interface { }  class { }

Difference between Interface and abstract class.

| Abstract class | Interface |
|---|---|
| i) It may contain abstract or non-abstracts methods. | i) It contains only abstract methods but since Java 8 we can have default and static methods. |
| ii) It doesn't support multiple inheritance. | ii) It supports multiple inheritance. |
| iii) It can provide implementation of interface. | iii) It cannot extend from class or abstract class. |
| iv) It can have final, non-final, static, static members. | iv) It can have final & static members. |
| v) It can have public, private, protected, default scope. | v) All members of a interface have public access specifier. (scope) |
| vi) It may have constructor. | vi) It can't have a construct |

| Abstract class | Interface. |
| --- | --- |
| It provides abstraction from 0 to 100%. | vii) It provides abstraction level of 100%. |

Write Java program that overrides method getUserValue(), and displayUserDetails().

```
interface Parent
{
  void getuser value();
  void display userDetails();
}
class Child implements Parent
{
      void getUser value()
     {  - - -
        - -
     }
      void display User Details()
     {  - - -
        - -
     }
}
```

Q) WAP to ~~user~~ program to take input from console.

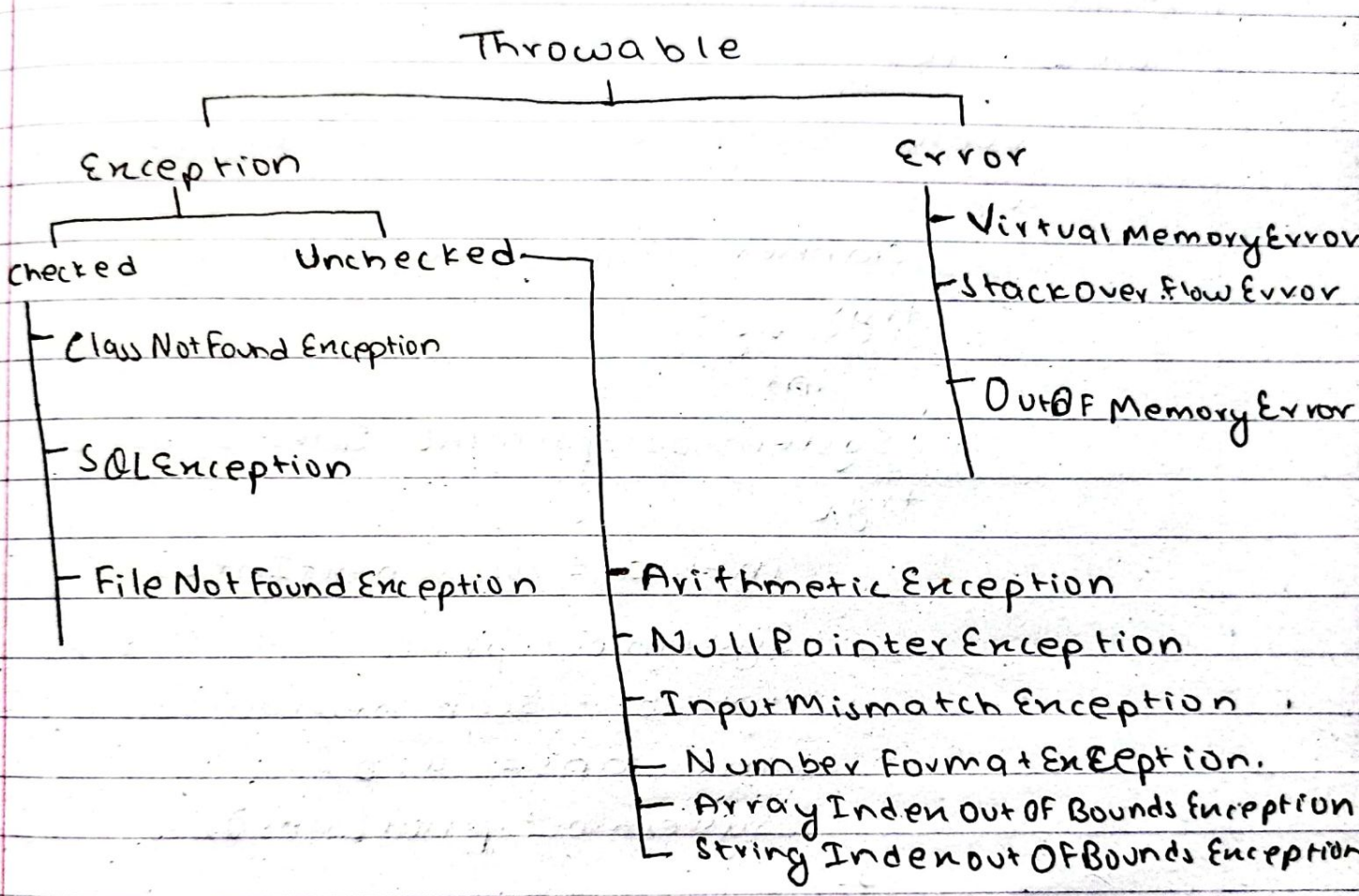
=>

```
import java.util.Scanner;
~~import java.util.Scanner;~~
class Example
{
    public static void main (String [] args)
    {
        Scanner scan = new Scanner(System.in);
        System.out.println(" Enter a string ");
        String s = scan.next();
// OR, scan.nextLine();

        System.out.println(" Enter a num ");
        int a = scan.nextInt();
```

## Exception / Errors

The events that disrupts the normal flow of program.

Exceptions can be recovered while errors can't be.




### Null Pointer Exception

object बनाया तो उसे initialize करते करते (init.. वाला fn call किया करते)

eg. String s;

s.length();

07/29 Thursday

WAP to get two integers from the user and perform division. catch all relevant exceptions.

=>

```
import java.util.*;
class Test
{
    public static void main(String[] args)
    {
        Scanner scan = new Scanner(System.in);
        try{ sy
            int
        System.out.println(" Enter a number");
        try{
            int a = scan.nextInt()
            System.out.print(" Enter 2nd no.");
            int b = scan.nextInt();
            int ans = a/b;
            System.out.print(ans);
        }
        catch(ArithmeticException e)
        {
            System.out.print(e.getMessage());
        }
        catch(InputMismatchException e)
        {
            System.out.print(" Enter only int");
        }
```

```
        finally
        {
            System.out.println("End");
        }
    }
}
```

throw

throws

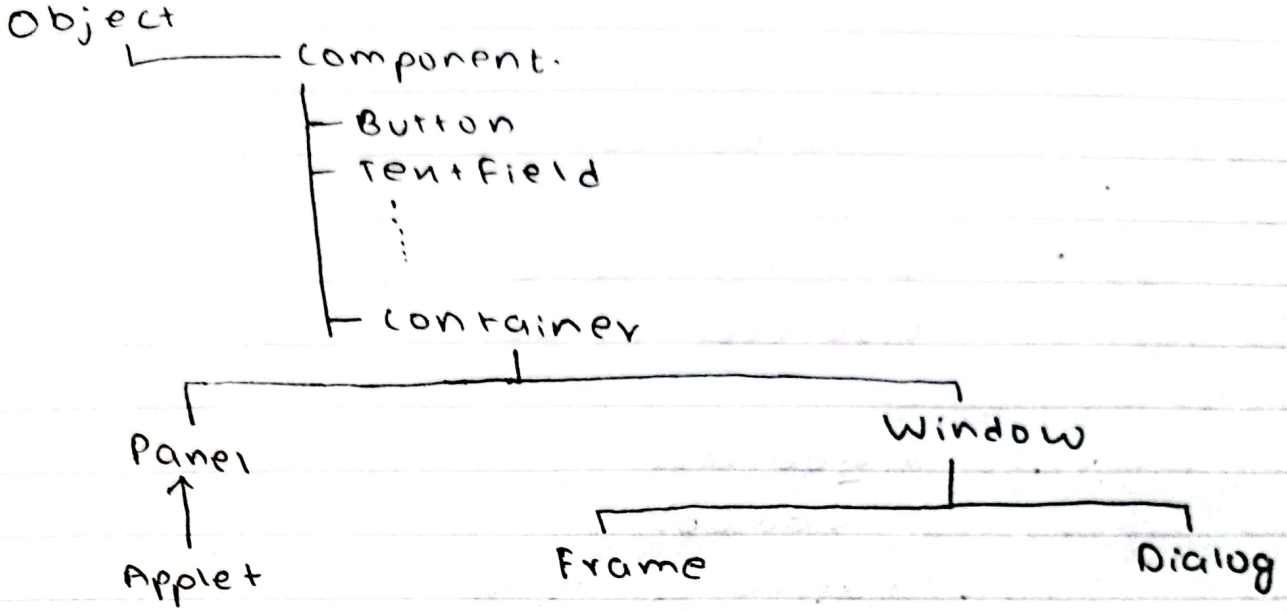
Q. Create a custom exception class SemesterExce-ption which can be thrown when a user tries to initialize semester with an invalid value.

```
class SemException extends Exception
{
    public SemException(String msg)
    {
        super(msg);
    }
}

class Student
{
    String name;
    int sem;
    public Student throws
    public Student(String name, int sem)
    throws SemException
    {
        if(sem<1 || sem>8)
        {
            throw new SemException("Invalid
        }
        else
        {
            this.name = name;
            this.sem = sem;
        }
    }
}
```

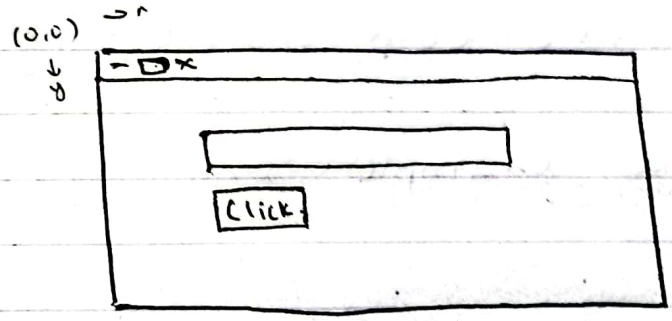
```
class Test
{
    public static void main(String [] args)
    {
        try{
            Student s = new Student("PQR", -3);
        }
        catch (SemException e)
        {
            System.out.print(e.getMessage());
        }
    }
}
```

AWT
Swing

07/30 Friday




Q) Use AWT to create:




```
import java.awt.*;
class Example
{
    Frame f;
    Button b;
    TextField t;

    public Example()
    {
        f = new Frame();
        f.setTitle("Java");
```

```
        f.setSize(500, 600);

        t = new TextField();
        t.setBounds(100, 150, 300, 100);

        b = new Button("click");
        b.setBounds(100, 300, 100, 100)

        f.add(t);
        f.add(b);
        f.setLayout(null);
        f.setVisible(true);
    }
    public static void main(String[] args)
    {
        new Example();

    }
}
```

If.

```
class Example extends Frame
{
    no need no need to write Frame f;
    f.add(t) = add(t)
    :
    -ll-y
}
```

## Using Swing

```
import javax.swing.*;
class Example extends JFrame
{
```

Component को अगाडि J लेख्ने

e.g. Frame = JFrame

TextField = JTextField.

Difference between AWT and Swing

| AWT | Swing |
| --- | --- |
| i) Abstract Window Toolkit | i) It is a part of JFC (Java Foundational Class) |
| ii) It is platform dependent since it, uses components of underlined OS. | ii) It is platform independent and has its own set of components. |
| iii) Its execution speed is slower. | iii) Its execution speed is faster. |
| iv) Components of AWT are heavy weighted. | iv) Components of swing are light weighted. |
| v) It doesn't have pluggable looks and feel. | v) It has pluggable looks & feel. |
| vi) It has limited no. of components compared to swing. | vi) It has extensive no. of components. |
| vii) Components of AWT are less customizable. | vii) components of swing are highly customizable. |

MVC: Model View Controller

| | |
|---|---|
| viii) AWT doesn't support MVC pattern. | viii) Swing supports MVC pattern. |
| ix) It is used to create GUI based application. | ix) It is used to create desktop app as well as web app. |
| x) All components of ~~swing~~ awt are in java awt package. | x) All components of swing are in javax.swing package |

Q Create a GUI application with a text field and a button. When the button is clicked change the text into uppercase.

⇒

```
import javax.swing.*;
import java.awt.event.*;
class Example implements ActionListener
{
    JFrame f;
    JButton b;
    JTextField t;
    public Example()
    {
        f = new JFrame();
        f.setTitle("Java");

        f.setSize(600,500);

        b = new JButton("Click");
        t = new JTextField();

        b.setBounds(150, 200, 150, 100);
        t.setBounds(150, 50, 400, 100);

        f.add(t);
        f.add(b);
        f.setLayout(null);
        f.setVisible(true);
```

```
    b.addActionListener(this);
}
public void actionPerformed(ActionEvent e)
{
        String text = t.getText();
        t.setText(text.toUpperCase());
}
public static void main(String[] args)
{
        new Example();
}
}
```

Q) Create a GUI application with 3 buttons representing your fav. colors. When the button is clicked change the background of the frame to the respective color.

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
class Example implements ActionListener
extends JFrame
{
    JButton b1, b2, b3;

    public Example()
    {
        setSize(600, 500)
        b1 = new JButton("Yellow");
        b2 = new JButton("Pink");
        b3 = new JButton("Blue");

        ~~b1.setB~~
        add(b1);
        add(b2);
        add(b3);
//place buttons horizontally */ setLayout(new FlowLayout())
        setVisible(true);
        setDefault(loseOperation(JFrame
                EXIT_ON_CLOSE);
                static
```

```
        b1.addActionListener(this);
        b2.addActionListener(this);
        b3.addActionListener(this);

    }

    public void actionPerformed(ActionEvent e
    {
        if(e.getSource()==b1)          → for awt
      { // setBackground(Color.YELLOW);

        getContentPane().setBackground(Color.
                                    YELLOW);
    }
    else if(e.getSource()==b2)
    {
        getContentPane().setBackground(
                        (Color.PINK));
    }
    else if(e.getSource()==b3)
    {
        getContentPane().setBackground(
                        Color.BLUE);
    }
}

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

Imp

Q) Create a GUI application with two buttons Red and Green. When the red button is clicked change the color of green to red and vice versa. Also print the message indicating button pressed.

=>

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

class Example implements ActionListener
 extends JFrame
{
        JButton r;
        JButton g;

        public Example()
        {
                setSize (500, 600);

                r= new JButton("Red");
                g= new JButton ("Green");

                add (r);
                add(g);
                set Layout (new FlowLayout());
                set visible(true);
        setDefaultCloseOperation(JFrame.EXIT_ON
                                        CLOSE);
```

```
        r.addActionListener(this);
        g.addActionListener(this);

    }
    @Override
    public void actionPerformed(ActionEvent e)
    {
        if(e.getSource() == r)
        {
            g.setBackground(Color.RED);
            System.out.println("Red is pressed");
        }
        else
        {
            r.setBackground(Color.GREEN);
            System.out.println("Green is pressed");
        }
    }
    public static void main(String[] args)
    {
        new Example();
    }
}


// g.setForeground(Color.RED);
// To change into text.
```

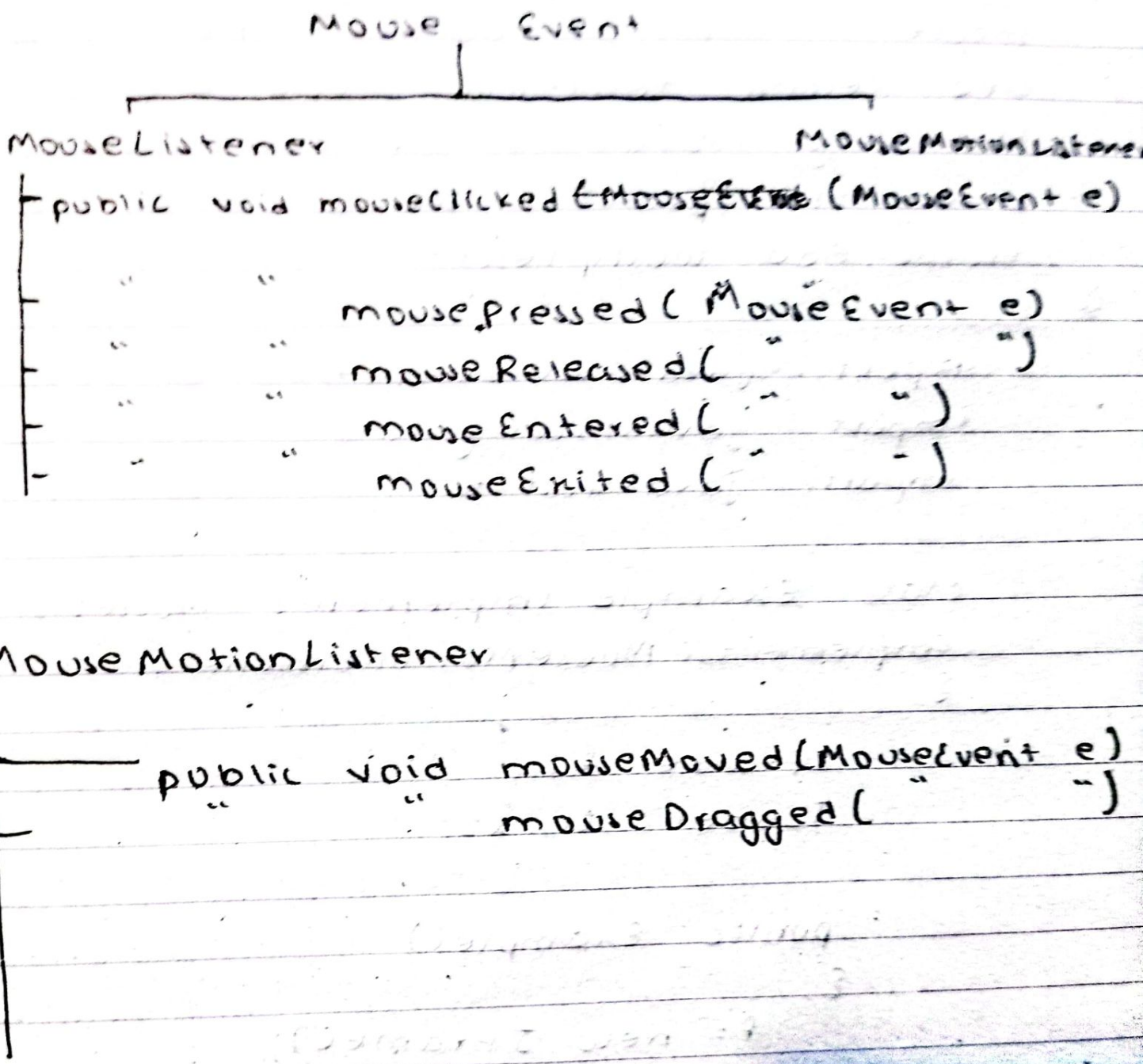
Q. Create a GUI with 3 text fields and 2 buttons (+ & -). Perform + or - based on button clicked and display the result in the third text field. Make the 3rd textfield uneditable.

=>

```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;
class Example implements ActionListener
 extends JFrame
{
        JButton plus, minus;
        JTextField first, second, third;

        public Example()
        {
                first = new JTextField();
                first.setColumns(10);
                second = new JTextField();
                second.setColumns(10);
                third = new JTextField();
                third.setColumns(10);
                third.setEditable(false);

                plus = new JButton("+");
                minus = new JButton("-");
```

```
        add(First);
        add(second);
        add(third);
        Add(plus);
        add(minus);
        pack();
        setLayout(new FlowLayout());
        setVisible(true);
        setDefaultCloseOperation(JFrame.
                                EXIT_ON_CLOSE);

        plus.addActionListener(this);
        minus.addActionListener(this);

    }
    public void actionPerformed(ActionEvent e)
    {
        int a = Integer.parseInt(First.getText()
        int b = Integer.parseInt(second.getText());
        int res = 0;
        if(e.getSource() == plus)
        {
            res = a+b;
            third.setText(b
        }
        else
        {
            res = a-b;
        }
        third.setText(""+res);
    }
    public static void main(String []args)
    {
        new example();
    }
}
```

# Mouse Event


Mouse Event
MouseListener
MouseMotionListener
public void mouseClicked (MouseEvent e)
" " mousePressed ( " " e)
" " mouseReleased ( " " )
" " mouseEntered ( " " )
" " mouseExited ( " " )
MouseMotionListener
public void mouseMoved (MouseEvent e)
" " mouseDragged ( " " )

Q) Create an application with two textfields. The 1st one should display whether the mouse is in or outside the frame and second textfield should display the x,y coordinate of the mouse pointer when the mouse is moved inside the frame. Add tooltiptext.

⇒

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

class Example implements MouseListener,
~~implements~~ MouseMotionListener
{
    JFrame f;
    JLabel l1, l2;

    public Example()
    {
        f= new JFrame();
        f.setSize(500,600);

        l1= new JLabel();
        l2= new JLabel();

        l1.setToolTipText("In/Out");
        l2.setToolTipText("coordinate");
```

```
        f.add(l1);
        f.add(l2);

        f.setLayout(FlowLayout());
        f.setVisible(true);
        f.setDefaultCloseOperation(JFrame.
                                   EXIT_ON_CLOSE);

        f.addMouseListener(this);
        f.addMouseMotionListener(this);
    }
    public void mouseEntered(MouseEvent e)
    {
        l1.setText("In");
    }
    public void mouseExited(MouseEvent e)
    {
        l1.setText("Out");
    }
    public void mouseMoved(MouseEvent e)
    {
        String loc = "X:" + e.getX() + "Y:" +
                             e.getY();
        l2.setText(loc);
    }
}
public static void main(String
    public void mouseClicked(MouseEvent e){}
    public " mouseDragged(MouseEvent e){}
    public void mousePressed(MouseEvent e){}
    public void mouseReleased(MouseEvent e){}
}
```

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

Adaptar Class

Every listener interface has an equivalent class called adapter which provides default implementation of methods of corresponding interface.

Advantage of using adapter class is that we can override only those which are necessary.

08/07 Friday

## LayoutManager

→ LayoutManager allows us to place the components in the container. Unlike using setBounds(), we don't have to specify x, y coordinate, width and height of any component.

→ It is responsive in nature i.e components are allocated and their size are adjusted according to the size of container.

→ Using LayoutManager, makes the code readable.

→ Parent LayoutManager is an interface and all other diff. Layout ~~Manager~~ manager are its children class.

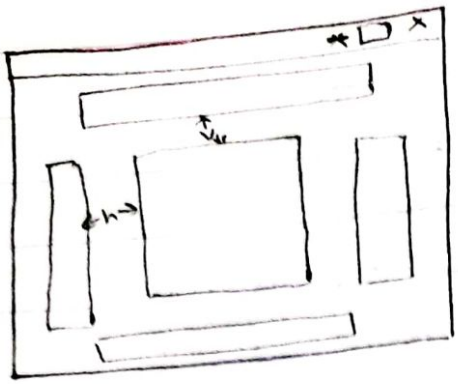
### 1) BorderLayout

→ It is the default layout of frame.

→ It allows us to place the components in five diff regions (East, West, North, South, Center).

**Constructor**

new BorderLayout()



new BorderLayout( int h_gap, int v_gap)

Fields

public final static int NORTH
public final static int SOUTH
" " " " EAST
" " " " WEST
" " " " CENTER

IF a region is not specified, components are placed at the center.

```
import java.awt.*;
import javax.swing.*;
class Example
{
    public Example()
    {
        JFrame f= new JFrame();
        f.setSize(500, 600);

        JButton n = new JButton("North");
        JButton s = new JButton("South");
        JButton e = new JButton("East");
```

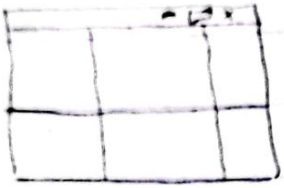
```
        JButton  w = new JButton("West");
        JButton  c = new JButton("Center");

        f.add(n);  f.add(n, BorderLayout.NORTH);
        f.add(s, BorderLayout.SOUTH);
        f.add(e, BorderLayout.EAST);
        f.add(w, BorderLayout.WEST);
        f.add(c);

        f.setVisible(true);
    }

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

}
```

## GridLayout

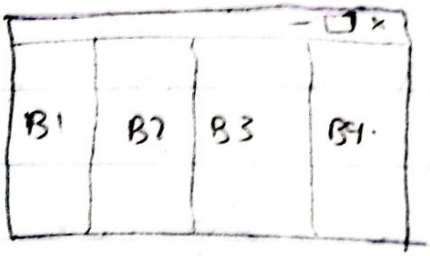
⇒ It is used to place the components in specified rows and columns.

### Constructors

→ new GridLayout( int rows, int columns)

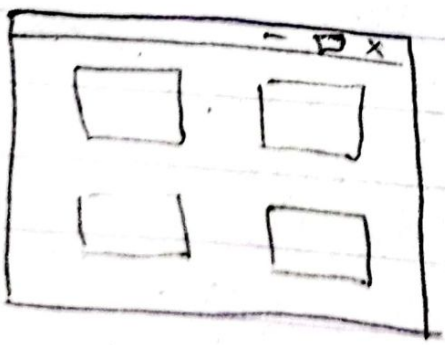


→ new GridLayout( )




If no. of rows and columns are not specified, it place components in separate columns.

→ new GridLayout( int rows, int cols, int h-gap, int v-gap )



```
import java.awt.*;
import javax.swing.*;
class GridDemo
{
    public GridDemo()
    {
        JFrame f = new JFrame();
        f.setSize(600,600);
        JButton [] btn = new JButton[6];
        for(int i=0; i<6; i++)
        {
            btn[i] = new JButton("B"+i+1);
            f.add(btn[i]);
        }
        f.setVisible(true);
        f.setLayout(new GridLayout(2,3));

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

## FlowLayout

→ It is used to place the components horizontally.

→ It provides default spacing of 5 units between the components.

### Constructors

→ new FlowLayout();
- central alignment

→ new FlowLayout(int align);

→ new FlowLayout(int align, int h_gap, int v

### fields

public final static int LEFT
public " " " " RIGHT
" " " " CENTER
" " " " LEADING
" " " " TRAILING

FlowLayout is the defaut layout of the panel.

```
import   java.awt.*;
import   javax.swing.*;
class    FlowDemo
{
        public   FlowDemo
        {
                JFrame   f = new JFrame();
                f.setSize(500, 500);
                JButton   B1 = new JButton("B1");
                JButton   B2 = new JButton("B2"
                f.add(B1);
                f.add(B2);
                f.setVisible(true);
                f.setLayout(new FlowLayout(
                                        FlowLayout.LE
        }
        public   static   void   main(String []
        {
                new   FlowDemo();
        }
}
```

## BoxLayout

→ It is used to place the components either horizontally or vertically.

## Constructors

→ new BoxLayout ( Container c, int axis)

## Fields

i) public static final int X_AXIS
ii) " " " " Y_AXIS

- i places components horizontally
  ii places components vertically.

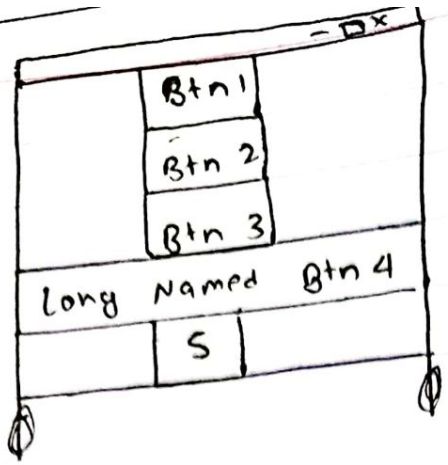
iii) " " " " LINE_AXIS

→ It places the components in the same manner the words are placed as in sentence as per the orientation property of the container.

iii) public static final int PAGE_AXIS

→ It places the components (lines) in the same manner sentences are put in a page.

```
import java.awt.*;
import javax.swing.*;
class BoxDemo
{
    public BoxDemo()
    {
        JFrame F = new JFrame();
        F.setSize(400, 500);
        JButton B1 = new JButton("B1");
        JButton B2 = new JButton("B2");

        f.add(B1);
        f.add(B2);
        F.setVisible(true);
        F.setLayout(new BoxLayout(
                    f.getContentPane(), BoxLayout.Y_AX
    }
    public static void main(String [] args)
    {
        new BoxDemo();
    }
}
```

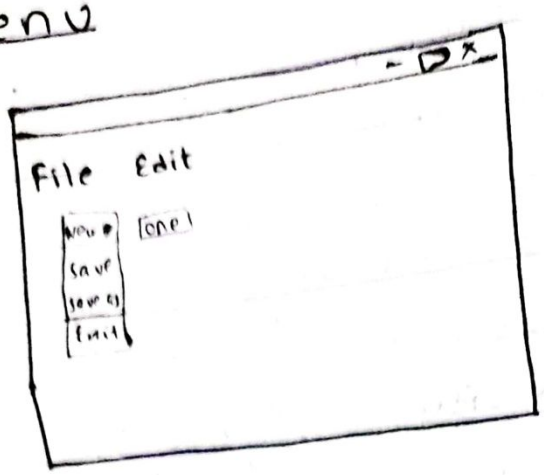



⇒

```
import javax.swing.*;
import java.awt.*;
class Box extends JFrame
{
    public Box()
    {
        JButton one = new JButton("Btn 1");
                  two =      "      ("Btn 2");
                  three =    "      ("Btn 3");
                  four =     "      ("Long
                                Named Btn 4");
                  five = new JButton("5");

        add(one);
        add(two);
        add(three);
        add(four);
        add(five);
        one.setAlignmentX(Component.CENTER_
                          ALIGNMENT);
        two.setAlignmentX(Component.CENTER_
                          ALIGNMENT);
```

```
three.setAlignmentX(Component.CENTER_ALIGNMENT);
four.setAlignmentX(Component.CENTER_ALIGNMENT);
five.setAlignmentX(Component.CENTER_ALIGNMENT);

setLayout(B.getContentPa
setLayout(new BoxLayout(getContentPane(),
                        BoxLayout.Y_AXIS));
setVisible(true);  pack();
}
public static void main(String[]args)
{
    new Box()
}
}
```

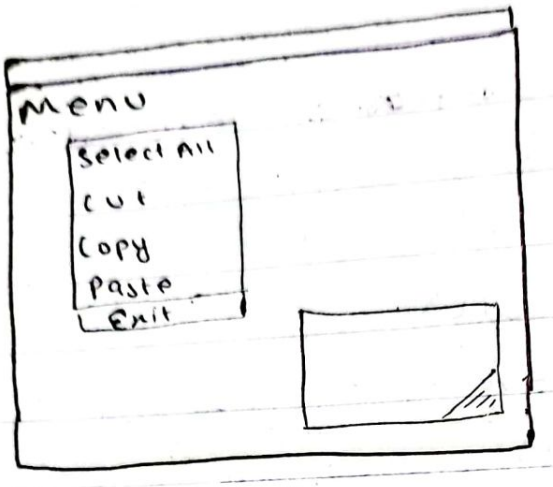
# Menu




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

class MenuEx extends JFrame
{
    public MenuEx()
    {
        setSize(400,500);
        JMenuBar bar = new JMenuBar();
        setJMenuBar(bar);

        JMenu file = new JMenu("File");
        bar.add(file);

        JMenu Edit = new JMenu("Edit");
        bar.add(Edit);

        JMenu n = new JMenu("New");
        file.add(n);
```

```
JMenuItem one = new JMenuItem("one");
n.add(one);

JMenuItem save = new JMenuItem("save");
file.add(save);

JMenuItem saveAs = new JMenuItem("saveas");
file.add(saveAs);

file.addSeparator();
JMenuItem exit = new JMenuItem("Exit");
file.add(exit);
setVisible(true);
}
public static void main(String [] args);
{
    new MenuEx();
}
}
```

Q. Create a GUI application having select All, cut, copy, paste, exit as items and add a text area. Also add functionalities to each item.




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

class MenuEx extends JFrame ~~exten~~
 implements ActionListener
{
    JTextArea ta;
    JMenu menu;
    JMenuItem sa, cut, cp, p, exit;
    JMenuBar bar;
    public MenuEx()
    {
        ~~setSize~~(
        setSize(500, 600);
```

```
bar = new JMenuBar()
setJMenuBar(bar);
menu = new JMenu("Menu");
bar.add(menu);

sa = new JMenuItem("Select All");
cut = new JMenuItem("Cut");
cp = new JMenuItem("Copy");
p = new JMenuItem("Paste");
exit = new JMenuItem("Exit");

menu.add(sa);
menu.add(cut);
menu.add(cp);
menu.add(p);
menu.addSeparator();
menu.add(exit);

ta = new JTextArea("Type here");
ta.setBounds(150, 150, 200, 200);
add(ta);

setLayout(null);
setVisible(true);

sa.addActionListener(this);
cut.addActionListener(this);
cp.addActionListener(this);
p.addActionListener(this);
```

```
exit.addActionListener(this);

}

public void actionPerformed(ActionEvent e
{
    String item= e.getActionCommand();
    if(item.equals("Select All"))
    {
        ta.selectAll();
    }
    else if(item.equals("Cut"))
    {
        ta.cut();
    }
    else if(item.equals("Copy"))
    {
        ta.copy();
    }
    else if(item.equals("Paste"))
    {
        ta.paste();
    }
    else
    {
        System.exit(0);
    }
}
Public static void main(String []args)
{
    new menuEx();
}
```

Q. Create a popup menu with few items, add a label. The menu should appear whereever the user clicks inside the frame. Also display the items being selected in the label.

⇒

```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;
class Popup implements ActionListener,
        MouseListener
{
    JFrame f;
    JLabel l;
    JPopupMenu pop;
    JMenuItem one, two;

    public Popup()
    {
        f = new JFrame();
        f.setSize(400,500);

        l = new JLabel();
        l.setColumns(15);

        f.add(l);
        f.setLayout(new FlowLayout());
        f.setVisible(true);
```

```
pop = new JPopupMenu();
f.add(pop);

one = new JMenuItem("One");
two = new JMenuItem("Two");
pop.add(one);
pop.add(two);

one.addActionListener(this);
two.addActionListener(this);

f.addMouseListener(this);
}

public void mouseClicked(MouseEvent e)
{
    pop.show(f, e.getX(), e.getY());
}

public void actionPerformed(ActionEvent
{
    if(e.getSource() == one)
    {
        l.setText("One is selected");
    }
    else
    {
        l.setText("Two is selected");
    }
}
```

```
public void mousePressed(MouseEvent e){ }
public void mouseReleased(MouseEvent e){ }
public void mouseEntered(MouseEvent e){ }
public void mouseExited(MouseEvent e){ }

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

08/12 Wednesday

WindowListener

1. public void windowOpened(WindowEvent e)
   public void windowClosing(WindowEvent e)
2. 
3. public void windowClosed(WindowEvent e)
4. public void windowActivated(WindowEvent e)
5. public void windowDeactivated(WindowEvent e)
6. public void windowIconified(WindowEvent e)
2. public void windowDeiconified(WindowEvent e)

1) This method is called window is opened at first. It is called only once.

2) This method is invoked when the window is in the process of being closed.

3) This method is called when the window is closed.

4) This method is called when window is in foreground or in active state.

5) This method is called when window is in background.

6) This method is called when window is minimized

7) This method is called when window is maximized.

Q) create a closable frame in awt.

A)

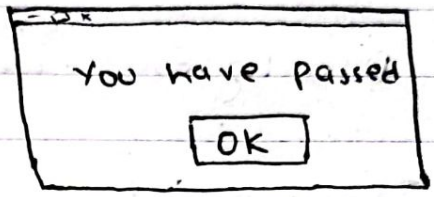
```
import java.awt.*;
import java.awt.event.*;

class Closable implements WindowListener
{
        Frame f;
        public Closable()
        {
                f = new Frame();
                f.setSize(500,600);
                f.setVisible(true);
                f.addWindowListener(this);

        }
        public void windowClosing(WindowEvent e
        {
                System.exit(0);
        }
        public void windowClosed(WindowEvent e){}
        public void windowOpened(WindowEvent e){
        public void windowActivated(WindowEvent e)
```

```
public void windowDeactivated(WindowEvent e)
public void windowIconified(WindowEvent e)
public void windowDeiconified(WindowEvent e)

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

## JOptionPane

It is a child of Component class that is used to display dialogBox or popup alert.

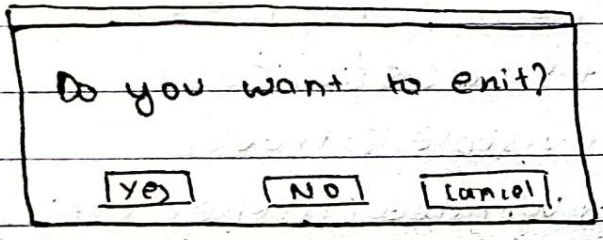
## MessageDialog




JOptionPane.showMessageDialog(Container c, String msg, String title, int icon)

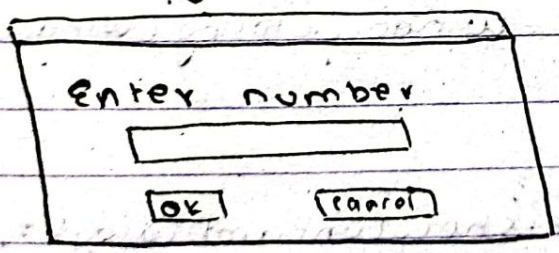
## Confirmation Dialog

(Returns Integer)




int choice = JOptionPane.showConfirmDialog(Container c, String msg)

## Input Dialog




It returns value of textfield as a String.

String val = JOptionPane.showInputDialog(Container c, String msg,

create a frame in swing, display confirmation box when user tries to close the frame

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

class DialogEx extends ~~implements~~ WindowAdapter
{
    JFrame f;

    public DialogEx()
    {
        f = new JFrame();
        f.setSize(500,600);
        f.setVisible(true);
        f.addWindowListener(this);
        f.setDefaultCloseOperation(JFrame.DO_NOTHING_M
    }

    public void windowClosing(WindowEvent e)
    {

int c = JOptionPane.showConfirmDialog(f, "Do you
            want to exit?");

    if(c == JOptionPane.YES_OPTION)
    {
    // System.exit(0);
    f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE
    }
```

```
}

public static void main(String [Jargs)
{
    new DialogEn();
}
}
```

Q) Create a GUI application with a textfield and a button, when the button is pressed do the following:

a) Change the color of text to Red.
b) Change the font of text to Arial.
c) Change the style of text to bold.
d) Change the font of text to 20 points.

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

class Example extends JFrame implements ActionListener
{
JTextField t;
    JButton b;
    public void showGUI()
    {
        setSize(500,600);
```

```
        t = new JTextField();
        b = new JButton("Click");
        add(t);
        add(b);
        setVisible(true);
        setLayout(new FlowLayout());
        b.addActionListener(this);
    }

    public void actionPerformed(ActionEvent e)
    {
        t.setForeground(Color.RED);
        Font font = new Font("Arial", Font.BOLD,
        t.setFont(font);
    }

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

# JavaFx

- JavaFx is a comprehensive set of graphics and media package bundled with JavaSE.
- It is used to create GUI application and RIA (Rich Internet Application).
- It uses FXML (Declarative Markup Language) to create and style UI components.

## Features of JavaFx

### FXML

- It is a declarative markup language used to create and style UI elements.

### Seen Builder

- It allows us to create UI components and style them and generates FXML which can be ported to any IDE.
- It provides high performance hardware accelerated graphic pipeline to render 2D and 3D elements.
- It supports web view which enables us to embed HTML, CSS, JS, SVG, etc.
- It allows CSS for styling UI elements.
- It allows audio and video multimedia rendering with low latency (delay).
- MVC pattern is supported (Model view Control).

- It has ensured extensive set of highly customizable UI elements
- Since, it is a Java Library, it ~~inherent~~ inherits all properties of Java by default.
- Built-in

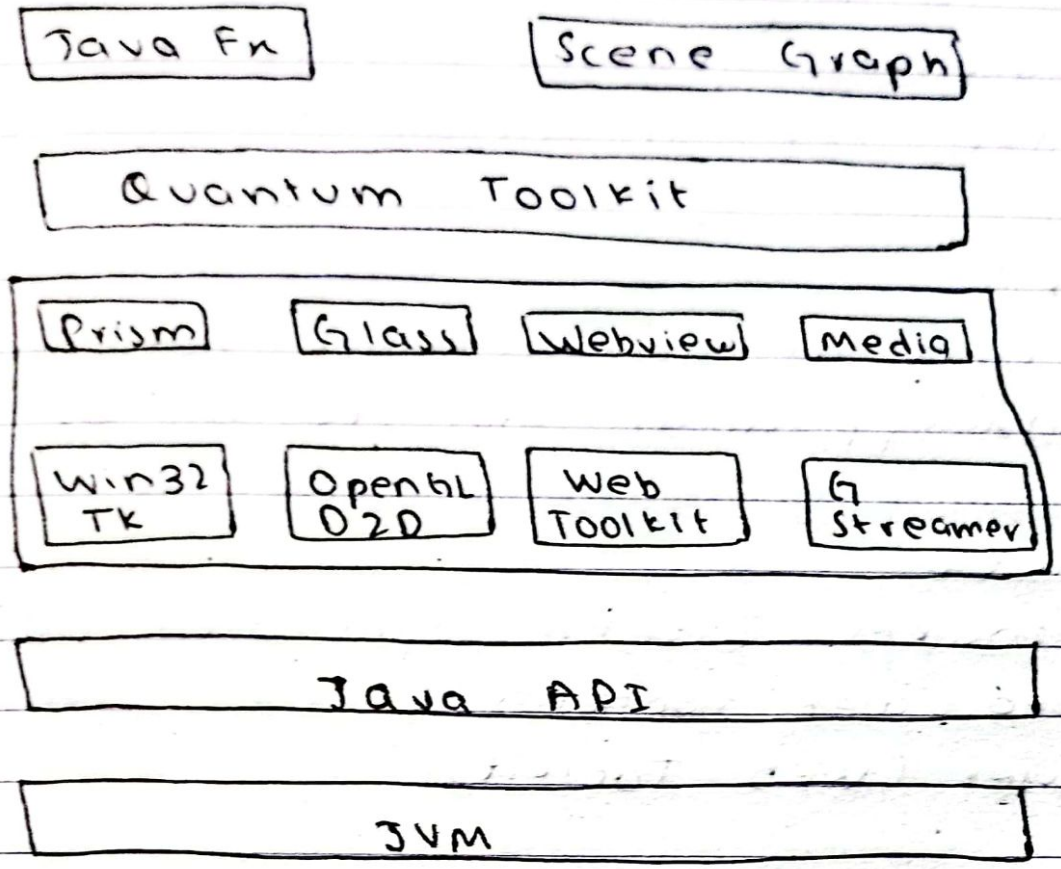
Storing Interoperability

- ~~If~~ Swing application can be converted into v application and Javafx elements Javafx can be embedded into swing application.

Built-in UI Controls

- It has own set of UI controls and is independent of OS (underlying OS).

# JavaFx Architecture




## Scene Graph

It is the starting point of any JavaFx application.
It is the hievarchical ~~tree~~ tree structure of nodes that represents all the visual elements of UI.
It is also reponsible for Event handling

## Quantum Toolkit

It combines prism and glass windowing toolkit and make them available to be used in upper level of stack.

Prism

→ It is a high performance hardware accelerated graphic pipeline that renders 2D, 3D animations and Fx.

Glass

→ It acts as an interface between JavaFx application and native OS.

Webview

→ It makes JavaFx application able to embed web contents. It uses open source browser (Web Toolkit).

Media

→ It uses open source G streamer engine to render audio and video media

Q. Create a JavaFx application with a label and Button. When button is pressed, display a string in label

```
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.layout.*;
import javafx.scene.control.*;
import javafx.event.*;

class First extends Application.
{
    @Override
    public void start(Stage s) throws Exception
    {
        Label l = new Label();
        Button b = new Button("Click");
        HBox root = new HBox();
        root.getChildren().addAll(l,b);
        Scene scene = new Scene(root, 300,300);
        s.setScene(scene);
        s.show();

        b.setOnAction(e -> l.setText("ncit"));
    }

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

Sunday

Create an application in JavaFX with a textfield and two buttons. The first button should change the text in uppercase and another should change in lowercase.

=>

```
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.layout.*;
import javafx.scene.control.*;
import javafx.event.*;

class Example extends Application
{
   public void start(Stage S) throws
    Exception
   {
      TextField t = new TextField();
      Button upper = new Button("Upper");
      Button lower = new Button("Lower");
     HBox root = new HBox();
     root.getChildren.addAll(t,upper,lower
     Scene scene = new Scene(root, 300,
     S.setScene(scene);
     S.show();

     upper.setOnAction(new EventHandler<Acti
                        -Event>()
```

```
        public void handle(ActionEvent e)
        {
            t.setText(t.getText().toUpperCase());
        }
    });

//  lower.setOnAction(new Even

    lower.setOnAction(e -> t.setText(t.getText().
                                toLowerCase()));
}

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

Q. Create a JavaFx application with two labels. The first one should display whether the mouse is in or outside the scene. The second one should display x and y coordinate of the pointer when it is moved inside the scene.

=>

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.stage.Stage;
import javafx.scene.layout.*;
import javafx.event.*;
import javafx.scene.control.*;

class MouseExample extends Application
{
    public void start(Stage s) throws Exceptio
    {
        Label l1 = new Label();
        Label l2 = new Label();

        HBox root = new HBox()
        root.getChildren().addAll(l1, l2);
        Scene scene = new Scene(root, 400
        s.setScene(scene);
        s.show();
```

```
scene.setOnMouseEntered(new EventHandler
        <MouseEvent>()
    {
        public void handle(MouseEvent e)
        {
            l1.setText("In");
        }
    });
```

~~scene.setOnMouseExited(e -> l1.setText("Out")~~

```
scene.setOnMouseExited(e -> l1.setText("out"));
```

```
scene.setOnMouseMoved(e ->{
                String loc = "x: " + e.getX()
                    + "y: " + e.getY();
            l2.setText(loc);
            });
    }
static public void main(String[] args)
    {
        launch(args);
    }
}
```

convenience method

Imp
Q.

# Layouts in JavaFx

## BorderPane

This layout allows the nodes to be placed in five different regions ie. top, bottom, left, right, center.

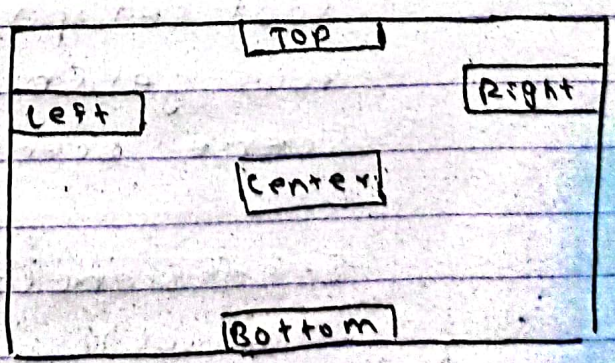
We can place a node on given directions using following methods:

a) setTop()
b) setBottom()
c) setLeft()
d) setRight()
e) setCenter()

```
import javafx.application.Application;
import javafx.stage.Stage();
import javafx.scene.Scene;
import javafx.scene.layout.*;
import javafx.scene.control.*;

class Example extends Application
{
    public void start(Stage s) throws
    Exception
    {
```

```
Button top = new Button("Top");
Button bottom = new Button("Bottom");
Button left = new Button("Left");
Button right = new Button("Right");
Button center = new Button("center");

BorderPane root = new BorderPane();
root.setTop(top);
root.setBottom(bottom);
root.setLeft(left);
root.setRight(right);
root.setCenter(center);

Scene scene = new Scene(root, 200, 400);
scene.show()
S.setScene(scene);
S.show();
}
public static void main(String []args)
{
    launch(args);
}
}
```

Output:



2) HBox

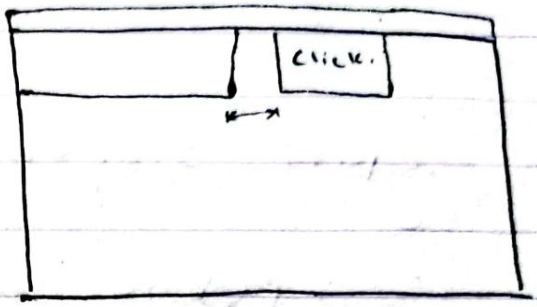
- This layout places the nodes horizontally in x axis
- By Default, there is no space between the nodes.
- We can provide spacing between the nodes by following methods:

```
public void setSpacing(double d)
```

Example

```
class HBoxDemo extends Application
{
    public void start(Stage s) throws
    Exception
    {
        TextField t = new TextField();
        Button b = new Button("Click");

        HBox root = new HBox();
        root.setSpacing(10.5);
        root.getChildren().addAll(t, b);
        Scene scene = new Scene(root,
                                300, 300);
        s.setScene(scene);
        s.show();
    }
```

```
public static void main(String s)
{
    launch(args);
}
}
```



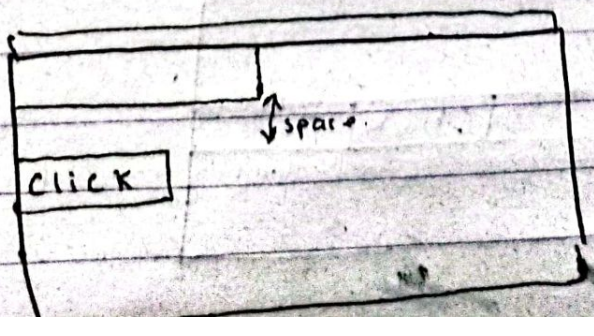

3) VBox

- This layout is used to place the nodes vertically.
- By default no spacing is between the nodes.
- We can provide space using the method:

```
public void setSpacing(double d)
```

```
class VBoxDemo extends Application
{
    :
    // Instead of HBox write VBox
    // other is same as HBox layout.
}
```

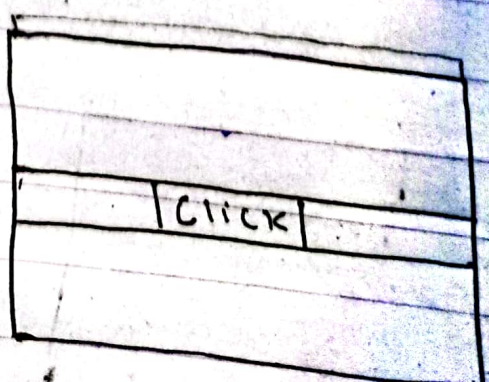
Output



4) StackPane

This layout places the nodes one on top of another as a stack.

```
class StackDemo extends Application
{
    public void start(Stage s) throws
    Exception
    {
        TextField t = new TextField();
        Button b = new Button("Click");

        StackPane root = new StackPane();
        root.getChildren().addAll(t,b);

        Scene scene = new Scene(root,500
        s.setScene(scene);
        s.show();
    }
static public void main(String [] args)
    {
        launch(args);
    }
}
```



5) GridPane

This layout allows us to place multiple nodes at multiple rows and columns.

By default, no spacing is between the nodes.

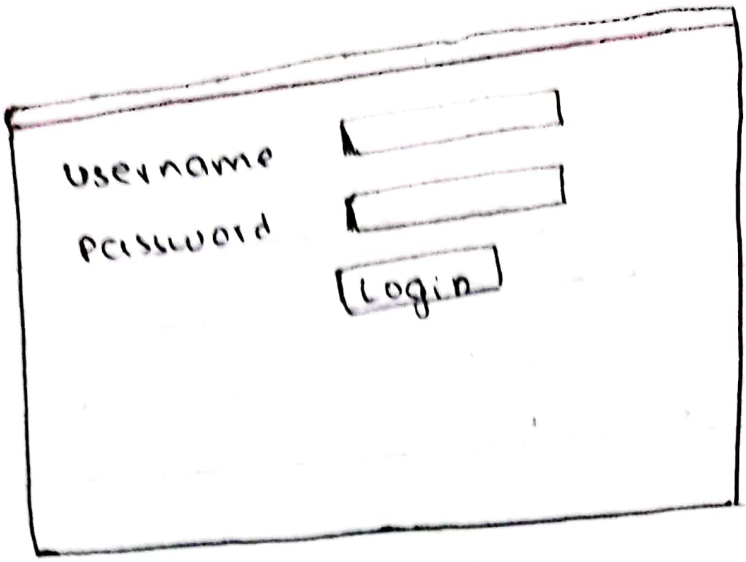
Methods

- public void vgap(double d)
  It allows us to specify vertical space between the nodes.

- public void hgap(double space)
  It allows us to specify horizontal space between the nodes

- public void gridLines(boolean b)
- It displays the gridlines which can be helpful while testing the position
- By default, it is false.

6)




```
class GridDemo extends Application
{
    public void start (Stage s) throws Exception
    {
        Label u = new Label ("Username");
        Label p = new Label ("Password");
        TextField uname = new
                          TextField ();
        PasswordField pwd = new PasswordField

        Button login = new Button ("Login

        GridPane root = new GridPane ();
        root.hgap(5);
        root.vgap(10);

        root.addColumn(0, u, p);
        root.addColumn(1, uname, pwd, logi

        Scene scene = new Scene (root, 500, 600);
        s.setScene(scene);
        s.show();
    }
```

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

6) Flow Pane

| Swing | JavaFx |
| --- | --- |
| i) Swing is a desktop focused GUI Library | i) It is a cross platform GUI library that works in web, desktop and mobile environments / applications. |
| ii) It partially supports MVC pattern. | ii) It completely supports MVC pattern. |
| iii) It has more compo-nents. | iii) It has components that are highly customizable |
| iv) It doesn't support hardware accelerated graphics. | iv) It supports high per-formance ~~highly~~ hardware accel-erated graphics pipelin |
| v) It has limited support for audio and video media. | v) It uses open source G streamer media engine to render aud and video multimedi |
| vi) It is straight forward and easy to use. | vi) It has steeper lea-ning curve. |

## JDBC : Java Database Connectivity

→ It is a java API that allows us to connect java application with database, execute query and fetch response

→ All the classes and interfaces of JDBC are in ~~Ja~~ java.sql package.

Imp

### JDBC Driver

It is a software that enables java application to communicate with vendor database.

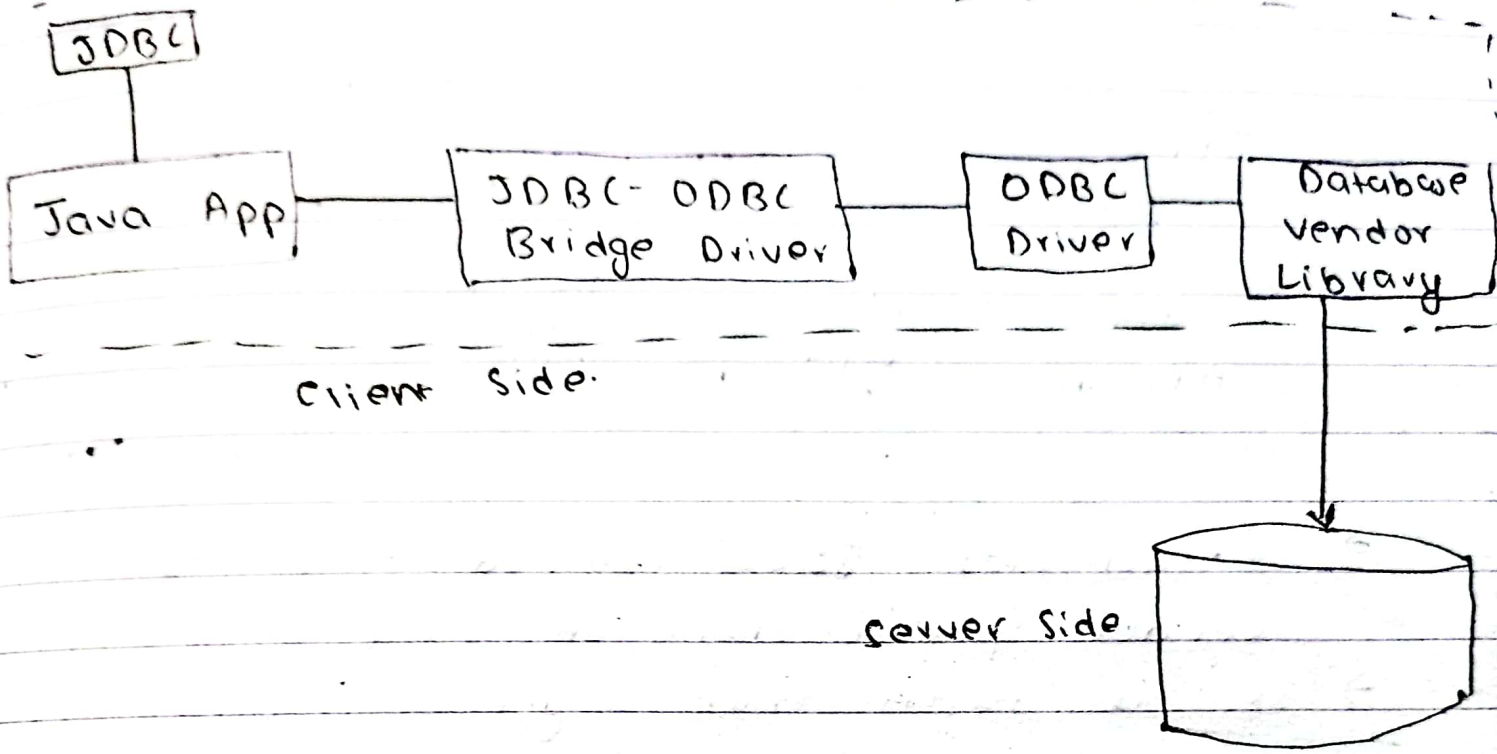
### Types

1) JDBC - ODBC Driver (Type 1)
2) Native API Driver (Type 2)
3) Network Protocol Driver (Type 3)
4) Thin Driver (Type 4)

### 1) JDBC - ODBC Driver

→ It uses ODBC driver.
→ ODBC is written in C, C++.
→ JDBC - ODBC Bridge Driver is used to convert Java method calls to ODBC

function calls; which then is converted to vendor specific library queries.




Advantages

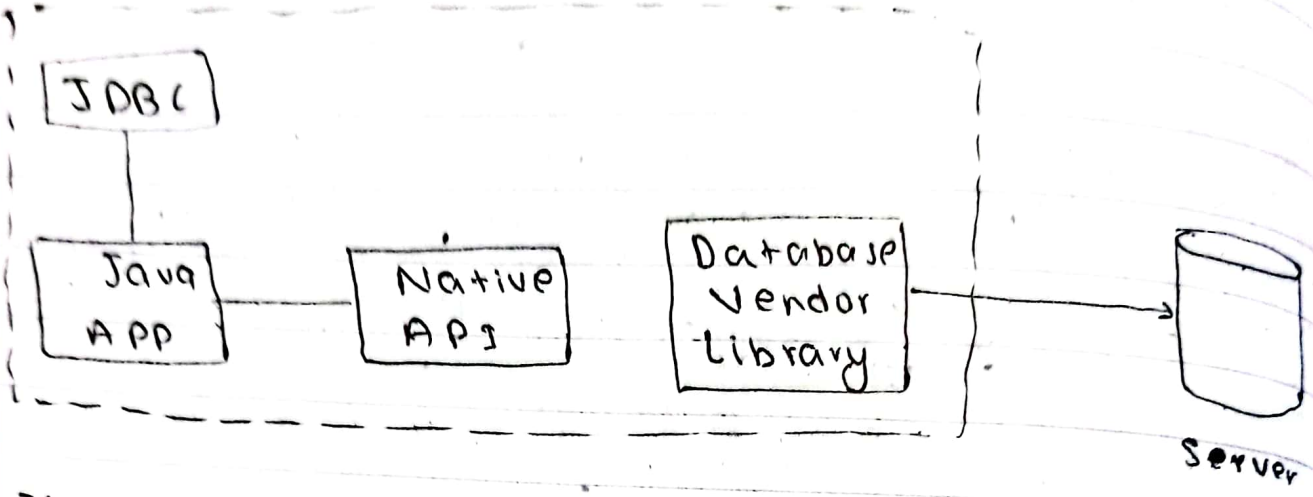
→ It is easy to use.
→ It can connect to any database

Disadvantages

→ Its performance is downgraded since every method call has to be converted to ODBC function calls.
→ Client has to install Bridge driver, ODBC driver and vendor library

2) Native API driver

This driver is partially written in Java.




→ It converts Java method calls to vendor specific function calls.

→ It is faster than type 1.

Advantage

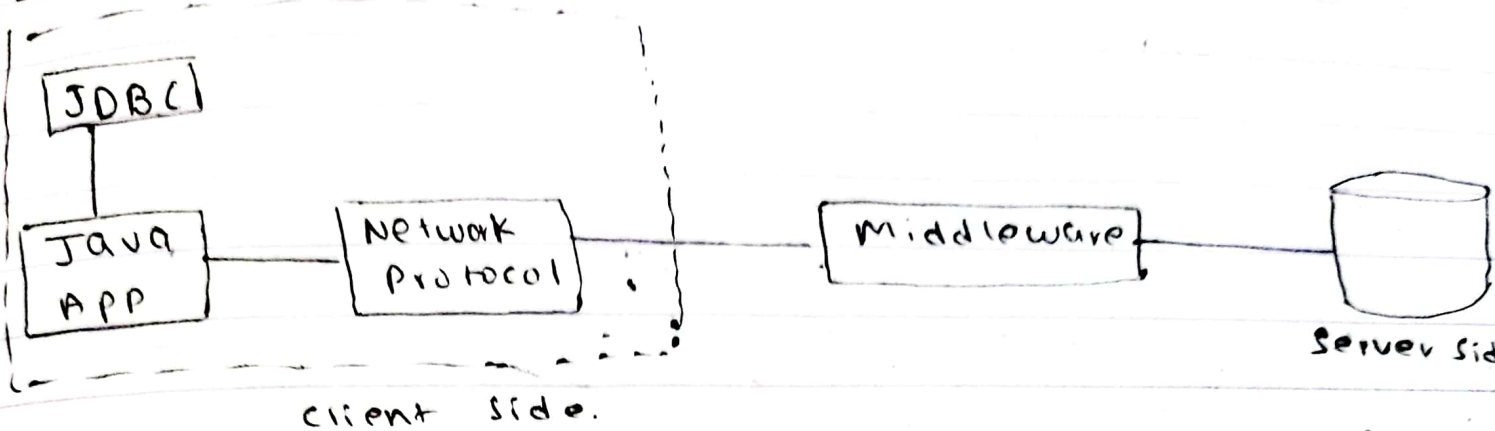
→ No need to install Bridge driver.

Disadvantage

→ Client has to install vendor library in their machine which has to be updated according to database.

3) Network Protocol Driver

→ It is completely written in Java.




→ It uses middleware (application server) that directly or indirectly converts java methods to specific database protocols.

Advantage

→ The middleware can be used for other extra works such as auditing, logging, load balancing.

→ Client doesn't have to install any vendor library in the machine.

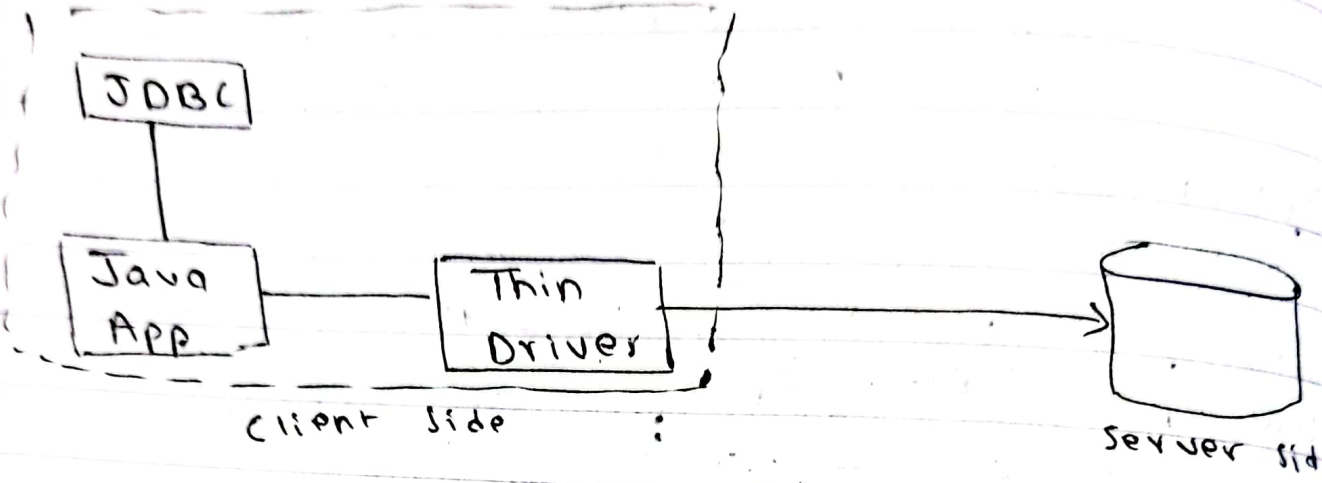
Disadvantage

→ Network support is req^d at the client side.

→ It increases the cost since middleware requires separate maintenance and support.

→ Vendor specific coding needs to be done at middleware

4) Thin Driver




→ It is completely written in Java.
→ It directly converts java method calls to database protocol

Advantage

→ It is the most efficient and the fastest driver.
→ Client doesn't have to install vendor library.

1. Register Driver
2. Connect to Database
3. Create statement
4. Execute Query
5. Close the connection.

08/20 Thursday

Q. write a program in java to display all the records from database.

=>

```
import java.sql.*;
class Select
{
    public static void main(String []args)
    throws ClassNotFoundException,
    SQLException
    {
        final String uname="root";
        final String pwd="xyz321";
        final String url="jdbc:mysql://localho
                         :3306/ncit";

        Class.forName("com.mysql.jdbc.Driver");

        Connection cn=DriverManager.getConnecti
                      (url,uname,pwd);

        Statement st=cn.createStatement();
```

factory method.

```
String q = "Select * from student";
st.executeQuery(q);

ResultSet rs = st.executeQuery(q);
while (rs.next())
{
    System.out.println(" Roll = " + rs.getInt(
        + "Name = " + rs.getString(2) +
        "Marks = " + rs.getInt(3));
}
cn.close();
}
}
```

Q) A database contains a table having Roll, Name and marks. Write a program to find avg marks.

→

```
import java.sql.*;
class CalculateAvg
{
    public static void main(String []arg
    throws Exception
    {
        String uname = "root";
        String pwd = "xy2123";
        String url = "jdbc:mysql://localhost:
                     3306/ncit";
```

```
Class.forName("com.mysql.jdbc.Driver");
Connection cn = DriverManager.getConnection
                (url, uname, pwd);
Statement st = cn.createStatement();

String q = "select AVG(marks) from
            student";

ResultSet rs = st.executeQuery(q);
rs.next();
System.out.println("Avg = " + rs.getFloat(1));
cn.close();
}
}
```

AVG

Q) WAP to find total no. of records in a database.

=>

```
" (All same upto ~~Connection cn~~ Statement st).

String q = "Select count(roll) From
            student";

ResultSet rs = st.executeQuery(q);
rs.next();
System.out.println("Total no = " + rs.getInt(1));
cn.close();
}
}
```

COUNT

(c) A database table employee contains id, name, post, salary. Change the salary to 50000 of only those employee who are manager.

⇒

```
String q = " Update employee
           set salary = 50000 where
           post = 'Manager' ";

int c = st.executeUpdate(q);
System.out.println(c + " records updated");
cn.close();
}
}
```

d) Delete all records whose post is Manager.

⇒

```
String q = " Delete from Employee
           where post = 'Manager' ";
int c = st.executeUpdate(q);
System.out.println(c + " records deleted");
cn.close();
}
}
```

Q. To insert a record, where id = 49, name = Ram, salary = 40000, post = Manager.

```
String q = "Insert into Employee
            values (49, 'Ram', 'Manager', 40000)";
st.executeUpdate(q);
```

Q) To ask user to enter id, name and Faculty and insert the record in database table

```
import java.util.*;
class Insert{
"   (same upto (connection.cn)

Scanner scan = new Scanner (System.in);

PreparedStatement ps = cn.prepareStatement(
        "Insert into student values (?,?,?)");

System.out.println ("Enter id:");
int id = scan.nextInt();
System.out.println ("Enter name:");
String name = scan.next();
System.out.println ("Enter Faculty:");
String Faculty = scan.next();

ps.setInt(1, id);                    ps.setString(3, Faculty)
ps.setString(2, name);
```

```
        ps.executeUpdate();
        System.out.println("Inserted");

        cn.close();
    }
}
```

05/21 Friday

Q) WAP to ask user to enter the details and insert it into database. After every successful insertion, prompt the user to continue or exit. Also, display the total no. of records inserted.

⇒

```
import java.util.*;
import java.sql.*;
class InsertExample
{
    public static void main(String []args)
    throws Exception{
        Scanner scan = new Scanner(System.in);

        String uname = "root";
        String pwd = "abczyx";
        String url = "jdbc:mysql://localhost:3
                      ncit";

        Class.forName("com.mysql.cj.jdbc.Driver
```

```
Connection cn = DriverManager.getConnection
                (url, uname, pwd);

PreparedStatement ps = cn.prepareStatement
                ("Insert into student
                 values (?,?,?)");

int count = 0;
do
{
    System.out.println("Enter roll:");
    int roll = scan.nextInt();
    System.out.println("Enter name:");
    String name = scan.next();
    System.out.println("Enter faculty:");
    String faculty = scan.next();

    ~~ps.getInt(1, roll);~~
    ~~ps.get~~
    ps.setInt(1, roll);
    ps.setString(2, name);
    ps.setString(3, faculty);
int i = ps.executeUpdate();
    count = count + i;

    System.out.println("Enter any key to
                continue and N/n to exit");
    String choice = scan.next();
    if (choice.toLowerCase().startsWith("n")
    { break; }
} while(true);
```

```
        System.out.println(" count + " records Inserted");
        cn.close();
    }
}
```

Create a menu based application to do the following:

```
***** Menu *****
1. search by name.
2. delete by roll
3. exit
* * * * * * * * * * * * *
```

```
import java.util.*;
import java.sql.*;
class MenuExample
{
    Connection cn;
    static Scanner scan = new Scanner(System.in);
    PreparedStatement ps;

    public void connect() throws Exception
    {
        Class.forName("com.mysql.cj.jdbc.Driver");
        cn = DriverManager.getConnection
            ("jdbc:mysql://localhost:3306/ncit",
             "root", "abc123");
```

```
    System.out.println("Database connected");
}

public void showMenu()
{
    System.out.println("******Menu******");
    System.out.println("1. Search by name");
    System.out.println("2. Delete by roll");
    System.out.println("3. exit");
    System.out.println("**********");
}

public void exit() throws SQLException
{
    cn.close();
    System.exit(0);
}

public void searchByName() throws SQLExceptio
{   System.out.println("Enter name");
    String name = scan.next();

    ps = cn.prepareStatement("select * from
                student where name = ?");
    ps.setString(1, name);
    ResultSet rs = ps.executeQuery();
    while (rs.next())
    {
        System.out.println(rs.getInt(1) + "  " +
            rs.getString(2) + "  " + rs.getString(3));
    }
    System.out.println("End");
}
```

```
public void deleteById() throws SQLExcep
{
    ps = cn.prepareStatement("Delete From
            student where roll id = ? ");

    System.out.println(" Enter id to be
            deleted : ");
    int id = scan.nextInt();
    ps.setInt(1, id);
int r=ps.executeUpdate();

    System.out.println(r+" records delete
}

public static void main(String []args)
throws Exception
{  MenuExample m= new MenuExample()
   m.connect();
while (true)
{  m.showMenu();

    System.out.println(" Enter your choice:");
    int choice = scan.nextInt()
    int choice = scan.nextInt();
    switch (choice)
    {
        case 1:
                m.searchByName()
                break;
```

```
case 2:
        m.deleteById();
        break;
case 4:
        m.exit();
        break;
default:
        System.out.println("Invalid input");
    }
  }
}
```

# Components of JDBC

1) Driver Manager

It is a class which is a part of java.sql package. It acts as an interface between the user and the driver.

- It manages the list of available drivers.
- It is also responsible for connecting the java application with database.
- It is a factory of Connection.

Methods

a) public static ~~synchro~~ synchronized registerDriver (Driver d) throws SQLExce

→ This method allows us to register the specified driver in the list o available drivers.

→ If the driver is already in the list, it does nothing

b) public static synchronized deregisterDr (Driver d) throws SQLException

→ This method removes the driver from the list of available driver.

→ If the driver is not in the list, it does nothing.

c) public static Driver getDriver(String url)
throws SQLException
{

→ This method returns the driver associa
-ted with the specified url

d) public static Connection getConnection
(String url, String uname, String pwd
throws SQLException

→ This method is used to connect ~~to~~
~~connect~~ to database specified by
the url using the username and
password.

e) public static Connection getConnection
(String url) throws SQLException

→ This method is used to connect to
database specified by the url.

f) public static int getLoginTimeout()

→ It returns the maximum duration of
time (in seconds) that the driver is
allowed to wait while trying
to make a connection.

8. public static void setLoginTimeout(int

→ This method is used to specify the max. duration of time the driver is allowed to wait while trying to make a connection.

→ If 0 is passed driver can wait indefinitely.

08/27 Thursday

2) Connection

It is an interface that represents session between java application and database.

→ It is a factory of Statement, PreparedStatement and DatabaseMetaData.

→ It also provides method for transaction management such as: rollback(), commit(), etc.

Methods

a) public Statement createStatement() throws SQL

→ This method creates an object of state-ment to execute SQL queries.

b) public Statement createStatement( int resultSetType, int resultSet(concurrency) throws SQLException.

→ This method is used to create a statement with given ResultSet type and concurrency.

c) public PreparedStatement prepareStatement (String query) throws SQLException.

→ This method is used to create an object of PreparedStatement which is used to execute dynamic ~~execute~~ queries.

d) public void close() throws SQLException.

→ It closes the connection and releases all the JDBC resources immediately.

3) Statement

→ It is an interface that provides us with methods to execute queries.

→ It is slow and inefficient since, queries are compiled every time before execution.

→ It doesn't allow placeholder for dynamic value insertion.

→ It is vulnarable to SQL injection.

Methods

a) public ResultSet executeQuery(String query) throws SQLException

→ It is used to execute DQL. (Select statement)

→ It returns an object of ResultSet that points to a row of a tabular data.

b) public ~~ResultSet~~ int executeUpdate(String query) throws SQLException.

→ It is used to execute DML such as; insert, update, delete, etc.

→ It returns an integer that represents no. of rows affected by query execution.

c) public boolean execute(String query) throws SQLException.

→ It is used to execute queries that may

return multiple responses.

d) public int[] executeBatch() throws SQLException

→ It is used to execute batch of commands.

4) PreparedStatement

→ It is a sub-interface of statement.
→ It is used to execute dynamic queries.
→ It is faster and efficient because queries are compiled only once.
→ We can use '?' as a placeholder for dynamic value insertion in a query.
→ Using ~~pr~~ PreparedStatement is one of the major to prevent SQL Injection.

Methods

a) public ResultSet executeQuery() throws SQLExcep

→ It is used to execute DQL.
→ It returns an object of ResultSet that points to a row of a tabular data

b) public int executeUpdate() throws SQLExceptio

→

c) public void setInt(int paramIndex, int va

→ It is used to set the given parameter Index with the specified integer value.

d) public void setString(int paramIndex, String va

→ It is used to set the given parameter Index with the specified string value.

Differences between

| Statement | PreparedStatement |
|---|---|
| i) used to execute static query. | i) used to execute dynamic query. |
| ii) queries are compiled every time before execution. | ii) queries are compiled only once. |
| iii) It is slower | iii) It is faster. |
| iv) We can use string concatenation for dynamic value query entry. | iv) It provides '?' as placeholder for dynamic value entry in a query. |

v) We cannot use statement to read or write binary files.

v) We can read or write binary files.

vi) It doesn't use binary protocol for communication.

vi) It uses binary protocol for communication.

vii) It is vulnarable to SQL Injection.

vii) It is used to prevent SQL Injection.

5) ResultSet

→ ResultSet is an interface whose object is created when DQL statement is executed.

→ It holds tabular data and maintains a cursor at a row at top.

→ By default, the cursor is maintained at position before the first tuple.

→ By default, it is not updatable and the cursor can be moved in the forward direction.

Methods

a) public boolean next()

) It moves the cursor one position forward from the current position.

b) public boolean previous()
→ It moves the cursor one position backward from the current position.

c) public boolean first()
→ It moves the cursor to the first row.

d) public boolean last()
→ It moves the cursor to the last row.

e) public boolean absolute(int row)
→ It moves the cursor directly to the specified row position irrespective of the current position.
(no. neg row no.)

f) public boolean relative(int row)
→ It moves the cursor forward or backward by the specified relative value w.r.t. the current position.

g) public int getInt(int columnIndex)
→ It returns the integer value of the specified columnIndex.

h) public int getInt(String columnName)
→ It returns the integer value of given column name.

# RowSet

→ It is a wrapper of ResultSet.
→ It is easier and more flexible than using ResultSet.
→ It is a Javabean component.
→ It provides mechanism to hold tabular data
→ It maintains connection with the database throughout the life cycle.

Types:-
RowSet is implemented by following child classes.

a) JdbcRowSet
b) WebRowset
c) JoinRowset
d) FilteredRowSet
e) CachedRowset

WAP to fetch all the records from database table using JdbcRowset.

```
import java.sql.*;
import javax.sql.*;

class Select
{
    public static void main(String []args)
    throws Exception
    {
        Class.forName(" com.mysql.cj.jdbc.Driv
        JdbcRowSet rs = RowSetProvider.
                newFactory().createJdbcRowSet(
        rs.setURL("
        rs.setUrl(" mysq
        rs.setUrl(" jdbc:mysql://localhost:3306/m
        rs.setUsername ("root");
        rs.setPassword(" abc ");
        rs.setCommand(" Select * from studen
        rs.execute();
        while (rs.next())
        {
        System.out.println(rs.getInt(1) + " " +
                            rs.getString(2));
        }
    }
}
```

08/28 Friday

## Jdbc Escapes

Jdbc Escapes are the syntax that provides flexibility to use features related to database that are not available through standard Jdbc methods and properties.

Syntax:

{ key 'parameters' }

example for keys:

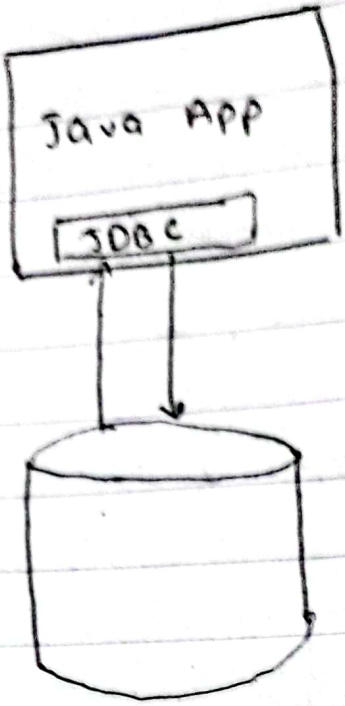
d (date)

t: (time)

ts (timestamp)

"Insert into students values ("xyz", { d '2024-12-12' }";

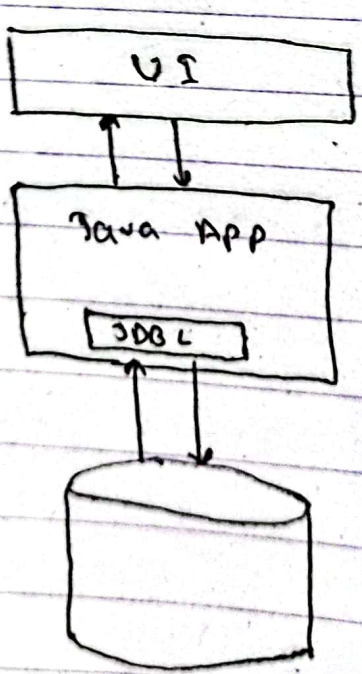
# Jdbc Architecture




2 Tier Architecture

→ In this type of architecture the java application is responsible for getting the user input, executing the queries and displaying the user

→ It is used in small scale desktop application




3-tier architecture

→ In this architecture, a separate UI is created that is responsible for getting user input and displaying results.

→ User can't interact directly with Java application (backend) and database.

→ It is used in mobile applications and web application.