Міністерство освіти і науки України

Черкаський державний технологічний університет

Кафедра програмного забезпечення автоматизованих систем

Звіт

З лабораторної роботи №9

З предмету «Об’єктно орієнтоване програмування»

Перевірив

Викладач кафедри

Програмного забезпечення автоматизованих систем

Долецький К. Ю.

Виконав

Студент І курсу

Група ПЗ-154

Кравченко Артем Олегович

Виконав

Студент ІІ курсу

Група ПЗ-154

Кравченко А. О.

ЧЕРКАСИ 2017

**Тема:** Абстрактні класи. Обробка виключних ситуацій. Робота з базами даних в Java з використанням JDBC.

**Мета:** Навчитись виконувати основні операції при роботі з базами даних в Java, використовуючи JDBC API. Зрозуміти реляційно-об’єктні та об’єктно-реляційні перетворення при використанні JDBC. Зрозуміти принципи використання абстрактних класів. Засвоїти правила обробки виключних ситуацій. Зміцнити знання стосовно поліморфізму.

**Постановка завдання:** Створити базу даних, в якій є одна таблиця з не менше, ніж 5 полями. В таблиці повинні бути поля типа integer, varchar, date.

Написати Java-програму, яка відображає таблицю бази даних на екрані в табличній формі та дозволяє редагувати дані.

Проаналізувати клас JDBCAdapter. Визначити, для чого в ньому використовується абстрактний клас. Визначити, де і як проявляється поліморфізм при роботі даної програми. Визначити, де і для чого в програмі проводиться обробка виключних ситуацій. Визначити, де і для чого в програмі використовується RTTI

**Теоретичні відомості.**

В бібліотеці javax.swing є клас JTable, який представляє таблицю. Для повноцінної роботи з таблицею для неї потрібно вказати модель даних – клас, об’єкт якого репрезентує ті дані, які будуть в таблиці.

При використанні таблиці потрібно створити об’єкт даного класу і вказати його як параметр при створенні об’єкта JTable.

Для того, щоб проводити обрахунки з даними, введеними в таблицю, потрібно перетворювати їх з одного типу в інший. Для цього іноді потрібно використовувати операцію приведення типу.

**Колекції.** Колекція – це деяка послідовність зв’язаних елементів. Прикладом колекції є масив. Також використовуються інші типи колекцій. Наприклад, в Java клас java.util.Vector реалізує список елементів. При цьому в даній колекції може бути довільна кількість елементів довільних типів. Основні методи, що використовуються для роботи з класом Vector:

void [add](file:///\\KTSERVER\Books\Programming\Java\j2sdk-1_4_1-doc\docs\api\java\util\Vector.html#add(int, java.lang.Object))(int index, [Object](file:///\\KTSERVER\Books\Programming\Java\j2sdk-1_4_1-doc\docs\api\java\lang\Object.html) element)

Object [elementAt](file:///\\KTSERVER\Books\Programming\Java\j2sdk-1_4_1-doc\docs\api\java\util\Vector.html#elementAt(int))(int index)

Enumeration elements()

int [indexOf](file:///\\KTSERVER\Books\Programming\Java\j2sdk-1_4_1-doc\docs\api\java\util\Vector.html#indexOf(java.lang.Object))([Object](file:///\\KTSERVER\Books\Programming\Java\j2sdk-1_4_1-doc\docs\api\java\lang\Object.html) elem)

int size()

**Зв’язок програми на Java з сервером баз даних InterBase/Firebird.** Для роботи програми з базою даних програма повинна насамперед під’єднатися (connect) до БД. Існують різні способи забезпечення зв’язку Java-програми з базами даних Firebird. Наприклад, можна використовувати ODBC або прямий зв’язок JDBC-Firebird. Для його забезпечення можна використати спеціальні бібліотеки роботи з БД, такі, як firebirdsql-full.jar.

**Виконання лабораторної роботи:**

**Лістинг програми:**

TraidingSystem.java

package ark;  
  
public class TradingSystem {  
  
 public static void main(String[] args) {  
 Window window = new Window();  
 window.drawWindow();  
 }  
}

Window.java

package ark;  
  
import jdbc.JDBCUtil;  
import tsobject.Mine;  
  
import javax.swing.\*;  
import javax.swing.table.DefaultTableModel;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.util.ArrayList;  
  
public class Window {  
 public Window() {  
 ArrayList<ArrayList> tableDataMines = jdbcUtil.getMines();  
  
 for (int i = 0; i < tableDataMines.size(); i++){  
 ArrayList row = tableDataMines.get(i);  
 this.minesBox.addItem((String) row.get(3));  
 }  
 }  
  
 private JDBCUtil jdbcUtil = new JDBCUtil();  
 private SellerBot sellerBot = new SellerBot(jdbcUtil);  
 private JComboBox minesBox = new JComboBox();  
  
 public void drawWindow(){  
 JFrame window = new JFrame();  
 window.setSize(800, 400);  
 window.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 window.setResizable(false);  
 window.setLayout(null);  
  
 //CITY PANEL  
 JPanel cityPanel = new JPanel();  
 cityPanel.setLayout(null);  
 setComponentPosition(cityPanel,new Point(350,window.getHeight()-100), new Point(0,100));  
 cityPanel.setBackground(Color.*ORANGE*);  
  
 String [] headerCity = {"Номер", "Назва", "Відстань", "Наявність золота", "Ціна золота"};  
  
 JTable cityTable = new JTable(getCityTableDataModel(),headerCity);  
 JScrollPane cityScrollPane = new JScrollPane(cityTable);  
 setComponentPosition(cityScrollPane,new Point(330,window.getHeight()-150), new Point(10,10));  
 cityPanel.add(cityScrollPane);  
  
 //RESULT PANEL  
 JPanel resultPanel = new JPanel();  
 resultPanel.setLayout(null);  
 setComponentPosition(resultPanel,new Point(window.getWidth()-350,window.getHeight()-100), new Point(350,100));  
 resultPanel.setBackground(Color.*RED*);  
  
 String [] resultTableHeader = {"ІНДЕКС МІСТА","НАЗВА МІСТА","КІЛЬКІСТЬ ЗОЛОТА ЗА ХІД","ЗАГАЛЬНА КІЛЬКІСТЬ ЗОЛОТА"};  
 DefaultTableModel resultTableModel = new DefaultTableModel(resultTableHeader,0);  
 JTable resultTable = new JTable(resultTableModel);  
  
 JScrollPane resultScrollPane = new JScrollPane(resultTable);  
 setComponentPosition(resultScrollPane,new Point(window.getWidth()-375,window.getHeight()-150), new Point(10,10));  
 resultPanel.add(resultScrollPane);  
  
 //MINE PANEL  
 JPanel minePanel = new JPanel();  
 minePanel.setLayout(null);  
 setComponentPosition(minePanel,new Point(window.getWidth(),100),new Point(0,0));  
 minePanel.setBackground(Color.*blue*);  
  
 JTextField inputName = new JTextField("Введіть назву Шахти");  
 setComponentPosition(inputName,new Point(230,25),new Point(70,10));  
 JTextField inputDistance = new JTextField("Введіть відстань");  
 setComponentPosition(inputDistance,new Point(115,25),new Point(70,35));  
 JTextField inputGoldValue = new JTextField("Введіть золото");  
 setComponentPosition(inputGoldValue,new Point(115,25),new Point(185,35));  
 JButton addMineButton = new JButton("Додати шахту");  
 setComponentPosition(addMineButton,new Point(230,30),new Point(70,60));  
 addMineButton.addActionListener(getAddButtonActionListener(inputName,inputDistance,inputGoldValue,resultTableModel));  
  
 JLabel comboBoxLabel = new JLabel("Оберіть вже існуючу шахту ⇓");  
 setComponentPosition(comboBoxLabel,new Point(230,25),new Point(470,10));  
 setComponentPosition(minesBox,new Point(230,25),new Point(450,50));  
 minesBox.addActionListener(getMineBoxActionListener(resultTableModel));  
 minePanel.add(inputName);  
 minePanel.add(inputDistance);  
 minePanel.add(inputGoldValue);  
 minePanel.add(addMineButton);  
 minePanel.add(comboBoxLabel);  
 minePanel.add(minesBox);  
  
 window.add(minePanel);  
 window.add(cityPanel);  
 window.add(resultPanel);  
 window.setVisible(true);  
 }  
  
 private void setComponentPosition(Component component, Point size, Point location){  
 component.setSize(size.x,size.y);  
 component.setLocation(location.x,location.y);  
 }  
  
 private ActionListener getAddButtonActionListener(JTextField inName, JTextField inDistance,JTextField inValue, DefaultTableModel model){  
 ActionListener action = new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 minesBox.addItem(inName.getText());  
 jdbcUtil.addMine(Integer.*parseInt*(inDistance.getText()),Integer.*parseInt*(inValue.getText()),inName.getText());  
 sellerBot.setSelectedMine(getNewMine(inName.getText()));  
 ArrayList<String[]> resultList = sellerBot.getChaneOfTownsWithBestPrice();  
 while(model.getRowCount() > 0) model.removeRow(0);  
 for (String[] row : resultList) model.addRow(row);  
 }  
 };  
 return action;  
 }  
  
 private ActionListener getMineBoxActionListener(DefaultTableModel model){  
 ActionListener action = new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 sellerBot = new SellerBot(jdbcUtil);  
 sellerBot.setSelectedMine(getNewMine((String) minesBox.getSelectedItem()));  
 ArrayList<String[]> resultList = sellerBot.getChaneOfTownsWithBestPrice();  
 while(model.getRowCount() > 0) model.removeRow(0);  
 for (String[] row : resultList) model.addRow(row);  
 }  
 };  
 return action;  
 }  
  
 private Mine getNewMine(String mineName){  
 ArrayList<ArrayList> tableData = jdbcUtil.getMineByName(mineName);  
 ArrayList row = tableData.get(0);  
  
 Mine mine = new Mine((int) row.get(1),(int) row.get(2),(String) row.get(3));  
 return mine;  
 }  
  
 private Object[][] getCityTableDataModel(){  
 Object[][] tableData = new Object[jdbcUtil.getTowns().size()][];  
  
 for (int i = 0; i < jdbcUtil.getTowns().size(); i++){  
 ArrayList row = jdbcUtil.getTowns().get(i);  
 tableData[i] = row.toArray(new Object[row.size()]);  
 }  
  
 return tableData;  
 }  
}

SellerBot.java

package ark;  
  
import jdbc.JDBCUtil;  
import tsobject.\*;  
import java.util.ArrayList;  
  
  
public class SellerBot {  
  
 public SellerBot(JDBCUtil jdbcUtil) {  
 this.jdbcUtil = jdbcUtil;  
  
 ArrayList<ArrayList> tableDataTowns = jdbcUtil.getTowns();  
  
 for (int i = 0; i < tableDataTowns.size(); i++){  
 ArrayList row = tableDataTowns.get(i);  
 townsList.add(new Town((String) row.get(1),(int) row.get(2),(int) row.get(3),(int) row.get(4)));  
 }  
 }  
  
 private JDBCUtil jdbcUtil;  
 private User user = new User(10);  
 private Mine selectedMine;  
 private ArrayList <Town> townsList = new ArrayList<>();  
  
 public void setSelectedMine(Mine selectedMine) {  
 this.selectedMine = selectedMine;  
 }  
  
 public ArrayList<Town> getTownsList() {  
 return townsList;  
 }  
  
 public ArrayList<String[]> getChaneOfTownsWithBestPrice(){  
 user.setStockGold(selectedMine.getPossibleGold());  
 ArrayList <String[]> chaneOfTownsForSel = new ArrayList<>();  
 int indexOfTheRichestTown = 0;  
  
 while (user.getStockGold() > 0){  
 sellGoldInTowns(user.getStockGold());  
 indexOfTheRichestTown = getIndexOfTownWithHigesCash();  
  
 if (townsList.get(indexOfTheRichestTown).getValueOfEarnedMoney() <= 0) break;  
  
 user.setStockGold(townsList.get(indexOfTheRichestTown).getRestOfUserGold());  
 user.setEarnedMoney(user.getEarnedMoney() + townsList.get(indexOfTheRichestTown).getValueOfEarnedMoney());  
 townsList.get(indexOfTheRichestTown).setAvailableValueOfGold(0);  
  
 String [] turnResult = {"" + indexOfTheRichestTown, townsList.get(indexOfTheRichestTown).getName(),"" + townsList.get(indexOfTheRichestTown).getValueOfEarnedMoney(),""+ user.getEarnedMoney()};  
 chaneOfTownsForSel.add(turnResult);  
 }  
 String [] turnResult = {"--", "Дорога до шахти","-" + user.payMoneyForRoad(selectedMine.getDistanceToMine()), user.getEarnedMoney() - user.payMoneyForRoad(selectedMine.getDistanceToMine()) + ""};  
 chaneOfTownsForSel.add(turnResult);  
  
 return chaneOfTownsForSel;  
 }  
  
 private void sellGoldInTowns(int valueOfGoldForSell){  
 for (int i = 0, priceForRoadToTown = 0; i < townsList.size(); i++){  
 priceForRoadToTown = user.payMoneyForRoad(townsList.get(i).getDistanceToTown());  
 townsList.get(i).setValueOfEarnedMoney(townsList.get(i).earnMoneyAtTown(valueOfGoldForSell) - priceForRoadToTown);  
 }  
 }  
  
 private int getIndexOfTownWithHigesCash(){  
 int index = 0;  
 for (int i = 0, max = 0; i < townsList.size(); i++){  
 if (townsList.get(i).getValueOfEarnedMoney() > max){  
 max = townsList.get(i).getValueOfEarnedMoney();  
 index = i;  
 }  
 }  
 return index;  
 }  
}

Mine.java

package tsobject;  
  
  
public class Mine {  
 public Mine(int distanceToMine, int possibleGold, String mineName){  
 this.distanceToMine = distanceToMine;  
 this.possibleGold = possibleGold;  
 this.mineName = mineName;  
 }  
  
 private int distanceToMine;  
 private int possibleGold;  
 private String mineName;  
  
 public int getDistanceToMine() {  
 return distanceToMine;  
 }  
 public int getPossibleGold() {  
 return possibleGold;  
 }  
 public String getMineName() {  
 return mineName;  
 }  
}

Town.java

package tsobject;  
  
  
public class Town {  
 public Town(String name, int distanceToTown, int availableValueOfGold, int priceOfGold){  
 this.name = name;  
 this.distanceToTown = distanceToTown;  
 this.availableValueOfGold = availableValueOfGold;  
 this.priceOfGold = priceOfGold;  
 this.valueOfEarnedMoney = 0;  
 }  
  
 private String name;  
 private int distanceToTown;  
 private int availableValueOfGold;  
 private int priceOfGold;  
 private int restOfUserGold;  
 private int valueOfEarnedMoney;  
  
 public String getName() {  
 return name;  
 }  
  
 public int getAvailableValueOfGold() {  
 return availableValueOfGold;  
 }  
  
 public int getPriceOfGold() {  
 return priceOfGold;  
 }  
  
 public int getDistanceToTown() {  
 return distanceToTown;  
 }  
  
 public int getRestOfUserGold() {  
 return restOfUserGold;  
 }  
  
 public void setAvailableValueOfGold(int availableValueOfGold) {  
 this.availableValueOfGold = availableValueOfGold;  
 }  
  
 public void setValueOfEarnedMoney(int valueOfEarnedMoney) {  
 this.valueOfEarnedMoney = valueOfEarnedMoney;  
 }  
  
 public int getValueOfEarnedMoney() {  
 return valueOfEarnedMoney;  
 }  
  
 public int earnMoneyAtTown(int valueForSell){  
 return getAvailableValueForSell(valueForSell) \* priceOfGold;  
 }  
  
 private int getAvailableValueForSell(int valueForSell){  
 if (valueForSell <= availableValueOfGold) {  
 restOfUserGold = 0;  
 return valueForSell;  
 }  
 else {  
 restOfUserGold = valueForSell - availableValueOfGold;  
 valueForSell = availableValueOfGold;  
 return valueForSell;  
 }  
 }  
}

User.java

package tsobject;  
  
  
public class User {  
 public User(int priceForKilometr) {  
 this.priceForKilometr = priceForKilometr;  
 this.stockGold = 0;  
 }  
  
 private int priceForKilometr;  
 private int stockGold;  
 private int earnedMoney;  
  
 public int getStockGold() {  
 return stockGold;  
 }  
  
 public void setStockGold(int stockGold) {  
 this.stockGold = stockGold;  
 }  
  
 public int getEarnedMoney() {  
 return earnedMoney;  
 }  
  
 public void setEarnedMoney(int earnedMoney) {  
 this.earnedMoney = earnedMoney;  
 }  
  
 public int payMoneyForRoad(int distance){  
 return distance \* priceForKilometr;  
 }  
}

JDBCAdapter.java

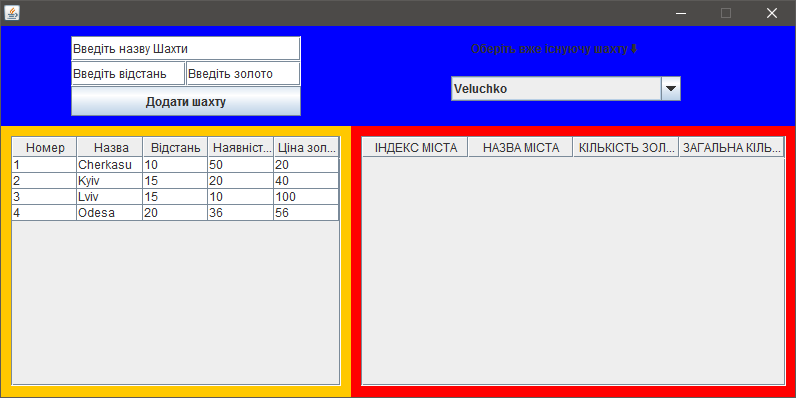
package jdbc;  
  
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.ResultSet;  
import java.sql.ResultSetMetaData;  
import java.sql.SQLException;  
import java.sql.Statement;  
import java.sql.Types;  
import java.util.ArrayList;  
import java.util.List;  
  
import javax.swing.table.AbstractTableModel;  
  
public class JDBCAdapter extends AbstractTableModel {  
 Connection connection;  
 Statement statement;  
 ResultSet resultSet;  
 String[] columnNames = {};  
 ArrayList<ArrayList> rows = new ArrayList();  
 ResultSetMetaData metaData;  
  
 public JDBCAdapter(String url, String driverName,  
 String user, String passwd) {  
 try {  
 Class.*forName*(driverName);  
 System.*out*.println("Opening db connection");  
  
 connection = DriverManager.*getConnection*(url, user, passwd);  
 statement = connection.createStatement();  
 }  
 catch (ClassNotFoundException ex) {  
 System.*err*.println("Cannot find the database driver classes.");  
 System.*err*.println(ex);  
 }  
 catch (SQLException ex) {  
 System.*err*.println("Cannot connect to this database.");  
 System.*err*.println(ex);  
 }  
 }  
  
 public ArrayList executeQuery(String query) {  
 if (connection == null || statement == null) {  
 System.*err*.println("There is no database to execute the query.");  
 return null;  
 }  
 try {  
 resultSet = statement.executeQuery(query);  
 metaData = resultSet.getMetaData();  
  
 int numberOfColumns = metaData.getColumnCount();  
 columnNames = new String[numberOfColumns];  
 // Get the column names and cache them.  
 // Then we can close the connection.  
 for(int column = 0; column < numberOfColumns; column++) {  
 columnNames[column] = metaData.getColumnLabel(column+1);  
 }  
  
 // Get all rows.  
 rows = new ArrayList();  
 while (resultSet.next()) {  
 ArrayList newRow = new ArrayList();  
 for (int i = 1; i <= getColumnCount(); i++) {  
 newRow.add(resultSet.getObject(i));  
 }  
 rows.add(newRow);  
 }  
 // close(); Need to copy the metaData, bug in jdbc:odbc driver.  
 fireTableChanged(null); // Tell the listeners a new table has arrived.  
 }  
 catch (SQLException ex) {  
 System.*err*.println(ex);  
 }  
  
 return rows;  
 }  
  
 public void close() throws SQLException {  
 System.*out*.println("Closing db connection");  
 resultSet.close();  
 statement.close();  
 connection.close();  
 }  
  
 protected void finalize() throws Throwable {  
 close();  
 super.finalize();  
 }  
  
 //////////////////////////////////////////////////////////////////////////  
 //  
 // Implementation of the TableModel Interface  
 //  
 //////////////////////////////////////////////////////////////////////////  
  
 // MetaData  
  
 public String getColumnName(int column) {  
 if (columnNames[column] != null) {  
 return columnNames[column];  
 } else {  
 return "";  
 }  
 }  
  
 public Class getColumnClass(int column) {  
 int type;  
 try {  
 type = metaData.getColumnType(column+1);  
 }  
 catch (SQLException e) {  
 return super.getColumnClass(column);  
 }  
  
 switch(type) {  
 case Types.*CHAR*:  
 case Types.*VARCHAR*:  
 case Types.*LONGVARCHAR*:  
 return String.class;  
  
 case Types.*BIT*:  
 return Boolean.class;  
  
 case Types.*TINYINT*:  
 case Types.*SMALLINT*:  
 case Types.*INTEGER*:  
 return Integer.class;  
  
 case Types.*BIGINT*:  
 return Long.class;  
  
 case Types.*FLOAT*:  
 case Types.*DOUBLE*:  
 return Double.class;  
  
 case Types.*DATE*:  
 return java.sql.Date.class;  
  
 default:  
 return Object.class;  
 }  
 }  
  
 public boolean isCellEditable(int row, int column) {  
 try {  
 return metaData.isWritable(column+1);  
 }  
 catch (SQLException e) {  
 return false;  
 }  
 }  
  
 public int getColumnCount() {  
 return columnNames.length;  
 }  
  
 // Data methods  
  
 public int getRowCount() {  
 return rows.size();  
 }  
  
 public Object getValueAt(int aRow, int aColumn) {  
 List row = (List)rows.get(aRow);  
 return row.get(aColumn);  
 }  
  
 public String dbRepresentation(int column, Object value) {  
 int type;  
  
 if (value == null) {  
 return "null";  
 }  
  
 try {  
 type = metaData.getColumnType(column+1);  
 }  
 catch (SQLException e) {  
 return value.toString();  
 }  
  
 switch(type) {  
 case Types.*INTEGER*:  
 case Types.*DOUBLE*:  
 case Types.*FLOAT*:  
 return value.toString();  
 case Types.*BIT*:  
 return ((Boolean)value).booleanValue() ? "1" : "0";  
 case Types.*DATE*:  
 return value.toString(); // This will need some conversion.  
 default:  
 return "\""+value.toString()+"\"";  
 }  
  
 }  
  
 public void setValueAt(Object value, int row, int column) {  
 try {  
 String tableName = metaData.getTableName(column+1);  
 // Some of the drivers seem buggy, tableName should not be null.  
 if (tableName == null) {  
 System.*out*.println("Table name returned null.");  
 }  
 String columnName = getColumnName(column);  
 String query =  
 "update "+tableName+  
 " set "+columnName+" = "+dbRepresentation(column, value)+  
 " where ";  
 // We don't have a model of the schema so we don't know the  
 // primary keys or which columns to lock on. To demonstrate  
 // that editing is possible, we'll just lock on everything.  
 for(int col = 0; col<getColumnCount(); col++) {  
 String colName = getColumnName(col);  
 if (colName.equals("")) {  
 continue;  
 }  
 if (col != 0) {  
 query = query + " and ";  
 }  
 query = query + colName +" = "+  
 dbRepresentation(col, getValueAt(row, col));  
 }  
 System.out.println(query);  
 statement.executeUpdate(query);  
 }  
 catch (SQLException e) {  
 e.printStackTrace();  
 System.err.println("Update failed");  
 }  
 List dataRow = (List)rows.get(row);  
 dataRow.set(column, value);  
 }  
  
 public void addRow(String s){  
 try{  
 // statement.execute("INSERT INTO mines VALUES (null," + mDistance + "," + mGoldValue + ",\""+ mName +"\")");  
  
 statement.executeUpdate(s);  
 }  
 catch (SQLException e){  
 System.out.println("Error in adding of data!");  
 }  
 }  
}

JDBCUtil.java

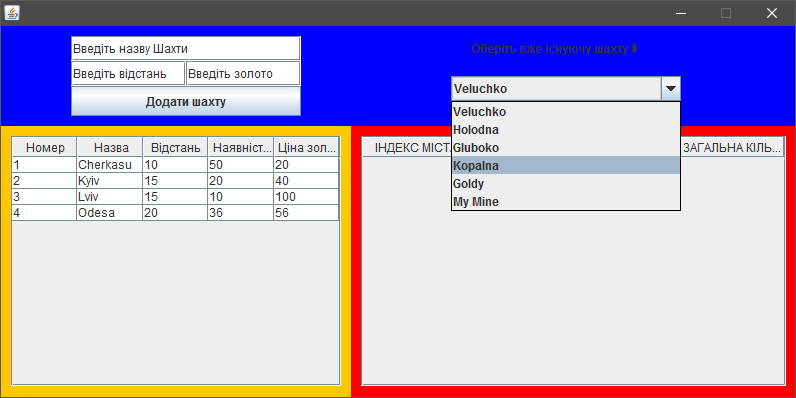
package jdbc;  
  
import java.sql.SQLException;  
import java.util.ArrayList;  
  
  
public class JDBCUtil {  
  
 JDBCAdapter jdbc = new JDBCAdapter("jdbc:mysql://localhost/gold\_industry?autoReconnect=true&useSSL=false","com.mysql.jdbc.Driver","root","04091998a");  
  
 public ArrayList<ArrayList> getTowns(){  
 ArrayList<ArrayList> tableData = jdbc.executeQuery("SELECT \* from towns");  
 return tableData;  
 }  
  
 public ArrayList<ArrayList> getMines(){  
 ArrayList<ArrayList> tableData = jdbc.executeQuery("SELECT \* from mines");  
 return tableData;  
 }  
  
 public ArrayList<ArrayList> getMineByName(String mineName){  
 ArrayList<ArrayList> tableData = jdbc.executeQuery("SELECT \* FROM mines WHERE name IN (\"" + mineName + "\")");  
 return tableData;  
 }  
  
 public void addMine(int mDistance, int mGoldValue, String mName){  
 jdbc.addRow("INSERT INTO mines VALUES (null," + mDistance + "," + mGoldValue + ",\""+ mName +"\")");  
 }  
  
 public void closeConnection(){  
 try {  
 jdbc.close();  
 }  
 catch (SQLException e){  
 System.*out*.println("Connecting cannot be close!");  
 }  
 }  
}

**Результат виконання програми:**

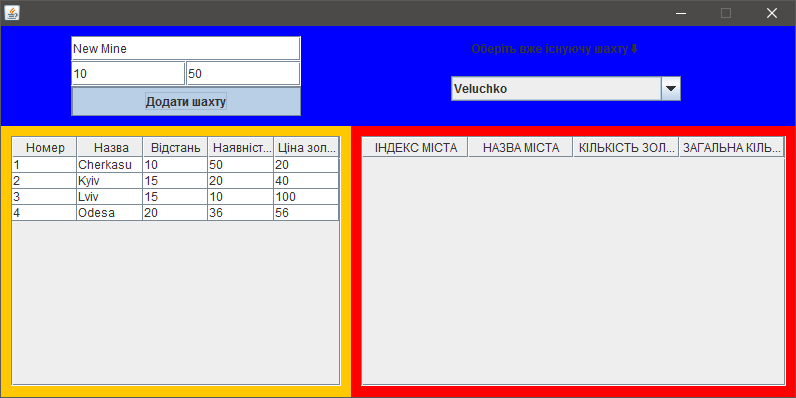
Головне вікно програми. Зліва містить інформація про міста, в яких можна продавати золото із шахт.



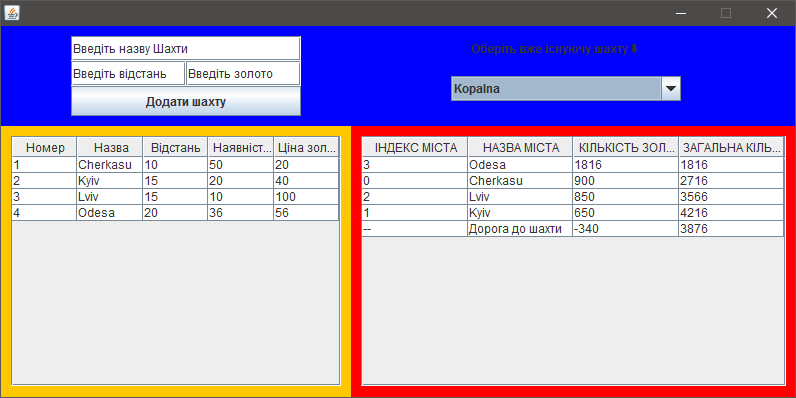
Користувач може обрати вже існуючу копальню.

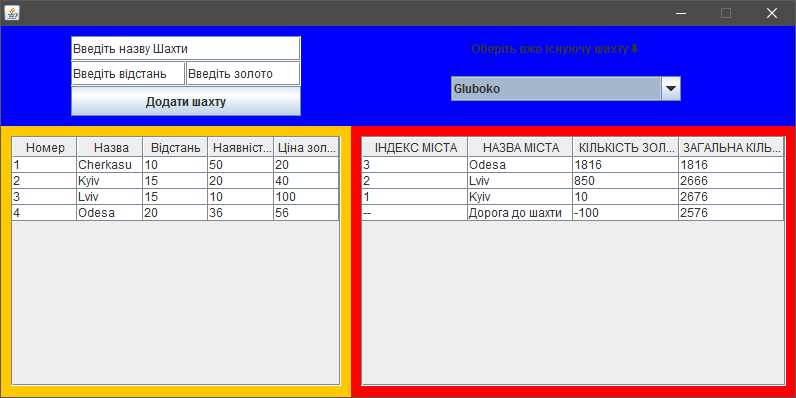


Або додати нову шахту із власною інформацією.



При виборі шахти, користувачу виведеться таблиця із ланцюгом міст, у яких почергово треба продати накопичене золото, щоб отримати найбільший прибуток.





**Висновок:** під час виконання лабораторної роботи я навчився використовувати базу даних у своїх додатках. Освоїв один із способів роботи із базою із допомогою технології JDBC. Навчився створювати додатки, що можуть показувати і дозволяють користувачеві редагувати інформацію із бази даних.