package atmsimulation;

atm application:

import java.awt.\*;

import javax.swing.\*;

public class ATMApplication {

boolean packFrame = false;

public ATMApplication(String datasource) {

try {

init(datasource);

} catch (Exception e) {

e.printStackTrace();

}

}

private void init(String datasource) throws Exception {

ATMFrame frame = new ATMFrame(datasource);

if (packFrame) {

frame.pack();

} else {

frame.validate();

}

Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();

Dimension frameSize = frame.getSize();

if (frameSize.height > screenSize.height) {

frameSize.height = screenSize.height;

}

if (frameSize.width > screenSize.width) {

frameSize.width = screenSize.width;

}

frame.setLocation((screenSize.width - frameSize.width) / 2,

(screenSize.height - frameSize.height) / 2);

frame.setVisible(true);

}

public static void main(String[] args) {

try {

new ATMApplication("ATM Application");

} catch (Exception e) {

e.printStackTrace();

}

}

}

Atm java:

package atmsimulation;

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

public class ATMFrame extends JFrame {

JPanel contentPane;

LayoutManager layout;

ATMMainPanel pnlATM;

public ATMFrame(String datasource) {

enableEvents(AWTEvent.WINDOW\_EVENT\_MASK);

try {

init(datasource);

}

catch(Exception e) {

e.printStackTrace();

}

}

private void init(String datasource) throws Exception {

contentPane = (JPanel) this.getContentPane();

layout = new BorderLayout();

pnlATM = new ATMMainPanel(datasource);

contentPane.setLayout(layout);

contentPane.add(pnlATM, BorderLayout.CENTER);

this.setSize(new Dimension(600, 300));

this.setTitle("ATM Application");

}

protected void processWindowEvent(WindowEvent e) {

super.processWindowEvent(e);

if (e.getID() == WindowEvent.WINDOW\_CLOSING) {

System.exit(0);

}

}

}

Atm main panel

package atmsimulation;

import java.awt.\*;

import java.awt.event.\*;

import java.sql.\*;

import javax.swing.\*;

public class ATMMainPanel extends JPanel implements ActionListener {

private LayoutManager layout;

private KeyPad keypad;

private Display display;

private String dbsource;

private Connection connection;

private Customer customer;

public ATMMainPanel(String datasource) {

try {

init(datasource);

} catch (Exception e) {

e.printStackTrace();

}

}

private void init(String datasource) throws Exception {

// dbsource = datasource;

String dbname = "illusion\_db";

String url = "jdbc:mysql://localhost:3306/";

Class.forName("com.mysql.jdbc.Driver");

connection = DriverManager.getConnection(url + dbname, "root", "");

layout = new GridBagLayout();

setLayout(layout);

initKeyPad();

add(keypad,

new GridBagConstraints(0, 0, 1, 1, 1.0, 1.0,

GridBagConstraints.CENTER,

GridBagConstraints.BOTH,

new Insets(3, 3, 3, 3), 0, 0));

initDisplay();

add(display,

new GridBagConstraints(1, 0, 1, 1, 3.0, 1.0,

GridBagConstraints.CENTER,

GridBagConstraints.BOTH,

new Insets(3, 3, 3, 3), 0, 0));

}

private void initKeyPad() {

keypad = new KeyPad();

keypad.addKeyActionListener(this);

}

private void initDisplay() {

display = new Display();

display.addControlActionListener(this);

}

public void actionPerformed(ActionEvent ae) {

String btntext = ((JButton) ae.getSource()).getText();

switch (display.getState()) {

case Display.PIN: {

String text = display.getPINInputLabel().getText();

if (btntext.equals("Enter")) {

try {

customer = new Customer(connection, Integer.parseInt(text.trim()));

display.setCustomer(customer);

display.setState(Display.ACCOUNTS);

} catch (Exception e) {

e.printStackTrace();

text = "Bad PIN";

}

} else if (btntext.equals("CE")) {

text = " ";

} else if (text.length() > 4) {

text = "Error";

} else if (btntext.length() == 1) {

text += btntext;

}

display.getPINInputLabel().setText(text);

}

break; //end case PIN

case Display.ACCOUNTS: {

if (btntext.equals("Deposit")) {

display.setState(Display.DEPOSIT);

} else if (btntext.equals("Withdraw")) {

display.setState(Display.WITHDRAW);

} else if (btntext.equals("Cancel")) {

display.setState(Display.PIN);

}

}

break; //end case ACCOUNTS

case Display.DEPOSIT:

case Display.WITHDRAW: {

boolean deposit = (display.getState() == Display.DEPOSIT);

boolean withdraw = (display.getState() == Display.WITHDRAW);

if (btntext.equals("Current")) {

if (deposit) {

display.setState(Display.DEPOSIT\_CHECKING);

} else if (withdraw) {

display.setState(Display.WITHDRAW\_CHECKING);

}

} else if (btntext.equals("Savings")) {

if (deposit) {

display.setState(Display.DEPOSIT\_SAVINGS);

} else if (withdraw) {

display.setState(Display.WITHDRAW\_SAVINGS);

}

} else if (btntext.equals("Cancel")) {

display.setState(Display.ACCOUNTS);

}

}

break; //end case DEPOSIT

case Display.DEPOSIT\_CHECKING:

case Display.DEPOSIT\_SAVINGS:

case Display.WITHDRAW\_CHECKING:

case Display.WITHDRAW\_SAVINGS: {

String text = display.getTransactInputLabel().getText();

boolean deposit = btntext.equals("Deposit");

boolean withdraw = btntext.equals("Withdraw");

if (deposit || withdraw) {

try {

if (deposit) {

switch (display.getState()) {

case Display.DEPOSIT\_CHECKING: {

customer.getChecking().deposit(

Double.parseDouble(text.trim()));

}

break; //end case DEPOSIT\_CHECKING

case Display.DEPOSIT\_SAVINGS: {

customer.getSavings().deposit(

Double.parseDouble(text.trim()));

}

break; //end case DEPOSIT\_SAVINGS

}

} else if (withdraw) {

switch (display.getState()) {

case Display.WITHDRAW\_CHECKING: {

customer.getChecking().withdraw(

Double.parseDouble(text.trim()));

}

break; //end case WITHDRAW\_CHECKING

case Display.WITHDRAW\_SAVINGS: {

customer.getSavings().withdraw(

Double.parseDouble(text.trim()));

}

break; //end case WITHDRAW\_SAVINGS

}

}

customer.update(connection);

display.setState(Display.ACCOUNTS);

} catch (Exception e) {

e.printStackTrace();

text = "Error";

}

} else if (btntext.equals("Cancel")) {

display.setState(Display.ACCOUNTS);

} else if (btntext.equals("CE")) {

text = " ";

} else if (btntext.length() > 10) {

text = "Error";

} else if (btntext.length() == 1) {

text += btntext;

}

display.getTransactInputLabel().setText(text);

}

break; //end transaction cases

}

}

public void finalize() {

try {

if (connection != null) {

connection.close();

}

} catch (Exception e) {

e.printStackTrace();

}

}

}

Bank account

package atmsimulation;

public class BankAccount {

private int account;

private double balance;

public BankAccount() {

this(0, 0);

}

public BankAccount(int number, double amount) {

setAccount(number);

setBalance(amount);

}

public void deposit(double amount) {

balance += amount;

}

public void withdraw(double amount) throws Exception {

boolean result = (balance > amount);

if (result)

balance -= amount;

else

throw new Exception();

}

public int getAccount() {

return account;

}

public double getBalance() {

return balance;

}

public void setAccount(int number) {

account = number;

}

public void setBalance(double amount) {

balance = amount;

}

}

Custom button

package atmsimulation;

import java.awt.\*;

import javax.swing.\*;

public class CustomButton extends JButton {

CustomGraphicInterface cgi;

public CustomButton() {

this("Custom Button");

}

public CustomButton(String text) {

try {

init(text);

}

catch (Exception e) {

e.printStackTrace();

}

}

private void init(String text) throws Exception {

cgi = new CustomGraphicInterface(this);

cgi.setHighlightable(true);

cgi.setSelectable(true);

setText(text);

setFont(new Font("Impact", 0, 15));

setForeground(Color.BLUE);

}

public void paintComponent(Graphics g) {

cgi.paintCGI(g);

super.paintComponent(g);

}

}

Custom graphic interface

package atmsimulation;

import java.awt.\*;

import java.awt.geom.\*;

import java.awt.event.\*;

import javax.swing.\*;

public class CustomGraphicInterface implements MouseListener {

Paint arrPaints[] = {new GradientPaint(0, 0, Color.lightGray,

0, 0, Color.blue),

new GradientPaint(0, 0, Color.lightGray.brighter(),

0, 0, Color.blue.brighter()),

new GradientPaint(0, 0, Color.cyan.darker(),

0, 0, Color.blue.darker()),

new GradientPaint(0, 0, Color.gray,

0, 0, Color.darkGray)};

Shape arrShapes[] = {new RoundRectangle2D.Float(0, 0, 0, 0, 5, 5)};

// Image arrImages[] = null;

Dimension dimArea = null;

JComponent component = null;

final static int DEFAULT = 0;

final static int HIGHLIGHT = 1;

final static int SELECTED = 2;

final static int DISABLED = 3;

int paintState = DEFAULT;

boolean highlightable = false;

boolean selectable = false;

public CustomGraphicInterface() {

this(null);

}

public CustomGraphicInterface(JComponent comp) {

if (comp != null) {

component = comp;

component.setOpaque(false);

if (component instanceof AbstractButton) {

AbstractButton button = (AbstractButton) component;

button.setFocusPainted(false);

button.setBorder(null);

button.setBorderPainted(false);

}

dimArea = comp.getSize();

}

}

public CustomGraphicInterface(JComponent component,

Paint paints[], Shape shapes[], Image images[]) {

this(component);

if (paints != null)

arrPaints = paints;

if (shapes != null)

arrShapes = shapes;

// if (images != null)

// arrImages = images;

}

public void updatePaints(Dimension area) {

for (int i = 0; i < arrPaints.length; i++)

if (arrPaints[i] instanceof GradientPaint) {

GradientPaint paint = (GradientPaint) arrPaints[i];

Color c1 = paint.getColor1();

Color c2 = paint.getColor2();

arrPaints[i] = new GradientPaint(0, 0, c1, area.width, area.height, c2);

}

}

public void updateShapes(Dimension area) {

for (int i = 0; i < arrShapes.length; i++) {

arrShapes[i] = new RoundRectangle2D.Float(0f, 0f,

(float) area.getWidth(),

(float) area.getHeight(),

5f, 5f);

}

}

public void updateCGI(Dimension area) {

if (dimArea.equals(area))

return;

else {

updatePaints(area);

updateShapes(area);

}

}

public void paintPSI(Graphics g, int pidx, int sidx, int iidx) {

Graphics2D g2 = (Graphics2D) g;

g2.setRenderingHint(RenderingHints.KEY\_TEXT\_ANTIALIASING,

RenderingHints.VALUE\_TEXT\_ANTIALIAS\_ON);

if (pidx > -1)

g2.setPaint(arrPaints[pidx]);

if (sidx > -1)

g2.fill(arrShapes[sidx]);

// if (iidx > -1)

// g2.drawImage(arrImages[iidx], 0, 0, null);

}

public void paintCGI(Graphics g) {

updateCGI(component.getSize());

paintPSI(g, paintState, 0, -1);

}

public void setHighlightable(boolean b) {

highlightable = b;

if (isHighlightable()) {

if (!isSelectable())

component.addMouseListener(this);

}

else if (!isSelectable()) {

component.removeMouseListener(this);

}

}

public boolean isHighlightable() {

return highlightable;

}

public void setSelectable(boolean b) {

selectable = b;

if (isSelectable()) {

if (!isHighlightable())

component.addMouseListener(this);

}

else if (!isHighlightable()) {

component.removeMouseListener(this);

}

}

public boolean isSelectable() {

return selectable;

}

public void mouseExited(MouseEvent e) {

if (isHighlightable()) {

paintState = DEFAULT;

component.repaint();

}

}

public void mouseEntered(MouseEvent e) {

if (isHighlightable()) {

paintState = HIGHLIGHT;

component.repaint();

}

}

public void mousePressed(MouseEvent e) {

if (isSelectable()) {

paintState = SELECTED;

component.repaint();

}

}

public void mouseReleased(MouseEvent e) {

if (isSelectable()) {

if (isHighlightable())

paintState = HIGHLIGHT;

else

paintState = DEFAULT;

component.repaint();

}

}

public void mouseClicked(MouseEvent e) {

}

}

Customer label

package atmsimulation;

import java.awt.\*;

import javax.swing.\*;

import javax.swing.border.Border;

public class CustomLabel extends JLabel {

CustomGraphicInterface cgi;

public CustomLabel() {

this("Custom Label", CENTER);

}

public CustomLabel(String text) {

this(text, CENTER);

}

public CustomLabel(String text, int alignment) {

try {

init(text, alignment);

} catch (Exception e) {

e.printStackTrace();

}

}

private void init(String text, int alignment) throws Exception {

cgi = new CustomGraphicInterface(this);

cgi.arrPaints[CustomGraphicInterface.DEFAULT]

= new GradientPaint(0, 0, Color.cyan.darker(),

0, 0, Color.blue.darker());

setText(text);

setHorizontalAlignment(alignment);

setBorder(BorderFactory.createEmptyBorder(2, 10, 2, 10));

setFont(new Font("Impact", 0, 15));

setForeground(Color.ORANGE);

}

public void paintComponent(Graphics g) {

cgi.paintCGI(g);

super.paintComponent(g);

}

}

Customer panel :

package atmsimulation;

import java.awt.\*;

import javax.swing.\*;

public class CustomPanel extends JPanel {

CustomGraphicInterface cgi;

public CustomPanel() {

try {

init();

}

catch (Exception e) {

e.printStackTrace();

}

}

private void init() throws Exception {

cgi = new CustomGraphicInterface(this);

}

protected void paintCustomItems(Graphics g) {}

public void paintComponent(Graphics g) {

cgi.paintCGI(g);

paintCustomItems(g);

super.paintComponent(g);

}

}

Customer

package atmsimulation;

import java.sql.\*;

public class Customer {

private int numberid;

private int pinid;

private BankAccount checking;

private BankAccount savings;

private boolean exist;

public Customer() {

this(0, 0, new BankAccount(0, 0), new BankAccount(0, 0));

}

public Customer(int number, int pin, BankAccount checkingAccount,

BankAccount savingsAccount) {

numberid = number;

pinid = pin;

checking = checkingAccount;

savings = savingsAccount;

exist = false;

}

public Customer(Connection connection, int pin) {

Customer customer = null;

try {

customer = isExist(connection, pin);

}

catch (Exception e) {

e.printStackTrace();

}

if (exist = (customer != null)) {

numberid = customer.numberid;

pinid = customer.pinid;

checking = customer.checking;

savings = customer.savings;

}

}

public int getNumber() {

return numberid;

}

public int getPIN() {

return pinid;

}

public BankAccount getChecking() {

return checking;

}

public BankAccount getSavings() {

return savings;

}

public void setNumber(int number) {

numberid = number;

}

public void setPIN(int pin) {

pinid = pin;

}

public void setChecking(BankAccount account) {

checking = account;

}

public void setSavings(BankAccount account) {

savings = account;

}

public boolean isExist() {

return exist;

}

public Customer isExist(Connection connection, int pin) throws Exception {

Customer record = null;

Statement sql = connection.createStatement();

ResultSet result = sql.executeQuery("SELECT \* FROM CUSTOMER\_TABLE " +

"WHERE PIN LIKE " + pin);

if (result != null)

if(result.next())

record = new Customer(result.getInt("CUSTOMER\_NUMBER"),

result.getInt("PIN"),

new BankAccount(

result.getInt("CHECKING\_ACCOUNT\_NUMBER"),

result.getDouble("CHECKING\_ACCOUNT\_BALANCE")),

new BankAccount(

result.getInt("SAVINGS\_ACCOUNT\_NUMBER"),

result.getDouble("SAVINGS\_ACCOUNT\_BALANCE")));

return record;

}

public void update(Connection connection) throws Exception {

Statement sql = connection.createStatement();

sql.executeUpdate("UPDATE CUSTOMER\_TABLE " +

"SET CHECKING\_ACCOUNT\_BALANCE = " +

checking.getBalance() +

" WHERE PIN LIKE " + pinid);

sql.executeUpdate("UPDATE CUSTOMER\_TABLE " +

"SET SAVINGS\_ACCOUNT\_BALANCE = " +

savings.getBalance() +

" WHERE PIN LIKE " + pinid);

}

}

Display

package atmsimulation;

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

public class Display extends CustomPanel {

public final static int PIN = 0;

public final static int ACCOUNTS = 1;

public final static int DEPOSIT = 2;

public final static int DEPOSIT\_CHECKING = 3;

public final static int DEPOSIT\_SAVINGS = 4;

public final static int WITHDRAW = 5;

public final static int WITHDRAW\_CHECKING = 6;

public final static int WITHDRAW\_SAVINGS = 7;

public final static int END\_TRANSACTION = 8;

private int state = PIN;

private final String controls[][] = {{"Enter", "...", "..."},

{"Deposit", "Withdraw", "Cancel"},

{"Current", "Savings", "Cancel"},

{"Deposit", "Cancel", "..."},

{"Deposit", "Cancel", "..."},

{"Current", "Savings", "Cancel"},

{"Withdraw", "Cancel", "..."},

{"Withdraw", "Cancel", "..."},

{"Yes", "No", "..."}};

LayoutManager layout;

JPanel pnlTitleBar;

JPanel pnlControl;

JPanel pnlPIN;

JPanel pnlAccounts;

JPanel pnlTransact;

JPanel pnlEnd;

JPanel panels[] = {pnlPIN, pnlAccounts, pnlAccounts, pnlTransact, pnlTransact,

pnlAccounts, pnlTransact, pnlTransact, pnlEnd};

Customer customer;

public Display() {

try {

init();

}

catch (Exception e) {

e.printStackTrace();

}

}

private void init() throws Exception {

layout = new GridBagLayout();

cgi.arrPaints[CustomGraphicInterface.DEFAULT] =

new GradientPaint(0, 0, Color.black,

0, 0, Color.darkGray.darker());

setLayout(layout);

initTitleBarPanel();

initControlPanel();

initPINPanel();

initAccountsPanel();

initTransactPanel();

setState(PIN);

add(pnlTitleBar,

new GridBagConstraints(0, 0, 1, 1, 3.3, 1.0,

GridBagConstraints.CENTER,

GridBagConstraints.BOTH,

new Insets(0, 0, 0, 0), 0, 0));

add(pnlControl,

new GridBagConstraints(1, 0, 1, 2, 0.7, 1.0,

GridBagConstraints.CENTER,

GridBagConstraints.BOTH,

new Insets(0, 0, 0, 0), 0, 0));

add(pnlPIN,

new GridBagConstraints(0, 1, 1, 1, 3.3, 1.0,

GridBagConstraints.CENTER,

GridBagConstraints.BOTH,

new Insets(0, 0, 0, 0), 0, 0));

add(pnlAccounts,

new GridBagConstraints(0, 1, 1, 1, 3.3, 1.0,

GridBagConstraints.CENTER,

GridBagConstraints.BOTH,

new Insets(0, 0, 0, 0), 0, 0));

add(pnlTransact,

new GridBagConstraints(0, 1, 1, 1, 3.3, 1.0,

GridBagConstraints.CENTER,

GridBagConstraints.BOTH,

new Insets(0, 0, 0, 0), 0, 0));

}

private void initTitleBarPanel() {

pnlTitleBar = new JPanel();

pnlTitleBar.setMaximumSize(new Dimension(100, 50));

pnlTitleBar.setMinimumSize(new Dimension(100, 50));

pnlTitleBar.setPreferredSize(new Dimension(100, 50));

pnlTitleBar.setOpaque(false);

}

private void initControlPanel() {

pnlControl = new JPanel();

pnlControl.setMaximumSize(new Dimension(80, 150));

pnlControl.setMinimumSize(new Dimension(80, 150));

pnlControl.setPreferredSize(new Dimension(80, 150));

pnlControl.setLayout(new GridBagLayout());

pnlControl.setOpaque(false);

for (int i = 0; i < controls[PIN].length; i++) {

CustomButton button = new CustomButton(controls[PIN][i]);

button.setFont(new Font("Impact", 0, 15));

pnlControl.add(button,

new GridBagConstraints(0, i, 1, 1, 1.0, 1.0,

GridBagConstraints.CENTER,

GridBagConstraints.BOTH,

new Insets(5, 5, 5, 5), 0, 0));

}

}

private void initPINPanel() {

JLabel label = new CustomLabel("Enter PIN", JLabel.CENTER);

JLabel input = new CustomLabel(" ", JLabel.CENTER);

pnlPIN = new JPanel();

pnlPIN.setMaximumSize(new Dimension(100, 100));

pnlPIN.setMinimumSize(new Dimension(100, 100));

pnlPIN.setPreferredSize(new Dimension(100, 100));

pnlPIN.setLayout(new GridBagLayout());

pnlPIN.setOpaque(false);

pnlPIN.setVisible(false);

Insets insets = new Insets(5, 5, 5, 5);

pnlPIN.add(label,

new GridBagConstraints(0, 0, 1, 1, 1.0, 1.0,

GridBagConstraints.NORTH,

GridBagConstraints.HORIZONTAL,

insets, 0, 0));

pnlPIN.add(input,

new GridBagConstraints(0, 1, 1, 1, 1.0, 1.0,

GridBagConstraints.NORTH,

GridBagConstraints.HORIZONTAL,

insets, 0, 0));

panels[PIN] = pnlPIN;

}

private void initAccountsPanel() {

JLabel label1 = new CustomLabel("Account Balances", JLabel.CENTER);

JLabel label2 = new CustomLabel("Current", JLabel.LEADING);

JLabel label3 = new CustomLabel("Savings", JLabel.LEADING);

JLabel label4 = new CustomLabel("", JLabel.RIGHT);

JLabel label5 = new CustomLabel("", JLabel.RIGHT);

pnlAccounts = new JPanel();

pnlAccounts.setMaximumSize(new Dimension(100, 100));

pnlAccounts.setMinimumSize(new Dimension(100, 100));

pnlAccounts.setPreferredSize(new Dimension(100, 100));

pnlAccounts.setLayout(new GridBagLayout());

pnlAccounts.setOpaque(false);

pnlAccounts.setVisible(false);

Insets insets = new Insets(5, 5, 5, 5);

pnlAccounts.add(label1,

new GridBagConstraints(0, 0, 2, 1, 1.0, 1.0,

GridBagConstraints.NORTH,

GridBagConstraints.HORIZONTAL,

insets, 0, 0));

pnlAccounts.add(label2,

new GridBagConstraints(0, 1, 1, 1, 1.0, 1.0,

GridBagConstraints.NORTH,

GridBagConstraints.HORIZONTAL,

insets, 0, 0));

pnlAccounts.add(label3,

new GridBagConstraints(0, 2, 1, 1, 1.0, 1.0,

GridBagConstraints.NORTH,

GridBagConstraints.HORIZONTAL,

insets, 0, 0));

pnlAccounts.add(label4,

new GridBagConstraints(1, 1, 1, 1, 1.0, 1.0,

GridBagConstraints.NORTH,

GridBagConstraints.HORIZONTAL,

insets, 0, 0));

pnlAccounts.add(label5,

new GridBagConstraints(1, 2, 1, 1, 1.0, 1.0,

GridBagConstraints.NORTH,

GridBagConstraints.HORIZONTAL,

insets, 0, 0));

panels[ACCOUNTS] = pnlAccounts;

panels[DEPOSIT] = pnlAccounts;

panels[WITHDRAW] = pnlAccounts;

}

private void initTransactPanel() {

JLabel label1 = new CustomLabel("Deposit To AccountType", JLabel.CENTER);

JLabel label2 = new CustomLabel("Balance", JLabel.LEADING);

JLabel label3 = new CustomLabel("Amount", JLabel.LEADING);

JLabel label4 = new CustomLabel("AccountBalance", JLabel.RIGHT);

JLabel label5 = new CustomLabel(" ", JLabel.RIGHT);

pnlTransact = new JPanel();

pnlTransact.setMaximumSize(new Dimension(100, 100));

pnlTransact.setMinimumSize(new Dimension(100, 100));

pnlTransact.setPreferredSize(new Dimension(100, 100));

pnlTransact.setLayout(new GridBagLayout());

pnlTransact.setOpaque(false);

pnlTransact.setVisible(false);

Insets insets = new Insets(5, 5, 5, 5);

pnlTransact.add(label1,

new GridBagConstraints(0, 0, 2, 1, 1.0, 1.0,

GridBagConstraints.NORTH,

GridBagConstraints.HORIZONTAL,

insets, 0, 0));

pnlTransact.add(label2,

new GridBagConstraints(0, 1, 1, 1, 1.0, 1.0,

GridBagConstraints.NORTH,

GridBagConstraints.HORIZONTAL,

insets, 0, 0));

pnlTransact.add(label3,

new GridBagConstraints(0, 2, 1, 1, 1.0, 1.0,

GridBagConstraints.NORTH,

GridBagConstraints.HORIZONTAL,

insets, 0, 0));

pnlTransact.add(label4,

new GridBagConstraints(1, 1, 1, 1, 1.0, 1.0,

GridBagConstraints.NORTH,

GridBagConstraints.HORIZONTAL,

insets, 0, 0));

pnlTransact.add(label5,

new GridBagConstraints(1, 2, 1, 1, 1.0, 1.0,

GridBagConstraints.NORTHEAST,

GridBagConstraints.HORIZONTAL,

insets, 0, 0));

panels[DEPOSIT\_CHECKING] = pnlTransact;

panels[DEPOSIT\_SAVINGS] = pnlTransact;

panels[WITHDRAW\_CHECKING] = pnlTransact;

panels[WITHDRAW\_SAVINGS] = pnlTransact;

}

public JLabel getPINInputLabel() {

return (JLabel) pnlPIN.getComponent(1);

}

public JLabel getTransactInputLabel() {

return (JLabel) pnlTransact.getComponent(4);

}

public void addControlActionListener(ActionListener listener) {

Component btns[] = pnlControl.getComponents();

for (int i = 0; i < btns.length; i++)

((JButton) btns[i]).addActionListener(listener);

}

public int getState() {

return state;

}

private void updateControls(int value) {

Component btns[] = pnlControl.getComponents();

for (int i = 0; i < btns.length; i++)

((JButton) btns[i]).setText(controls[value][i]);

}

private void updatePanels(int value) {

panels[state].setVisible(false);

panels[value].setVisible(true);

switch(value) {

case PIN: {

getPINInputLabel().setText(" ");

} break;

case ACCOUNTS: {

((JLabel) pnlAccounts.getComponent(3)).setText("" +

customer.getChecking().getBalance());

((JLabel) pnlAccounts.getComponent(4)).setText("" +

customer.getSavings().getBalance());

} break;

case DEPOSIT\_CHECKING: case DEPOSIT\_SAVINGS:

case WITHDRAW\_CHECKING: case WITHDRAW\_SAVINGS: {

String transaction = "";

String balance = "";

switch(value) {

case DEPOSIT\_CHECKING:

transaction = "Deposit Current";

balance = "" + customer.getChecking().getBalance();

break;

case DEPOSIT\_SAVINGS:

transaction = "Deposit Savings";

balance = "" + customer.getSavings().getBalance();

break;

case WITHDRAW\_CHECKING:

transaction = "Withdraw Current";

balance = "" + customer.getChecking().getBalance();

break;

case WITHDRAW\_SAVINGS:

transaction = "Withdraw Savings";

balance = "" + customer.getSavings().getBalance();

break;

}

((JLabel) pnlTransact.getComponent(0)).setText(transaction);

((JLabel) pnlTransact.getComponent(3)).setText(balance);

((JLabel) pnlTransact.getComponent(4)).setText(" ");

} break;

}

}

public void setCustomer(Customer record) throws Exception {

if (record.isExist())

customer = record;

else

throw new Exception();

}

public void setState(int value) {

updateControls(value);

updatePanels(value);

state = value;

}

protected void paintCustomItems(Graphics g) {

Graphics2D g2 = (Graphics2D) g;

g2.setPaint(Color.blue);

g2.setFont(new Font("Impact", Font.BOLD, 60));

g2.translate(12, 53);

g2.drawString("ATM", 80, 30);

g2.translate(-12, -53);

g2.setPaint(Color.yellow);

g2.setFont(new Font("Impact", Font.BOLD, 35));

g2.translate(37, 57);

g2.drawString("", 0, 0);

g2.translate(-37, -57);

g2.setPaint(Color.orange);

g2.setFont(new Font("Impact", Font.BOLD, 20));

g2.translate(85, 25);

g2.drawString("", 0, 0);

g2.translate(-85, -25);

}

}

Keypad package atmsimulation;

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

public class KeyPad extends CustomPanel {

LayoutManager layout;

JPanel keygrid;

public KeyPad() {

try {

init();

}

catch (Exception e) {

e.printStackTrace();

}

}

private void init() throws Exception {

setMaximumSize(new Dimension(100, 150));

setMinimumSize(new Dimension(100, 150));

setPreferredSize(new Dimension(100, 150));

layout = new GridBagLayout();

setLayout(layout);

cgi.arrPaints[0] = new GradientPaint(0, 0, Color.black,

0, 0, Color.darkGray.darker());

initKeyGrid();

add(keygrid,

new GridBagConstraints(0, 0, 1, 1, 1.0, 1.0,

GridBagConstraints.CENTER,

GridBagConstraints.BOTH,

new Insets(5, 5, 5, 5), 0, 0));

}

private void initKeyGrid() {

keygrid = new JPanel();

keygrid.setOpaque(false);

keygrid.setLayout(new GridLayout(4, 3, 5, 5));

String text[][] = {{"7", "8", "9"},

{"4", "5", "6"},

{"1", "2", "3"},

{"0", ".", "CE"}};

for (int y = 0; y < text.length; y++)

for (int x = 0; x < text[y].length; x++)

keygrid.add(new CustomButton(text[y][x]));

}

public void addKeyActionListener(ActionListener listener) {

Component buttons[] = keygrid.getComponents();

for (int i = 0; i < buttons.length; i++)

((JButton) buttons[i]).addActionListener(listener);

}

}