

# Hibernate: Интерфейс пользователя

# Дорожная карта

Для тестирования модели необходимо создать в главном файле проекта подключение к БД и поработать с данными.

# Цели

Создать приложение, работающее с БД через **Hibernate** ORM:

1. Создать визуальный интерфейс пользователя с помощью библиотеки Swing
2. Просмотр данных в виде таблицы
3. Реализация входа в систему
4. Изменение и добавление данных
5. Удаление данных

# Исходные данные

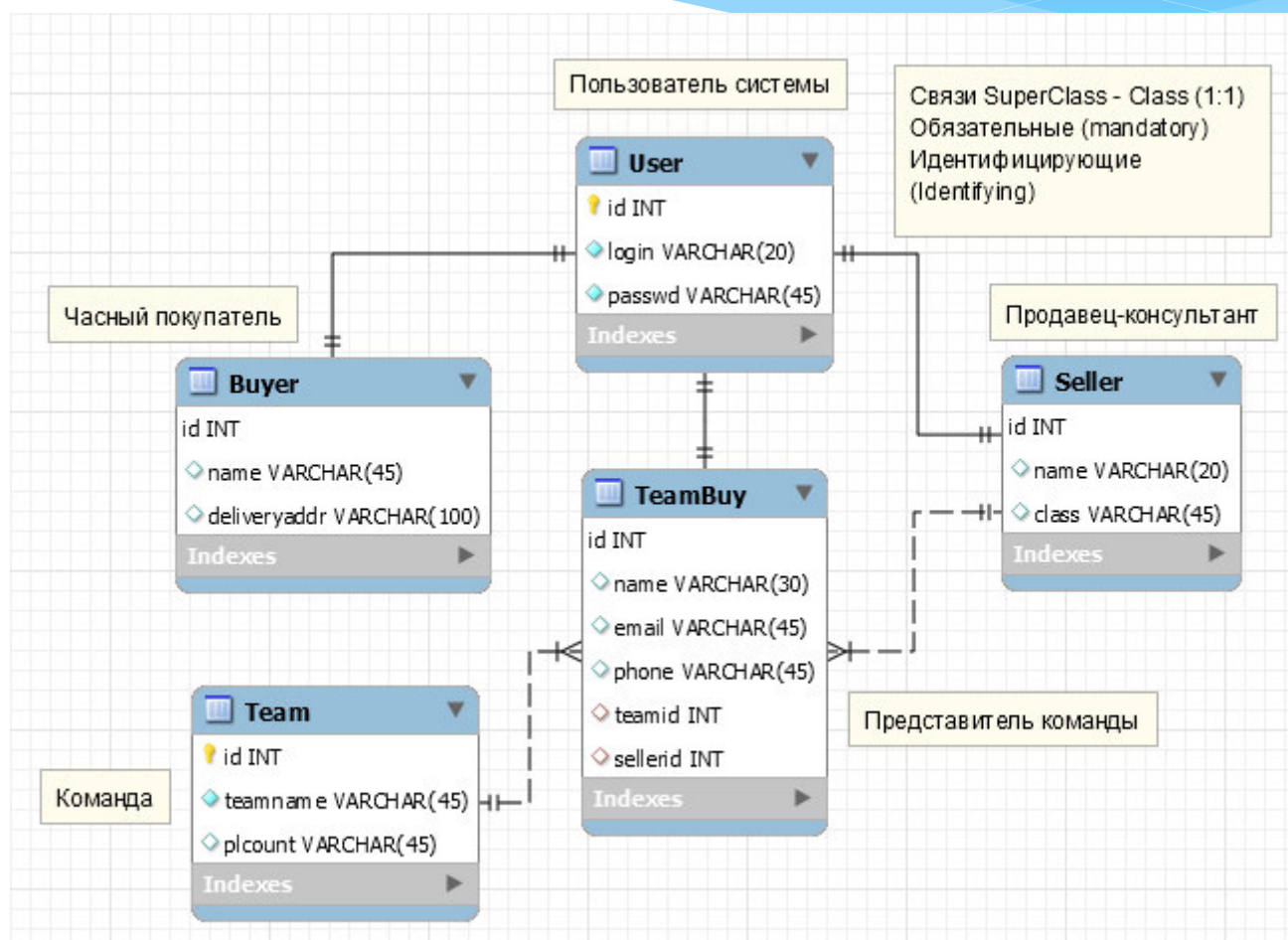
У нас имеется БД-электронный справочник хранящая данные о покупателях, корпоративных клиентах и продавцах-консультантах некой торговой фирмы (OBL).

В БД присутствуют следующие **сущности**:

1. **User** – любой пользователь электронного справочника
2. **Buyer** – покупатель, частное лицо
3. **Seller** – продавец-консультант торговой фирмы
4. **TeamBuy** – корпоративный клиент, представляющий команду
5. **Team** – команда, имеющая своего представителя

# Исходные данные

ER-диаграмма БД в физическом представлении в **MySQL**



# Создание проекта

Приступаем к созданию каркаса приложения.

1. Создаем новый проект Java в NetBeans **File->New Project** (Файл -> Новый проект)
2. Входим в меню и выбираем **File->New File** (Файл->Новый файл)
3. Конфигурируем проект для использования Hibernate, добавляем классы сущностей
4. Добавляем класс формы **File->New File** категория **Swing GUI Forms** тип **JFrame Form**. Название класса формы - **JMainFrame**

Помещаем созданный файл в отдельный контейнер (например UI, GUI, View, Forms и т.п.)

# Настройка Hibernate

Так как визуальный интерфейс событийно управляемый, нам необходимо обеспечить корректный запуск и закрытие соединения с БД. Для этого обеспечим централизованное управление сессией.

1. В **HibernateUtil**, где находится **sessionFactory** добавляем еще одно **public** свойство для хранения сессии
2. Добавляем **get'тер** для нового свойства
3. Добавляем метод **done** закрывающий транзакции, сессии а так же сам **sessionFactory** (корневой класс взаимодействия с БД).

**Get'тер должен создавать сессию, если она еще не создана**

# Настройка Hibernate

1. Добавляем новое свойство в класс **HibernateUtil** сразу после **SessionFactory**. Оно будет хранить текущую активную сессию для того, чтобы она была доступна из любой точки программы.

```
private static Session sess=null;
```

2. get'тер позволяет получить текущую сессию. Если сессия не создана (первый вызов), она автоматически создается

```
public static Session getSession() {  
    if (sess==null)  
        sess=getSessionFactory().openSession();  
    return sess; }  

```



# Настройка Hibernate

3. Метод **done** закрывает активные транзакции и сессии (если таковые есть) и закрывает **sessionFactory**. После его работы программа полностью отключается от БД и закрывает **Hibernate** (**getSession** уже не работает)

```
public static void done() {  
    // Есть открытые сессии?  
    if (sess!=null) {  
        // Транзакция активна?  
        if (sess.getTransaction().isActive())  
            sess.getTransaction().commit();  
        sess.close();  
    }  
    sessionFactory.close();  
}
```

# Создание формы

Так как приложение работает с БД, настроим форму таким образом, чтобы она **не закрывалась автоматически** по нажатию крестика. В главном методе main запустим Hibernate и создадим форму.

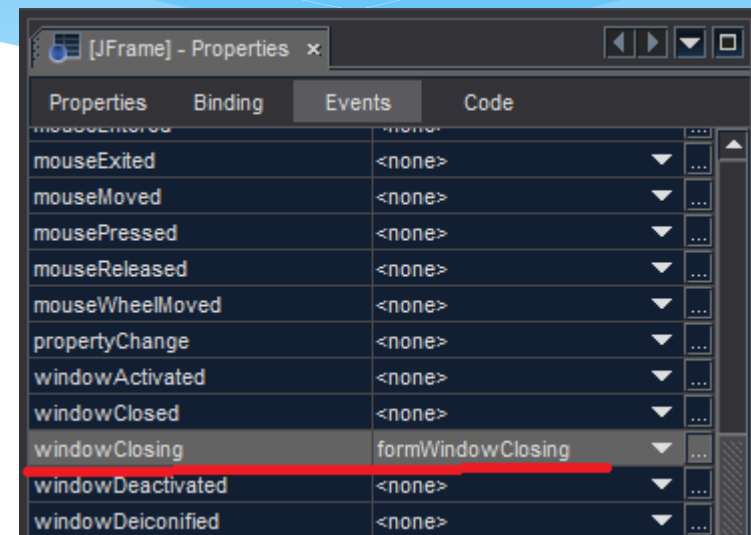
```
HibernateUtil.getSession(); // Открываем соединение
// Создаем форму
JMainFrame MainFRM=new JMainFrame();
// Запрещаем закрываться по кнопке закрытия
MainFRM.setDefaultCloseOperation(
                                JFrame.DO_NOTHING_ON_CLOSE);

// Выводим на экран
MainFRM.setVisible(true);
```

# Настройка формы

В таком виде форму закрыть невозможно, поэтому добавим обработчик **formWindowClosing**.

*Для добавления обработчика необходимо перейти в режим **Design (Конструктор)**.*



Появится следующая заготовка для обработчика кода

```
private void formWindowClosing(java.awt.event.WindowEvent evt) {  
    // TODO add your handling code here:  
}
```

# Настройка формы

С помощью статического метода **showOptionDialog** класса **JOptionPane** выведем запрос на возможность закрытия формы

// Варианты ответа на вопрос

```
Object[] options = {"OOO!! ДААА!!", "НЕЕЕТ!!!"};
```

// В n будет помещен номер выбранной опции

```
int n=JOptionPane.showOptionDialog(evt.getWindow(),  
    "Выйти из программы?", "Выход",  
    JOptionPane.YES_NO_OPTION,  
    JOptionPane.QUESTION_MESSAGE, null, options,  
    options[0]);
```

# Настройка формы

Рассмотрим параметры **showOptionDialog**

1. Первый параметр – форма к которой принадлежит диалог
2. Два параметра: текст вопроса и заголовок диалога
3. Тип вариантов возвращаемого значения и тип диалога (влияет на иконки и звук его появления)
4. Иконка которая будет рядом с вопросом (здесь **null**)
5. Варианты ответов, которые будут размещены на кнопках
6. Вариант выбора по умолчанию

В зависимости от выбора пользователя, n примет соответствующее значение. Его надо проанализировать и выполнить соответствующие действия.

# Настройка формы

При выборе “Да”, программа должна корректно завершиться. Для этого помимо самой формы, необходимо закрыть соединения с БД и все что создано **Hibernate**.

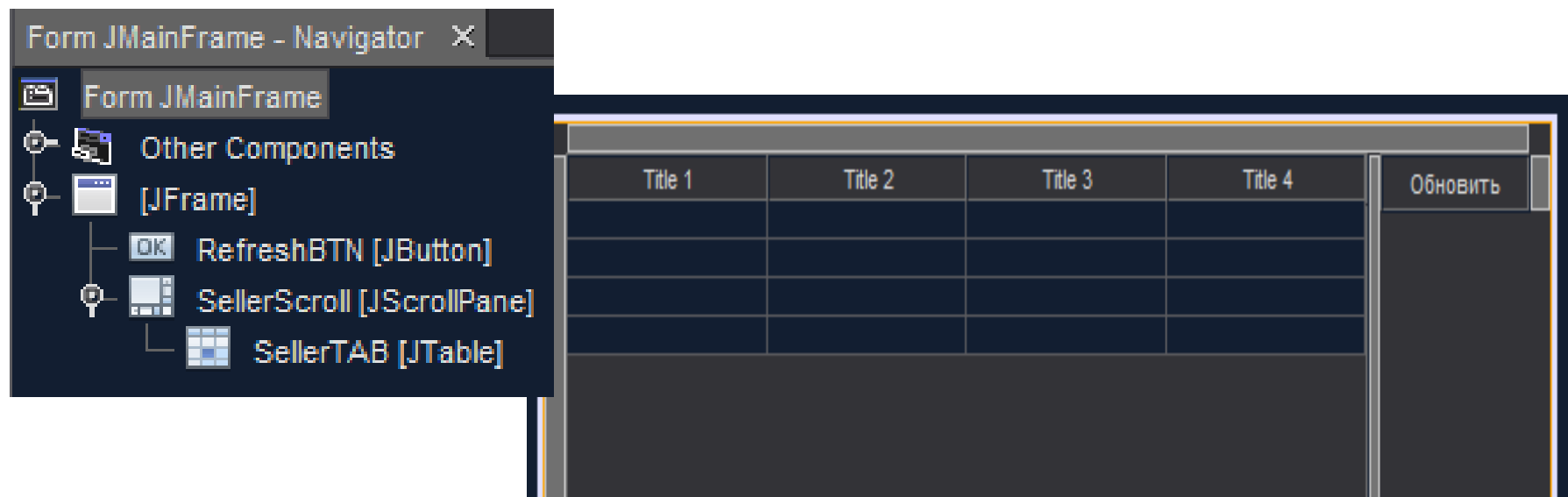
Для этого можно использовать созданный ранее метод **done**.

```
if (n==JOptionPane.YES_OPTION) {  
    // Выставляем действие выти при закрытии  
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
        System.out.print("Closing .. ");  
    // Закрываем сессии и соединения с БД  
        HibernateUtil.done();  
        System.out.println(" done");  
}
```

# Выборка данных

Попробуем вывести содержимое **Seller** в табличном виде. Для этого используем компонент **JTable**.

1. Разместим на форму кнопку (имя – **RefreshBTN**, заголовок – “Обновить”)
2. Разместим компонент **Jtable** (имя – **SellerTAB**)



# Выборка данных

Добавим обработчик нажатия на кнопку **Refresh**. В нем необходимо:

1. Создать экземпляр **DefaultTableModel**, в котором содержаться все данные таблицы (не зависимо отображаются ли они на форме)
2. Настроить колонки таблицы (добавить **Column**)
3. Сделать запрос к БД и получить список объектов **Seller**
4. Внести список в соответствующие столбцы **TableModel**
5. Привязать созданную **TableModel** к **Jtable**
6. Скрыть столбец первичного ключа

***isCellEditable** необходим для запрета редактирования ячеек*



# Выборка данных

1. Создать экземпляр **DefaultTableModel**

```
DefaultTableModel datatab=  
    new DefaultTableModel() {  
// Добавляем событие запрета редактирования  
    @Override  
    public boolean isCellEditable  
        (int row, int column) {  
        return false;  
    }  
};
```

# Выборка данных

## 2. Настроить колонки таблицы (добавить **Column**)

```
datatab.addColumn("id");  
datatab.addColumn("Логин");  
datatab.addColumn("Пароль");  
datatab.addColumn("Имя");  
datatab.addColumn("Класс");
```

## 3. Сделать запрос к БД и получить список объектов **Seller**

```
Query qry=HibernateUtil.getSession().  
    createQuery("FROM Seller");
```

**Сессию получаем как уже созданную из класса *HibernateUtil*!**

# Выборка данных

4. Внести список в соответствующие столбцы **TableModel**

```
for(Seller sl: (ArrayList<Seller>)qry.list())  
{ // Формируем строку  
    Object[] row={ sl,  
        sl.getLogin(), sl.getPassword(),  
        sl.getName(), sl.getClass_() };  
    // Заполняем строку  
    datatab.addRow(row);  
}
```

# Выборка данных

5. Привязать созданную **TableModel** к **JTable**

```
SellerTAB.setModel (datatab) ;
```

6. Скрыть столбец первичного ключа

```
SellerTAB.removeColumn (  
    SellerTAB.getColumnModel () .getColumn (0) ) ;
```

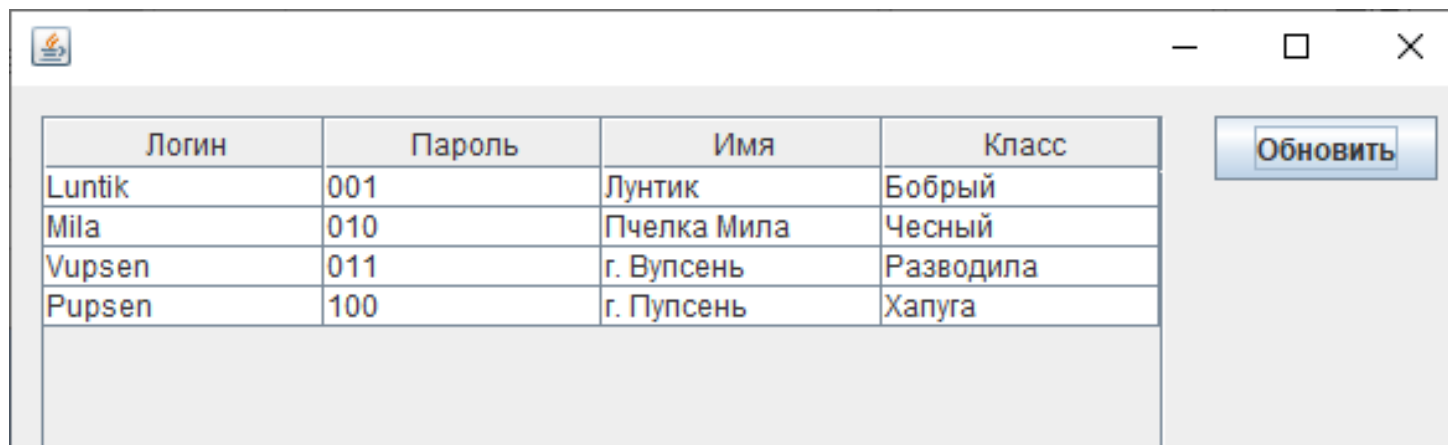
*К невидимому столбцу можно получить доступ через*

***TableModel:***

```
SellerTAB.getModel () .getValueAt (row, column) ;
```

# Выборка данных

В результате пир нажатии на кнопку появляется список продавцов

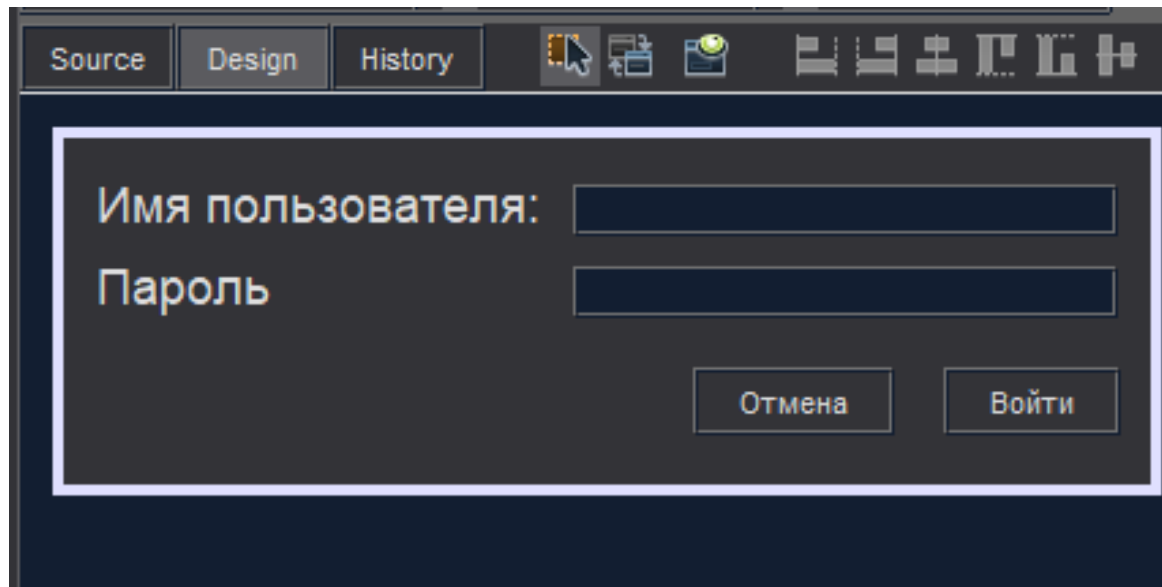
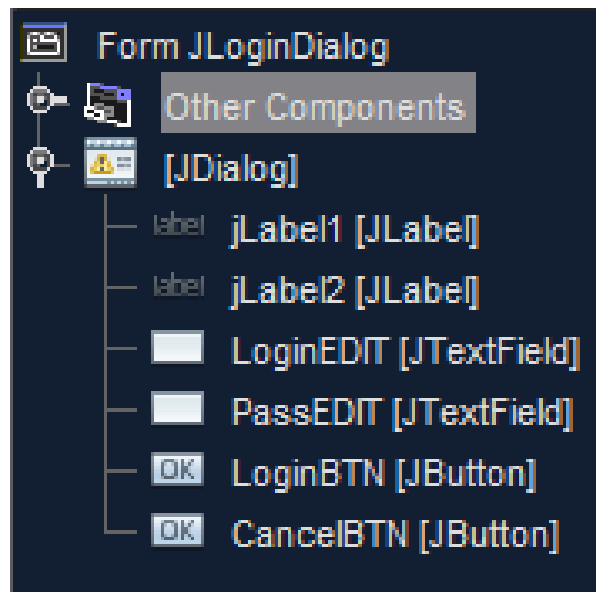


The screenshot shows a software window with a title bar containing a small icon and standard Windows window controls (minimize, maximize, close). Inside the window, there is a table with four columns: 'Логин' (Login), 'Пароль' (Password), 'Имя' (Name), and 'Класс' (Class). The table contains four rows of data. To the right of the table is a button labeled 'Обновить' (Update).

Логин	Пароль	Имя	Класс
Luntik	001	Лунтик	Бобрый
Mila	010	Пчелка Мила	Чесный
Vupsen	011	г. Вупсень	Разводила
Pupsen	100	г. Пупсень	Хапуга

# Выборка данных: Авторизация

Сделаем вход по логину и паролю. Для этого добавим форму запроса и будем вызывать ее перед выводом главной формы.



# Выборка данных: Авторизация

Зададим реакцию на нажатия кнопок диалога. Для этого внутри класса **JLoginDialog** опишем переменную, которая будет хранить результат авторизации.

```
public int Result;
```

Добавим код для кнопки “**Отмена**”. Данный код сообщает, что пользователь не вошел в систему и прячет диалог

```
Result=JOptionPane.NO_OPTION;  
setVisible(false);
```

# Выборка данных: Авторизация

Зададим реакцию на нажатие кнопки “Войти”

```
Criteria cqr=HibernateUtil.getSession().
    createCriteria(User.class);
cqr.add(Restrictions.eq("login",LoginEDIT.getText()));
cqr.add(Restrictions.eq("passwd",PassEDIT.getText()));
ArrayList<User> usr=(ArrayList<User>)cqr.list();
if (usr.size()==0) {    // Логин/пароль не верны
    JOptionPane.showMessageDialog(null,"Фигушки!!!")
    return;
}
// Пользователь авторизовался
Result=JOptionPane.YES_OPTION;
setVisible(false);
```



# Выборка данных: Авторизация

Добавим вызов диалога в главный файл проекта

```
JLoginDialog login=new JLoginDialog(null,true);  
login.setVisible(true);  
if (login.Result==JOptionPane.NO_OPTION) {  
    HibernateUtil.done();  
    System.exit(0);  
}  
login.dispose();
```

*Последняя строка заставляет Garbage Collector (GC) очистить память, занимаемую диалогом.*

# Удаление

Добавим на главную форму кнопку удаления записи. Для удаления необходимо произвести следующие действия :

1. Проверить, есть ли выделенная строка
2. Найти объект, который мы собираемся удалить. Метод **getValueAt** позволяет получить его, зная номер выделенной строки (**getSelectedRow**). При заполнении **TableModel** формируется так, что нулевой столбец указывает на объект.
3. Удалить объект из БД
4. Удалить строку из таблицы

# Удаление

Код, осуществляющий это представлен ниже

**// Пункт 1**

```
if (SellerTAB.getSelectedRow()==-1) return;
```

**// Пункт 2**

```
Seller sl=(Seller)SellerTAB.getModel().  
    getValueAt(SellerTAB.getSelectedRow(), 0);
```

**// Пункт 3**

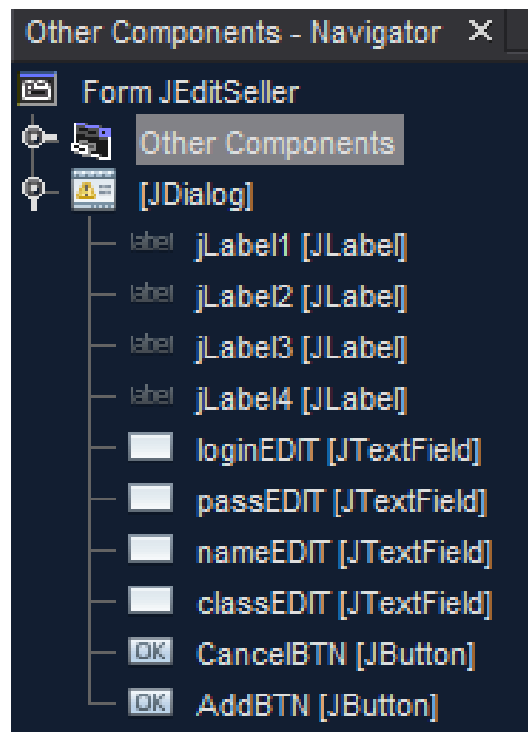
```
HibernateUtil.getSession().beginTransaction();  
HibernateUtil.getSession().delete(sl);  
HibernateUtil.getSession().getTransaction().commit();
```

**// Пункт 4**

```
((DefaultTableModel)SellerTAB.getModel()).  
    removeRow(SellerTAB.getSelectedRow());
```

# Добавление

Добавим на главную форму кнопку добавления записи. Для добавления необходимо создать форму, в которую будут вводиться добавляемые данные.



*Всем полям ввода изменить модификатор на **public** в code!!!*

# Добавление

На обработчик кнопки отмена, необходимо просто закрыть форму. По кнопке добавить, необходимо создать новый объект и сохранить его в БД.

**// Создаем продавца**

```
Seller sl=new Seller(loginEDIT.getText(),  
    passEDIT.getText(),  
    nameEDIT.getText(),classEDIT.getText());
```

**// Получаем сессию**

```
Session sess=HibernateUtil.getSession();  
sess.beginTransaction();
```

**// Добавляем**

```
sess.save(sl);  
sess.getTransaction().commit();
```

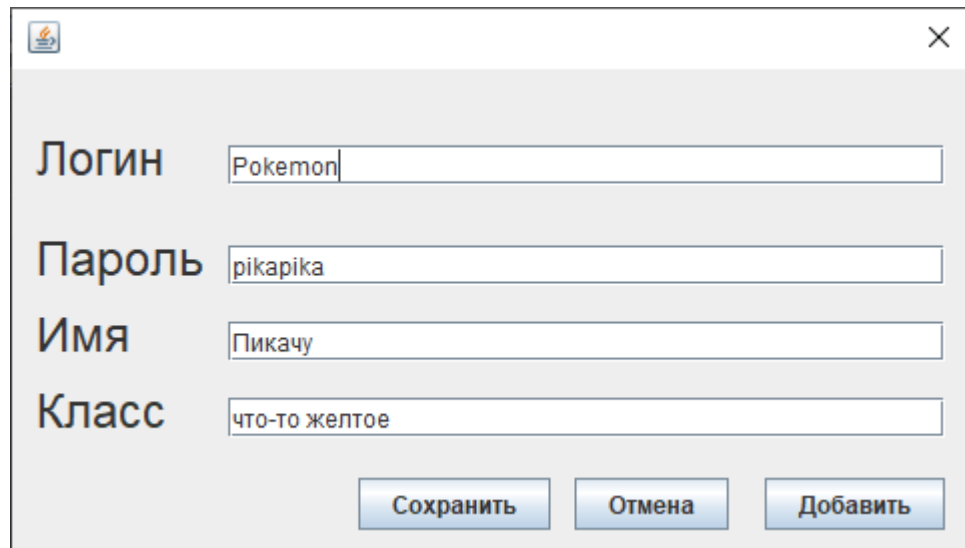
```
setVisible(false);          // Закрываем форму
```

# Добавление

Для вызова формы добавления на кнопку **добавить** запишем следующий код

**// Создаем форму и отображаем ее**

```
JEditSeller AddSeller=new JEditSeller(this,true);  
AddSeller.setVisible(true);
```



Логин

Пароль

Имя

Класс

# Редактирование

Для возможности редактирования записи необходимо:

1. Получить текущий объект
2. Открыть форму редактирования
3. Записать значения полей объекта в соответствующие окна редактирования
4. Для сохранения необходимо скопировать данные из полей редактирования в поля текущего объекта
5. Сделать **commit()**

Для реализации этого добавим в наш диалог **JSellerEdit** еще одно поле:

```
public Seller selledit;
```

# Редактирование

Для вызова диалога напомним следующий код:

**// Получаем текущий объект**

```
Seller sl=(Seller)SellerTAB.getModel().  
        getValueAt(SellerTAB.getSelectedRow(),0);
```

**// Создаем диалог редактирования**

```
JEditSeller EditSeller=new JEditSeller(this,true);
```

**// Задаем параметры**

```
EditSeller.loginEDIT.setText(sl.getLogin());  
EditSeller.passEDIT.setText(sl.getPasswd());  
EditSeller.nameEDIT.setText(sl.getName());  
EditSeller.classEDIT.setText(sl.getClass_());  
EditSeller.selledit=sl;
```

**// Открываем его**

```
EditSeller.setVisible(true);
```



# Редактирование

Для сохранения результата добавим код на кнопку **Сохранение**

**// Получаем сессию и открываем транзакцию**

```
Session sess=HibernateUtil.getSession();
```

```
sess.beginTransaction();
```

**// Меняем значения на текущие**

```
selledit.setLogin(loginEDIT.getText());
```

```
selledit.setPasswd(passEDIT.getText());
```

```
selledit.setName(nameEDIT.getText());
```

```
selledit.setClass_(classEDIT.getText());
```

**// Сохраняемся и закрываем форму**

```
sess.getTransaction().commit();
```

```
setVisible(false);
```

# Задание

По аналогии сделать формы для просмотра и редактирования всех таблиц.