**ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**"ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ"**

Лабораторная работа № 1

Тема: «ПОСТОЯННЫЕ ТАБЛИЦЫ»

Проверил: Выполнил:

асс. каф. ПИ ст. гр. ПИ-18б

Бакаленко В.С. Моргунов А. Г.

\_\_\_\_.\_\_\_\_.2019г. \_\_\_\_.\_\_\_\_.2019г.

асс. каф. ПИ

Серёженко О.А.

\_\_\_\_.\_\_\_\_.2019г.

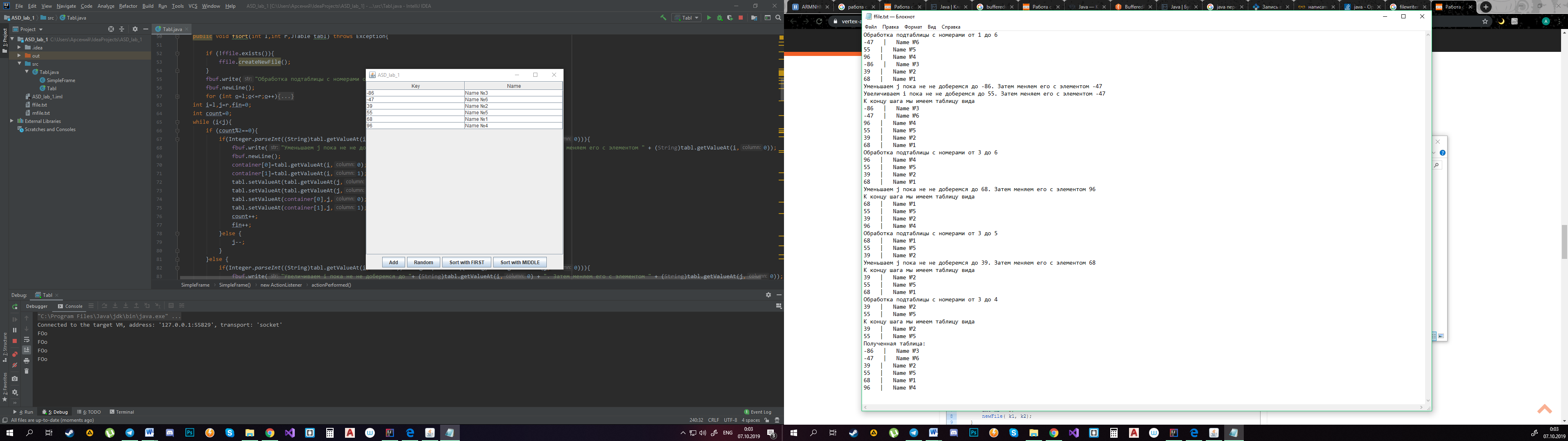
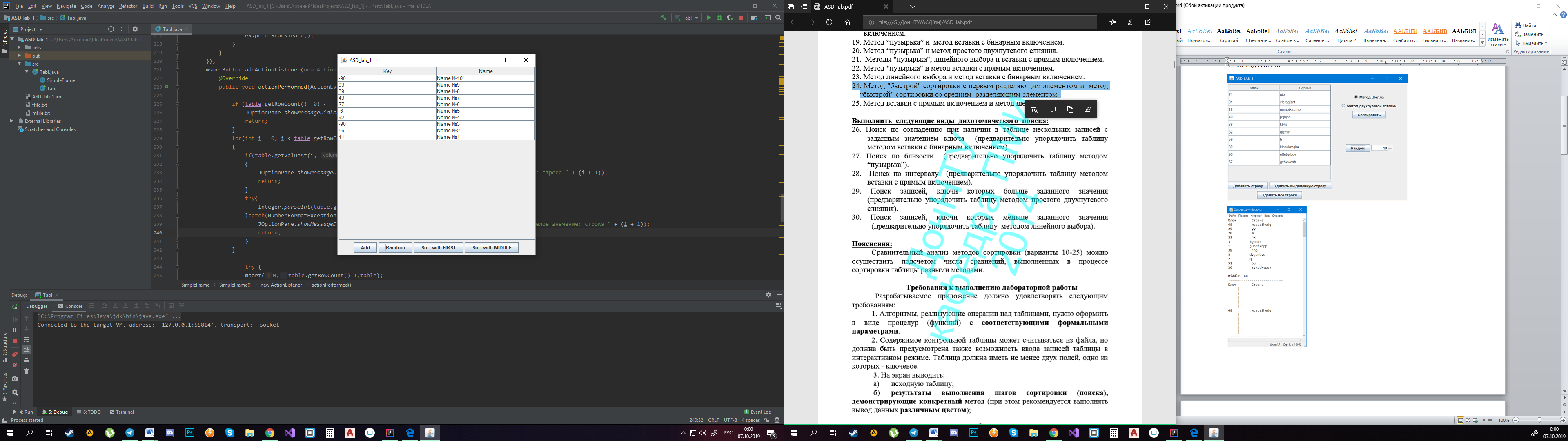
Донецк – 2019

ЗАДАНИЕ

Разработать приложение, содержащее процедуры, реализующие основные операции над таблицами: формирование таблицы, сортировка или поиск (согласно указанному варианту), вывод таблицы.

Упорядочить таблицу, используя:

24. Метод "быстрой" сортировки с первым разделяющим элементом и метод “быстрой” сортировки со средним разделяющим элементом.



*/\*\*  
 \* Java lab 1 asd  
 \*  
 \** ***@author*** *Арсений  
 \*/*import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.util.Arrays;  
import java.io.\*;  
import javax.swing.\*;  
import javax.swing.table.DefaultTableModel;  
import java.lang.Exception;  
  
  
public class Tabl {  
 public static void main(String[] args) throws Exception{  
 EventQueue.*invokeLater*(() ->  
 {  
 SimpleFrame frame = null;  
 try {  
 frame = new SimpleFrame();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 frame.setVisible(true);  
 frame.setTitle("ASD\_lab\_1");  
 });  
 }  
}  
  
  
class SimpleFrame extends JFrame{  
 private static final int *DEFAULT\_WIDTH* = 1000;  
 private static final int *DEFAULT\_HEIGHT* = 1000;  
 private DefaultTableModel tableModel;  
 private Object[] columnsHeader = new String[] {"Key","Name"};  
 private Object[] container = new Object[2];  
 private int arr[]=new int[100];  
 File ffile = new File("ffile.txt");  
 FileWriter ffilew = new FileWriter(ffile);  
 BufferedWriter fbuf = new BufferedWriter(ffilew);  
 File mfile = new File("mfile.txt");  
 FileWriter mfilew = new FileWriter(mfile);  
 BufferedWriter mbuf = new BufferedWriter(mfilew);  
  
  
 public void fsort(int l,int r,JTable tabl) throws Exception{  
  
 if (!ffile.exists()){  
 ffile.createNewFile();  
 }  
 fbuf.write("Обработка подтаблицы с номерами от "+(l+1)+" до "+(r+1));  
 fbuf.newLine();  
 for (int o=l;o<=r;o++){  
  
 fbuf.write((String)tabl.getValueAt(o,0)+" | "+(String)tabl.getValueAt(o,1));  
 fbuf.newLine();  
  
 }  
 int i=l,j=r,fin=0;  
 int count=0;  
 while (i<j){  
 if (count%2==0){  
 if(Integer.*parseInt*((String)tabl.getValueAt(i,0))>Integer.*parseInt*((String) tabl.getValueAt(j,0))){  
 fbuf.write("Уменьшаем j пока не не доберемся до "+ (String)tabl.getValueAt(j,0) + ". Затем меняем его с элементом " + (String)tabl.getValueAt(i,0));  
 fbuf.newLine();  
 container[0]=tabl.getValueAt(i,0);  
 container[1]=tabl.getValueAt(i,1);  
 tabl.setValueAt(tabl.getValueAt(j,0),i,0);  
 tabl.setValueAt(tabl.getValueAt(j,1),i,1);  
 tabl.setValueAt(container[0],j,0);  
 tabl.setValueAt(container[1],j,1);  
 count++;  
 fin++;  
 }else {  
 j--;  
 }  
 }else {  
 if(Integer.*parseInt*((String)tabl.getValueAt(i,0))>Integer.*parseInt*((String) tabl.getValueAt(j,0))){  
 fbuf.write("Увеличиваем i пока не не доберемся до "+ (String)tabl.getValueAt(i,0) + ". Затем меняем его с элементом " + (String)tabl.getValueAt(j,0));  
 fbuf.newLine();  
 container[0]=tabl.getValueAt(i,0);  
 container[1]=tabl.getValueAt(i,1);  
 tabl.setValueAt(tabl.getValueAt(j,0),i,0);  
 tabl.setValueAt(tabl.getValueAt(j,1),i,1);  
 tabl.setValueAt(container[0],j,0);  
 tabl.setValueAt(container[1],j,1);  
 count++;  
 fin++;  
 }else {  
 i++;  
 }  
 }  
 }//while  
 fbuf.write("К концу шага мы имеем таблицу вида");  
 fbuf.newLine();  
 for (int o=l;o<=r;o++){  
 fbuf.write((String)tabl.getValueAt(o,0)+" | "+(String)tabl.getValueAt(o,1));  
 fbuf.newLine();  
  
 }  
 if(fin>0){  
 if ((i-1)>l){  
 fsort(l,i-1,tabl);  
 }  
 if ((i+1)<r){  
 fsort(i+1,r,tabl);  
 }  
 }  
 System.*out*.println("FOo");  
  
}//fsort  
  
 public void msort(int l,int r,JTable tabl) throws Exception{  
 if (!mfile.exists()){  
 mfile.createNewFile();  
 }  
 int x,i=l,j=r,count=0,sup,fin,ipr=0,jpr=0;  
 fin=0;  
 mbuf.write("Обработка подтаблицы с номерами от "+(l+1)+" до "+(r+1));  
 mbuf.newLine();  
 for (int o=l;o<=r;o++){  
  
 mbuf.write((String)tabl.getValueAt(o,0)+" | "+(String)tabl.getValueAt(o,1));  
 mbuf.newLine();  
  
 }  
 if(l>=r)  
 return;  
 x=Integer.*parseInt*((String)tabl.getValueAt((l+(r-l)/2),0));  
 mbuf.write("Средний элемент "+((l+(r-l)/2)+1)+"й сверху:"+x+"(в общей таблице)");  
 mbuf.newLine();  
 while (i<=j){  
 while(Integer.*parseInt*((String)tabl.getValueAt(i,0))<x){  
 i++;  
 }  
 while(Integer.*parseInt*((String)tabl.getValueAt(j,0))>x){  
 j--;  
 }  
  
 mbuf.write("Увеличиваем i пока не не доберемся до "+ (String)tabl.getValueAt(i,0));  
 mbuf.newLine();  
 mbuf.write("Уменьшаем j пока не не доберемся до "+ (String)tabl.getValueAt(j,0) + ". Затем меняем его с элементом " + (String)tabl.getValueAt(i,0));  
 mbuf.newLine();  
 if (i<=j){  
 container[0]=tabl.getValueAt(i,0);  
 container[1]=tabl.getValueAt(i,1);  
 tabl.setValueAt(tabl.getValueAt(j,0),i,0);  
 tabl.setValueAt(tabl.getValueAt(j,1),i,1);  
 tabl.setValueAt(container[0],j,0);  
 tabl.setValueAt(container[1],j,1);  
 i++;  
 j--;  
 }  
 }  
 if(l<j)  
 msort(l,i,tabl);  
 if(r>i)  
 msort(i,r,tabl);  
  
  
  
  
  
  
 }//end of msort  
  
 public SimpleFrame () throws Exception{  
 setSize (*DEFAULT\_WIDTH*, *DEFAULT\_HEIGHT*);  
 tableModel = new DefaultTableModel();  
 tableModel.setColumnIdentifiers(columnsHeader);  
 JTable table = new JTable(tableModel);  
 JButton addButton = new JButton("Add");  
 JButton randButton = new JButton("Random");  
 JButton fsortButton = new JButton("Sort with FIRST");  
 JButton msortButton = new JButton("Sort with MIDDLE");  
 fsortButton.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 if (table.getRowCount()==0) {  
 JOptionPane.*showMessageDialog*(null, "Нет строк в таблице");  
 return;  
 }  
 for(int i = 0; i < table.getRowCount(); i++)  
 {  
 if(table.getValueAt(i, 0) == null | table.getValueAt(i, 0) == "")  
 {  
 JOptionPane.*showMessageDialog*(null, "Не вся таблица заполнена: строка " + (i + 1));  
 return;  
 }  
 try{  
 Integer.*parseInt*(table.getValueAt(i, 0).toString());  
 }catch(NumberFormatException e1){  
 JOptionPane.*showMessageDialog*(null, "Ключ имеет не числовое целое значение: строка " + (i + 1));  
 return;  
 }  
  
 }  
 try {  
 fsort(0,table.getRowCount()-1,table);  
 fbuf.write("Полученная таблица: ");  
 fbuf.newLine();  
 for(int h=0;h<table.getRowCount();h++) {  
 fbuf.write((String) table.getValueAt(h, 0) + " | " + (String) table.getValueAt(h, 1));  
 fbuf.newLine();  
 }  
  
  
 fbuf.flush();  
 ffilew.flush();  
 fbuf.close();  
 ffilew.close();  
 } catch (Exception ex) {  
 ex.printStackTrace();  
 }  
 }  
 });  
 msortButton.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
  
 if (table.getRowCount()==0) {  
 JOptionPane.*showMessageDialog*(null, "Нет строк в таблице");  
 return;  
 }  
 for(int i = 0; i < table.getRowCount(); i++)  
 {  
 if(table.getValueAt(i, 0) == null | table.getValueAt(i, 0) == "")  
 {  
 JOptionPane.*showMessageDialog*(null, "Не вся таблица заполнена: строка " + (i + 1));  
 return;  
 }  
 try{  
 Integer.*parseInt*(table.getValueAt(i, 0).toString());  
 }catch(NumberFormatException e1){  
 JOptionPane.*showMessageDialog*(null, "Ключ имеет не числовое целое значение: строка " + (i + 1));  
 return;  
 }  
 }  
  
 try {  
 msort(0,table.getRowCount()-1,table);  
 mbuf.write("Полученная таблица: ");  
 mbuf.newLine();  
 for(int h=0;h<table.getRowCount();h++) {  
 mbuf.write((String) table.getValueAt(h, 0) + " | " + (String) table.getValueAt(h, 1));  
 mbuf.newLine();  
 }  
 mbuf.flush();  
 mfilew.flush();  
 mbuf.close();  
 mfilew.close();  
 } catch (Exception ex) {  
 ex.printStackTrace();  
 }  
 }  
 });  
 randButton.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 for (int i=0;i<table.getRowCount();i++){  
  
 table.setValueAt(Integer.*toString*((int)(Math.*random*()\*200-100)),i,0);  
  
 }  
 }  
 });  
  
 addButton.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 int idx = table.getSelectedRow();  
 tableModel.insertRow(idx+1,new String[]{  
 "","Name №"+String.*valueOf*(table.getRowCount()+1),  
 });  
 }  
 });  
 Box contents = new Box(BoxLayout.*Y\_AXIS*);  
 contents.add(new JScrollPane(table));  
 getContentPane().add(contents);  
 setVisible(true);  
  
 JPanel buttons = new JPanel();  
 buttons.add(addButton);  
 buttons.add(randButton);  
 buttons.add(fsortButton);  
 buttons.add(msortButton);  
 getContentPane().add(buttons,"South");  
 setSize(500,500);  
 setVisible(true);  
 }  
}