Syeda Reeha Quasar

14114802719

4C7

Topics Covered

Swings

Lab - 11

Java Programming Lab

# **EXPERIMENT – 11.1**

## **Aim:**

Create runnable jar file in java.

## **Theory:**

**Java 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.

**JFrame :** The javax.swing.JFrame class is a type of container which inherits the java.awt.Frame class. JFrame works like the main window where components like labels, buttons, textfields are added to create a GUI. Unlike Frame, JFrame has the option to hide or close the window with the help of setDefaultCloseOperation(int) method.

## **Source Code:**

import javax.swing.\*;

public class IOFile {

IOFile() {

JFrame f = new JFrame("My Jar File");

JButton b = new JButton("Click Me!");

b.setBounds(100, 150, 100, 40);

JLabel l1 = new JLabel("Syeda Reeha Quasar : 14114802719");

l1.setBounds(100, 50, 170, 100);

f.add(b);

f.add(l1);

f.setSize(300, 400);

f.setLayout(null);

f.setVisible(true);

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

}

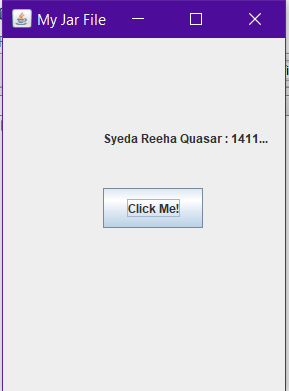
public static void main(String[] args) {

new IOFile();

}

}

## **Output:**



# **EXPERIMENT – 11.2**

## **Aim:**

Display image on a button in swing.

## **Theory:**

**Java 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.

**JFrame :** The javax.swing.JFrame class is a type of container which inherits the java.awt.Frame class. JFrame works like the main window where components like labels, buttons, textfields are added to create a GUI. Unlike Frame, JFrame has the option to hide or close the window with the help of setDefaultCloseOperation(int) method

**Jbutton :** The JButton class is used to create a labeled button that has platform independent implementation. The application result in some action when the button is pushed. It inherits AbstractButton class.

## **Source Code:**

import javax.swing.ImageIcon;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.SwingUtilities;

import javax.swing.WindowConstants;

import java.awt.FlowLayout;

public class exp2 extends JFrame {

public exp2() {

initComponents();

}

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> new exp2().setVisible(true));

}

private void initComponents() {

setTitle("Image inside Button");

setSize(500, 500);

setDefaultCloseOperation(WindowConstants.EXIT\_ON\_CLOSE);

getContentPane().setLayout(new FlowLayout(FlowLayout.CENTER));

JButton helloButton = new JButton("Hello", new ImageIcon(

this.getClass().getResource("/images/hello.png")));

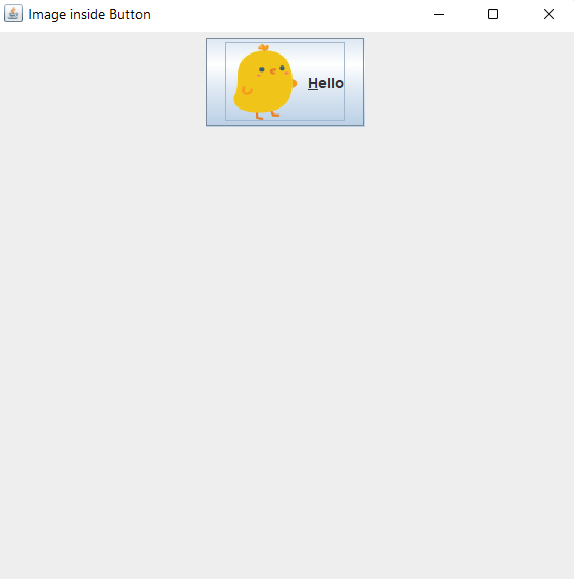
helloButton.setMnemonic('H');

getContentPane().add(helloButton);

}

}

## **Output:**



# **EXPERIMENT – 11.3**

## **Aim:**

Change the component color by choosing a color from ColorChooser.

## **Theory:**

**Java 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.

**JFrame :** The javax.swing.JFrame class is a type of container which inherits the java.awt.Frame class. JFrame works like the main window where components like labels, buttons, textfields are added to create a GUI. Unlike Frame, JFrame has the option to hide or close the window with the help of setDefaultCloseOperation(int) method

**Jbutton :** The JButton class is used to create a labeled button that has platform independent implementation. The application result in some action when the button is pushed. It inherits AbstractButton class.

**JColorChooser :** The JColorChooser class is used to create a color chooser dialog box so that user can select any color. It inherits JComponent class.

## **Source Code:**

import java.awt.event.\*;

import java.awt.\*;

import javax.swing.\*;

public class IOFile extends

JFrame implements ActionListener {

JButton b = new JButton("color");

protected JLabel label;

Container c = getContentPane();

IOFile() {

label = new JLabel("Syeda Reeha Quasar : 14114802719", JLabel.CENTER);

label.setForeground(Color.BLACK);

label.setBackground(Color.WHITE);

label.setOpaque(true);

label.setFont(new Font("SansSerif", Font.BOLD, 25));

c.setLayout(new FlowLayout());

b.addActionListener(this);

c.add(label);

c.add(b);

}

public void actionPerformed(ActionEvent e) {

Color initialcolor = Color.RED;

Color color = JColorChooser.showDialog(this, "Select a color", initialcolor);

label.setForeground(color);

}

public static void main(String[] args) {

IOFile ch = new IOFile();

ch.setSize(400, 400);

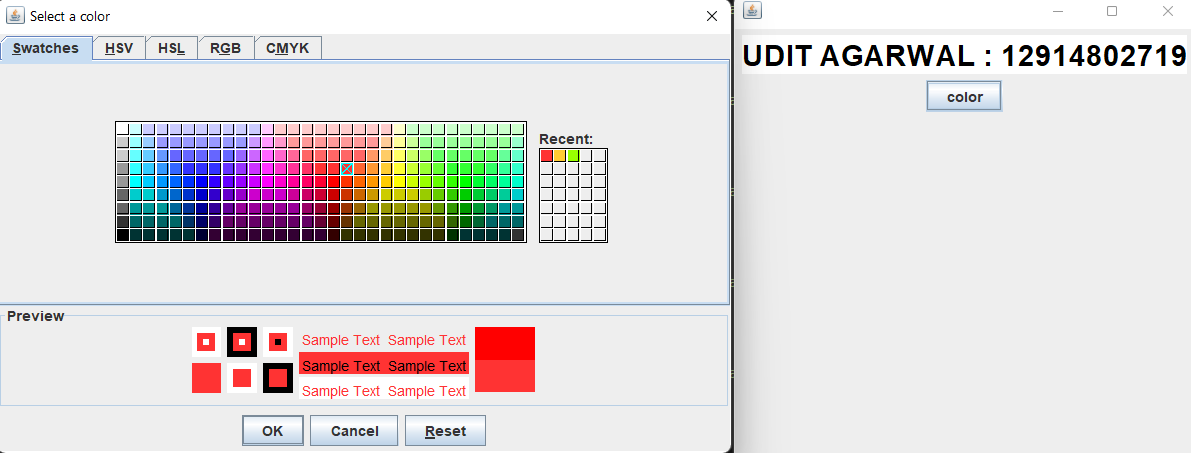
ch.setVisible(true);

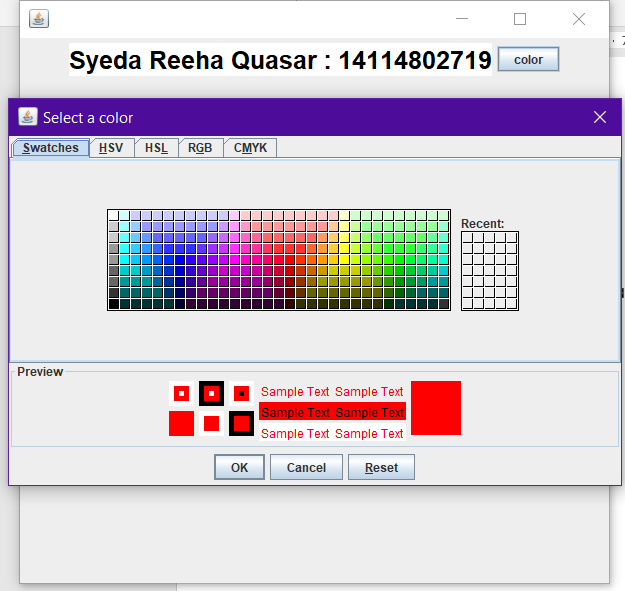
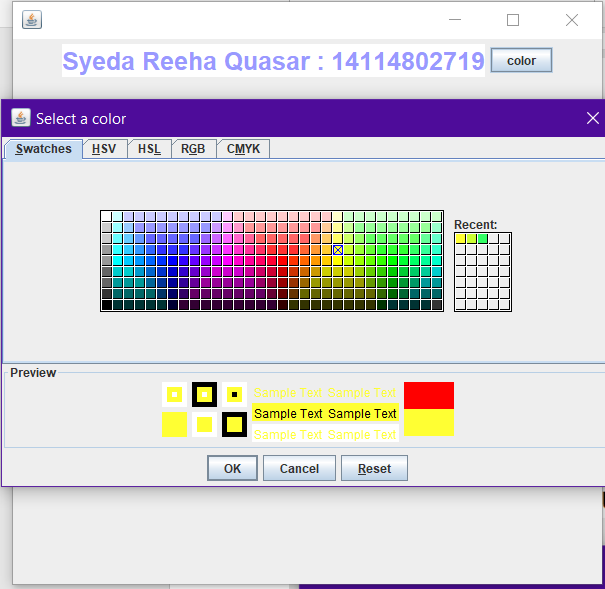
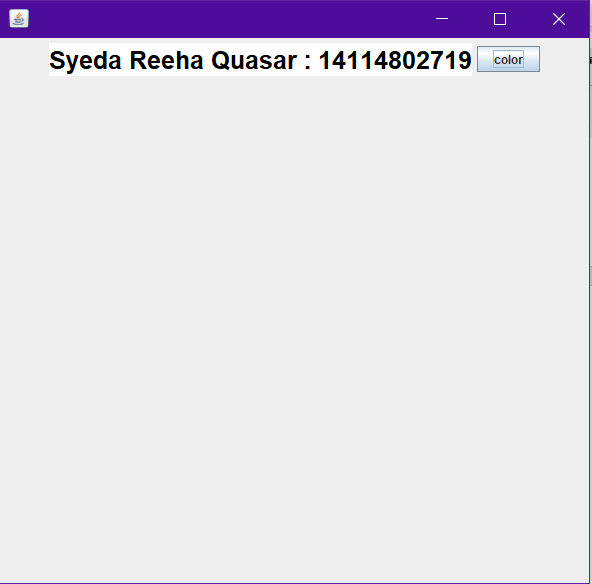
ch.setDefaultCloseOperation(EXIT\_ON\_CLOSE);

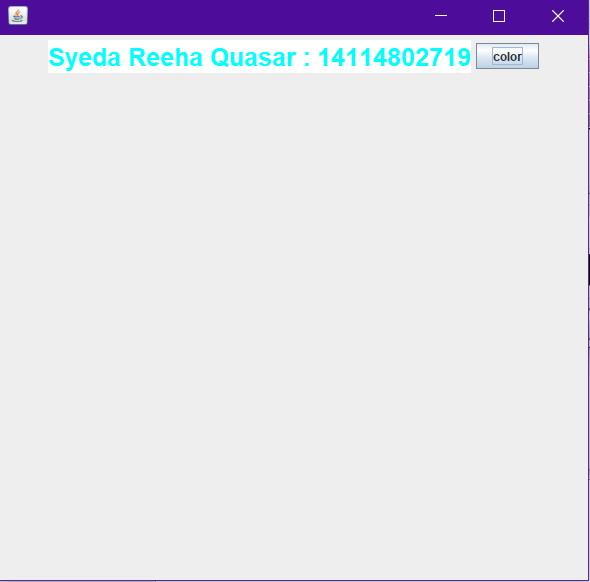
}

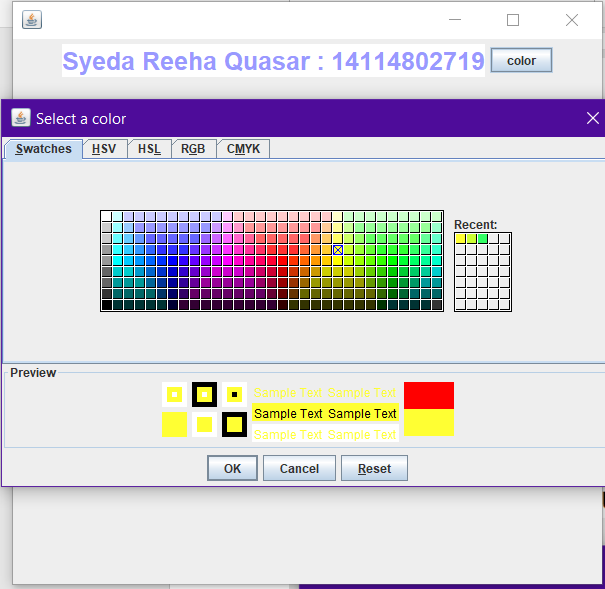
}

## **Output:**









# **EXPERIMENT – 11.4**

## **Aim:**

Display the digital watch in swing tutorial.

## **Theory:**

**Java 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.

**Thread :** Threads allows a program to operate more efficiently by doing multiple things at the same time. Threads can be used to perform complicated tasks in the background without interrupting the main program.

**JFrame :** The javax.swing.JFrame class is a type of container which inherits the java.awt.Frame class. JFrame works like the main window where components like labels, buttons, textfields are added to create a GUI. Unlike Frame, JFrame has the option to hide or close the window with the help of setDefaultCloseOperation(int) method

**Jbutton :** The JButton class is used to create a labeled button that has platform independent implementation. The application result in some action when the button is pushed. It inherits AbstractButton class.

## **Source Code:**

import javax.swing.\*;

import java.awt.\*;

import java.text.\*;

import java.util.\*;

public class IOFile implements Runnable {

JFrame f;

Thread t = null;

int hours = 0, minutes = 0, seconds = 0;

String timeString = "";

JButton b;

JLabel l1;

IOFile() {

f = new JFrame();

t = new Thread(this);

t.start();

b = new JButton();

b.setBounds(100, 100, 100, 50);

l1 = new JLabel("Reeha : 14114802719");

l1.setBounds(100, 10, 170, 100);

f.add(l1);

f.add(b);

f.setSize(300, 400);

f.setLayout(null);

f.setVisible(true);

}

public void run() {

try {

while (true) {

Calendar cal = Calendar.getInstance();

hours = cal.get(Calendar.HOUR\_OF\_DAY);

if (hours > 12)

hours -= 12;

minutes = cal.get(Calendar.MINUTE);

seconds = cal.get(Calendar.SECOND);

SimpleDateFormat formatter = new SimpleDateFormat("hh:mm:ss");

Date date = cal.getTime();

timeString = formatter.format(date);

printTime();

t.sleep(1000);

}

} catch (Exception e) {

}

}

public void printTime() {

b.setText(timeString);

}

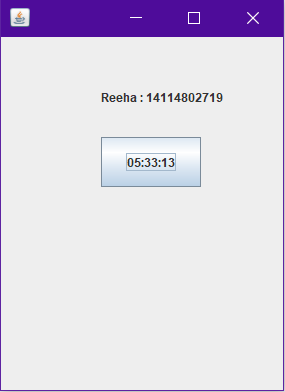
public static void main(String[] args) {

new IOFile();

}

}

## **Output:**



# **EXPERIMENT – 11.5**

## **Aim:**

Create a notepad in swing.

## **Theory:**

**Java 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.

**JFrame :** The javax.swing.JFrame class is a type of container which inherits the java.awt.Frame class. JFrame works like the main window where components like labels, buttons, textfields are added to create a GUI. Unlike Frame, JFrame has the option to hide or close the window with the help of setDefaultCloseOperation(int) method

**Jbutton :** The JButton class is used to create a labeled button that has platform independent implementation. The application result in some action when the button is pushed. It inherits AbstractButton class.

## **Source Code:**

import java.awt.\*;

import java.awt.datatransfer.Clipboard;

import java.awt.datatransfer.DataFlavor;

import java.awt.datatransfer.Transferable;

import java.awt.event.\*;

import java.io.File;

import java.io.PrintWriter;

import java.util.Scanner;

import javax.swing.\*;

public class Notepad extends JFrame {

    private static final long serialVersionUID = 1L;

    JFrame frame;

    JMenuBar menuBar;

    JMenu file;

    JMenu edit;

    JMenuItem open, newFile, save, exit;

    JMenuItem undo, paste, selectAll;

    JMenu format;

    JMenu help;

    JFileChooser fileChooser;

    JTextArea textArea;

    Clipboard clip;

    Notepad() {

        frame = new JFrame("Notepad Application");

        file = new JMenu("File");

        edit = new JMenu("Edit");

        format = new JMenu("Format");

        help = new JMenu("Help");

        newFile = new JMenuItem("New");

        open = new JMenuItem("Open");

        save = new JMenuItem("Save");

        exit = new JMenuItem("Exit");

        undo = new JMenuItem("Undo                 Ctrl+Z");

        paste = new JMenuItem("Paste                Ctrl+V");

        selectAll = new JMenuItem("Select All       Ctrl+A ");

        textArea = new JTextArea();

        fileChooser = new JFileChooser();

        menuBar = new JMenuBar();

        frame.setLayout(new BorderLayout());

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

        frame.add(textArea);

        file.add(open);

        file.add(newFile);

        file.add(save);

        file.add(exit);

        edit.add(undo);

        edit.add(paste);

        edit.add(selectAll);

        menuBar.add(file);

        menuBar.add(edit);

        menuBar.add(format);

        menuBar.add(help);

        frame.setJMenuBar(menuBar);

        OpenListener openL = new OpenListener();

        NewListener NewL = new NewListener();

        SaveListener saveL = new SaveListener();

        ExitListener exitL = new ExitListener();

        open.addActionListener(openL);

        newFile.addActionListener(NewL);

        save.addActionListener(saveL);

        exit.addActionListener(exitL);

        // UndoListener UndoL = new UndoListener();

        PasteListener pasteL = new PasteListener();

        // EditListener EditL = new EditListener();

        // SelectListener SelectL = new SelectListener();

        // undo.addActionListener(UndoL);

        // paste.addActionListener(EditL);

        // selectAll.addActionListener(SelectL);

        frame.setSize(800, 600);

        frame.setVisible(true);

    }

    class OpenListener implements ActionListener {

        public void actionPerformed(ActionEvent e) {

            if (JFileChooser.APPROVE\_OPTION == fileChooser.showOpenDialog(frame)) {

                File file = fileChooser.getSelectedFile();

                textArea.setText("");

                Scanner in = null;

                try {

                    in = new Scanner(file);

                    while (in.hasNext()) {

                        String line = in.nextLine();

                        textArea.append(line + "\n");

                    }

                } catch (Exception ex) {

                    ex.printStackTrace();

                } finally {

                    in.close();

                }

            }

        }

    }

    class SaveListener implements ActionListener {

        public void actionPerformed(ActionEvent e) {

            if (JFileChooser.APPROVE\_OPTION == fileChooser.showSaveDialog(frame)) {

                File file = fileChooser.getSelectedFile();

                PrintWriter out = null;

                try {

                    out = new PrintWriter(file);

                    String output = textArea.getText();

                    System.out.println(output);

                    out.println(output);

                } catch (Exception ex) {

                    ex.printStackTrace();

                } finally {

                    try {

                        out.flush();

                    } catch (Exception ex1) {

                    }

                    try {

                        out.close();

                    } catch (Exception ex1) {

                    }

                }

            }

        }

    }

    class NewListener implements ActionListener {

        public void actionPerformed(ActionEvent e) {

            textArea.setText("");

            // frame.add(newFile);

            // textArea.(newFile+"\n");

        }

    }

    class ExitListener implements ActionListener {

        public void actionPerformed(ActionEvent e) {

            System.exit(0);

        }

    }

    class PasteListener implements ActionListener {

        public void actionPerformed(ActionEvent e) {

            Transferable cliptran = clip.getContents(Notepad.this);

            try {

                String sel = (String) cliptran.getTransferData(DataFlavor.stringFlavor);

                textArea.replaceRange(sel, textArea.getSelectionStart(), textArea.getSelectionEnd());

            } catch (Exception exc) {

                System.out.println("not string flavour");

            }

        }

    }

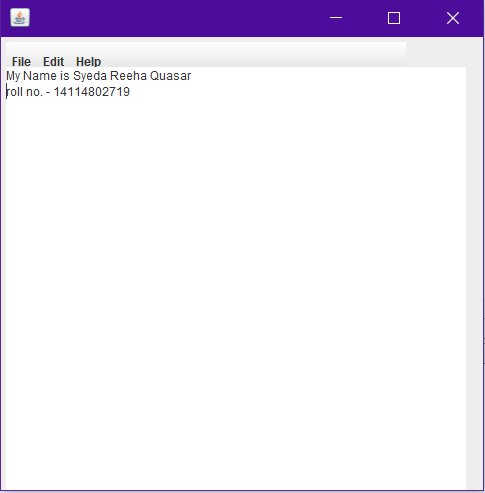
    public static void main(String args[]) {

        Notepad n = new Notepad();

    }

}

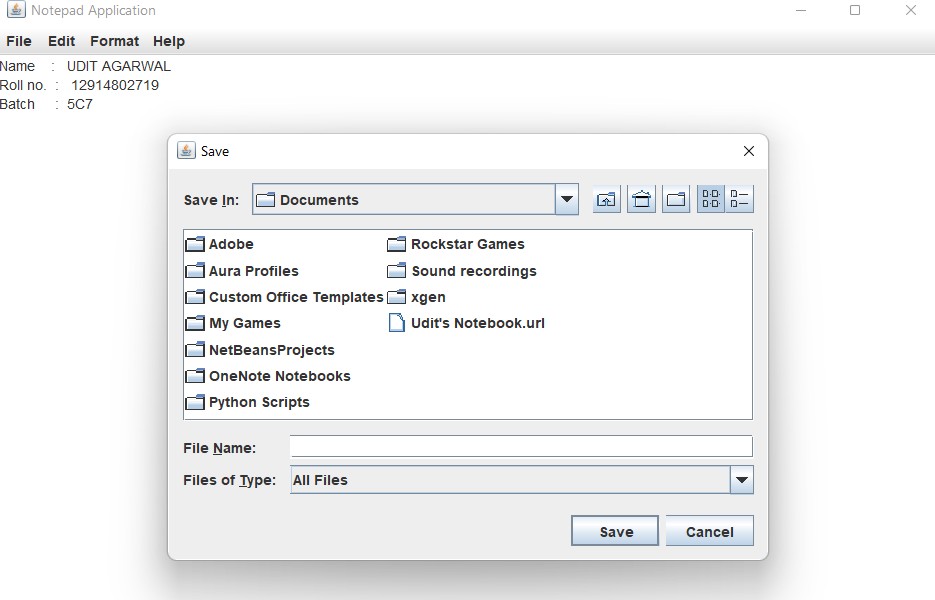
## **Output:**





Syeda Reeha Quasar

14114802719



Syeda Reeha Quasar

14114802719

# **Viva Questions**

### **1. What is Java Swing?**

### Ans.

It is a part of JFC (Java Foundation Classess) that is used to create window-based applications.

Java Swing components are platform independent and lightweight .

### **2. What are the methods of component class in Java Swing?**

### Ans.

There are four types of methods of component class are:

* public void add(Component c)
* public void setSize(int width, int height)
* public void setLayout(LayoutManager m)
* public void setVisible(boolean b)

**3. How many ways to create a frame in Java Swing ?** Ans.

There are two ways to create a frame:

* By Association(creating the object of Frame class)
* By Inheritance(extending Frame class)

### **4. What are differences between Swing and AWT?**

Ans.

There is couple of differences between swing and AWT.

### AWT component are considered to be heavyweight while Swing component are lightweights.

### Swing has plug gable look and feel.

### AWT is platform dependent same GUI will look different platform while Swing is developed in Java and is platform dependent.

### **5. True of false: An applet can run multiple threads.**

### Ans.

True. The paint and update methods are always called from the AWT drawing and event handling thread. You can have your applet create additional threads, which is recommended for performing time-consuming tasks.