JAVA AWT PROGRAMS

Name : Kathasagaram Aishwarya
USN : 1BM22CS123
Section : 3C

---

1. ButtonDemo.java

Code:

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;

/*
 <applet code="ButtonDemo" width=250 height=150>
 </applet>
*/

public class ButtonDemo extends Applet implements ActionListener {
 String msg = "";
 Button yes, no, maybe;
 public void init() {
 yes = new Button("Yes");
 no = new Button("No");
 maybe = new Button("Undecided");
 add(yes);
 add(no);
 add(maybe);
 yes.addActionListener(this);
 no.addActionListener(this);
 maybe.addActionListener(this);
 }
 public void actionPerformed(ActionEvent ae) {
 String str = ae.getActionCommand();
 if(str.equals("Yes")) {
 msg = "You pressed Yes.";
 }
 else if(str.equals("No")) {
 msg = "You pressed No.";
 }
 else {
 msg = "You pressed Undecided.";
 }
 repaint();
 }
 public void paint(Graphics g) {
 g.drawString(msg, 6, 100);
 }}
```

Output:

Could not execute

Functionality:

The **ButtonDemo** program is compiled first. A HTML file with the same filename is made and executed on **appletviewer ButtonDemo.html**. This when implemented will display a window with three buttons ("Yes", "No", "Undecided") using AWT. These buttons, when clicked display an appropriate message.

2. buttondrag.java

Code:

```
import java.awt.*;
import java.awt.event.*;
import java.util.Collections;
import javax.swing.JPanel;
import java.util.Random;


public class buttondrag extends Frame implements ActionListener
{
    int n = 3;
    int m=n*n;
    Boolean clicked=false,doneFlag=false;
    String cLabel;
    int cI;
    JPanel buttonPanel = new JPanel();
    JPanel optionPanel =new JPanel();
    Button b[]=new Button[n*n];
    Button start,reset,restart;
    String msg="";
    timecalc total;
    int totalTime;

    public buttondrag()
    {
        addWindowListener(new MyWindowAdapter());
        setLayout(new BorderLayout());
        buttonPanel.setLayout(new GridLayout(n, n));
        setFont(new Font("Arial", Font.BOLD, 24));
        buttonPanel.setSize(300, 300);
        buttonPanel.setEnabled(false);
        optionPanel.setLayout(new FlowLayout());
        add(buttonPanel,BorderLayout.CENTER);
        add(optionPanel,BorderLayout.SOUTH);
        for(int i = 0; i < n; i++)
        {
            for(int j = 0; j < n; j++)
            {
```

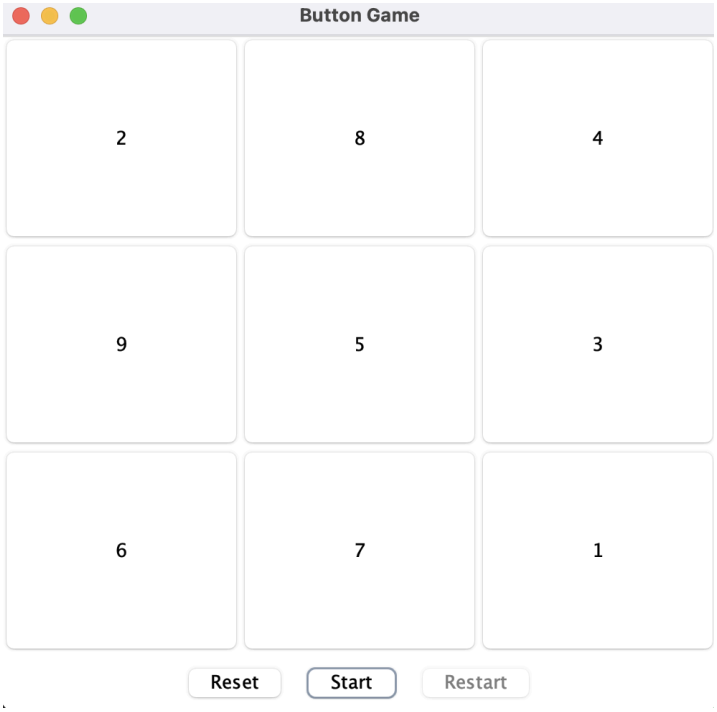
```
            int k = i * n + j;
              if(k > 0)
                 {
                   buttonPanel.add(b[k]=new Button("" + k));
                 }

         }
     }
     buttonPanel.add(b[0]=new Button("9"));
      for(int i=0;i<m;i++)
      {
         b[i].addActionListener(this);
      }
      optionPanel.add(reset=new Button("Reset"));
      optionPanel.add(start=new Button("Start"));
      optionPanel.add(restart=new Button("Restart"));
      start.addActionListener(this);
      reset.addActionListener(this);
      restart.addActionListener(this);
      restart.setEnabled(false);
      reset.setEnabled(false);
      //restart.setVisible(false);
      Component[] com = buttonPanel.getComponents();
     for (int a = 0; a < com.length; a++)
     com[a].setEnabled(false);
}

public void actionPerformed(ActionEvent ae)
{

     if(ae.getSource()==start && (!doneFlag))
     {
         Component[] com = buttonPanel.getComponents();
         for (int a = 0; a < com.length; a++)
         com[a].setEnabled(true);
         shuffleStart();
         reset.setEnabled(true);
         total=new timecalc();
     }
     else
     if(ae.getSource()==reset  && (!doneFlag) )
     {
         reSet();
         totalTime=0;
         reset.setEnabled(false);
     }
     else
     if(ae.getSource()==restart  && (doneFlag) )
     {
         reStart();
         totalTime=0;
         reset.setEnabled(true);
     }
```

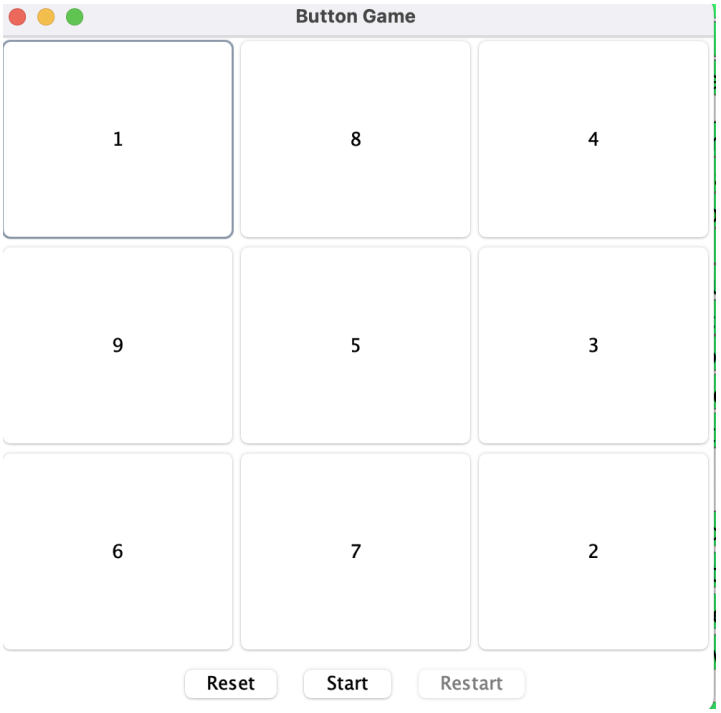
```
     {
        for(int i=0;i<m;i++)
        {
            if(ae.getSource()==b[i] && (!clicked))
            {
                b[i].setVisible(false);
                cLabel=b[i].getLabel();
                cI=i;
                clicked=!clicked;
            }
            else
            if(ae.getSource()==b[i] && (clicked))
            {
                b[cI].setLabel(b[i].getLabel());
                b[cI].setVisible(true);
                b[i].setLabel(""+cLabel);
                clicked=!clicked;
                checkCorrect();
            }
        }
      }
    }
    public void checkCorrect()
    {
       int checkComI=0;
       for(int i=1;i<m;i++)
       {
            if(b[i].getLabel().equals(String.valueOf(i)))
                checkComI+=1;
       }
       if(checkComI==8)
        {
          totalTime=total.getTimeInSeconds();
          for(int i=0;i<m;i++)
             b[i].setVisible(false);
          doneFlag=true;
          restart.setEnabled(true);
          restart.setVisible(true);
          reset.setEnabled(false);
          msg="Congradulations!, you Finished it in "+ totalTime+ " seconds
!!";
          repaint();
         }

    }
    public void paint(Graphics g)
    {
        if(doneFlag)
        {
            setBackground(Color.BLACK);
            setForeground(Color.WHITE);
        }
        else
```

```
        setBackground(Color.WHITE);
    g.setFont(new Font("Serif", Font.PLAIN, 24));
    g.drawString(msg,30,250);
}
public void shuffleStart()
{
    for(int i=0; i<m;i++)
    {
        Random number=new Random();
        int num=number.nextInt(9);
        swap(num,i);
    }
}
 public void reStart()
 {
    for(int i=1; i<m;i++)
    {
         b[i].setVisible(true);
         b[i].setLabel(String.valueOf(i));
    }
    b[0].setVisible(true);
    b[0].setLabel("9");
    doneFlag=false;
    Component[] com = buttonPanel.getComponents();
    for (int a = 0; a < com.length; a++)
    com[a].setEnabled(false);
    restart.setEnabled(false);
    repaint();
 }
  public void reSet()
 {
    for(int i=1; i<m;i++)
    {
        b[i].setLabel(String.valueOf(i));
    }
    b[0].setLabel("9");
    Component[] com = buttonPanel.getComponents();
    for (int a = 0; a < com.length; a++)
    com[a].setEnabled(false);
 }
public void swap(int x,int y)
{
    String temp=b[x].getLabel();
    b[x].setLabel(b[y].getLabel());
    b[y].setLabel(temp);
}
public static void main(String ar[])
{
    buttondrag cd=new buttondrag();
    cd.setSize(new Dimension(500,500));
    cd.setTitle("Button Game");
    cd.setVisible(true);
}
```

```
}

class MyWindowAdapter extends WindowAdapter
{
    public void windowClosing(WindowEvent we)
    {
        System.exit(0);
     }
}
class timecalc
{
    private final long startedMillis = System.currentTimeMillis();

    public int getTimeInSeconds()
    {
        long nowMillis = System.currentTimeMillis();
        return (int)((nowMillis - this.startedMillis) / 1000);
    }
}
```

Output:





Functionality:

The **buttondrag** Java program implements a sliding tile puzzle game using AWT. It features buttons arranged in a grid layout, with options to start, reset, and restart the game. Players can click tiles to move them, aiming to rearrange them into the correct order. The game tracks completion status, disables tile interaction upon completion, and calculates the time taken to solve the puzzle. Additionally, it manages window closing events for proper application exit.

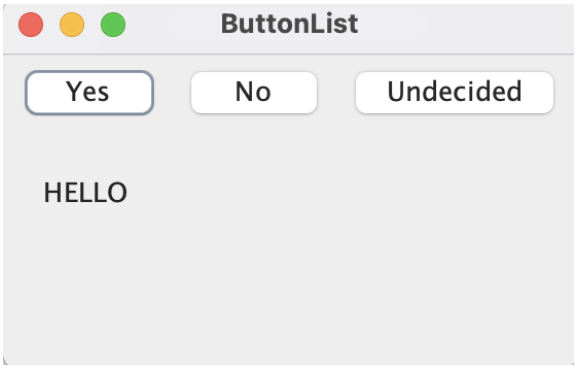
3. ButtonList.java

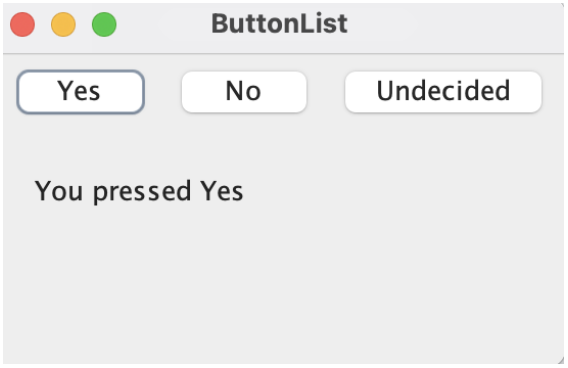
Code:

```
import java.awt.*;
import java.awt.event.*;

public class ButtonList extends Frame implements ActionListener {
 String msg = "HELLO";
 Button bList[]=new Button[3];
 public ButtonList() {
 setLayout(new FlowLayout());
 Button yes = new Button("Yes");
 Button no = new Button("No");
 Button maybe = new Button("Undecided");


 //Syntax- Component add(Component compRef)

 bList[0]=(Button) add(yes);
 bList[1]=(Button) add(no);
 bList[2]=(Button)add(maybe);
```

```
 for(int i=0;i<3;i++)
 {
 bList[i].addActionListener(this);
 }
 /*
 addWindowListener(new WindowAdapter()
 {public void windowClosing(WindowAdapter we)
       {System.exit(0);}
 });*/



 addWindowListener(new MyWindowAdapter());

 }



 public void actionPerformed(ActionEvent ae) {
 for(int i=0;i<3;i++){
 if(ae.getSource()==bList[i])
       {
 msg = "You pressed "+bList[i].getLabel();
 }
 }
  repaint();
 }
 public void paint(Graphics g) {
 g.drawString(msg, 20, 100);
 }

 public static void main(String ar[])
 {
       ButtonList aa=new ButtonList();
       aa.setSize(new Dimension(230,150));
       aa.setTitle("ButtonList");
       aa.setVisible(true);
 }
}

class MyWindowAdapter extends WindowAdapter {
public void windowClosing(WindowEvent we) {
 System.exit(0);
}}
```

Output:





Functionality:

The **ButtonList** Java program creates a GUI with three buttons ("Yes", "No", "Undecided") using AWT. It implements the **ActionListener** interface to handle button clicks, updating a message displayed on the frame accordingly.

4. ButtonListD.java

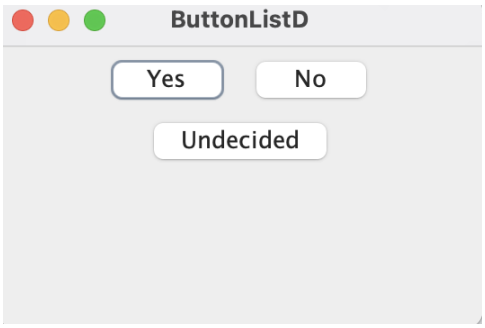
Code:

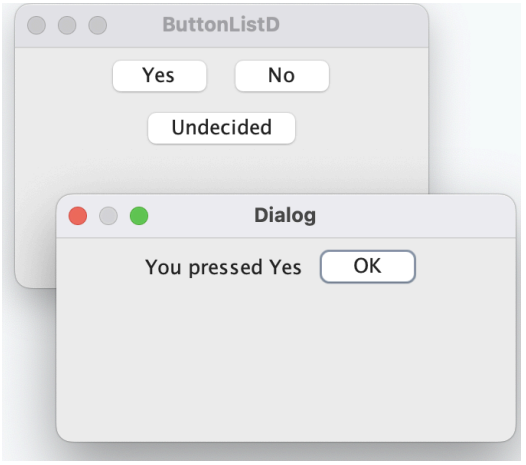
```
import java.awt.event.*;
import java.awt.*;


class SampleDialog extends Dialog implements ActionListener {
      ButtonListD bld;
  SampleDialog(Frame parent, String title) {
 /*String msg=title;
 title="Dialog window";*/
 //modal(child dominates parent(true) / modalless (both the windows can be
accessed)false)
 super(parent, title, false);
  bld=(ButtonListD)parent;
 setLayout(new FlowLayout());
 setSize(300, 200);
 add(new Label(bld.msg));
 Button b;
 add(b = new Button("OK"));
 b.addActionListener(this);
 }
  public void actionPerformed(ActionEvent ae) {
 dispose();
 }
 /*
 public void paint(Graphics g) {
 g.drawString(bld.msg, 10, 70);
 }
 */
}
```

```
public class ButtonListD extends Frame implements ActionListener {
 String msg = "";
 Button bList[]=new Button[3];
 public ButtonListD() {
 setLayout(new FlowLayout());
 Button yes = new Button("Yes");
 Button no = new Button("No");
 Button maybe = new Button("Undecided");


 //Syntax- Component add(Component compRef)

 bList[0]=(Button) add(yes);
 bList[1]=(Button) add(no);
 bList[2]=(Button)add(maybe);
 for(int i=0;i<3;i++)
 {
 bList[i].addActionListener(this);
 }

 addWindowListener(new MyWindowAdapter());

 }



 public void actionPerformed(ActionEvent ae) {
 for(int i=0;i<3;i++){
 if(ae.getSource()==bList[i])
       {
 msg = "You pressed "+bList[i].getLabel();
 }
 }
  //repaint();
  SampleDialog d = new
 SampleDialog(this, "Dialog");
  d.setVisible(true);
 }

 /*
 public void paint(Graphics g) {
 g.drawString(msg, 20, 100);
 }
 */
 public static void main(String ar[])
 {
       ButtonListD aa=new ButtonListD();
       aa.setSize(new Dimension(230,150));
       aa.setTitle("ButtonListD");
       aa.setVisible(true);
 }
}
```

```
class MyWindowAdapter extends WindowAdapter {
public void windowClosing(WindowEvent we) {
 System.exit(0);
}}
```

Output:





Functionality:

The program features two classes, **ButtonListD** and **SampleDialog**, in Java AWT. **ButtonListD** creates a frame with three buttons: "Yes", "No", and "Undecided". When clicked, these buttons update a message and open a dialog box using **SampleDialog**, which closes upon clicking "OK".

5. DivisionMain.java

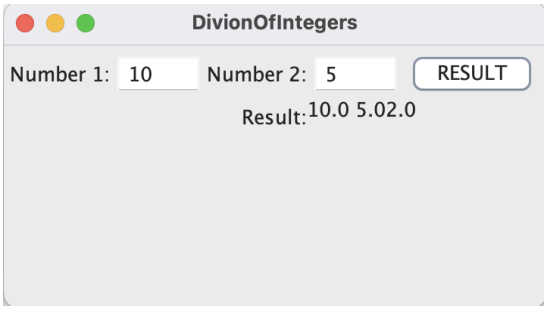
Code:

```
import java.awt.*;
import java.awt.event.*;

public class DivisionMain extends Frame implements ActionListener
{
	TextField num1,num2;
	Button dResult;
	Label outResult;
	String out="";
	double resultNum;
	int flag=0;

	public DivisionMain()
	{
		setLayout(new FlowLayout());

		dResult = new Button("RESULT");
		Label number1 = new Label("Number 1:",Label.RIGHT);
		Label number2 = new Label("Number 2:",Label.RIGHT);
```

```
        num1=new TextField(5);
        num2=new TextField(5);
        outResult = new Label("Result:",Label.RIGHT);

        add(number1);
        add(num1);
        add(number2);
        add(num2);
        add(dResult);
        add(outResult);

        num1.addActionListener(this);
        num2.addActionListener(this);
        dResult.addActionListener(this);
        addWindowListener(new WindowAdapter()
        {
              public void windowClosing(WindowEvent we)
              {
                    System.exit(0);
              }
        });
}

public void actionPerformed(ActionEvent ae)
{
        double n1,n2;
        try
        {
              if (ae.getSource() == dResult)
              {
                    n1=Double.parseDouble(num1.getText());
                    n2=Double.parseDouble(num2.getText());

                    /*if(n2==0)
                          throw new ArithmeticException();*/
                    out=n1+" "+n2;
                    resultNum=n1/n2;
                    out+=String.valueOf(resultNum);
                    repaint();

              }
        }
        catch(ArithmeticException e2)
        {
              flag=1;
              out="Divide by 0 Exception!  "+e2;
              repaint();
        }
        catch(NumberFormatException e1)
        {
              flag=1;
              out="Number Format Exception!  "+e1;
              repaint();
```

```
		}

	}

	public void paint(Graphics g)
	{
		if(flag==0)

g.drawString(out,outResult.getX()+outResult.getWidth(),outResult.getY()+outRes
ult.getHeight()-8);
		else
		g.drawString(out,100,200);
		flag=0;
	}

	public static void main(String[] args)
	{
		DivisionMain dm=new DivisionMain();
		dm.setSize(new Dimension(800,400));
		dm.setTitle("DivionOfIntegers");
		dm.setVisible(true);
	}

}
```

Output:



Functionality:

DivisionMain is a Java GUI application for performing division on floating-point numbers. It features text fields for input, a button to trigger the division, and a label to display the result. The program handles exceptions like division by zero and incorrect number formats, ensuring smooth functionality within a graphical interface.

6. DivisionMain1.java

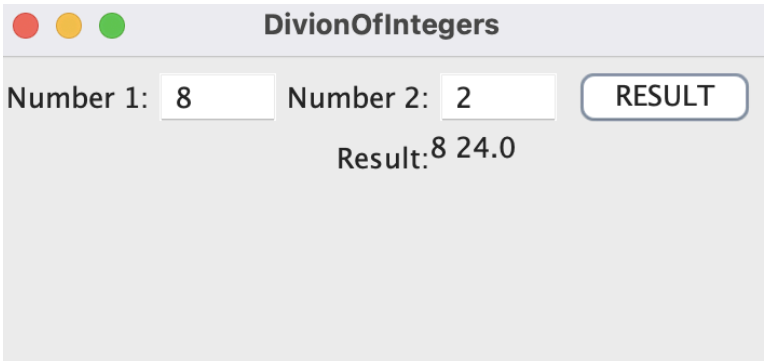
Code:

```
import java.awt.*;
import java.awt.event.*;

public class DivisionMain1 extends Frame implements ActionListener
{
	TextField num1,num2;
	Button dResult;
	Label outResult;
	String out="";
	double resultNum;
	int flag=0;

	public DivisionMain1()
	{
		setLayout(new FlowLayout());

		dResult = new Button("RESULT");
		Label number1 = new Label("Number 1:",Label.RIGHT);
		Label number2 = new Label("Number 2:",Label.RIGHT);
		num1=new TextField(5);
		num2=new TextField(5);
		outResult = new Label("Result:",Label.RIGHT);

		add(number1);
		add(num1);
		add(number2);
		add(num2);
		add(dResult);
		add(outResult);

		num1.addActionListener(this);
		num2.addActionListener(this);
		dResult.addActionListener(this);
		addWindowListener(new WindowAdapter()
		{
			public void windowClosing(WindowEvent we)
			{
				System.exit(0);
			}
		});
	}
```

```
	public void actionPerformed(ActionEvent ae)
	{
		int n1,n2;
		try
		{
			if (ae.getSource() == dResult)
			{
				n1=Integer.parseInt(num1.getText());
				n2=Integer.parseInt(num2.getText());

				/*if(n2==0)
					throw new ArithmeticException();*/
				out=n1+" "+n2;
				resultNum=n1/n2;
				out+=String.valueOf(resultNum);
				repaint();

			}
		}
		catch(NumberFormatException e1)
		{
			flag=1;
			out="Number Format Exception!  "+e1;
			repaint();
		}
		catch(ArithmeticException e2)
		{
			flag=1;
			out="Divide by 0 Exception!  "+e2;
			repaint();
		}

	}

	public void paint(Graphics g)
	{
		if(flag==0)

g.drawString(out,outResult.getX()+outResult.getWidth(),outResult.getY()+outRes
ult.getHeight()-8);
		else
		g.drawString(out,100,200);
		flag=0;
	}
```

```
	public static void main(String[] args)
	{
		DivisionMain1 dm=new DivisionMain1();
		dm.setSize(new Dimension(800,400));
		dm.setTitle("DivionOfIntegers");
		dm.setVisible(true);
	}

}
```

Output:



Functionality:

The program creates a GUI for performing integer division. It takes two integer inputs from the user, performs the division when the "RESULT" button is clicked, and displays the result or exception messages accordingly.

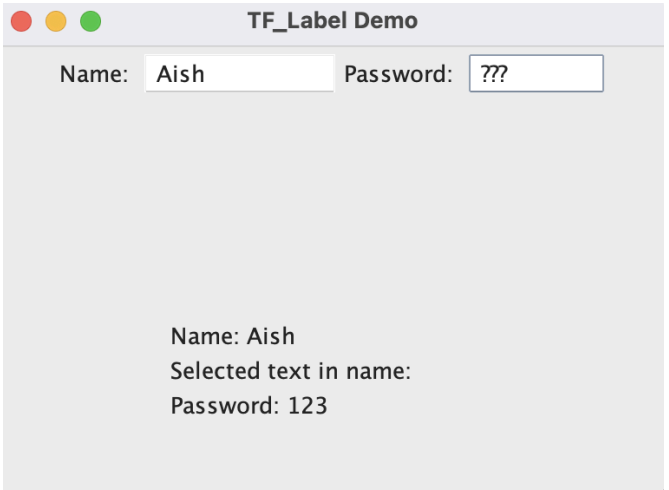
7. TextFieldDemo.java

Code:

```
// Demonstrate text field.
import java.awt.*;
import java.awt.event.*;

public class TextFieldDemo extends Frame
 implements ActionListener {
 TextField name, pass;
 public TextFieldDemo() {
setLayout(new FlowLayout());
 Label namep = new Label("Name: ", Label.RIGHT);
 Label passp = new Label("Password: ", Label.RIGHT);
 name = new TextField(12);
 pass = new TextField(8);
 pass.setEchoChar('?');
```

```
 add(namep);
 add(name);
 add(passp);
 add(pass);
 // register to receive action events
 name.addActionListener(this);
 pass.addActionListener(this);
 addWindowListener(new MyWindowAdapter());
 }
 // User pressed Enter.
 public void actionPerformed(ActionEvent ae) {
 repaint();
 }
 public void paint(Graphics g) {
 g.drawString("Name: " + name.getText(), 100, 200);
 g.drawString("Selected text in name: "
 + name.getSelectedText(), 100, 220);
 g.drawString("Password: " + pass.getText(), 100, 240);
 }
 public static void main(String ar[])
 {
       TextFieldDemo awin=new TextFieldDemo();
       awin.setSize(new Dimension(700,700));
       awin.setTitle("TF_Label Demo");
       awin.setVisible(true);
 }
}

class MyWindowAdapter extends WindowAdapter {
public void windowClosing(WindowEvent we) {
 System.exit(0);
}}
```

Output:



Functionality:

The Java program **TextFieldDemo** creates a graphical user interface (GUI) with two text fields for entering a name and password. It extends the **Frame** class and implements the **ActionListener** interface to handle user input events. The program uses AWT components such as **TextField** and **Label** to create the GUI layout. It registers event listeners for text fields and window closing events. When the user interacts with the text fields, the program updates the display accordingly.

---------- Thank You ----------