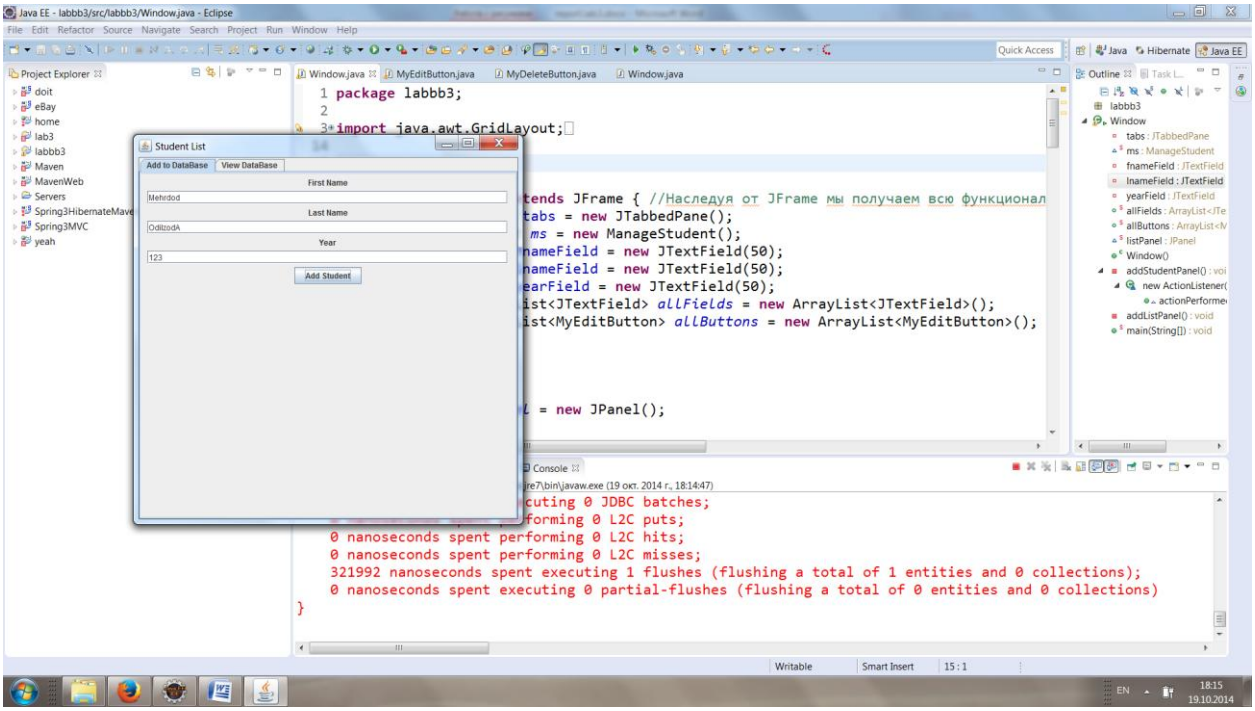
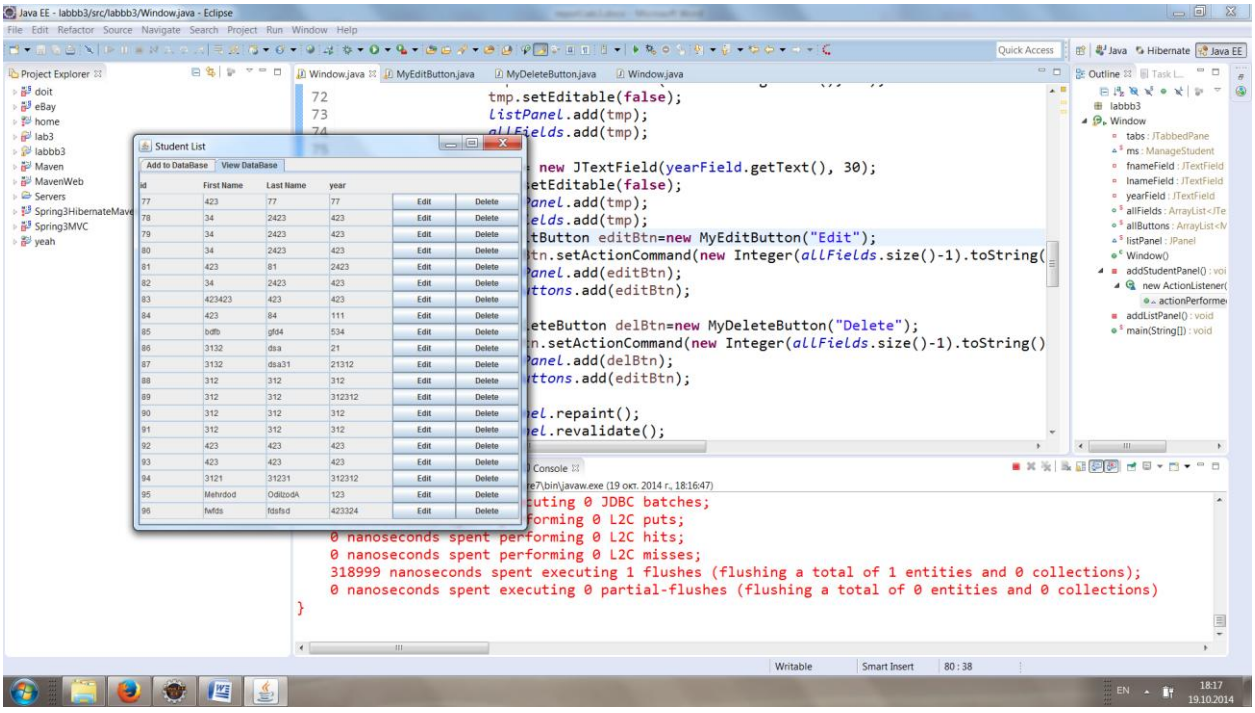
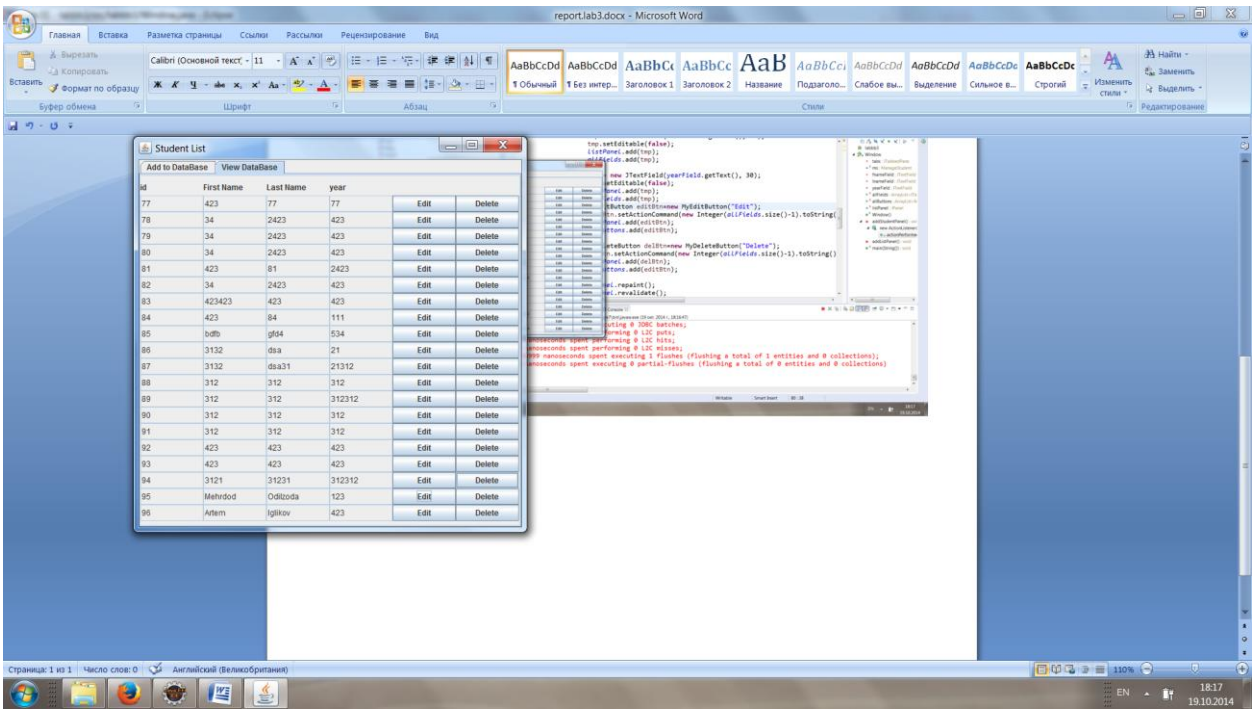
Мехрдод Одилзода. Тут некоторые скрины третей лабы







Расскажу как это работает:

Дизайн: использовал 2 таба, один для добавления другой для просмотра списка студентов. Использовал JTabbedPane.

1) Таб добавления: использовал JLabel, JTextField и один Jbutton. Использовал FlowLayout для JPanel.
2) Таб просмотров: тоже самое, но использовал GridLayout для JPanel.

Button Add: при клике вызывается ф-я actionPerformed. Основная часть кода это класс Manage Student, который работает с Hibernate-ом.(тут в основном копи паст с ссылки, которая в условии лабы есть) + настраиваем xml.

Button Edit and Delete: Так как это button-ы мы добавлем когда запускаем прогу и когда делаем Add Student, т.е. в двух местах, чтобы не было одинакового кода я создал 2 класса MyDeleteButton и MyEditButton. Они наследуются от JButton и имеют собственные `actionPerformed`. Чтобы удалять или редактировать нужные JTextField пришлось завести массив. А находим по какой кнопочке тыкали с помощью .getSource().

Расскажу про копипаст со ссылки с лабы: Как работает Hibernate. Hibernate.cfg мы устанавливаем связь с MySql и указываем xml файл в котором описан объект

транзакций(Student). В Student.hbm.XML мы описываем объект Student(Hibernate умеет делать автоматически).

ManageStudent, класс которым мы управляем БД. В конструкторе у нас создается sessionfactory.

addStudent. Основное код

```
tx = session.beginTransaction();
Student Student = new Student(fname, lname, year);
 StudentID = (Integer) session.save(Student);
 tx.commit();
```

List<Student> listStudents( ) Основной код

```
Session session = factory.openSession();
      Transaction tx = null;
      try{
         tx = session.beginTransaction();
         List<Student> students = session.createQuery("FROM
Student").list();
         tx.commit();
         return students;
```

Для других ф-ий типа Update, delete логика и код почти такой же.