Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа № 2

По дисциплине ГИИС

Тема: «Разработка приложения «Адресная книга»»

Выполнил:

Студент 4 курса

Группы ИИ-16

Журавлёв В.А.

Проверил:

Михно Е.В.

Брест, 2021

**Цель работы:** разработать кроссплатформенный графический интерфейс.

**Ход работы:**

**Задание:** разработать приложение «Адресная книга» со след функционалом:

1. Добавление контакта.
2. Редактирование контакта.
3. Удаление контакта.
4. Поиск определенного контакта по имени.
5. Загрузка адресной книги.
6. Сохранение адресной книги.
7. Возможность экспорта контакта в формате визитки.

**Код программы:**

**Address:**

public class Address {

private String name;

private String address;

public Address(String name, String address) {

this.name = name;

this.address = address;

}

public void edit(String name, String address){

setName(name);

setAddress(address);

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

}

**Application:**

public class Application extends JFrame{

ArrayList<Address> addressBook;

Integer currentIndex = 0;

Integer mode = null;

JLabel name = new JLabel("Name:");

JLabel address = new JLabel("Address:");

JButton add = new JButton("Add");

JButton edit = new JButton("Edit");

JButton remove = new JButton("Remove");

JButton find = new JButton("Find");

JButton load = new JButton("Load...");

JButton save = new JButton("Save...");

JButton export = new JButton("Export");

JButton next = new JButton("Next");

JButton previous = new JButton("Previous");

JButton submit = new JButton("Submit");

JButton cancel = new JButton("Cancel");

JTextField nameField = new JTextField(16);

JTextArea addressField = new JTextArea(1, 16);

GridBagConstraints gbc = new GridBagConstraints();

public Application () {

JPanel p = new JPanel();

p.setLayout(new GridBagLayout());

addressBook = new ArrayList<>();

addressField.setLineWrap(true);

addressField.setWrapStyleWord(true);

addressField.setEnabled(false);

nameField.setEnabled(false);

add.addActionListener(e -> {

nameField.setEnabled(true);

addressField.setEnabled(true);

nameField.setText("");

addressField.setText("");

mode = 0;

add.setEnabled(false);

edit.setVisible(false);

remove.setVisible(false);

find.setVisible(false);

load.setVisible(false);

save.setVisible(false);

export.setVisible(false);

next.setEnabled(false);

previous.setEnabled(false);

func(p, submit,5,5,1,1, true);

submit.setVisible(true);

func(p, cancel,5,6,1,1, true);

cancel.setVisible(true);

});

submit.addActionListener(e ->{

if(nameField.getText().isEmpty() || addressField.getText().isEmpty()){

JOptionPane.showMessageDialog(Application.this,"Please enter a name and address.",

"Error", JOptionPane.OK\_OPTION);

return;

}

if(mode.equals(0) || (!mode.equals(1) && addressBook.get(currentIndex).getName() != nameField.getText()))

if(addressBook.stream()

.filter(t -> t.getName().equals(nameField.getText()))

.collect(Collectors.toCollection(ArrayList::new)).size() != 0){

JOptionPane.showMessageDialog(Application.this,"Sorry, " + nameField.getText() + " is already in your address book",

"Error", JOptionPane.OK\_OPTION);

return;

}

if(mode == 0)

addNote(nameField.getText(), addressField.getText());

else

editNote(currentIndex, nameField.getText(), addressField.getText());

if(mode == 0)

JOptionPane.showMessageDialog(Application.this,"Adding success");

else

JOptionPane.showMessageDialog(Application.this,"Edit success");

activeButtons();

});

cancel.addActionListener(e ->{

if(addressBook.size() > 0) {

nameField.setText(addressBook.get(currentIndex).getName());

addressField.setText(addressBook.get(currentIndex).getAddress());

}

if(mode == 0)

JOptionPane.showMessageDialog(Application.this,"Adding cancel");

else

JOptionPane.showMessageDialog(Application.this,"Edit cancel");

if(addressBook.size() > 0)

activeButtons();

else{

add.setEnabled(true);

edit.setEnabled(false);

remove.setEnabled(false);

find.setEnabled(false);

save.setEnabled(false);

export.setEnabled(false);

next.setEnabled(false);

previous.setEnabled(false);

edit.setVisible(true);

remove.setVisible(true);

find.setVisible(true);

load.setVisible(true);

save.setVisible(true);

export.setVisible(true);

next.setVisible(true);

previous.setVisible(true);

submit.setVisible(false);

cancel.setVisible(false);

nameField.setEnabled(false);

addressField.setEnabled(false);

}

});

next.addActionListener(e -> {

currentIndex = currentIndex + 1 != addressBook.size() ? currentIndex + 1: 0;

Address current = addressBook.get(currentIndex);

nameField.setText(current.getName());

addressField.setText(current.getAddress());

});

previous.addActionListener(e -> {

currentIndex = (currentIndex - 1) >= 0 ? currentIndex - 1: addressBook.size() - 1;

Address current = addressBook.get(currentIndex);

nameField.setText(current.getName());

addressField.setText(current.getAddress());

});

edit.addActionListener(e -> {

nameField.setEnabled(true);

addressField.setEnabled(true);

mode = 1;

add.setVisible(false);

edit.setEnabled(false);

remove.setVisible(false);

find.setVisible(false);

load.setVisible(false);

save.setVisible(false);

export.setVisible(false);

next.setEnabled(false);

previous.setEnabled(false);

func(p, submit,5,5,1,1, true);

submit.setVisible(true);

func(p, cancel,5,6,1,1, true);

cancel.setVisible(true);

});

remove.addActionListener(e -> {

int res = JOptionPane.showConfirmDialog(Application.this, "Are you sure?", "Warning",JOptionPane.YES\_NO\_OPTION, JOptionPane.WARNING\_MESSAGE);

if(res == JOptionPane.YES\_OPTION)

removeNote(currentIndex);

});

find.addActionListener((ActionEvent e) -> {

JFrame j = new JFrame();

JPanel pa = new JPanel();

pa.setLayout(new FlowLayout());

JLabel l = new JLabel("Enter the name of a contact:");

JTextField t = new JTextField(10);

JButton b = new JButton("Find");

b.addActionListener(e1 -> {

int indexFind = findByName(t.getText());

if(indexFind == -1)

JOptionPane.showMessageDialog(Application.this,"Sorry, " + t.getText() + " is not in your address book.",

"", JOptionPane.OK\_OPTION);

else{

currentIndex = indexFind;

nameField.setText(addressBook.get(indexFind).getName());

addressField.setText(addressBook.get(indexFind).getAddress());

}

Application.this.setEnabled(true);

j.setVisible(false);

j.dispose();

});

pa.add(l);

pa.add(t);

pa.add(b);

j.add(pa);

j.setSize(500,500);

j.addWindowListener(new WindowAdapter() {

public void windowClosing(WindowEvent e) {

Application.this.setEnabled(true);

j.dispose();

}

});

j.setVisible(true);

j.setResizable(false);

j.pack();

setEnabled(false);

});

export.addActionListener((e -> {

JFileChooser c = new JFileChooser();

c.addChoosableFileFilter(new FileNameExtensionFilter("vCard Files","vcf"));

int rVal = c.showSaveDialog(Application.this);

if (rVal == JFileChooser.APPROVE\_OPTION) {

String temp = "";

if (c.getFileFilter().getDescription().equals("vCard Files") && !c.getSelectedFile().toString().contains(".vcf"))

temp += ".vcf";

exportAddress(c.getSelectedFile().toString() + temp);

JOptionPane.showMessageDialog(Application.this,"Export success");

}

if (rVal == JFileChooser.CANCEL\_OPTION)

JOptionPane.showMessageDialog(Application.this,"Export cancel",

"", JOptionPane.OK\_OPTION);

}));

load.addActionListener(e -> {

JFileChooser c = new JFileChooser();

int rVal = c.showOpenDialog(Application.this);

if (rVal == JFileChooser.APPROVE\_OPTION) {

JOptionPane.showMessageDialog(Application.this,"Load success");

loadAddressBook(c.getSelectedFile().toString());

}

if (rVal == JFileChooser.CANCEL\_OPTION)

JOptionPane.showMessageDialog(Application.this,"Load cancel",

"", JOptionPane.OK\_OPTION);

});

save.addActionListener(e -> {

JFileChooser c = new JFileChooser();

c.addChoosableFileFilter(new FileNameExtensionFilter("Text document","txt"));

int rVal = c.showSaveDialog(Application.this);

if (rVal == JFileChooser.APPROVE\_OPTION) {

String temp = "";

if (c.getFileFilter().getDescription().equals("Text document") && !c.getSelectedFile().toString().contains(".txt"))

temp += ".txt";

saveAddressBook(c.getSelectedFile().toString() + temp);

JOptionPane.showMessageDialog(Application.this,"Save success");

}

if (rVal == JFileChooser.CANCEL\_OPTION)

JOptionPane.showMessageDialog(Application.this,"Save cancel",

"Error", JOptionPane.OK\_OPTION);

});

gbc.insets = new Insets(5,5,5,5);

gbc.anchor = GridBagConstraints.NORTH;

func(p,name,0,0,1,1,true);

func(p,nameField,1,0,4,1,true);

func(p,address,0,1,1,1,true);

func(p,addressField,1,1,4,9,false);

gbc.anchor = GridBagConstraints.CENTER;

func(p,add,5,1,1,1,true);

func(p,edit,5,2,1,1,true);

func(p,remove,5,3,1,1,true);

func(p,find,5,4,1,1,true);

func(p,load,5,5,1,1,true);

func(p,save,5,6,1,1,true);

func(p,export,5,7,1,1,true);

func(p,new JLabel(" "),0,2,1,1,true);

func(p,new JLabel(" "),0,3,1,1,true);

func(p,new JLabel(" "),0,4,1,1,true);

func(p,new JLabel(" "),0,5,1,1,true);

func(p,new JLabel(" "),0,6,1,1,true);

func(p,new JLabel(" "),0,7,1,1,true);

func(p,new JLabel(" "),0,8,1,1,true);

func(p,new JLabel(" "),0,9,1,1,true);

func(p,previous,1,10,2,1,true);

func(p,next,3,10,2,1,true);

edit.setEnabled(false);

remove.setEnabled(false);

find.setEnabled(false);

save.setEnabled(false);

export.setEnabled(false);

next.setEnabled(false);

previous.setEnabled(false);

add(p);

}

public void activeButtons(){

if(mode == 0) {

add.setEnabled(true);

edit.setEnabled(true);

edit.setVisible(true);

}

else {

edit.setEnabled(true);

add.setEnabled(true);

add.setVisible(true);

}

remove.setVisible(true);

remove.setEnabled(true);

find.setEnabled(true);

load.setEnabled(true);

save.setEnabled(true);

export.setEnabled(true);

find.setVisible(true);

load.setVisible(true);

save.setVisible(true);

export.setVisible(true);

next.setEnabled(true);

previous.setEnabled(true);

submit.setVisible(false);

cancel.setVisible(false);

nameField.setEnabled(false);

addressField.setEnabled(false);

}

public void func(JPanel p, JComponent c, Integer x, Integer y, Integer w, Integer h, boolean b){

gbc.gridx = x;

gbc.gridy = y;

gbc.gridwidth = w;

gbc.gridheight = h;

gbc.fill = b ? GridBagConstraints.HORIZONTAL : GridBagConstraints.VERTICAL;

p.add(c, gbc);

}

public void addNote(String name, String address){

addressBook.add(new Address(name,address));

currentIndex = addressBook.size() - 1;

}

public void editNote(Integer index, String name, String address){

addressBook.get(index).edit(name, address);

}

public void removeNote(int index){

addressBook.remove(index);

if(index != addressBook.size()){

nameField.setText(addressBook.get(index).getName());

addressField.setText(addressBook.get(index).getName());

}

else

if( index - 1 < 0 ){

remove.setEnabled(false);

edit.setEnabled(false);

find.setEnabled(false);

save.setEnabled(false);

export.setEnabled(false);

next.setEnabled(false);

previous.setEnabled(false);

nameField.setText("");

addressField.setText("");

}

else {

currentIndex -= 1;

nameField.setText(addressBook.get(index - 1).getName());

addressField.setText(addressBook.get(index - 1).getName());

}

}

public int findByName(String name){

for (int i =0; i < addressBook.size(); i++)

if(addressBook.get(i).getName().equals(name))

return i;

return -1;

}

public void loadAddressBook(String fileAddress){

addressBook = new ArrayList<>();

try {

Files.lines(Paths.get(fileAddress), StandardCharsets.UTF\_8).forEach(e -> {

ArrayList<String> list = Arrays.stream(e.split(" && ")).collect(Collectors.toCollection(ArrayList::new));

addressBook.add(new Address(list.get(0), list.get(1)));

});

}catch (IOException e){

e.printStackTrace();

}

if(addressBook.size() > 0){

edit.setEnabled(true);

remove.setEnabled(true);

find.setEnabled(true);

save.setEnabled(true);

export.setEnabled(true);

next.setEnabled(true);

previous.setEnabled(true);

currentIndex = addressBook.size() -1;

nameField.setText(addressBook.get(currentIndex).getName());

addressField.setText(addressBook.get(currentIndex).getAddress());

}

}

public void saveAddressBook(String fileDestination){

try {

File file = new File(fileDestination);

file.createNewFile();

try (FileWriter writer = new FileWriter(file)) {

for (Address address : addressBook)

writer.write(address.getName() + " && " + address.getAddress() + "\n");

writer.flush();

}

}catch (IOException e){

e.printStackTrace();

}

}

public void exportAddress(String fileDestination){

try {

File file = new File(fileDestination);

file.createNewFile();

try (FileWriter writer = new FileWriter(file)) {

writer.write("BEGIN:VCARD\n");

writer.write("VERSION:2.1\n");

String name = nameField.getText();

String address = addressField.getText();

String fname;

String sname;

ArrayList<String> listName;

int i = name.indexOf(" ");

if(i != -1){

listName = Stream.of(name.split("\\s+")).collect(Collectors.toCollection(ArrayList::new));

fname = listName.get(0);

sname = listName.get(listName.size() - 1);

}

else {

listName = new ArrayList<>();

fname = name;

sname = "";

}

writer.write("N:" + sname + ";" + fname +"\n");

String temp = "";

for (String s: listName)

temp += s + " ";

if(!listName.isEmpty())

writer.write("FN:" + temp.trim() + "\n");

else

writer.write("FN:" + fname +"\n");

writer.write("ADR;HOME:;" + address + "\n");

writer.write("END:VCARD\n");

writer.flush();

}

}catch (IOException e){

e.printStackTrace();

}

}

public static void main(String[] args) {

EventQueue.invokeLater(() -> {

Application frame = new Application ();

frame.setSize(500,500);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setVisible(true);

frame.setTitle("Address book");

frame.setResizable(false);

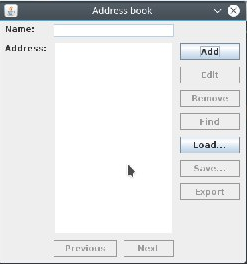
frame.pack();

});

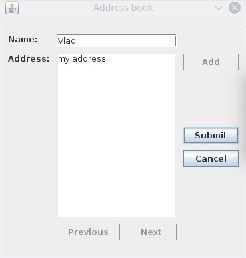
}

**Результат:**

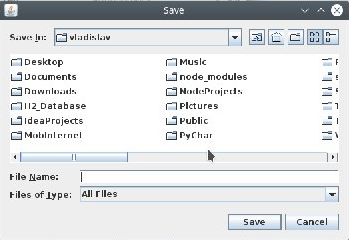
1). При запуске приложения мы видим окно:

****

2). При нажатии на Add:



3). При нажатии на Save



4) Кнопки Previous Next отвечают за перемещение по адресной книге

**Вывод:** разработал кроссплатформенный графический интерфейс.