**Sprawozdanie 6 – Programowanie Obiektowe**

Wykonał: Łukasz Borsuk Automatyka i Robotyka Gr. 1

1. Klasa LoginPanel

package com.baselukasz.ui;  
  
import com.baselukasz.core.User;  
import com.baselukasz.dao.DBConnection;  
import com.jgoodies.forms.factories.FormFactory;  
import com.jgoodies.forms.layout.ColumnSpec;  
import com.jgoodies.forms.layout.FormLayout;  
import com.jgoodies.forms.layout.RowSpec;  
import org.jasypt.digest.config.SimpleDigesterConfig;  
import org.jasypt.util.password.ConfigurablePasswordEncryptor;  
  
import javax.swing.\*;  
import javax.swing.border.EmptyBorder;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.awt.event.WindowAdapter;  
import java.awt.event.WindowEvent;  
import java.sql.ResultSet;  
import java.sql.SQLException;  
  
public class LoginPanel extends JFrame implements ActionListener {  
  
 private static final long *serialVersionUID* = 1L;  
  
 private DBConnection con;  
 private User user;  
  
 private JPanel userPanel;  
  
 private JTextField username;  
 private JPasswordField password;  
  
 private JButton confirm;  
 private JButton cancel;  
  
  
 public LoginPanel(DBConnection conInit){  
 con = conInit;  
  
 // Reakcja na zamkniecie okna  
 this.addWindowListener(new WindowAdapter() {  
 @Override  
 public void windowClosing(WindowEvent e) {  
 con.disconnect(true, null);  
 System.*exit*(0);  
 }  
 });  
 setTitle("Panel logowania");  
 setBounds(100,100,450,168);  
  
 // Utworzenie layoutu  
 getContentPane().setLayout(new BorderLayout());  
 userPanel = new JPanel();  
 userPanel.setBorder(new EmptyBorder(5,5,5,5));  
 getContentPane().add(userPanel, BorderLayout.*CENTER*);  
 userPanel.setLayout(new FormLayout(new ColumnSpec[]{  
 FormFactory.*RELATED\_GAP\_COLSPEC*,  
 FormFactory.*DEFAULT\_COLSPEC*,  
 FormFactory.*RELATED\_GAP\_COLSPEC*,  
 ColumnSpec.*decode*("default:grow")  
 },  
 new RowSpec[]{  
 FormFactory.*RELATED\_GAP\_ROWSPEC*,  
 FormFactory.*DEFAULT\_ROWSPEC*,  
 FormFactory.*RELATED\_GAP\_ROWSPEC*,  
 FormFactory.*DEFAULT\_ROWSPEC*,  
 }  
 ));  
  
 // Dane uzytkownika  
 JLabel lblUser = new JLabel("Nazwa");  
 // dodanie do pola JPanel etykietki z prawej  
 userPanel.add(lblUser, "2, 2, right, default");  
 username = new JTextField();  
 // dodanie do pola JPanel pola tekstowego wypełniającego komórkę  
 userPanel.add(username, "4, 2, fill, default");  
  
 // Dane haslo  
 JLabel lblPassword = new JLabel("Haslo");  
 // dodanie do pola JPanel etykietki z prawej  
 userPanel.add(lblPassword, "2, 4, right, default");  
 password = new JPasswordField();  
 // dodanie do pola JPanel pola tekstowego wypełniającego komórkę  
 userPanel.add(password, "4, 4, fill, default");  
  
 // Panel z przyciskami  
 JPanel buttonPane = new JPanel();  
 buttonPane.setLayout(new FlowLayout(FlowLayout.*RIGHT*));  
 getContentPane().add(buttonPane, BorderLayout.*SOUTH*);  
  
 // Przycisk potwierdzajacy wprowadzone dane  
 confirm = new JButton("Potwierdz");  
 confirm.addActionListener(this);  
 confirm.setActionCommand("Potwierdz");  
 buttonPane.add(confirm);  
  
 // Przycisk zamykajacy okno  
 cancel = new JButton("Anuluj");  
 cancel.addActionListener(this);  
 cancel.setActionCommand("Anuluj");  
 buttonPane.add(cancel);  
 }  
  
 public void setCon(DBConnection conInit){  
 con = conInit;  
 }  
  
 public boolean checkLoginData() {  
  
 SimpleDigesterConfig md5Config = new SimpleDigesterConfig();  
 md5Config.setAlgorithm("MD5");  
 md5Config.setIterations(1);  
 md5Config.setSaltSizeBytes(0);  
  
 ConfigurablePasswordEncryptor md5Encryptor = new ConfigurablePasswordEncryptor();  
 md5Encryptor.setConfig(md5Config);  
 md5Encryptor.setStringOutputType("hexadecimal");  
  
 String encryptedPassword = md5Encryptor.encryptPassword( new String(password.getPassword()));  
  
 String Query = "SELECT \* FROM Users WHERE Users.name = '"+username.getText()+"' AND Users.password = '"+encryptedPassword+"'";  
 // String Query = "SELECT \* FROM Users WHERE Users.name = '"+username.getText()+"'";  
  
 ResultSet rs = con.load(Query);  
 if(rs != null){  
 try{  
 if(rs.next()) {  
 int id = rs.getInt("id");  
 String nazwa = rs.getString("name");  
 String haslo = rs.getString("password");  
 user = new User(id, nazwa, haslo);  
 con.destroyRS(rs);  
 return true;  
 }  
 else  
 {  
 con.destroyRS(rs);  
 return false;  
 }  
 } catch (SQLException e) {  
 System.*out*.println("SprawdzDane: Problem z przetworzeniem danych");  
 System.*out*.println("SQLException: " + e.getMessage());  
 System.*out*.println("SQLState: " + e.getSQLState());  
 System.*out*.println("Vendor Error: " + e.getErrorCode());  
 return false;  
 }  
 }  
 else{  
 return false;  
 }  
 }  
  
 private void login() {  
 // Ukryj okno  
 setVisible(false);  
 dispose();  
  
 // Otworz okienko glowne aplikacji  
 TaskList ts = new TaskList(con, user);  
 ts.setVisible(true);  
 }  
  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 if(e.getActionCommand() == "Potwierdz"){  
 if(checkLoginData()){  
 System.*out*.println("sprawdzDane() == true");  
 login();  
 }  
 else{  
 System.*out*.println("sprawdzDane() == false");  
 JOptionPane.*showMessageDialog*(LoginPanel.this, "Bledne dane uzytkownika", "Error", JOptionPane.*ERROR\_MESSAGE*);  
 return;  
 }  
 } else if(e.getActionCommand() == "Anuluj"){  
 con.disconnect(true, null);  
 System.*exit*(0);  
 }  
 }  
}

Klasa TaskTableModel

package com.baselukasz.ui;  
  
import com.baselukasz.core.Task;  
  
import javax.swing.table.AbstractTableModel;  
import java.util.List;  
  
public class TaskTableModel extends AbstractTableModel {  
  
 private static final long *serialVersionUID* = 1L;  
 public static final int *OBJECT\_COL* = -1;  
 private static final int *TITLE\_COL* = 0;  
 private static final int *STATUS\_COL* = 1;  
  
 private String[] columnNames = { "Tytul", "Status zadania"};  
  
 private List<Task> tasks;  
  
 public TaskTableModel(List<Task> tasks) {  
 this.tasks = tasks;  
 }  
  
 @Override  
 public int getRowCount() {  
 return tasks.size();  
 }  
  
 @Override  
 public int getColumnCount() {  
 return columnNames.length;  
 }  
  
 @Override  
 public String getColumnName(int kol){  
 return columnNames[kol];  
 }  
  
 @Override  
 public Object getValueAt(int wie, int kol) {  
 Task zad = tasks.get(wie);  
  
 switch (kol){  
 case *TITLE\_COL*:  
 return zad.getTitle();  
 case *STATUS\_COL*:  
 if(zad.isStatus()){  
 return "tak";  
 }  
 else {  
 return "nie";  
 }  
 case *OBJECT\_COL*:  
 return zad;  
 default:  
 return zad.getTitle();  
 }  
 }  
  
 @Override  
 public Class<? extends Object> getColumnClass(int c){  
 return getValueAt(0, c).getClass();  
 }  
}

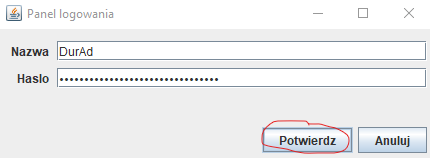
Klasa TaskList

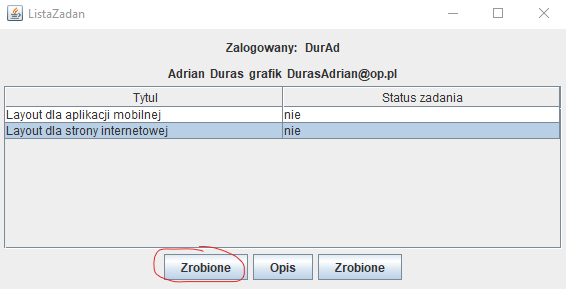
package com.baselukasz.ui;  
  
import com.baselukasz.core.Employee;  
import com.baselukasz.core.Task;  
import com.baselukasz.core.User;  
import com.baselukasz.dao.DBConnection;  
  
import javax.swing.\*;  
import javax.swing.border.EmptyBorder;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.awt.event.WindowAdapter;  
import java.awt.event.WindowEvent;  
import java.sql.ResultSet;  
import java.sql.SQLException;  
import java.util.ArrayList;  
  
public class TaskList extends JFrame implements ActionListener{  
  
 private static final long *serialVersionUID* = 1L;  
  
 private DBConnection con;  
 private User user;  
 private Employee employee;  
  
 private final JPanel taskListPanel;  
 // private final JScrollPane panelZadania;  
  
 private JTable tasksTable;  
 private JLabel lblLogged;  
 private JLabel loggedUser;  
 private JLabel lblFirstName;  
 private JLabel lblLastName;  
 private JLabel lblEmail;  
 private JLabel lblPosition;  
  
 private JButton btnDone;  
 private JButton btnDescription;  
 private JButton btnLogout;  
  
 public static void main(String[] args) {  
 DBConnection komInit = DBConnection.*getInstance*();  
 *loginWindow*( komInit);  
 }  
  
 public TaskList(DBConnection c, User u){  
 con = c;  
 user = u;  
  
 this.addWindowListener(new WindowAdapter() {  
 @Override  
 public void windowClosing(WindowEvent e) {  
 con.disconnect(true, null);  
 System.*exit*(0);  
 }  
 });  
  
 setTitle("ListaZadan");  
  
 setBounds(100, 100, 584, 300);  
 getContentPane().setLayout(new BorderLayout());  
  
 taskListPanel = new JPanel();  
 taskListPanel.setBorder(new EmptyBorder(5,5,5,5));  
 taskListPanel.setLayout(new BorderLayout());  
 setContentPane(taskListPanel);  
  
 JPanel panelGora = new JPanel();  
 taskListPanel.add(panelGora, BorderLayout.*NORTH*);  
 panelGora.setLayout(new BorderLayout());  
  
 // Panel z nazwa uzytkownika  
 JPanel panelUzytkownik = new JPanel();  
 panelGora.add(panelUzytkownik, BorderLayout.*NORTH*);  
  
 lblLogged = new JLabel("Zalogowany: ");  
 panelUzytkownik.add(lblLogged);  
  
 loggedUser = new JLabel(user.getUsername());  
 panelUzytkownik.add(loggedUser);  
  
 // Panel z danymi pracownika  
 JPanel panelPracownika = new JPanel();  
 panelGora.add(panelPracownika);  
  
 lblFirstName = new JLabel();  
 panelPracownika.add(lblFirstName);  
  
 lblLastName = new JLabel();  
 panelPracownika.add(lblLastName);  
  
 lblEmail = new JLabel();  
 panelPracownika.add(lblEmail);  
  
 lblPosition = new JLabel();  
 panelPracownika.add(lblPosition);  
  
 // Panel z zadaniami  
 JScrollPane panelZadania = new JScrollPane();  
 taskListPanel.add(panelZadania, BorderLayout.*CENTER*);  
  
 tasksTable = new JTable();  
 panelZadania.setViewportView(tasksTable);  
  
 // Dane do widoku  
 loadEmployeeData();  
 updateView();  
  
 // Panel opcji  
 JPanel panelMenu = new JPanel();  
 taskListPanel.add(panelMenu, BorderLayout.*SOUTH*);  
  
 btnDone = new JButton("Zrobione");  
 btnDone.addActionListener((ActionListener) this);  
 panelMenu.add(btnDone);  
  
 btnDescription = new JButton("Opis");  
 btnDescription.addActionListener((ActionListener) this);  
 panelMenu.add(btnDescription);  
  
 btnLogout = new JButton("Zrobione");  
 btnLogout.addActionListener((ActionListener) this);  
 btnLogout.setActionCommand("Wyloguj");  
 panelMenu.add(btnLogout);  
 }  
  
 private TaskList() {  
 this.taskListPanel = new JPanel();  
 }  
  
 static void loginWindow(DBConnection conInit){  
 LoginPanel log = new LoginPanel(conInit);  
  
 log.setDefaultCloseOperation(JFrame.*DISPOSE\_ON\_CLOSE*);  
 log.setVisible(true);  
 }  
  
 public void loadEmployeeData(){  
  
 String QueryPrac = "SELECT \* FROM `Employees` WHERE user\_id= "+user.getId();  
 ResultSet rsPrac = con.load(QueryPrac);  
 if(rsPrac != null) {  
 try {  
 if(rsPrac.next()){  
 int id = rsPrac.getInt("id");  
 String imie = rsPrac.getString("first\_name");  
 String nazwisko = rsPrac.getString("last\_name");  
 String email = rsPrac.getString("email");  
 String position = rsPrac.getString("position");  
 employee = new Employee(id, imie, nazwisko, email, position);  
 con.destroyRS(rsPrac);  
 }  
 else{  
 con.destroyRS(rsPrac);  
 }  
 } catch (SQLException e) {  
 System.*out*.println("ListaZadanFirmy: Problem z przetworzeniem danych");  
 System.*out*.println("SQLException: " + e.getMessage());  
 System.*out*.println("SQLState: " + e.getSQLState());  
 System.*out*.println("Vendor Error: " + e.getErrorCode());  
 return ;  
 }  
 }  
 // Wczytaj dane z tabeli zadania  
 String QueryZad = "SELECT \* FROM `Tasks` WHERE employee\_id= " +employee.getId();  
 ResultSet rsZad = con.load(QueryZad);  
 if(rsZad != null) {  
 try {  
 ArrayList<Task> listazad = new ArrayList<>();  
 while (rsZad.next()){  
 int id = rsZad.getInt("id");  
 String tytul = rsZad.getString("title");  
 String opis = rsZad.getString("description");  
 boolean status = rsZad.getBoolean("status");  
 listazad.add(new Task(id, tytul, opis, status));  
 }  
 employee.setTasks(listazad);  
 con.destroyRS(rsZad);  
  
 } catch (SQLException e) {  
 System.*out*.println("ListaZadanFirmy: Problem z przetworzeniem danych");  
 System.*out*.println("SQLException: " + e.getMessage());  
 System.*out*.println("SQLState: " + e.getSQLState());  
 System.*out*.println("Vendor Error: " + e.getErrorCode());  
 return ;  
 }  
 }  
 }  
  
 public void updateView() {  
  
 lblFirstName.setText(employee.getFirstName());  
 lblLastName.setText(employee.getLastName());  
 lblEmail.setText(employee.getEmail());  
 lblPosition.setText(employee.getPosition());  
  
 TaskTableModel model = new TaskTableModel(employee.getTasks());  
 tasksTable.setModel(model);  
 }  
  
 @Override  
 public void actionPerformed(ActionEvent e ){  
 if(e.getActionCommand() == "Wyloguj"){  
 *loginWindow*(con);  
 setVisible(false);  
 dispose();  
 } else if(e.getActionCommand() == "Opis"){  
  
 // Wybrany wiersz  
 int wiersz = tasksTable.getSelectedRow();  
  
 // Jezeli nie zostal wybrany  
 if (wiersz < 0){  
 JOptionPane.*showMessageDialog*(TaskList.this, "Wybierz zadanie", "Error",  
 JOptionPane.*ERROR\_MESSAGE*);  
 return;  
 }  
  
 // Wybierz zadanie  
 Task task = (Task) tasksTable.getValueAt(wiersz, TaskTableModel.*OBJECT\_COL*);  
  
 TaskDescription opis = new TaskDescription(task);  
 opis.setVisible(true);  
 }  
 else if(e.getActionCommand() == "Zrobione"){  
  
 // Wybrany wiersz  
 int wiersz = tasksTable.getSelectedRow();  
  
 // Jezeli nie zostal wybrany  
 if (wiersz < 0){  
 JOptionPane.*showMessageDialog*(TaskList.this, "Wybierz zadanie", "Error",  
 JOptionPane.*ERROR\_MESSAGE*);  
 return;  
 }  
  
 // Wybierz zadanie  
 Task zad = (Task) tasksTable.getValueAt(wiersz, TaskTableModel.*OBJECT\_COL*);  
  
 int nowyStatus = (!zad.isStatus()) ? 1 : 0;  
  
 String Query = "UPDATE `Tasks` SET `status` = "+nowyStatus+" WHERE Tasks.id = "+zad.getId()+";";  
  
 if(con.update(Query)) {  
 employee.getTasks().get(wiersz).setStatus( !zad.isStatus());  
 updateView();  
 }  
 else {  
 JOptionPane.*showMessageDialog*(TaskList.this, "Blad przy modyfikacji", "Error",  
 JOptionPane.*ERROR\_MESSAGE*);  
 return;  
 }  
 }  
 }  
}

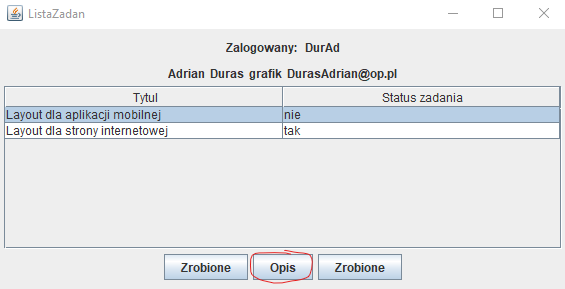
Klasa TaskDescription

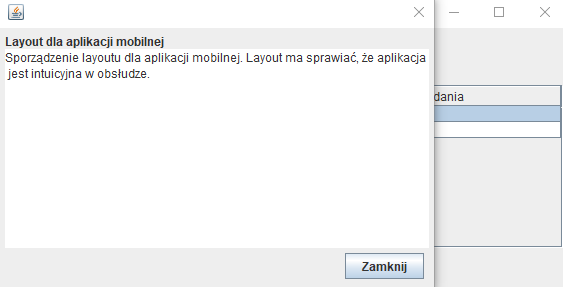
package com.baselukasz.ui;  
  
import com.baselukasz.core.Task;  
  
import javax.swing.\*;  
import javax.swing.border.EmptyBorder;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
  
public class TaskDescription extends JDialog implements ActionListener {  
  
 private final JPanel panelDescription;  
 private JLabel lblTitle;  
 private JTextArea txtrDescription;  
 private JButton btnClose;  
  
 */\*\*  
 \* Create the dialog  
 \*/* public TaskDescription(Task task){  
  
 setBounds(100,100,450,300);  
 getContentPane().setLayout(new BorderLayout());  
 panelDescription = new JPanel();  
 panelDescription.setBorder(new EmptyBorder(5,5,5,5));  
 getContentPane().add(panelDescription, BorderLayout.*CENTER*);  
 panelDescription.setLayout(new BorderLayout(0,0));  
 {  
 lblTitle = new JLabel(task.getTitle());  
 panelDescription.add(lblTitle, BorderLayout.*NORTH*);  
 }  
 {  
 txtrDescription = new JTextArea(task.getDescription());  
 txtrDescription.setLineWrap(true);  
 panelDescription.add(txtrDescription, BorderLayout.*CENTER*);  
 }  
 {  
 JPanel panelPrzyciskow = new JPanel();  
 panelPrzyciskow.setLayout(new FlowLayout(FlowLayout.*RIGHT*));  
 panelDescription.add(panelPrzyciskow, BorderLayout.*SOUTH*);  
 {  
 btnClose = new JButton("Zamknij");  
 btnClose.setActionCommand("Zamknij");  
 btnClose.addActionListener(this);  
 panelPrzyciskow.add(btnClose);  
 }  
 }  
 }  
  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 if (e.getActionCommand() == "Zamknij") {  
 setVisible(false);  
 dispose();  
 }  
 }  
}

1. Wynik działania programu testowego

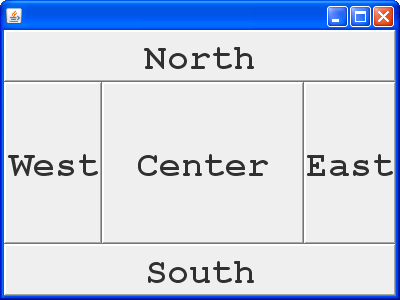




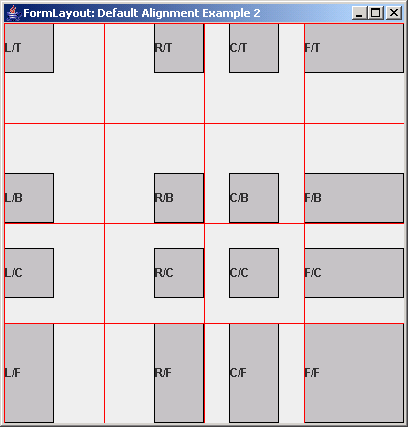


-------------------------------------------------------------------------------------------------------------

1. Odpowiedzi na pytania:
   1. Border Layout



Form Layout



* 1. Content Pane jest podstawową warstwą layoutu w bibliotece Java Swing, służy do przechowywania różnych obiektów.
  2. *Salt* jest to losowa sekwencja bitów, która jest generowana dla każdego nowego *hash.* Wprowadzając tę losowość chronimy naszą baze danych przed pre-kompilowaną liczbą hashy znaną jako *rainbow tables.*

