→ Java's AWT is responsible for communicating these actions b/w the program and the user.

PAGE:
DATE: / /

Chapter - 5 - (2Q) - 15 marks
Events, Handling Events and AWT / Swing

1) Events. Types of events.

Ans: In Java, an event is an object which specifies the change of state in the source. Events are generated as result of user interaction with the graphical user interface components. For example, clicking on a button, moving the mouse, entering a character through keyboard, selecting an item from the list are the activities that causes an event to happen.

The events are broadly classified into two categories :-

i) Foreground events

→ Those events which require the direct interaction of user. They are generated as consequences of a person interacting with the graphical components in Graphical User Interface. For eg:- clicking on a button, moving the mouse, entering a character through keyboard, selecting an item from the list, scrolling the page, etc.

ii) Background events

→ Those events that require the interaction of end user are known as background events. Operating System interrupts, hardware or software failure, timer expires, ~~etc are the~~ operation completion etc are the example of background events.

Scanned with CamScanner

PAGE :
DATE : / /

2) Event handling models available for event handling.

Ans: Event handling is the mechanism that controls the event and decides what should happen if an event occurs. Java packages such as java.util, java.awt, java.awt.event and javax.swing support event handling mechanism.

Event handling mechanism has the code which is known as event handler that is executed when an event occurs. Java uses the Delegation ~~model~~ Event model to handle the events. This model defines the standard mechanism to generate and handle the events.

It has the following key participants namely:-

i) Source - The source is an object on which event occurs. It is responsible for providing information of the occurred event to it's handler.

ii) Listener - It is also known as event handler. It is responsible for generating response to an event.

→ This approach is an efficient way of handling the event because the event notifications are sent only to those listeners that want to receive them.

Scanned with CamScanner

(11) user interface logic is separated from event handler logic

PAGE :
DATE : / /

Steps involved in event handling

1. The user clicks the button and the event is generated.
2. Now the object ~~is~~ of concerned event class is created automatically and information about the source and the event get populated with in same object.
3. Event object is forwarded to the method of registered listener class.
4. the method ~~is~~ now gets executed and returns.

Examples of Event Handling

i) Action Listener
→ interface
→ used to handle events caused by sources like Buttons, MenuItems, etc.

ii) Item Listener
→ used for RadioButton, list, choice and checkbox.

iii) KeyListener
→ handles events generated from the keys of the keyboard

iv) Mouse Motion Listener
→ handles events generated from Mouse.

v) Mouse Listener
→ ~~mouseClicked, mouseReleased~~ mouseClicked, mouseReleased, etc.

Scanned with CamScanner

PAGE :
DATE : / /

3) Events & Event classes

Ans Changing the state of an object is known as event. for example, click on button, dragging mouse etc. The java.awt.event package provides many event classes and listener interfaces for event handling.

Event Classes and Listener Interfaces

	Event Classes	Listener Interfaces
1.	ActionEvent	ActionListener
2.	MouseEvent	MouseListener and MouseMotionListener
3.	KeyEvent	KeyListener
4.	ItemEvent	ItemListener
5.	TextEvent	TextListener
6.	WindowEvent	WindowListener
7.	ComponentEvent	ComponentListener
8.	ContainerEvent	ContainerListener

Scanned with CamScanner

PAGE :
DATE : / /

4)	Event Class	Description
1.	ActionEvent	Generated when a button is pressed, a menu item is double clicked, item in combo box is clicked.
2.	Component Event	Generated when a component is hidden, moved, resized or shown
3.	Container Event	Generated when a component is added to/removed from a container.
4.	ItemEvent	Generated when a check box, list item or combo box item is clicked.
5.	KeyEvent	Generated when input is received from the keyboard.
6.	MouseEvent	Generated when the mouse is clicked, pressed, or released.
7.	MouseWheelEvent	Generated when the mouse wheel is moved.
8.	TextEvent	Generated when the value of text area or text field is changed.
9.	WindowEvent	Generated when a window is activated, closed, deactivated etc.

Scanned with CamScanner

PAGE :
DATE : / /

MVC - Model View Controller

3) Difference between

① ~~Frame~~ AWT & Swing

AWT	Swing
1. AWT is an API to develop GUI applications in Java.	1. Swing is a part of Java foundation classes and is used to create various applications.
2. AWT components are platform-dependent	2. Java swing components are platform-independent.
3. It's components are heavy weight.	3. It's components are light weight.
4. It doesnot support pluggable look and feel.	4. It supports pluggable look and feel.
5. It provides comparatively less powerful components.	5. It provides more powerful components.
6. It's execution time is more than swing.	6. It's execution time is less than AWT.
7. It doesn't follow MVC.	7. It follows MVC.
8. Programmer has to import the 'javax.awt' package to develop an AWT-based GUI.	8. Programmer has to import 'javax.swing' package to write a Swing application.

Scanned with CamScanner

PAGE :
DATE : / /

iii) Frame & Panel

Frame	Panel
1. It is an awt component which is a top level window with border and title.	1. It is an AWT component which represents a simple container that can attach other GUI components including other panels.
2. A subclass of ~~container~~ window	2. A subclass of container.
3. It has a title bar.	3. It doesnot have a title bar.
4. It has a border	4. It doesnot have a border.
5. It is not possible to add ~~another~~ a frame to another frame.	5. It is possible to add another panel to an existing panel.
6. It is an independent output window.	6. It depends on the frame.
7. It uses BorderLayout as default layout manager.	7. It uses FlowLayout as default layout manager.

Scanned with CamScanner

PAGE :
DATE : / /

Java Swing

Ans. Java Swing is used to create window-based application. It is built on the top of AWT 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.

Scanned with CamScanner

PAGE :
DATE : / /

4) What is a frame? Create a frame with following attributes.
Height = 400
Width = 400
Title = Myframe

Ans: In java, a frame is a window that has borders, various buttons, and other features. A frame is a container object so GUI components can be placed in it.

labels
textfields

Code:

```
// frame in awt

import java.awt.*;

public class frame1 {
        frame f;
    frame1() {
        f= new frame("My frame");
        f.setSize(400,400);
        f.setlayout(null);
        f.setVisible(true);
    }

public static void main(String[] args) {
        frame1 f = new frame1();
    }
}
```

We can only write {new frame();}

Scanned with CamScanner

PAGE :
DATE : / /

or,

~~import java.awt.*;~~

~~class frame1 extends frame{~~
~~frame1() {~~

```
import java.awt.*;

class first extends frame {
first() {
setTitle("My frame");
setSize(400,400);
setLayout(null);
setVisible(true);
}
public static void main(String[] args) {
first f = new first();
}
}
```

Scanned with CamScanner

PAGE :
DATE : / /

5) WAP to create a closable frame in the front and centre of your desktop.

Ans: A JFrame is a subclass of frame class. A JFrame is a like a window with border, title and buttons. We can ~~use~~ implement most of the java swing application using JFrame.

By default, a JFrame ~~is~~ can be displayed at the top-left position of a screen. We can display the center position of JFrame using the ~~[illegible]~~ setLocationRelativeTo() method of window class.

```
import javax.swing.*;
import java.awt.*;

public class Example extends JFrame{
    public Example(){
        setTitle("My frame");
        setSize(400,300);
        setDefaultCloseOperation(JFrame.
                                EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setVisible(true);
    }

public static void main(String[] args){
Example f=new Example();
    }
}
```

Scanned with CamScanner

PAGE :
DATE : / /

11(B) WAP using frame which contains (i) 3 text fields and one button (ii) accept 2 numbers (iii) when user clicks the button, it should calculate the sum of two numbers entered in two textfield and display the result in third textfield.

->

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

class Calculator implements ActionListener {
    Frame f;
    TextField t1, t2, t3;
    Button b1;

    Calculator() {
        f= new Frame("Calculate");
        f.setSize (500,500);
        t1= new TextField("Number1");
        t1.setBounds (100,100,75,25);
        t2= new TextField("Number 2");
        t2.setBounds (100, 150,75,25);
        t3=new TextField("Result");
        t3.setBounds(100,200, 75,25);
        f.add(t1);
        f.add(t2);
        f.add(t3);
        b1= new Button("+");
        b1.setBounds (200,250,50,25);
```

Scanned with CamScanner

PAGE :
DATE : / /

```
f.add(b1);
f.setLayout(null);
f.setVisible(true);
b1.addActionListener(this);
b1.setActionCommand("add");   // can run without this
}
public void actionPerformed(ActionEvent e) {
    int sum=0;
    int a = Integer.parseInt(t1.getText());
    int b= Integer.parseInt(t2.getText());

    sum=a+b;

    String r = String.valueOf(sum);
    t3.setText(r);
  }
public static void main(String[] args) {
    new Calculator();
  }
}
```

Scanned with CamScanner

PAGE :
DATE : / /

19) Create a GUI that contains two button (add and subtract) and three text fields. When the buttons are clicked sum or difference of value of first two fields should be displayed in the third text field respectively.

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

public class AddSubtract implements ActionListener{
    JFrame f;
    JTextfied t1,t2,t3;
    JButton b1,b2;
  AddSubtract(){
    f = new frame();
    t1=new JTextfield();
    t1.setBounds(50,150,150,20);
    t2=new JTextfield();
    t2.setBounds(50,100,150,20);
    t3=new JTextfield();
    t3.setBounds(50,150,150,20);
    b1=new JButton("+");
    b1.setBounds (50,200,50,50);
    b2=new JButton("-");
    b2.setBounds(120,200,50,50);
    ~~b1.add~~
    b1.addActionListener(this);
    b2.addActionListener(this);
```

Scanned with CamScanner

PAGE :
DATE : / /

```
f.add(t1);
f.add(t2);
f.add(t3);
f.add(b1);
f.add(b2);
f.setSize(300,300);
f.setLayout(null);
f.setVisible(true);
f.setDefaultCloseOperation(f.EXIT_ON_CLOSE);
}
public void actionPerformed(ActionEvent e) {
    String s1= t1.getText();
    String s2= t2.getText();
    int a = Integer.parseInt(s1);
    int b = Integer.parseInt(s2);
    int c=0;
    if (e.getSource() ==b1) {
        c=a+b;
    }
    else if (e.getSource()==b2) {
        c=a-b;
    }
String result= String.valueOf(c);
    t3.setText(result);
    }
public static void main(String[] args) {
        new AddSubtract();
    }
}
```

Scanned with CamScanner

PAGE :
DATE : / /

1) WAP to create a frame in swing. The frame should contain three text fields with labels arg1, arg2, and result respectively, a menu called file with sub menus add, subtract and close.

```
import javax.swing.*;
import java.awt.*;

class MenuBar {
    JFrame f;
    JTextfield t1, t2, t3;
    JMenu file;
    JMenuBar mbar;
    JMenuItem add, subtract, close;
  MenuBar() {
    f=new JFrame();
    f.setSize(400,400);
    t1=new JTextfield("arg1");
    t1.setBounds(100,100,75,25);
    t2=new JTextfield("arg2");
    t2.setBounds(100,150,75,25);
    t3=new JTextfield("result");
    t3.setBounds(100,200,75,25);
    f.add(t1);
    f.add(t2);
    f.add(t3);
```

Scanned with CamScanner

PAGE :
DATE : / /

```
mbar = new JMenuBar();
f.setJMenuBar(mbar);
file = new JMenu("file");
mbar.add(file);
add = new JMenuItem("Add");
subtract = new JMenuItem("Subtract");
close = new JMenuItem("Close");
file.add(add);
file.add(subtract);
file.add(close);

f.setLayout(null);
f.setVisible(true);
f.setDefaultCloseOperation(f.EXIT_ON_CLOSE)
}
public static void main(String[] args){
    new MenuBar();
}
}
```

Scanned with CamScanner

PAGE :
DATE : / /

11) What is layout manager? State the diff types of layout manager in swing.

Ans: Layout Managers are used for arranging the components in order. Layout Manager is an interface that is implemented by all the classes of layout manager. There are 6 types of layout manager. They are described below:-

End

11 i) Border layout

= The Border Layout is used to arrange the components in five regions: north, south, east, west and center. Each region/area may contain one component only. It is the default layout of frame or window.

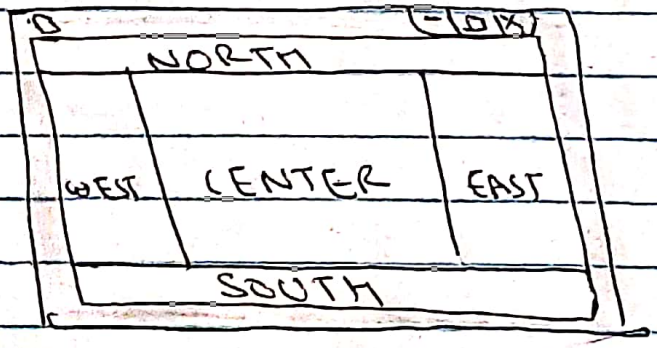


fig:- Border layout

There are 2 types of constructor in Border layout.

1. Border layout() &
2. JBorder layout (int hgap, int vgap)

Scanned with CamScanner

PAGE :
DATE : / /

Example code:

```
import java.awt.*;
import javax.swing.*;

public class Border {
    JFrame f;
    JButton b1,b2,b3,b4,b5;
  Border () {
      f=new JFrame();
      b1=new JButton("NORTH");
      b2=new JButton("SOUTH");
      b3=new JButton("EAST");
      b4=new JButton("WEST");
      b5=new JButton("CENTER");

      f.add(b1, BorderLayout.NORTH);
      f.add(b2, BorderLayout.SOUTH);
      f.add(b3, BorderLayout.EAST);
      f.add(b4, BorderLayout.WEST);
      f.add(b5, BorderLayout.CENTER);

      f.setSize(400,400);
      f.setVisible(true);
      f.setDefaultCloseOperation(f.EXIT_ON_CLOSE);
  }

public static void main(String[] args) {
      new Border();
  }
}
```

Scanned with CamScanner

PAGE :
DATE : / /

Not asked

ii) Flow layout

= Flowlayout is used when we want to arrange the components in a sequence one after another. It is the default layout of applet or panel.

There are 3 types of constructors in Flow layout. They are :-

1. FlowLayout()
2. FlowLayout(int align)
3. FlowLayout(int align, int hgap, int vgap)

Example code :-

```
import java.awt.*;
import javax.swing.*;

public class FlowDemo {
    JFrame f;
    JButton b1,b2,b3,b4,b5;

    FlowDemo() {
        ~~frame~~ = f = new JFrame();

        b1 = new JButton("1");
        b2 = new JButton("2");
        b3 = new JButton("3");
        b4 = new JButton("4");
        b5 = new JButton("5");
```

Scanned with CamScanner

PAGE :
DATE : / /

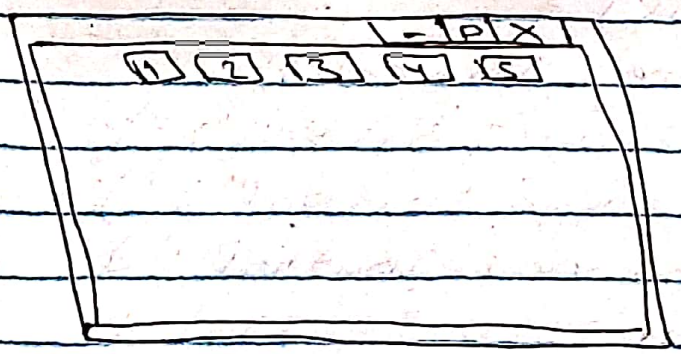
```
f.add(b1);
f.add(b2);
f.add(b3);
f.add(b4);
f.add(b5);
f.setLayout(new FlowLayout(FlowLayout.LEFT));

f.setSize(400,400);
f.setVisible(true);

f.setDefaultCloseOperation(f.EXIT_ON_CLOSE);
}

public static void main(String[] args){
        new FlowDemo();
    }
}
```

Output:-



PAGE :
DATE : / /

iii) Gridlayout

Ans: Gridlayout is used when we want to arrange the component in a rectangular grid.

Types of Constructor:-

1. Gridlayout();
2. Gridlayout(int rows, int columns);
3. Gridlayout(int rows, int columns, int hgap, int vgap)

Example Code:-

```
import java.awt.*;
import javax.swing.*;

public class GridDemo {
    JFrame f;
    JButton b1,b2,b3,b4,b5,b6;
  GridDemo() {
      f = new JFrame();
      b1 = new JButton("1");
      b2 = new JButton("2");
      b3 = new JButton("3");
      b4 = new JButton("4");
      b5 = new JButton("5");
      b6 = new JButton("6");
```

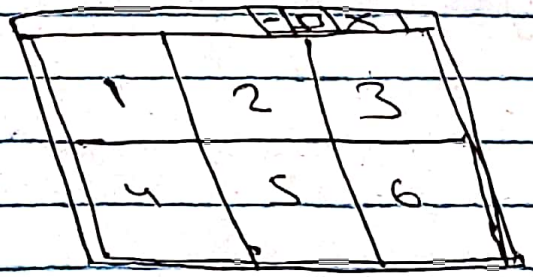
Scanned with CamScanner

PAGE :
DATE : / /

```
f.add(b1);
f.add(b2);
f.add(b3);
f.add(b4);
f.add(b5);
f.add(b6);

f.setLayout(new GridLayout(2,3));
f.setSize(400,400);
f.setVisible(true);
setDefaultCloseOperation(EXIT_ON_CLOSE);
}
public static void main(String[] args){
        new GridDemo();
    }
}
```

output:

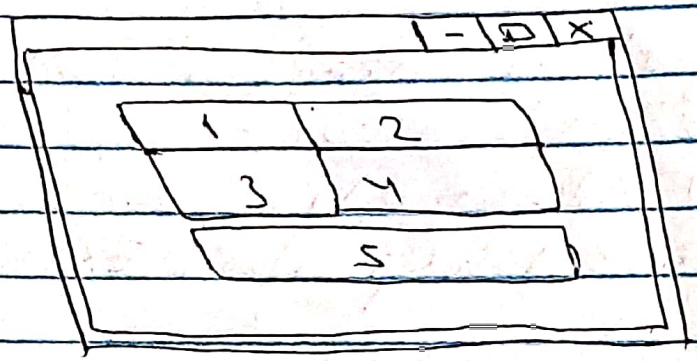


Scanned with CamScanner

PAGE :
DATE : / /

4) GridBag layout

Ans: GridBag layout is used to align components vertically, horizontally or along the baseline. The components may not not be of same size. Each GridBag layout object maintains a dynamic, rectangular gird of cells



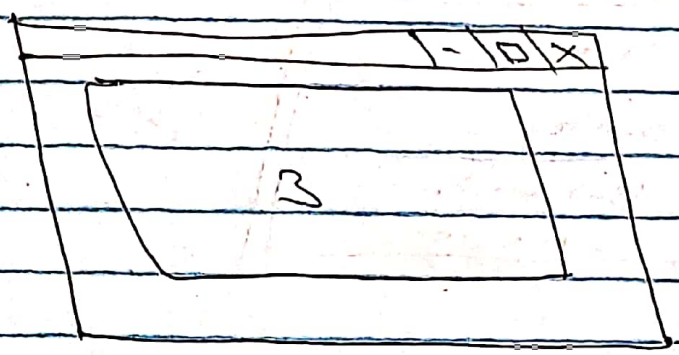
Scanned with CamScanner

PAGE :
DATE : / /

5) Cardlayout

= Cardlayout manages the component in such a manner that only one component is visible at a time. The two 2 types of constructor in Card layout are:-

1. Cardlayout()
2. Cardlayout(int hgap, int vgap)



2
3

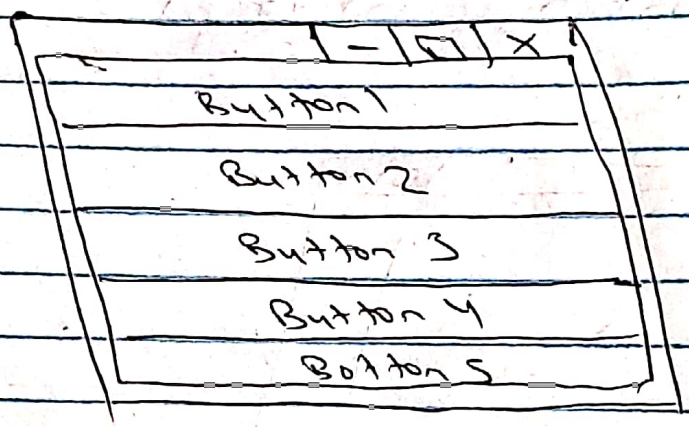
Scanned with CamScanner

PAGE :
DATE : / /

5) Box layout

-> The Box layout is used to arrange the components either vertically or horizontally.

Box layout(Container c, int axis) is the only constructor in Box layout.



Scanned with CamScanner

PAGE :
DATE : / /

1.b) Difference between ~~Gridbag layout~~ Grid layout

Grid layout	Grid Bag layout
1. It puts all the components in a rectangular grid and is divided into equal-sized rectangles and each component is placed inside the rectangle.	1. It is flexible layout manager that aligns the components vertically and horizontally without requiring that the component be of the same size.
2. It is a simple layout manager which arranges components in a grid of rows and columns.	2. It is much more complex and much more flexible than Grid layout.
3. All components are of same size.	3. Size of the components may vary.
4. Fig	4. Fig

Scanned with CamScanner

PAGE :
DATE : / /

13) Define dialogue box? ~~Write~~ Explain it's types? Way to create your own dialogue box.

Ans: The JOptionPane class is used to provide standard dialog boxes. These dialog boxes are used to display information or get input from the user. Unlike frame, it does not have a maximize and minimize buttons.

There are 3 types of dialog boxes and they are :-

i) Message dialog box
ii) Confirm dialog box
iii) Input dialog box

P.T.O →

Scanned with CamScanner

PAGE :
DATE : / /

Example:

```
import javax.swing.*;

public class ~~Option~~ Example {
    JFrame f;
    Example() {
        f=new JFrame();
        JOptionPane.showMessageDialog(f,"Hello");
    }
public static void main(String[] args) {
        new Example();
    }
}
```

Scanned with CamScanner

PAGE :
DATE : / /

Chapter 6
GUI / Animation / Multimedia

1) What is GUI programming?
-> Swing in Java is a Graphical User Interface that includes the GUI components. Swing provides a rich set of widgets and packages to make sophisticated GUI components for Java applications. Swing is a ~~part of~~ API for Java programs that provide GUI.

The Java Swing library is built on top of the AWT, an older, platform dependent GUI toolkit. You can use the Java GUI programming components like buttons, textbox, etc. from the library and do not have to create the components from scratch.

2) What is a Container Class?
-> Container Classes are classes that can have other components on it. So for creating a Java GUI, we need at least one container object. There are 3 types of Java Swing containers.

1. Panel : It is a pure container and is not a window in itself. The sole purpose of a Panel is to organize the components onto to a window.

Scanned with CamScanner

PAGE :
DATE : / /

2. Frame : It is a fully functioning window that its title and icons.

3. Dialog : It can be thought of like pop-up window that pops out when a message has to be displayed. It is not fully functioning window like the frame.

GUI in Java

→ GUI in Java is an easy-to-use visual experience builder for Java applications. It is mainly made of graphical components like buttons, labels, window etc. through which the user can interact with an application. GUI plays an important role to build easy interfaces for Java applications.

Scanned with CamScanner

PAGE :
DATE : / /

V.V. Imp 10 90% sure

1) Create a swing GUI that contains a button and two text fields, when the button is clicked the value of first text field should be checked ~~the~~ and display "odd number" or "even number" in 2nd text field.

```
import javax.swing.*;
import java.awt.event.*;
import java.awt.Font;
import java.awt.Color;
```

{ Color & Font Capital }

```
public class Example implements ActionListener {
    JFrame f;
    TextField t1,t2;
    JButton b1;
    String s;
  Example() {
    f=new Frame();
    t1=new JTextField();
    t1.setBounds(50,150,150,20);
    t2=new JTextField();
    t2.setBounds(50,100,150,20);
    b1=new JButton("Check");
    b1.setBounds(50,200,150,20);
    b1.addActionListener(this);
    f.add(t1);
    f.add(t2);
    f.add(b1);
    f.setSize(300,300);
```

Scanned with CamScanner

PAGE :
DATE : / /

```
f.setLayout(null);
f.setVisible(true);
f.setDefaultCloseOperation(f.EXIT_ON_CLOSE);
}
public void actionPerformed(ActionEvent e) {
    if(e.getSource()==b1) {
        s=t1.getText();
        int n= Integer.valueOf(s);
        if(n%2==0)
        {
            t2.setText("EVEN");
        }
        else {
            t2.setText("ODD");
        }
    }
}
    public static void main(String[] args) {
        new Example();
    }
}
```

Scanned with CamScanner

DATE : / /

2. Create a frame with one button and one text field, when user clicks on the button the text entered on textfield should be changed to uppercase.

```
~~[illegible]~~        ⋮  s=tf.getText();
                       ⋮

public void actionPerformed (ActionEvent e) {
    if (e.getSource() == b1) {
        tf.setText(s.toUpperCase());
    }
}
                       ⋮
```

rest of the process is similar to Q.no.1

PAGE :
DATE : / /

3. Create a swing GUI that contains a text field and a button. When the ~~text~~ button is pressed the content in the textfield should be changed into uppercase and the background color of the text field should be changed.

```
public void actionPerformed(ActionEvent e) {

    if(e.getSource() == b1) {

        s = t1.getText();
        t1.setText(s.toUpperCase());
        t1.setBackground(Color.red);
    }
}
```

Scanned with CamScanner

PAGE :
DATE : / /

4. WAP to create a ~~text~~ text field and a button. When user clicks on the button the color of the text changes to red and font changes to Arial, Italic and size 15.

```
public void actionPerformed(ActionEvent e){
    if(e.getSource()==b1) {
        s=t1.getText();
        Font ft=new Font("Arial",Font.Italic,15);
        t1.setFont(ft);
        t1.getForeground(Color.RED);
    }
}
```

~~To change background color~~

PAGE :
DATE : / /

5. Create a swing GUI which contains a text field (to input radius of circle), a label and a button. When the button is clicked area of circle should be calculated and displayed in the label.

```
public void actionPerformed (ActionEvent e) {
    if (e.getSource() == b1) {
        int r = Integer.parseInt (t1.getText());
        float area = 0;
        area = (22/7) * r * r;
        String s = String.valueOf (area);
        l1.setText(s);
    }
}
```

Scanned with CamScanner

PAGE :
DATE : / /

6 Closable frames using AWT

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

public class Example extends WindowAdapter {
    frame f;
    Example() {
        f = new Frame();
        f.addWindowListener(this);
        f.setSize(400,400);
        f.setLayout(null);
        f.setVisible(true);
    }
    public void windowClosing(WindowEvent e){
        f.dispose();
    }
    public static void main(String[] args) {
        new Example();
    }
}
```

Scanned with CamScanner

PAGE :
DATE : / /

Closable frames using Swing

```
import javax.swing.* ;

public class Example {
    public static void main (String[] args) {
        JFrame f=new JFrame("2018");
        f.setSize (500,500);
        f.setvisible(true);
        f.setlayout(null);
        f.setDefaultCloseOperation(f.
                                EXIT_ON_CLOSE);
    }
}
```

Scanned with CamScanner