate;

}

}