Міністерство освіти і науки України

Черкаський державний технологічний університет

Кафедра програмного забезпечення автоматизованих систем

Звіт

З лабораторної роботи №10

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:** Використання ORM-систем при роботі з базами даних. ORM Hibernate.

**Мета:** Зрозуміти призначення ORM-систем. Навчитись використовувати ORM-системи при створенні програм, що працюють з базами даних. Навчитись використовувати ORM-систему Hibernate.

**Постановка завдання:** Створити базу даних, в якій є дві пов'язані між собою таблиці, кожна з яких налічує не менше 4-х полів.

Написати Java-програму, яка відображає дві дані таблиці бази даних на екрані в табличному вигляді, причому так, що в кожному рядку основної таблиці показані всі рядки підлеглої таблиці. Читання з бази даних повинно відбуватись з використанням ORM Hibernate. Entity-класи повинні вміщувати зв'язки ManyToOne та OneToMany.

**Теоретичні відомості.**

Hibernate — засіб відображення між об'єктами та реляційними структурами (object-relational mapping, ORM) для платформи Java. Hibernate є вільним програмним забезпеченням, яке поширюється на умовах GNU Lesser General Public License. Hibernate надає легкий для використання каркас (фреймворк) для відображення між об'єктно-орієнтованою моделлю даних і традиційною реляційною базою даних.

Метою Hibernate є звільнення розробника від значних типових завдань із програмування взаємодії з базою даних. Розробник може використовувати Hibernate як при розробці з нуля, так і для вже існуючої бази даних.

Hibernate піклується про зв'язок класів з таблицями бази даних (і типів даних мови програмування із типами даних SQL), і надає засоби автоматичної побудови SQL запитів й зчитування/запису даних, і може значно зменшити час розробки, який зазвичай витрачається на ручне написання типового SQL і JDBC коду. Hibernate генерує SQL виклики і звільняє розробника від ручної обробки результуючого набору даних, конвертації об'єктів і забезпечення сумісності із різними базами даних.

Hibernate забезпечує прозору підтримку збереження даних, тобто їхньої персистентності (англ. persistence) для «POJO»-об'єктів, себто для звичайних Java-об'єктів; єдина сувора вимога до класу, що зберігається — конструктор за замовчанням (Для коректної поведінки у деяких застосуваннях потрібно приділити особливу увагу до методів equals() і hashCode()[1]).

Mapping Java класів з таблицями бази даних здійснюється за допомогою конфігураційних XML файлів або Java анотацій. При використанні файлу XML, Hibernate може генерувати скелет вихідного коду для класів тривалого зберігання (persistent). У цьому немає необхідності, якщо використовується анотація. Hibernate може використовувати файл XML або анотації для підтримки схеми бази даних.

Забезпечуються можливості з організації відношення між класами «один-до-багатьох» і «багато-до-багатьох». На додаток до управління зв'язками між об'єктами, Hibernate також може керувати рефлексивними асоціаціями, де об'єкт має зв'язок «один-до-багатьох» з іншими примірниками свого власного типу даних.

Hibernate підтримує відображення користувацьких типів значень. Це робить можливим такі сценарії:

Перевизначення типу за замовчуванням SQL, який Hibernate вибирає при відображенні стовпчика властивості.

Картування перераховуваного типу Java до колонок БД, так ніби вони є звичайними властивостями.

Картування однієї властивості в декілька колонок.

**Виконання лабораторної роботи:**

TraidingSystem.java

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

SellerBot.java

package ark;  
  
import database.\*;  
import tsobject.\*;  
import java.util.ArrayList;  
  
  
public class SellerBot {  
  
 public SellerBot(ArrayList<Town> townsList) {  
 this.townsList = townsList;  
 }  
  
 private User user = new User(10);  
 private Mines selectedMine;  
 private ArrayList <Town> townsList;  
  
 public void setSelectedMine(Mines selectedMine) {  
 this.selectedMine = selectedMine;  
 }  
  
 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;  
 }  
}

Window.java

package ark;  
  
import database.HibernateUtil;  
import database.Mines;  
  
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;  
import java.util.List;  
  
public class Window {  
 public Window() {  
 List<Mines> mineList = hibernateUtil.getMinesList();  
 for(Mines mines : mineList) minesBox.addItem(mines.getMineName());  
 }  
  
 private HibernateUtil hibernateUtil = new HibernateUtil();  
 private SellerBot sellerBot = new SellerBot(hibernateUtil.getTownArray());  
 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);  
  
 JTable cityTable = new JTable(hibernateUtil.getTownTableModel());  
 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());  
 hibernateUtil.addNewMine(Integer.parseInt(inDistance.getText()),Integer.parseInt(inValue.getText()),inName.getText());  
 sellerBot.setSelectedMine(hibernateUtil.getMineByName(inName.getText()));  
 preparingOfResult(model);  
 }  
 };  
 return action;  
 }  
  
 private ActionListener getMineBoxActionListener(DefaultTableModel model){  
 ActionListener action = new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 sellerBot = new SellerBot(hibernateUtil.getTownArray());  
 sellerBot.setSelectedMine(hibernateUtil.getMineByName((String) minesBox.getSelectedItem()));  
 preparingOfResult(model);  
 }  
 };  
 return action;  
 }  
  
 private void preparingOfResult(DefaultTableModel model){  
 ArrayList<String[]> resultList = sellerBot.getChaneOfTownsWithBestPrice();  
 while(model.getRowCount() > 0) model.removeRow(0);  
 for (String[] row : resultList) model.addRow(row);  
 }  
}

Town.java

package tsobject;  
  
import database.Towns;  
  
public class Town {  
 public Town(Towns town){  
 this.name = town.getName();  
 this.distanceToTown = town.getDistanceToTown();  
 this.availableValueOfGold = town.getAvailableValueOfGold();  
 this.priceOfGold = town.getPriceOfGold();  
 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;  
 }  
}

hibernate.cfg.xml

<?xml version='1.0' encoding='utf-8'?>  
<!DOCTYPE hibernate-configuration PUBLIC  
"-//Hibernate/Hibernate Configuration DTD//EN"  
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">  
<hibernate-configuration>  
<session-factory>  
 <property name="connection.driver\_class">   
 com.mysql.jdbc.Driver  
 </property>  
 <property name="connection.url">  
 jdbc:mysql://localhost:3306/gold\_industry?characterEncoding=utf8  
 </property>  
 <property name="connection.username">root</property>  
 <property name="connection.password">04091998a</property>  
 <property name="connection.pool\_size">10</property>  
 <property name="show\_sql">true</property>  
 <property name="dialect">org.hibernate.dialect.MySQL5Dialect</property>  
 <property name="useUnicode">true</property>  
 <property name="hibernate.hbm2ddl.auto">update</property>  
 <!-- Mapping files -->  
 <mapping class="database.Mines"/>  
 <mapping class="database.Towns"/>  
</session-factory>  
</hibernate-configuration>

HibernateUtil.java

package database;  
  
  
import org.hibernate.Session;  
import org.hibernate.SessionFactory;  
import org.hibernate.Transaction;  
import org.hibernate.cfg.Configuration;  
import tsobject.Town;  
  
import javax.swing.table.DefaultTableModel;  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Vector;  
  
public class HibernateUtil {  
 public HibernateUtil(){  
 try {  
 sessionFactory = new Configuration().configure("database/hibernate.cfg.xml").buildSessionFactory();  
 session = sessionFactory.openSession();  
 } catch (Throwable ex) {  
 System.*err*.println("Unfortenatly Initial SessionFactory creation failed." + ex);  
 throw new ExceptionInInitializerError(ex);  
 }  
 }  
  
 private SessionFactory sessionFactory;  
 private Session session;  
  
  
 private List<Towns> getTownsList(){  
 return session.createQuery("from Towns").list();  
 }  
  
 public ArrayList<Town> getTownArray(){  
 ArrayList<Town> townArray = new ArrayList<>();  
 List<Towns> townsList = getTownsList();  
 for(Towns towns : townsList) townArray.add(new Town(towns));  
  
 return townArray;  
 }  
  
 public DefaultTableModel getTownTableModel(){  
 String [] headerCity = {"Номер", "Назва", "Відстань", "Наявність золота", "Ціна золота"};  
 DefaultTableModel dfTableModel = new DefaultTableModel(headerCity,0);  
 List<Towns> townsList = getTownsList();  
  
 for(Towns towns : townsList){  
 Vector<Object> row = new Vector();  
 row.add(towns.getId());  
 row.add(towns.getName());  
 row.add(towns.getDistanceToTown());  
 row.add(towns.getAvailableValueOfGold());  
 row.add(towns.getPriceOfGold());  
  
 dfTableModel.addRow(row);  
 }  
 return dfTableModel;  
 }  
  
 public List<Mines> getMinesList(){  
 return session.createQuery("from Mines").list();  
 }  
  
 public Mines getMineByName (String mineName){  
 return (Mines) session.createQuery("from Mines where mineName = :mineName").setParameter("mineName",mineName).uniqueResult();  
 }  
  
 public void addNewMine(int distance, int possibleGold, String mineName){  
 Transaction transaction = session.beginTransaction();  
  
 Mines mines = new Mines();  
 mines.setDistanceToMine(distance);  
 mines.setPossibleGold(possibleGold);  
 mines.setMineName(mineName);  
  
 session.save(mines);  
 transaction.commit();  
 }  
}

Mines.java

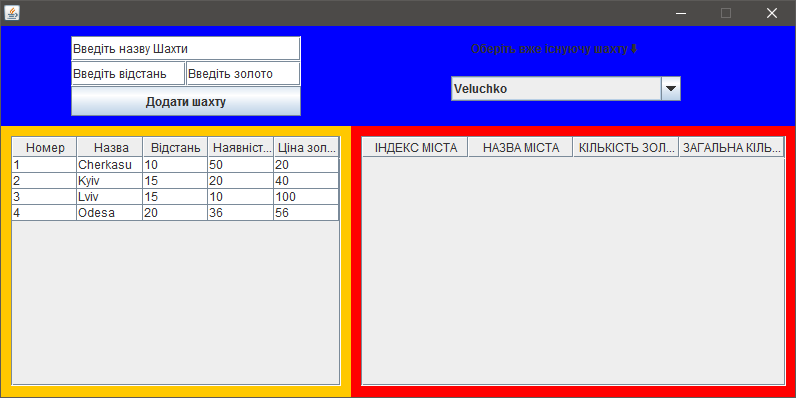
package database;  
  
import javax.persistence.Entity;  
import javax.persistence.Id;  
  
@Entity  
public class Mines {  
  
 @Id  
 private int id;  
 private int distanceToMine;  
 private int possibleGold;  
 private String mineName;  
  
 public int getDistanceToMine() {  
 return distanceToMine;  
 }  
  
 public int getPossibleGold() {  
 return possibleGold;  
 }  
  
 public String getMineName() {  
 return mineName;  
 }  
  
 public void setDistanceToMine(int distanceToMine) {  
 this.distanceToMine = distanceToMine;  
 }  
  
 public void setPossibleGold(int possibleGold) {  
 this.possibleGold = possibleGold;  
 }  
  
 public void setMineName(String mineName) {  
 this.mineName = mineName;  
 }  
}

Towns.java

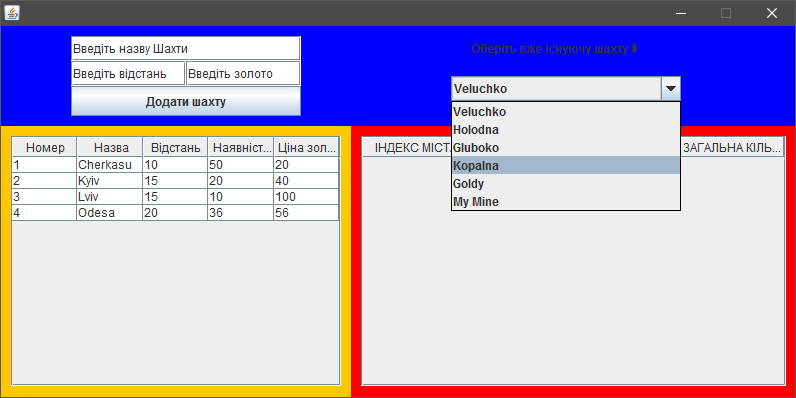
package database;  
  
import javax.persistence.Entity;  
import javax.persistence.Id;  
  
@Entity  
public class Towns {  
 @Id  
 private int id;  
 private String name;  
 private int distanceToTown;  
 private int availableValueOfGold;  
 private int priceOfGold;  
  
 public int getId() {  
 return id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public int getDistanceToTown() {  
 return distanceToTown;  
 }  
  
 public int getAvailableValueOfGold() {  
 return availableValueOfGold;  
 }  
  
 public int getPriceOfGold() {  
 return priceOfGold;  
 }  
}

**Результат виконання програми:**

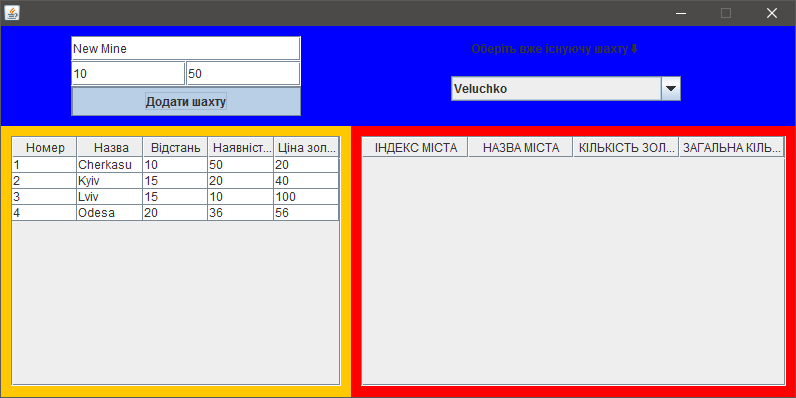
Головне вікно програми. Зліва містить інформація про міста, в яких можна продавати золото із шахт.



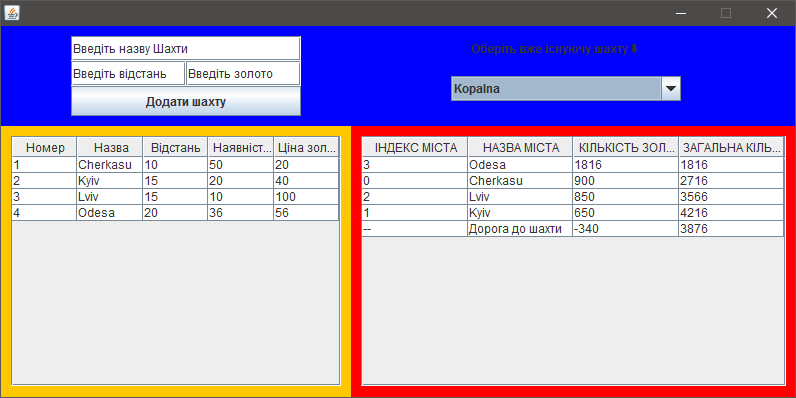
Користувач може обрати вже існуючу копальню.

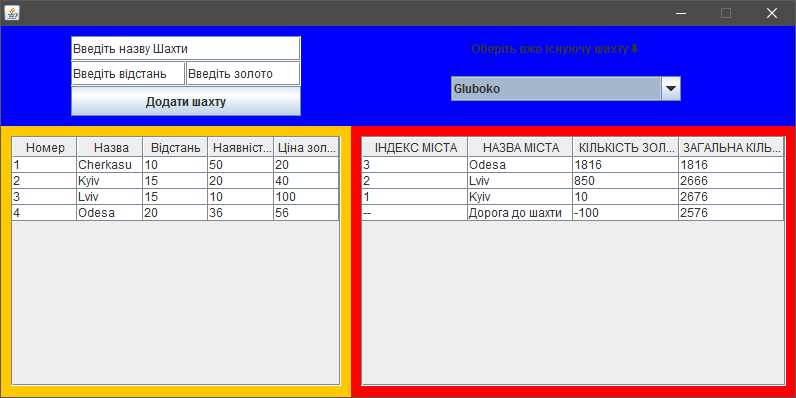


Або додати нову шахту із власною інформацією.



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 застосовувати ORM системи для зв'язку бази даних і додатку. В якості ORM використовував Hibernate і освоїв базові принципи роботи із даною системою. Навчився представляти і змінювати дані із бази даних із допомогою Hibernate.