Hindi Vidya Prachar Samiti’s

# RAMNIRANJAN JHUNJHUNWALA COLLEGE

Ghatkopar (W), Mumbai-86

**Certificate**



This is to certify that **Pandey Ashwini Ashish,** roll no- **629** of **M.S.C.(I.T.)** class has completed the required number of experiments in the subject of **Cloud Computing** in the department of Information Technology during the academic year 2022-2023.

Signature of Internal Guide Sign of Co-Ordinator

College Seal and Date Examiner

*Mini Project: PhoneBook Application*

*(Java)*

*Abstract:*

*I represent the design and implementation of PhoneBook. When People were started using Phone so they need to manage their contacts in phone, as maintaining the dairy of contacts was difficult so an application was needed to maintain contacts in phone. Contacts are easily maintained by their names in ascending order in PhoneBook application using Comparator in Java.*

*Declaration:*

I declare that this written submission represents my own ideas in my own words and where other’s ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/fact/data/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission.

## ASHWINI ASHISH PANDEY

*Index:*

|  |  |
| --- | --- |
| Sr No. | Topics |
| 1 | Introduction |
| 2 | Objective |
| 3 | Technology Used |
| 4 | Application Working |
| 5 | Requirements |
| 6 | Project Images |

Introduction:

My PhoneBook Application is simple application in which user can “Add” or “View” contact in application. In this Project, I have used Java Code technology to build it and some libraries of Java.

Objective:

* To add the contact and at same time to sort them before saving into the application.
* To provide very simple operation and minimal look.

Technology Used:

Java is a programming language and a platform. Java is a high level, robust, object-oriented, and secure programming language. Java was developed by *Sun Microsystems* (which is now the subsidiary of Oracle) in the year 1995. *James Gosling* is known as the father of Java.

Platform:

Any hardware or software environment in which a program runs, is known as a platform. Since Java has a runtime environment (JRE) and API, it is called a platform.

AWT:

Java AWT (Abstract Window Toolkit) is *an API to develop Graphical User Interface (GUI) or windows-based applications* in Java.

Java AWT components are platform-dependent i.e., components are displayed according to the view of operating system.

The java.awt package provides classes for AWT API such

as TextField, Label, TextArea,

RadioButton, CheckBox, Choice, List etc.

For example, an AWT GUI with components like TextField, label and button will have different look and feel for the different platforms like Windows, MAC OS, and Unix. The reason for this is the platforms have different view for their native components and AWT directly calls the native subroutine that creates those components.

In simple words, an AWT application will look like a windows application in Windows OS whereas it will look like a Mac application in the MAC OS.

SWING:

Java Swing tutorial is a part of Java Foundation Classes (JFC) that is *used to create window-based applications*. It is built on the top of AWT (Abstract Windowing Toolkit) API and entirely written in java.

Unlike AWT, Java Swing provides platform-independent and lightweight components.

The javax.swing package provides classes for java swing API such as JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser etc.

TREESET:

Java TreeSet class implements the Set interface that uses a

tree for storage. It inherits AbstractSet class and implements the NavigableSet interface. The objects of the TreeSet class are stored in ascending order.

Access and retrieval times are quite fast, which makes TreeSet an excellent choice when storing large amounts of sorted information that must be found quickly.

The ordering of the elements is maintained by a set using their natural ordering whether or not an explicit comparator is provided.

Requirement:

* Windows 10/11
* JDK 17/18
* Java IDE Platform

Application Working:

In this project user usually Add Contact while adding contact they need to provide “Name” and “Contact Number” of whose contact is to be added. After it will ask the user to continue adding another number.

Code of Adding Contact:

*@Override*

public void actionPerformed(ActionEvent e) {

String name=txt\_name.getText();

String cno=txt\_cno.getText();

if(name.equals("") || (cno.equals("")))

{

JOptionPane.*showMessageDialog*(jf\_AddContact, "pl fill your details","Warning Message",JOptionPane.***WARNING\_MESSAGE***);

} else

{

Contact obj=new Contact(); obj.setName(name); obj.setCt\_No(cno);

contactList.add(obj);

ArrayList<Contact> tempList=new

ArrayList<Contact>(contactList);

Collections.*sort*(tempList,new NameSort()); dlm\_contact.removeAllElements();

for(Contact c: tempList)

{

dlm\_contact.addElement(c);

}

txt\_name.setText(""); txt\_cno.setText("");

JOptionPane.*showMessageDialog*(jf\_AddContact, "Contact has been added","Success

Message",JOptionPane.***INFORMATION\_MESSAGE***);

int status=JOptionPane.*showConfirmDialog*(jf\_AddContact, "want to continue?","Confirmation",JOptionPane.***YES\_NO\_OPTION***); if (status==0)

{

jf\_AddContact.setVisible(true);

} else { jf\_AddContact.dispose();

}

}

}

Also, we can view the contact that we have been added in application.

Code of Viewing Contact:

public void addComponentToView()

{

jf\_ViewContact = new JFrame("View Contact") jf\_ViewContact.setLayout(null); jf\_ViewContact.setSize(300, 220); jf\_ViewContact.setVisible(true);

jf\_ViewContact.setLocationRelativeTo(null);

jl\_contact.setModel(dlm\_contact); scroll = new JScrollPane(jl\_contact);

lbl\_ct.setBounds(10, 20, 100, 20); jf\_ViewContact.add(lbl\_ct);

scroll.setBounds(10, 50, 265, 100); jf\_ViewContact.add(scroll);

}

Main Code:

package phonebookapplication;

import phonebook.\*;

public class PhoneBookApplication {

public static void main(String[] args) {

Menu mobj = new Menu(); mobj.addComponent();

}

}

Images of Project Output:

