МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра информационной безопасности

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«11» ноября 2017г.

СОЗДАНИЕ ПРИЛОЖЕНИЯ ДЛЯ ДОСТУПА К БАЗЕ ДАННЫХ С ИСПОЛЬЗОВАНИЕМ ТЕХНОЛОГИИ JDBC

Методические указания по выполнению практических работ по дисциплине «Проектирование защищенных телекоммуникационных систем» для студентов специальности 10.05.02

Курск 2017

УДК 004.056.55

Составители: А.Л. Марухленко

Рецензент
Кандидат технических наук, доцент А.Г. Спеваков

Создание приложения для доступа к базе данных с использованием технологии JDBC: методические указания к выполнению практических работ / Юго-Зап. гос. ун-т; сост.: А. Л. Марухленко Курск, 2017. - 19 с.

Указывается порядок выполнения практической работы, правила оформления, содержание отчета.

Методические указания по выполнению практических работ по дисциплине «Проектирование защищенных телекоммуникационных систем» соответствуют требованиям программы, утвержденной учебно-методическим объединением и предназначены для студентов направления подготовки 10.05.02.

Текст печатается в авторской редакции

Подписано в печать 01.11.2017. Формат 60х84 1/16.
Усл.печ. л. 1,1. Уч.-изд.л. 1,0. Тираж 30 экз. Заказ _____. Бесплатно.
Юго-Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

СОДЕРЖАНИЕ

1. ЦЕЛЬ РАБОТЫ

Разработать приложение для автоматизации учета товаров на складе. Приложение должно иметь диалоговый пользовательский интерфейс и позволять пользователю просматривать, добавлять, изменять и удалять информацию о товарах из таблицы products. Таблица products должна содержать следующие поля: идентификатор товара (id), описание (description), цена (rate), количество (quantity).

2. ВЫПОЛНЕНИЕ РАБОТЫ

Для решения поставленной задачи необходимо выполнить следующие шаги:

1. Создать новый проект.
2. Создать в БД таблицу `products` для хранения данных о товарах.
3. Реализовать Java-класс `Product` для представления и обработки данных о товаре.
4. Реализовать Java-класс обработки данных `ProductDAO`, в котором обеспечить получение соединения БД и реализовать методы выборки, создания, обновления и удаления товара.
5. Разработать графический интерфейс приложения (Swing) и обеспечить вызов методов обработки данных.
6. Запустить и проверить работу приложения.

2.1 Подготовительный этап

Для реализации проекта необходимо установить и настроить среду разработки Ecplise, Apache Derby и Derby Plugins (см. п. «Установка и настройка программного обеспечения»).

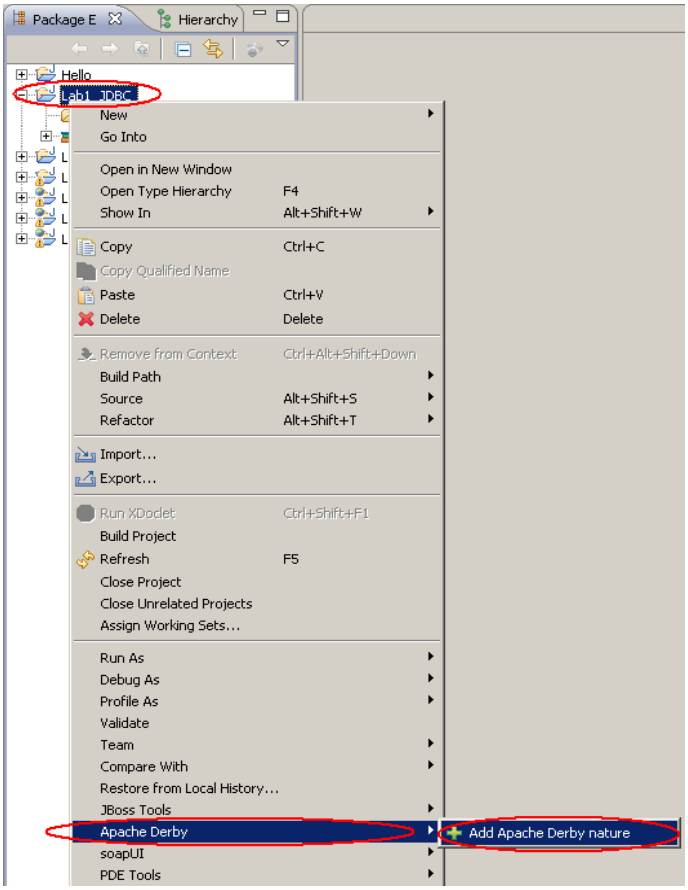
2.2 Создание нового проекта

1) Выберите пункт меню File/New/Project, в окне выбора типа проекта укажите other/Java Project и нажмите Next.

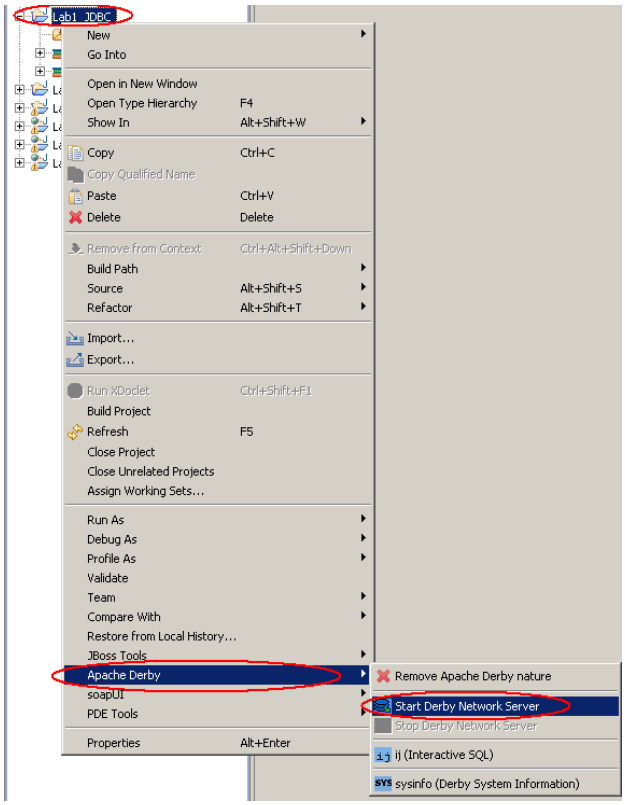
2) Укажите имя проекта Lab1_JDBC, проверьте параметры подраздела JRE – должна быть указана JRE 1.6. (или 1.5, если используется предыдущая версия, но лучше 1.6) Нажмите Finish.

2.3 Создание таблицы `products`

1) Подключите к проекту окружение БД Derby и запустите сервер БД. Для этого, щелкните правой кнопкой мыши на созданный проект в представлении Package Explorer и выберите пункт меню Apache Derby/Add Apache Derby Nature.



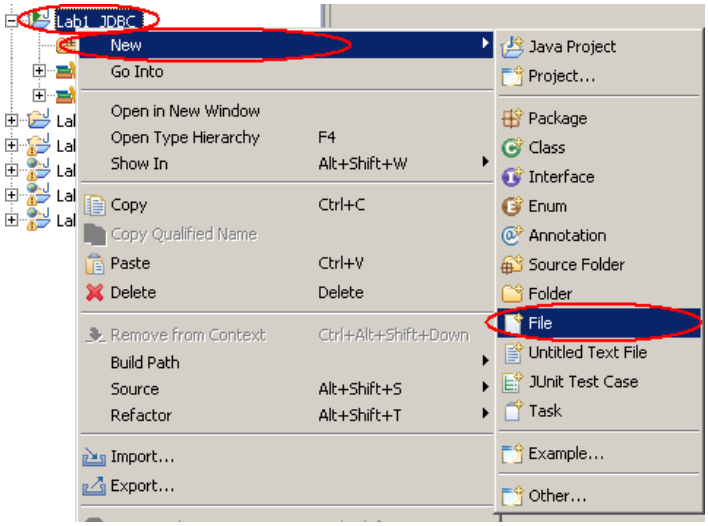
Затем повторно щелкните правой кнопкой мыши на проект и выберите Apache Derby/Start Derby Network Server



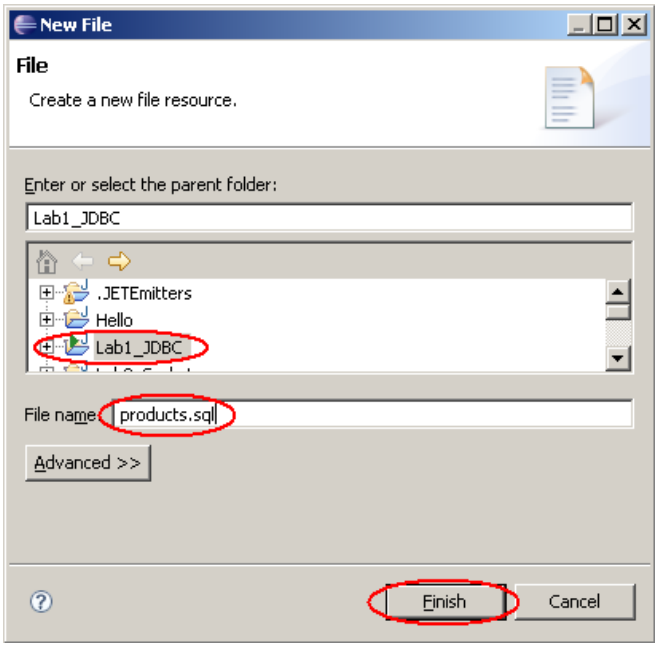
В случае успешного запуска сервера БД в консоли (представление Concole) появляется следующее сообщение:

```
DRDA_SecurityInstalled.I
Сетевой сервер Apache Derby Network Server - 10.3.2.1 - (599110) запущен и готов принимать соединения на порту 1527
```

2) Для хранения SQL-скриптов создадим новый файл products.sql. В окне Package Explorer щелкните правой кнопкой мыши на значок проекта и выберите New/File.



3) В появившемся окне укажите имя файла products.sql и нажмите Finish.

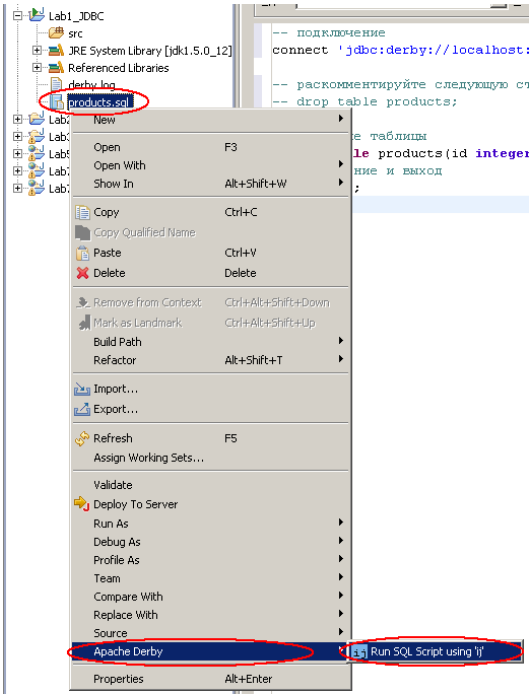


4) SQL-скрипт должен содержать команду создания таблицы `products`. Скопируйте в файл следующие команды:

```
-- подключение
connect 'jdbc:derby://localhost:1527/myDB;
           create=true;user=me;password=mine';

-- раскомментируйте следующую строку, если требуется пересоздать таблицу
-- drop table products;

-- создание таблицы
create table products(id integer, description varchar(100), rate float,
quantity integer);

-- отключение и выход
disconnect;
exit;
```

5) Сохраните файл нажатием на Ctrl-S.

6) Щелкните правой кнопкой мыши на файл products.sql в окне Package Explorer и выберите Apache Derby/Run SQL Script using ‘ij’.



7) В случае успешного выполнения скрипта в консоли выводится следующее:

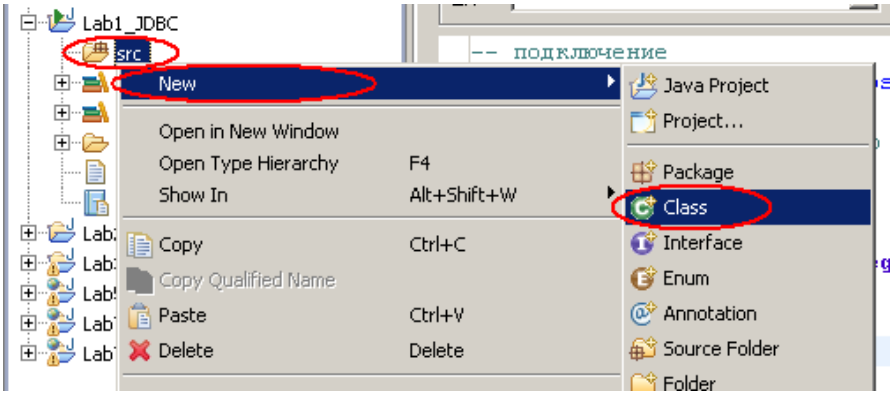
```
      ij version 10.3
      ij> -- подключение
      connect
'jdbc:derby://localhost:1527/myDB;create=true;user=me;password=mine';
      ij> -- раскомментируйте следующую строку, если требуется пересоздать
таблицу
      -- drop table products;

      -- создание таблицы
      create table products(id integer, description varchar(100), rate float,
quantity integer);
      0 rows inserted/updated/deleted
```

```
ij> -- отключение и выход
disconnect;
ij> exit;
```

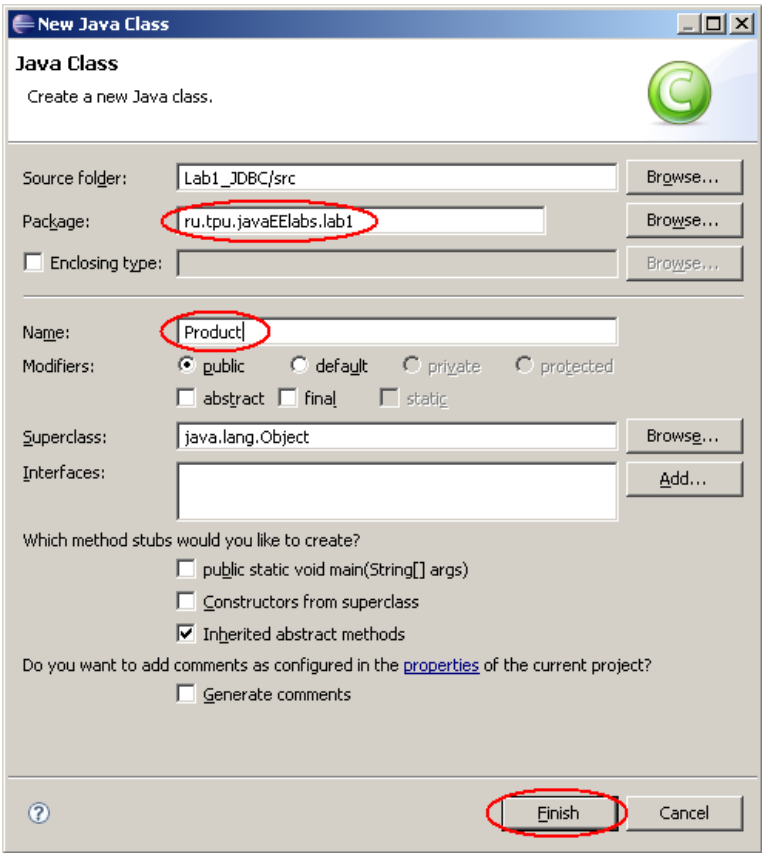
2.4 Создание класса Product

Класс Product является обычным Java-классом и будет использоваться для передачи и обработки данных о товаре. Класс содержит четыре поля – `id`, `description`, `rate`, `quantity`, соответствующие полям таблицы `products`.

1) В окне Package Explorer щелкните правой кнопкой мыши на каталог src проекта Lab1_JDBC и выбрав пункт меню New/Class.



2) Назовите класс Product и задайте имя пакета (Package) `ru.tpu.javaEElabs.lab1`. Нажмите Finish.



3) В созданном классе необходимо реализовать четыре поля `id`, `description`, `rate`, `quantity`, создать конструктор и методы get/set для доуступа к этим полям. Полный код класса `Product` приведен ниже:

```
package ru.tpu.javaEElabs.lab1;

public class Product {

private int id;
private String description;
private float rate;
private int quantity;

public Product(int id, String description, float rate, int quantity) {
	this.id = id;
	this.description = description;
	this.rate = rate;
	this.quantity = quantity;
}

public int getId() {
	return id;
}

	public void setId(int id) {
		this.id = id;
	}

	public String getDescription() {
		return description;
	}

	public void setDescription(String description) {
		this.description = description;
	}

	public float getRate() {
		return rate;
	}

	public void setRate(float rate) {
		this.rate = rate;
	}

	public int getQuantity() {
		return quantity;
	}

	public void setQuantity(int quantity) {
		this.quantity = quantity;
	}
}
```

2.5 Создание класса обработки данных `ProductDAO`

На следующем этапе создадим класс `ProductDAO` (DAO – Data Access Object – объект доступа к данным), который скрывает детали реализации логики работы с БД – получение соединения, выполнение операций выборки, создания, обновления и удаления.

1) В окне Package Explorer щелкните правой кнопкой мыши на пакет `ru.tpu.javaEElabs.lab1` проекта Lab1_JDBC и выбрав пункт меню New/Class.

2) Назовите класс ProductDAO и убедитесь, что имя пакета (Package) задано в `ru.tpu.javaEElabs.lab1`. Нажмите Finish.

Код класса `ProductDAO` приведен ниже:

```
package ru.tpu.javaEElabs.lab1;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;

/**
 */
public class ProductDAO {
	// Вспомогательный метод получения соединения
	private Connection getConnection() throws Exception {
		// Подгрузка драйвера БД Derby
	Class.forName("org.apache.derby.jdbc.ClientDriver").newInstance();
		// Получение соединения с БД
		return DriverManager.getConnection(
"jdbc:derby://localhost:1527/myDB;create=true;user=me;password=mine");
	}

	/**
	 * Возвращает список идентификаторов товаров
	 *
	 * @return
	 */
	public List<Integer> getProductIds() throws Exception {
		List<Integer> productIds = new ArrayList<Integer>();
		// Получение соединения с БД
		Connection con = getConnection();

		// Выполнение SQL-запроса
		ResultSet rs = con.createStatement().executeQuery(
				"Select id From products");
		// Перечисляем результаты выборки
		while (rs.next()) {
			// Из каждой строки выборки выбираем
			// результат и помещаем в коллекцию
			productIds.add(rs.getInt(1));
		}
		// Закрываем выборку и соединение с БД
		rs.close();
		con.close();
		return productIds;
	}

	/**
	 * Возвращает товар по идентификатору
	 *
	 * @return
	 */
	public List<Product> getProductById(int id) throws Exception {
		List<Product> products = new ArrayList<Product>();
```

```
        // Получение соединения с БД
        Connection con = getConnection();

        // Подготовка SQL-запроса
        PreparedStatement st = con.prepareStatement(
                        "Select description, rate, quantity " +
                        "From products " +
                        "Where id = ?");
        // Указание значений параметров запроса
        st.setInt(1, id);

        // Выполнение запроса
        ResultSet rs = st.executeQuery();

        Product product = null;
        // Перечисляем результаты выборки
        while (rs.next()) {
              // Из каждой строки выборки выбираем результаты,
              // формируем новый объект Product
              // и помещаем его в коллекцию
              product = new Product(
                        id,
                        rs.getString(1),
                        rs.getFloat(2),
                        rs.getInt(3));
              products.add(product);
        }
        // Закрываем выборку и соединение с БД
        rs.close();
        con.close();
        return products;
}

/**
 * Добавляет новый товар
 * @param product
 * @throws Exception
 */
public void addProduct(Product product) throws Exception {
        // Получение соединения с БД
        Connection con = getConnection();

        // Подготовка SQL-запроса
        PreparedStatement st = con.prepareStatement(
                    "Insert into products " +
                    "(id, description, rate, quantity) " +
                    "values (?, ?, ?, ?)");
        // Указание значений параметров запроса
        st.setInt(1, product.getId());
        st.setString(2, product.getDescription());
        st.setFloat(3, product.getRate());
        st.setInt(4, product.getQuantity());

        // Выполнение запроса
        st.executeUpdate();

        con.close();
}

/**
 * Обновляет данные о товаре
 * @param product
 * @throws Exception
```

```
     */
    public void setProduct(Product product) throws Exception {
        // Получение соединения с БД
        Connection con = getConnection();

        // Подготовка SQL-запроса
        PreparedStatement st = con.prepareStatement(
                "Update products " +
                "Set description=?, rate=?, quantity=? " +
                "Where id=?");
        // Указание значений параметров запроса
        st.setString(1, product.getDescription());
        st.setFloat(2, product.getRate());
        st.setInt(3, product.getQuantity());
        st.setInt(4, product.getId());

        // Выполнение запроса
        st.executeUpdate();
        con.close();
    }

    public void removeProduct(int id) throws Exception {
        // Получение соединения с БД
        Connection con = getConnection();

        // Подготовка SQL-запроса
        PreparedStatement st = con.prepareStatement(
                "Delete from products " +
                "Where id = ?");
        // Указание значений параметров запроса
        st.setInt(1, id);

        // Выполнение запроса
        st.executeUpdate();
        con.close();
    }

}
```

2.6 Разработка клиентского приложения

Клиентское приложения для просмотра/редактирования товаров будет представлять собой диалоговое приложение Swing, содержащее список для выбора товара по идентификатору и полей, отображающих информацию по выбранному товару. Эти же поля будут использоваться для ввода/редактирования информации по добавляемому/изменяемому товару. Ниже будут располагаться кнопки «Добавить», «Обновить» и «Удалить». Обработчики нажатия на эти кнопки обращаются к полям формы и методам класса `ProductDAO` для выполнения соответствующих действий.

1) Создайте новый Java-класс с именем ProductInfoClient в пакете `ru.tpu.javaEElabs.lab1`. Класс наследует `JDialog`.

Код класса `ProductInfoClient` приведен ниже:

```
package ru.tpu.javaEElabs.lab1;

import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
```

```
import java.util.List;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JDialog;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JTextField;

public class ProductInfoClient extends JDialog  {

	/**
	 *
	 */
	private static final long serialVersionUID = 1L;

	// Создаем DAO-объект
	ProductDAO productDAO = new ProductDAO();

	// Объявление элементов управления
	private JLabel lbSelectId = new JLabel("Выбор товара по Id");
	private JLabel lbId = new JLabel("Id");
	private JLabel lbDescription = new JLabel("Описание");
	private JLabel lbRate = new JLabel("Цена");
	private JLabel lbQuantity = new JLabel("Остаток");

	private JComboBox comboId = new JComboBox();
	private JTextField txtId = new JTextField();
	private JTextField txtDescription = new JTextField();
	private JTextField txtRate = new JTextField();
	private JTextField txtQuantity = new JTextField();

	private JButton btnAdd = new JButton("Добавить");
	private JButton btnUpdate = new JButton("Обновить");
	private JButton btnRemove = new JButton("Удалить");
	private JButton btnClear = new JButton("Очистить");


	/**
	 * Создает экземпляр диалога
	 * @param args
	 */
	public static void main(String[] args) {
		new ProductInfoClient();
	}
	/**
	 * Конструктор диалога
	 */
	public ProductInfoClient() {
		this.setTitle("Информация о товарах");
		this.setDefaultCloseOperation(JDialog.DISPOSE_ON_CLOSE);
		this.setLayout(new GridLayout(8, 2));
		this.setBounds(100, 50, 400, 200);

		// Добавление элементов управления в диалог
		this.add(lbSelectId);
		this.add(comboId);
		this.add(lbId);
		this.add(txtId);
		this.add(lbDescription);
		this.add(txtDescription);
		this.add(lbRate);
		this.add(txtRate);
		this.add(lbQuantity);
```

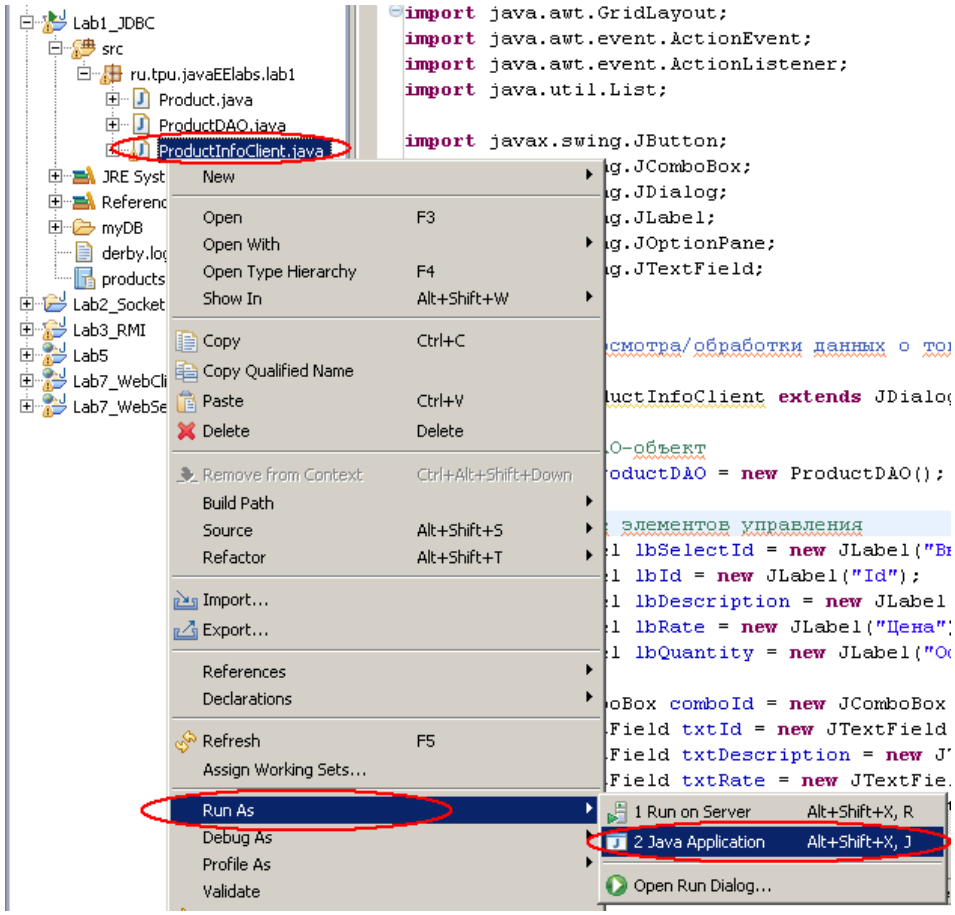
```
        this.add(txtQuantity);
        this.add(btnAdd);
        this.add(btnUpdate);
        this.add(btnRemove);
        this.add(btnClear);

        // Описание обработчиков событий
        comboId.addActionListener(new ActionListener() {
              public void actionPerformed(ActionEvent e) {
                    showProductData();
              }
        });

        btnAdd.addActionListener(new ActionListener() {
              public void actionPerformed(ActionEvent e) {
                    addProduct();
              }
        });

        btnUpdate.addActionListener(new ActionListener() {
              public void actionPerformed(ActionEvent e) {
                    updateProduct();
              }
        });

        btnRemove.addActionListener(new ActionListener() {
              public void actionPerformed(ActionEvent e) {
                    removeProduct();
              }
        });

        btnClear.addActionListener(new ActionListener() {
              public void actionPerformed(ActionEvent e) {
                    clearProductInfo();
              }
        });

        // Обновляем список идентификаторов товаров
        refreshIdList();

        // Отображаем диалог на экране
        this.setVisible(true);
  }

  /**
   * Считывает список идентификаторов товаров
   * и заполняет список
   */
  private void refreshIdList() {
        try {
              // получаем список идентификаторов
              List<Integer> productIds = productDAO.getProductIds();
              // очищаем список
              comboId.removeAllItems();
              // заполняем список полученными данными
              for (Integer productId: productIds) {
                    comboId.addItem(productId);
              }
        } catch (Exception e) {
              e.printStackTrace();
              JOptionPane.showMessageDialog(this, e.getMessage());
        }
  }
```

```
	/**
	 * Отображает данные о товаре по
	 * выбранному в списке идентификатору
	 */
	protected void showProductData() {
	  try {
	      // забираем значение, выбранное в списке
// идентификаторов товаров
	      Integer productId = (Integer)comboId.getSelectedItem();

	      if (productId != null) {
	            // получаем товар по идентификатору

	            List<Product> product =
productDAO.getProductById(productId);
	            // заполняем текстовые поля значениями
// параметров товара
	            for(int i=0; i<product.size(); i++){
	                  txtId.setText(String.valueOf(product.get(i).getId()));

	txtDescription.setText(product.get(i).getDescription());

	txtRate.setText(String.valueOf(product.get(i).getRate()));

	txtQuantity.setText(String.valueOf(product.get(i).getQuantity()));
	            }
	      }
	    } catch (Exception e) {
	      e.printStackTrace();
	      JOptionPane.showMessageDialog(this, e.getMessage());
	    }
	}

	/**
	 * Добавляет новый товар на основе
	 * данных текстовых полей
	 */
	protected void addProduct() {
	      try {
	            // создаем новый объект-товар
	            // на основе данных диалога
	            Product product = new Product(
	                  Integer.parseInt(txtId.getText()),
	                  txtDescription.getText(),
	                  Float.parseFloat(txtRate.getText()),
	                  Integer.parseInt(txtQuantity.getText()));
	            // сохраняем товар в БД
	            productDAO.addProduct(product);
	            // обновляем список идентификаторов
	            refreshIdList();
	            // устанавливаем текущим добавленный товар
comboId.setSelectedItem(
Integer.parseInt(txtId.getText()));
	      } catch (Exception e) {
	            e.printStackTrace();
	            JOptionPane.showMessageDialog(this, e.getMessage());
	      }
	}

	/**
	 * Обновляет информацию о товаре на основе
	 * данных текстовых полей
```

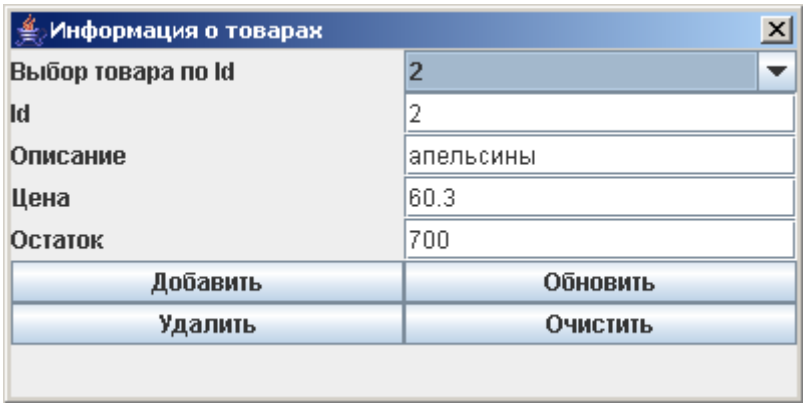
```
	 */
	protected void updateProduct() {
		try {
			// формируем объект-товар
			// на основе данных диалога
			Product product = new Product(
					Integer.parseInt(txtId.getText()),
					txtDescription.getText(),
					Float.parseFloat(txtRate.getText()),
					Integer.parseInt(txtQuantity.getText()));
			// обновляем данные о товаре в БД
			productDAO.setProduct(product);
		} catch (Exception e) {
			e.printStackTrace();
			JOptionPane.showMessageDialog(this, e.getMessage());
		}
	}

	/**
	 * Удаляет выбранный товар
	 */
	protected void removeProduct() {
		try {
			// удаляем товар из БД
			productDAO.removeProduct(
Integer.parseInt(txtId.getText()));
			// обновляем список идентификаторов товаров
			refreshIdList();
			// отображаем данные по первому товару в списке
			showProductData();
		} catch (Exception e) {
			e.printStackTrace();
			JOptionPane.showMessageDialog(this, e.getMessage());
		}
	}

	/**
	 * Очищает данные в текстовых полях
	 */
	protected void clearProductInfo() {
		try {
			txtId.setText("");
			txtDescription.setText("");
			txtRate.setText("");
			txtQuantity.setText("");
		} catch (Exception e) {
			e.printStackTrace();
			JOptionPane.showMessageDialog(this, e.getMessage());
		}
	}
}
```

2.7 Запуск и тестирование приложения

1) Щелкните правой кнопкой мыши на класс ProductInfoClient в окне Package Explorer и выберите команду Run As/Java Application.



2) Проверьте работу приложения – добавьте несколько товаров, проверьте функции обновления и удаления.



3. ВАРИАНТЫ ЗАДАНИЙ

1. Необходимо разработать приложение для учета компакт-дисков в музыкальном салоне. Приложение должно позволять пользователю просматривать, добавлять, изменять и удалять информацию о дисках. Описание диска должно содержать следующие поля: идентификатор (id), жанр, исполнитель, название, год. Необходимо обеспечить отображение

списка дисков в виде таблицы и возможность просмотра дисков по жанру, выбранному пользователем из выпадающего списка.

2. Необходимо разработать приложение для учета книг в книжном магазине. Приложение должно позволять пользователю просматривать, добавлять, изменять и удалять информацию о книгах. Описание книги должно содержать следующие поля: идентификатор (id), жанр, ФИО автора, название, год. Необходимо обеспечить отображение списка книг в виде таблицы и возможность фильтрации книг по ФИО автора, введенной пользователем в текстовом поле.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1) Шувалов В.П., Величко В.В., Субботин Е.А., Ярославцев А.Ф. Телекоммуникационные системы и сети. Том 3. Мультисервисные сети (2005)

2) Петраков А.В. Основы практической защиты информации. 2-е изд. Учебн. пособие. – М.: Радио и связь. 2000. – 368 с.

3) Цифровые и аналоговые системы передачи: Учебник для вузов/ В.И.Иванов, В.Н.Гордиенко, Г.Н.Попов и др.; Под ред. В.И.Иванова. – 2-е изд. – М.: Горячая линия – Телеком, 2003. – 232 с.

4) Гольдштейн Б.С., Соколов Н.А., Яновский Г.Г. Сети связи: Учебник для ВУЗов. - СПб.: БХВ-Петербург, 2010. - 400 с.

5) Башарин Г.П. Лекции по математической теории телетрафика: Учеб. пособие. Изд. 3-е, испр. и доп. - М.: РУДН, 2009. - 342 с.

6) Буч Г. Объектно-ориентированный анализ и проектирование. – М.: Вильямс, 2008.

7) Леоненков А.В. Самоучитель языка UML. – СПб.: БХВ-Петербург, 2004.

8) Розенберг Д., Скотт К. Применение объектного моделирования с использованием UML и анализ прецедентов. – М.: ДМК Пресс, 2002.

9) А.В. Росляков. Виртуальные частные сети. Основы построения и применения. - М.: Эко-Трендз, 2006. - 242 с.