Частное учреждение образования
Колледж бизнеса и права

УТВЕРЖДАЮ
Заведующий
методическим кабинетом
_____Е.В. Паскал
«___»_____2021

Специальность: 2-40 01 01 «Программное обеспечение информационных технологий»	Дисциплина: «Основы кроссплатформенного программирования»

ЛАБОРАТОРНАЯ РАБОТА № 17
Инструкционно-технологическая карта

Тема: «Работа с файлами, JList, JScrollPane, HashMap. Создание запускного jar-файла.»

Цель:

Время выполнения: 2 часа

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

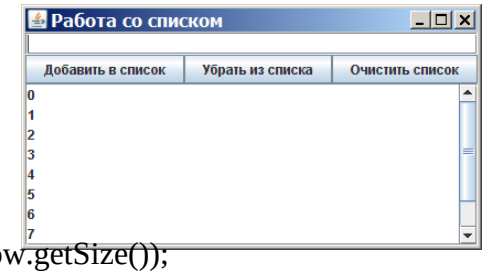
1. Изучить теоретические сведения;
2. Ответить на контрольные вопросы;
3. Откомпилировать примеры программ из раздела «Теоретические сведения»;
4. Выполнить ИДЗ.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ ДЛЯ ВЫПОЛНЕНИЯ РАБОТЫ

Компонент JList в Java используется для отображения данных в виде списка. При этом используется парадигма Модель-Вид-Контроллер. Она позволяет не сваливать весь код приложения в кучу, а разделять его на три большие блока. Модель – обработка данных и всё, что с ними связано. Вид – внешность приложения. Определяет то, как будет выглядеть приложение, и что будет отображаться пользователю. Вид показывает данные, которые ему предоставляет модель. Контроллер – обработка всего, что приходит от пользователя.

Для списка работа этой парадигмы выглядит следующим образом: за Модель, т.е. обработку данных, отвечает класс DefaultListModel, за Вид – класс JList, а в качестве Контроллера могут выступать слушатели. Например, напишем программу, в которой создается список с числами от 0

до 9, в этот список можно добавлять и удалять элементы, при этом сам список расположим на панели с прокруткой (JScrollPane):



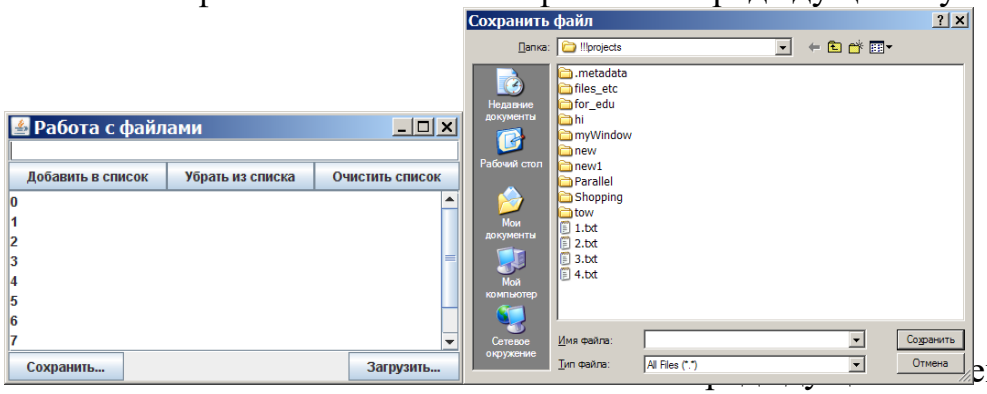

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class ListWork extends JFrame{
public static void main(String[] args) {
ListWork window= new ListWork("Работа со списком"); window.setVisible(true);
window.pack(); window.setMinimumSize(window.getSize());
}
public ListWork(String s){
super(s); //задаем название окна
final DefaultListModel myListModel = new DefaultListModel(); //создаем модель для нашего
for (int i=0;i<10;i++){//списка и заполняем ее элементами, модификатор final делает
myListModel.addElement(""+i); //объект доступным для вложенных слушателей }
final JList myList=new JList(); //создаем объект, отвечающий за вид нашего списка JScrollPane myScroll = new JScrollPane(myList); //создаем панель с прокруткой myList.setModel(myListModel); //задаем для списка созданную модель
Box myBox1=new Box(BoxLayout.Y_AXIS);//создаем компоновку
final JTextField myText=new JTextField();//текстовое поле для ввода значений элементов myBox1.add(myText); //добавляем поле на компоновку
Box box1=new Box(BoxLayout.X_AXIS);//создаем еще одну компоновку JButton button1=new JButton("Добавить в список"); //создаем кнопку box1.add(button1); //добавляем кнопку на компоновку
button1.addActionListener(new ActionListener() {//создаем слушатель,
public void actionPerformed(ActionEvent e) {//отвечающий за добавление myListModel.addElement(myText.getText()); //элемента списка
}
});
JButton button2=new JButton("Убрать из списка"); //создаем кнопку button2.addActionListener(new ActionListener() {//создаем слушатель, отвечающий
public void actionPerformed(ActionEvent e) {//за удаление элементов списка while (myListModel.contains(myText.getText())){ //равных значению
myListModel.removeElement(myText.getText()); //в текстовой строке }
}
});
box1.add(button2); //добавляем кнопку на компоновку
JButton buttonClear=new JButton("Очистить список");//еще кнопка для очистки списка buttonClear.addActionListener(new ActionListener() {//к ней слушатель
@Override
public void actionPerformed(ActionEvent e) { myListModel.clear(); //очищаем список
}
});
box1.add(buttonClear); //добавляем кнопку на компоновку
myBox1.add(box1); //вставляем компоновку в другую компоновку
add(myScroll,BorderLayout.CENTER);//добавляем панель с прокруткой в центр
```

окна add(myBox1,BorderLayout.NORTH); //добавляем компоновку в верхнюю часть окна
}

Теперь добавим возможность сохранения в файл и загрузки из файла нашего списка.

Для работы с файлами в языке Java используется несколько классов. Стратегия работы с текстовыми файлами такая:

- создаем объект для записи в файл или чтения из файла (класс FileWriter или FileReader), соединяя его с нужным файлом;
- создаем объект для потока буферизации записи или чтения (класс BufferedWriter или BufferedReader), связывая его с объектом для записи в файл или чтения из файла из предыдущего пункта;



- ...екта;
- все вышеописанные пункты помещаем в конструкцию:
  try {
  //наши действия с файлами }
  catch (IOException e1) { e1.printStackTrace();
  }

Данная конструкция позволяет безопасно использовать код внутри try, а все возникающие ошибки (исключения) обрабатывать в секции catch, а не выдавать на обработку операционной системе.

Загрузка и сохранение списка в файл. Добавим следующий код к предыдущему примеру

Box myBox2=new Box(BoxLayout.X_AXIS); //новая компоновка

JButton button3=new JButton("Сохранить..."));//кнопка для сохранения списка myBox2.add(button3);

final FileDialog fdlg=new FileDialog(this, "");//создаем диалоговое окно для чтения и записи //файла

button3.addActionListener(new ActionListener() { //слушатель для сохранения public void actionPerformed(ActionEvent e) {

fdlg.setMode(FileDialog.SAVE); //делаем созданный диалог диалогом сохранения

fdlg.setTitle("Сохранить файл"); //задаем ему заголовок

fdlg.setVisible(true); //делаем видимым

FileWriter myWriter = null;//создаем объект типа FileWriter и приравниваем его к null try { //секция, в которой можно выполнять небезопасные действия созданному объекту типа FileWriter задаем новый объект с параметрами каталога и файла, //выбранного пользователем в диалоге сохранения файла

myWriter=new FileWriter(fdlg.getDirectory()+fdlg.getFile());

//создаем объект типа BufferedWriter, соединяя его с созданным объектом myWriter BufferedWriter myBWriter=new BufferedWriter(myWriter);

```
        for(int i=0;i<myListModel.getSize();i++){   //в цикле сохраняем каждый элемент в файл
        myBWriter.write(""+myListModel.getElementAt(i));//используя
        BufferedWriter myBWriter.newLine(); //и вставляем символ перехода на новую строку
        }
        myBWriter.close();//закрываем все соединения myWriter.close();
        } catch (IOException e1) {
        e1.printStackTrace(); //если произойдет ошибка, будет выведено сообщение }
        }
}
)
;
        myBox2.add(Box.createHorizontalGlue()); //вставляем «пружину», чтобы кнопки были по краям окна
        JButton button4=new JButton("Загрузить...");//кнопка для загрузки списка из файла
        button4.addActionListener(new ActionListener() { //слушатель для загрузки из файла
        public void actionPerformed(ActionEvent e) {
        fdlg.setMode(FileDialog.LOAD); //делаем созданный диалог диалогом загрузки
        fdlg.setTitle("Загрузить файл");//задаем ему заголовок
        fdlg.setVisible(true); //делаем видимым
        FileReader myReader = null;
        try { //секция, в которой можно выполнять небезопасные действия
        // созданному объекту типа FileReader задаем новый объект с параметрами каталога и файла, //выбранного пользователем в диалоге загрузки файла
        myReader=new FileReader(fdlg.getDirectory()+fdlg.getFile());
        myListModel.clear(); //очищаем список, т.к. в него будут помещены новые данные
        //создаем объект типа BufferedReader, соединяя его с созданным объектом
        myReader BufferedReader myBReader=new BufferedReader(myReader);
        String s; //строка для временного хранения данных
        //в s записываем строку из файла, и если она не пустая (а пустой она будет, если файл закончился //или пустой), то добавляем в список новый элемент с параметром s
        while ((s=myBReader.readLine())!=null){ myListModel.addElement(s);
        }
        myBReader.close();//закрываем все соединения myReader.close();
        } catch (IOException e1) { e1.printStackTrace();
        }
 }
        });
        myBox2.add(button4);//добавляем кнопку на компоновку
        add(myBox2,BorderLayout.SOUTH);//вставляем компоновку в нижнюю область окна
```

Можно добавить к проекту загрузку рисунков из файлов, связанных с названиями элементов списка. Самый простой способ – когда имя файла совпадает с названием элемента списка. Для этого нужно добавить панель, на которую можно будет выводить изображения. Примерный код, который нужно добавить к нашему проекту, может быть следующим:

```
JPanel panel1=null;
private static Image image;
static File f1;
```

Эта часть вставляется перед

public static void main(String[] args) {

в ней описываются панель для изображений, объект типа Image для хранения изображения и объект типа File для доступа к

файлу Для списка добавляем слушателя:

myList.addListSelectionListener(new ListSelectionListener() {//слушатель выделения элемента списка public void valueChanged(ListSelectionEvent e) {

JList tempList=(JList)e.getSource();//создаем временный объект, равный объекту, который //вызвал слушателя во время выполнения программы

f1=new File(tempList.getSelectedValue().toString()+".jpg"); //файл f1 связываем с //именем выбранного элемента списка и с расширением jpg (если не задан // каталог - значит файл находится в текущем каталоге проекта), таким

//образом, создаем связь с физическим файлом в каталоге проекта try { //секция для небезопасных операций с файлами

image=ImageIO.read(f1); //объект типа Image связываем с файлом

loadImage(image); //вызываем функцию для загрузки изображения на панель

} catch (IOException e1) {//в случае ошибки

showDialog(); //вызываем функцию с сообщением об ошибке

panel1.repaint(); //перерисовываем панель

}

}

});

Создаем панель для вывода изображения и меняем компоновку центральной части окна:

panel1=new JPanel();

Box centerBox=new Box(BoxLayout.X_AXIS); //создаем компоновку

centerBox.add(myScroll); //вставляем в компоновку панель с прокруткой

centerBox.add(panel1); //и панель для изображений

add(centerBox,BorderLayout.CENTER); //добавляем компоновку в центральную область окна

В конце класса описываем новые функции:

public void loadImage(Image im){ //функция для загрузки изображения на панель

Graphics2D g = (Graphics2D)panel1.getGraphics(); //получаем графический контекст панели g.drawImage(im, 0, 0, null); //и выводим на него изображение

}

public void showDialog(){ //функция с сообщением об ошибке

JDialog myDialog=new JDialog(); //создаем окно для вывода сообщения

myDialog.setModal(true); //делаем его модальным, т.е. пока не закрыть его, основное окно будет недоступным

myDialog.add(new JLabel("No file!!!")); //вставляем в окно метку с текстом

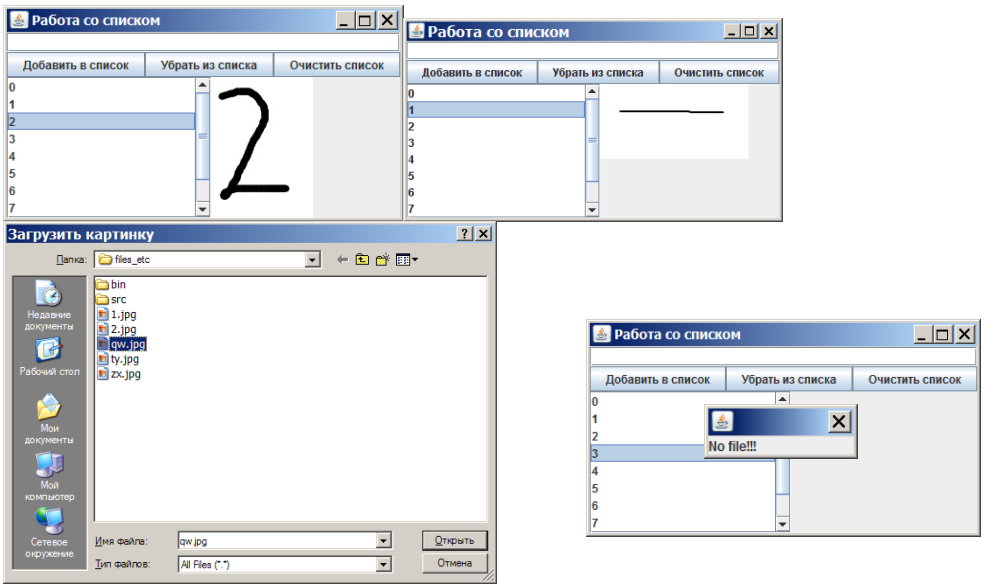
myDialog.pack(); //упаковываем окно по размерам выведенного текста //задаем расположение окна – посередине основного окна

myDialog.setLocation(getLocation().x+getWidth()/2, getLocation().y+getHeight()/2);

myDialog.setVisible(true);//делаем окно видимым

}

Таким образом, в каталоге с проектом должны быть файлы с названиями 1.jpg, 2.jpg и т.д., если файла для какого-либо числа из списка не будет, то появится окно с сообщением.



Список, связанный с графическими файлами.

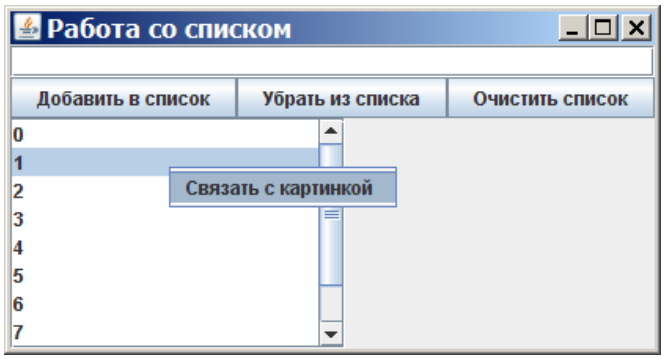
Но описанный выше способ подойдет, если название файла совпадает с названием элемента списка. А если нам нужно связать с элементом списка произвольную картинку, то необходимо где-то хранить эту связь. Для этих целей можно использовать класс HashMap. Этот класс представляет собой хранилище связанных пар <ключ, значение>. Т.е. по ключу можно найти нужное значение. Например, создадим хранилище связанных пар <Фамилия, Номер телефона>:

```
import java.util.HashMap;
public class SampleHash {
public static void main(String[] args) {
//создадим объект типа HashMap, в котором оба значения будут
строкового типа
HashMap<String, String> myHash=new HashMap<String, String>();
myHash.put("Иванов", "891234567");   //записываем телефон Иванова
myHash.put("Петров", "892233345");   //записываем телефон Петрова
myHash.put("Сидоров", "893137567"); //записываем телефон Сидорова
System.out.println(myHash); //выводим на экран все записи
System.out.println(myHash.get("Иванов")); //выводим значение, связанное
с Ивановым
}
}
```

При запуске получим:

{Иванов=891234567, Петров=892233345, Сидоров=893137567}
891234567

Теперь добавим к ранее созданному проекту возможность связывания произвольных изображений с нашим списком, реализовав это при помощи контекстного меню (класс JPopupMenu) и HashMap



Связывание произвольных изображений с элементами списка.

Добавим 2 новых члена класса

private static HashMap<String, String> myHash=new HashMap<String, String>(); //наш HashMap static JPopupMenu myPopup; //контекстное меню

Добавим создание контекстного меню и его обработку

myPopup=new JPopupMenu(); //создаем контекстное меню

JMenuItem myItem1=new JMenuItem("Связать с картинкой"); //создаем пункт контекстного меню

myItem1.addActionListener(new ActionListener() { //добавляем к нему слушателя

public void actionPerformed(ActionEvent e) {

loadFromFile(myList.getSelectedValue().toString()); //вызываем функцию, которую создадим ниже, //ей передаем название выбранного пункта списка

}

}

}

;

myPopup.add(myItem1); //добавляем созданный пункт к контекстному меню

myList.setComponentPopupMenu(myPopup); //устанавливаем созданное контекстное меню для списка

myList.addMouseListener(new MouseAdapter() { //добавляем к списку слушатель событий мыши

public void mousePressed(MouseEvent e) { //обрабатываем нажатие на клавишу мыши

myList.setSelectedIndex(myList.locationToIndex(e.getPoint())); //устанавливаем текущим //элементом списка тот, который ближе к позиции курсора мыши в момент нажатия кнопки мыши

}

}

;

Упростим

myList.addListSelectionListener: myList.addListSelectionListener(new ListSelectionListener() {

public void valueChanged(ListSelectionEvent e) {

//запишем в переменную path значение, связанное в myHash с текущим выделенным элементом списка String

path=myHash.get(myList.getSelectedValue().toString());

loadImage(path); //вызов loadImage с путем к графическому файлу }

});

Перепишем метод loadImage:

public void loadImage(String path){ try {

```
if (path!=null) { //если переменная не нулевая
f1=new File(path); //создаем файл
image=ImageIO.read(f1); //считываем из него изображение
Graphics2D g = (Graphics2D)panel1.getGraphics(); //получаем графический контекст панели
g.setColor(panel1.getBackground()); //устанавливаем цвет текущим цветом фона панели
g.clearRect(0, 0, panel1.getWidth(), panel1.getHeight()); //и закрашиваем панель, чтобы//стереть предыдущее изображение
g.drawImage(image, 0, 0, null); //выводим картинку на панель
} else throw new IOException(); //иначе кидаем исключение
catch (IOException e1) { // и обрабатываем его,
panel1.repaint(); //перерисовывая панель }
}
```

Добавим новую функцию loadFromFile, которая принимает в качестве параметра строку с текстом выбранного элемента списка, вызывает диалог открытия файла и соединяет через myHash элемент списка с выбранным пользователем графическим файлом:

```
public void loadFromFile(String s){
FileDialog fdlg=new FileDialog(this, "Загрузить картинку",FileDialog.LOAD);
fdlg.setFile("*.jpg"); //задаем видимость только файлам с расширением *.jpg
fdlg.setVisible(true);
myHash.put(s, fdlg.getDirectory()+fdlg.getFile()); //записываем в myHash текст выбранного //элемента списка и путь к выбранному файлу
loadImage(fdlg.getDirectory()+fdlg.getFile());//показываем выбранное изображение
}
```

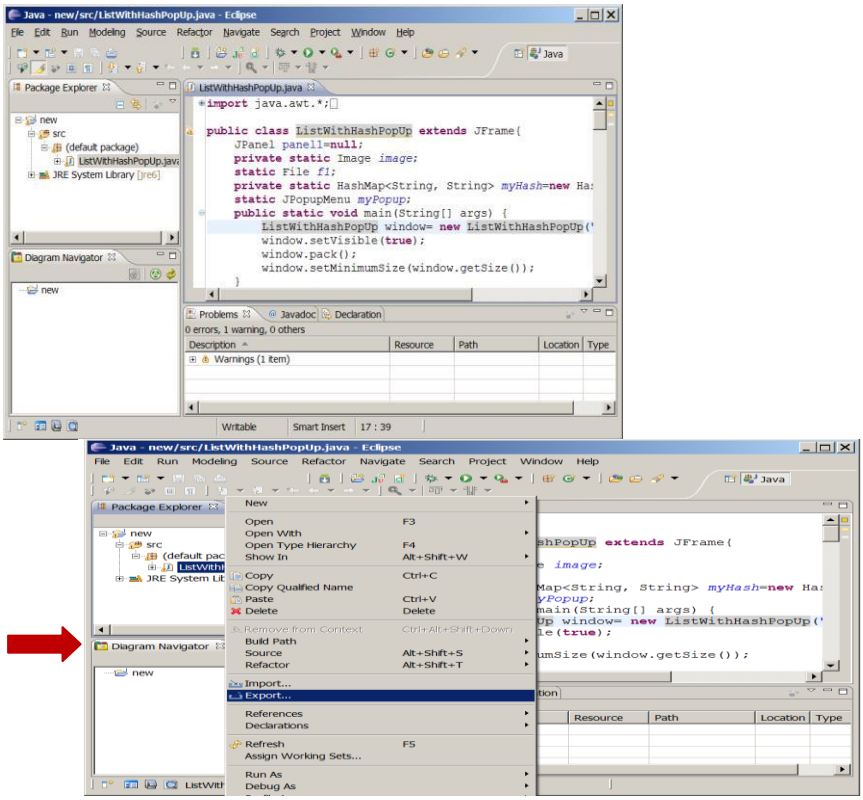
Полностью новый класс будет выглядеть следующим образом:

```
import java.awt.*;
import java.awt.event.*; import java.io.*;
import java.util.HashMap;
import javax.imageio.ImageIO; import javax.swing.*;
import javax.swing.event.*;
public class ListWithHashPopUp extends JFrame{ JPanel panel1=null;
private static Image image; static File f1;
private static HashMap<String, String> myHash=new HashMap<String, String>();
static JPopupMenu myPopup;
public static void main(String[] args) {
ListWithHashPopUp window= new ListWithHashPopUp("Работа со списком");
window.setVisible(true);
window.pack(); window.setMinimumSize(window.getSize());
}
public ListWithHashPopUp(String s){ super(s);
final DefaultListModel myListModel = new DefaultListModel(); for (int i=0;i<10;i++){
myListModel.addElement(""+i); }
final JList myList=new JList();
JScrollPane myScroll = new JScrollPane(myList); myList.setModel(myListModel);
myPopup=new JPopupMenu();
JMenuItem myItem1=new JMenuItem("Связать с картинкой");
myItem1.addActionListener(new ActionListener() {
```

```
        public void actionPerformed(ActionEvent e) {
        loadFromFile(myList.getSelectedValue().toString());
        } });
        myPopup.add(myItem1);
        myList.setComponentPopupMenu(myPopup);
        myList.addMouseListener(new MouseAdapter() { public void
        mousePressed(MouseEvent e) {
        myList.setSelectedIndex(myList.locationToIndex(e.getPoint())); }
        });
        myList.addListSelectionListener(new ListSelectionListener() { public void
        valueChanged(ListSelectionEvent e) {
        String path=myHash.get(myList.getSelectedValue().toString());
        loadImage(path);
        } });
        Box myBox1=new Box(BoxLayout.Y_AXIS);
        final JTextField myText=new JTextField(); myBox1.add(myText);
        Box box1=new Box(BoxLayout.X_AXIS);
        JButton button1=new JButton("Добавить в список"); box1.add(button1);
        button1.addActionListener(new ActionListener() { @Override
        public void actionPerformed(ActionEvent e) {
        myListModel.addElement(myText.getText());
        } });
        JButton button2=new JButton("Убрать из списка");
        button2.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
        while (myListModel.contains(myText.getText())){
        myListModel.removeElement(myText.getText());
        } }
        }); box1.add(button2);
        JButton buttonClear=new JButton("Очистить список");
        buttonClear.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
        myListModel.clear();
        } });
        box1.add(buttonClear); myBox1.add(box1);
        panel1=new JPanel();
        Box centerBox=new Box(BoxLayout.X_AXIS); centerBox.add(myScroll);
        centerBox.add(panel1); add(centerBox,BorderLayout.CENTER);
        add(myBox1,BorderLayout.NORTH);
        }
        public void loadImage(String path){ try {
        if (path!=null) {
        f1=new File(path); image=ImageIO.read(f1);
        Graphics2D g = (Graphics2D)panel1.getGraphics();
        g.setColor(panel1.getBackground());
        g.clearRect(0, 0, panel1.getWidth(), panel1.getHeight()); g.drawImage(image, 0, 0,
null);
        }
```

```
else throw new IOException(); } catch (IOException e1) {
panel1.repaint(); }
}
public void loadFromFile(String s){
FileDialog fdlg=new FileDialog(this, "Загрузить картинку",FileDialog.LOAD);
fdlg.setFile("*.jpg");
fdlg.setVisible(true);
myHash.put(s, fdlg.getDirectory()+fdlg.getFile());
loadImage(fdlg.getDirectory()+fdlg.getFile());
}

}
```



До этого мы запускали наши проекты только в среде Eclipse, но для распространения программы нужно, чтобы она запускалась и вне среды создания Java классов. Для этого нужно создать запускной jar-файл (аналог exe-файла). Сделаем это на примере последнего проекта:

После нажатия кнопки Finish может появиться окно, в котором сказано, что экспорт завершился с предупреждениями. Нажимаем ОК и запускной файл готов.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что представляет пространственная модель box (блок)?
2. Какие основные свойства у блока позиционирования?
3. В чем заключается фиксированное позиционирование?

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Задание 1.

Переделать пример со списком, чтобы в нем хранился список группы, а выбрав студента, в другом поле появлялись его данные (возраст, адрес). Обязательно использовать HashMap, запись в файл и чтение из файла.

Задание 2.

Создать запускные jar-файлы из своих лабораторных, начиная с 14-ой.

1. ДОМАШНЕЕ ЗАДАНИЕ

Индивидуальное задание

2. ЛИТЕРАТУРА

Альфред В., Ахо Компиляторы. Принципы, технологии и инструментарий, Вильямс, 2015.

Преподаватель А.С.Кибисова

Рассмотрено на заседании цикловой комиссии программного обеспечения информационных технологий
Протокол № ____ от «___» _____ 2021
Председатель ЦК _____В.Ю.Михалевич