Java

Data Types

1) Primitive Data Type

a) Numbers

- byte (1)
- short (2)
- int (4)
- long (8)
- Float (4)
- double (8)

b) Character

- char (2)

c) boolean (1) bit.

Data Type Conversion

a) → सानोबाट ठुलो
→ Implicit
→ widening.

e.g.

```
int a = 5;
float f = a;
```

b) ठुलोबाट सानो (narrowing)
(explicit).

```
// float f = 5.5;
// int a = f
```

error आउँछ

Solution

```
float f = 5.5;
int a = (int) f;
```

Integer a = 's'; is diff. from int a = 5;

(Integer: capital. Integer = class, a = obj)

wrapper class.

Byte
Short
Integer
Long
Float
Double

Character

Boolean

(Data type को corresponding classes पनि हुन्छ।)
i.e wrapper class.

Ruby: pure OOP. everything

Integer a; → Unboxing (wrapper class → primitive data type) conversion

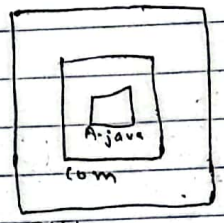
int a → Autoboxing.

Packages: collection of related classes and interfaces.

a) Built in
- java.sql
- java.net
- java.awt

b) User defined

```
package ncit.com;
import java.sql.*;
class A
{

}
```



files भर्नेको classes

```
package ncit;

class A
{
    ---
    ---
}

class B
{

}
```

Importing (All)

```
package sw;
import ncit.*;
class D
{
    A a = new A();
    B b = new B();
}
```

Importing To make faster (Specified class)

```
import ncit.A;
class D
{
    A a = new A();
    // B b = new B();
    // error
```

Also,

Fully qualified without importing

```
package sw;
// no import
class B
{
    ncit.A a = new A();
}
```

Access Modifier (Specifier)

	Same class	Same package	Diff (folder package) with sub(class)	Diff. Package
Private	✓	X	X	X
default	✓	✓	X	X
protected	✓	✓	✓	X
public	✓	✓	✓	✓

```
class A
{
    private class B
    {
    }
}
```

class can be made private if it is inner class (Nesting class)

```
class student
{
    String name;
    int id;
    public void setName( String name )
    {
        this.name = name;
    }
    public void setRoll(int id)
    {
        this.id = id;
    }

    public String getName()
    {
        return name;
    }
    public int getRoll()
    {
        return roll;
    }
}

class Test
{
    public static void main( String [] args)
    {
        student obj = new student();
        obj.setName("krishna");
        obj.setRoll(1);
System.out.print obj.getName());
        System.out.println("Name = "+obj.getName());
        System.out.println("Roll = "+obj.getRoll());
    }
}
```

```
class Student
{
    String name;
    int roll;        } Instance variable
    static String college; } class variable.
}

public static void main(String [] args)
```

Create a class student with name and id as instance variable. Initialize those variables using parametrized constructor.

```
class shape
{
    int x, y, l, h;
```

create a class shape with x, y, l, h as attributes. create constructors that allows us to create an object by passing all the 4 attributes, only two attributes or no attributes.

```
class shape
{
    int x, y, l, h;
    public shape(int x, int y, int l, int h)
    {
        this.x = x;
        this.y = y;
        this.l = l;
        this.h = h;
    }
    public shape(int l, int h)
    {
        x = 0;
        y = 0;
        this.l = l;
        this.h = h;
    }
    public shape()
    {
        x = 0;   l = 1;
        y = 0;   h = 1;
    }
```

```
    public static void main(String []args)
    {
        shape s1 = new shape(1,3,5,5);
        shape s2 = new shape(10,20);
        shape s3 = new shape();
    }
}
```

Also,

```
class shape
{
    int x, y, l, h;
    public shape(int x, int y, int l, int h)
    {
        this.x = x;
        this.y = y;
        this.l = l;
        this.h = h;
    }
    public shape(int l, int h)
    {
        this(0, 0, l, h);
    }
    public shape()
    {
        this(0, 0, 1, 1);
    }
```

Every class is child of

Create a class student with name and roll, initialize them using parametrized constructor. Also, override toString() method to display the detail of the student.

⇒

```
class Student
{
    String name;
    int roll;
    public Student(String name, int roll)
    {
        this.name=name;
        this.roll = roll;
    }
    @Override
    public String toString()
    {
        return this.name+" "+this.roll;
    }
}

class Test
{
static public void main(String [] arg)
{
    Student s = new student("krishna", 21);
    System.out.println(s);
}
}
```

7/26 Monday

Inheritance

```
class A          class B extends A
{                {
}                }
```

a) Create a class employee having name, ~~id and~~ monthly salary. Implement a method to calculate yearly salary. Also, create a class teacher having name, monthly salary and no. of subjects. Also, create a method to calculate yearly salary of a teacher (monthly salary * 12 + 10*20). Implement method to display the details.

⇒

```
class Employee
{
    String name;
    int mSalary;
    public Employee(String name,
                    int mSalary)
    {
        this.name = name;
        this.mSalary = mSalary;
    }
    public int getYearly()
    {
        return (mSalary*12);
    }
```

```
public String showDetail()
{
    return "Name" + this.name +
           "Monthly" + this.monthly
           + "Yearly" + this.getYearly();
}

class Teacher extends Employee
{
    int numClasses;
    public Teacher(String name,
        int mSalary, int numClasses)
    {
        super(name, mSalary);
        this.numClasses = numClasses;
    }

    public int getYearly()
    {
        return super.getYearly() +
        this.numClasses *100;
    }

    public String showDetail()
    {
        return super.showDetail()+
        "Class = " + this.numClasses;
    }
}
```

```
class Test
{
    public static void main(String [] arg
    {
        Employee E = new Employee("John
                                  10000)
        System.E.
        System.out.println(E.showDetails());

        Teacher T = new Teacher("Jivan",
                                10000, 5);
        System.out.println(T.showDetails());
    }
}
```

Interface

```
interface Animal
{
    void makeNoise()
    {
    }
}
class Dog implements Animal
{
    void makeNoise()
    {
        System.out.println("Bark");
    }
}
```

7/28

class
interface
abstract class

- class मा f^n खाली हुन हुँदैन।
- Interface मा ~~class~~ f^n खाली हुनुपर्छ।
- कुनै f^n खाली, कुनै f^n contais code support गर्न abstract class चाहिन्छ।

```
abstract class Payment
{
    void auth();
    void deduct()
    {
        this.ap = this.ap - this.b;
    }
}
```

```
abstract class Payment
{
    abstract void auth();
}
```

interface A
{
}

Difference between Interface and abstract class.

Abstract class	Interface
i) It may contain abstract or non-abstract 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 constructor
vii) It provides abstraction from 0 to 100%.	vii) It provides abstraction level of 100%

Q) Write Java program that overrides methods getUserValue(), and displayUserDetails()

```
Interface Parent
{
    void getUserValue();
    void displayUserDetails();
}
class Child implements Parent
{
    void getUserValue()
    {
        ...
    }
    void displayUserDetails()
    {
        ...
    }
}
```

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

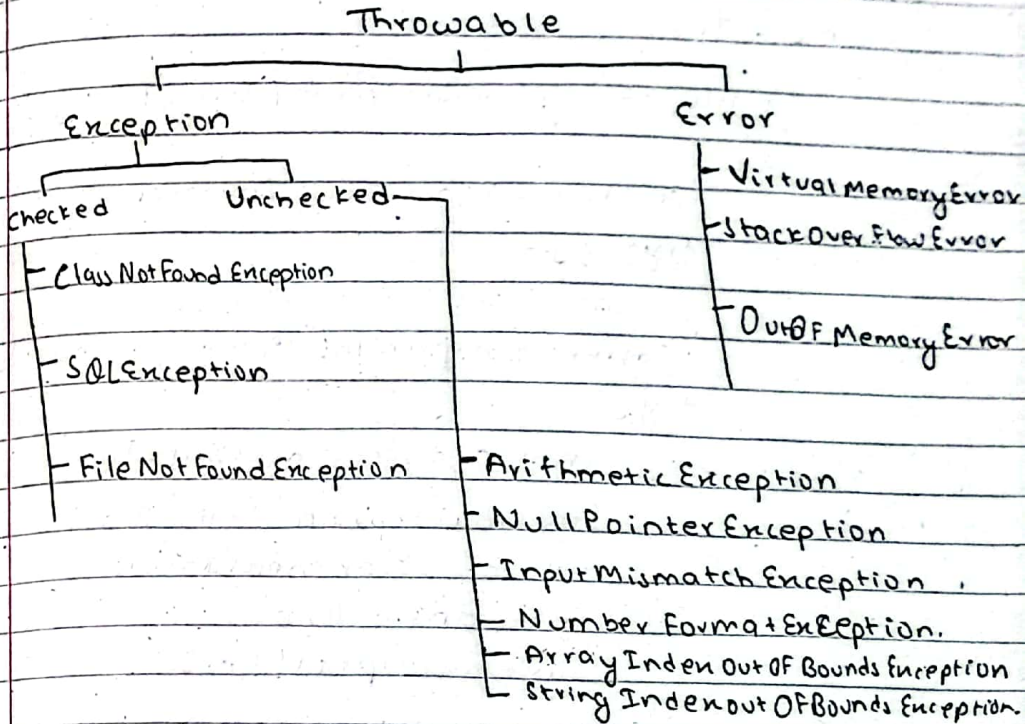
⇒

```
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.



NullPointerException

object बनाया है initialize नहीं किया (init. बाद में call नहीं 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{ try
        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.

⇒

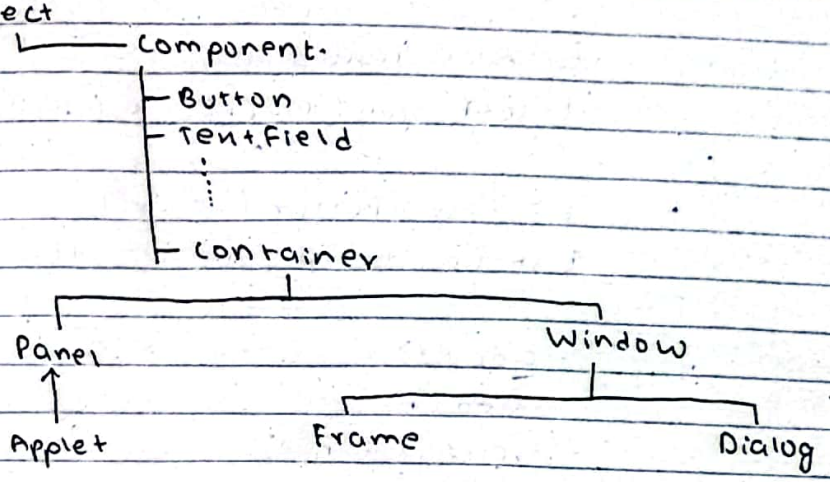
```
class SemException extends Exception
{
    public SemException(String msg)
    {
        super(msg);
    }
}

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());
        }
    }
}
```

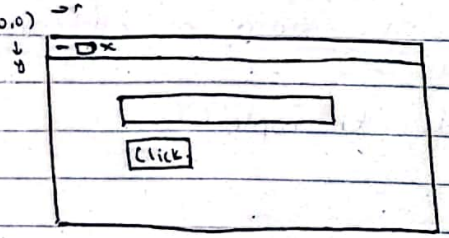
Q) WAP with custom exception handler that handles Arithmetic Exception.

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");
```

```
setBounds ( x, y,

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 nee~~ no need to write Frame f;

f.add (t) = add(t)

:
-"-

}

Using Swing

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

Component को अगाडि J लगाउने

e.g. Frame = JFrame

TextField = JTextField.

Difference between AWT and Swing.

	AWT	Swing
i)	Abstract Window Toolkit.	It is a part of JFC (Java Foundational Class)
ii)	It is platform dependent since it uses components of underlined OS.	It is platform independent and has its own set of components.
iii)	Its execution speed is slower.	Its execution speed is faster.
iv)	Components of AWT are heavy weighted.	components of swing are light weighted.
v)	It doesn't have pluggable looks and feel.	It has pluggable looks & feel.
vi)	It has limited no. of components compared to swing.	It has extensive no. of components.
vii)	Components of AWT are less customizable.	components of swing are highly customizable.
viii)	AWT doesn't support MVC pattern.	Swing supports MVC pattern
ix)	It is used to create GUI based application.	It is used to create desktop app as well as web app
x)	All components of ~~swing~~ awt are in java.awt package.	All components of swing are in javax.swing package

Q Create a GUI application with a textfield 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();
    }
}
```

java.awt

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)
    { // 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();
}
}
```

08/10 Thursday

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 message indicating the button pressed.

=>

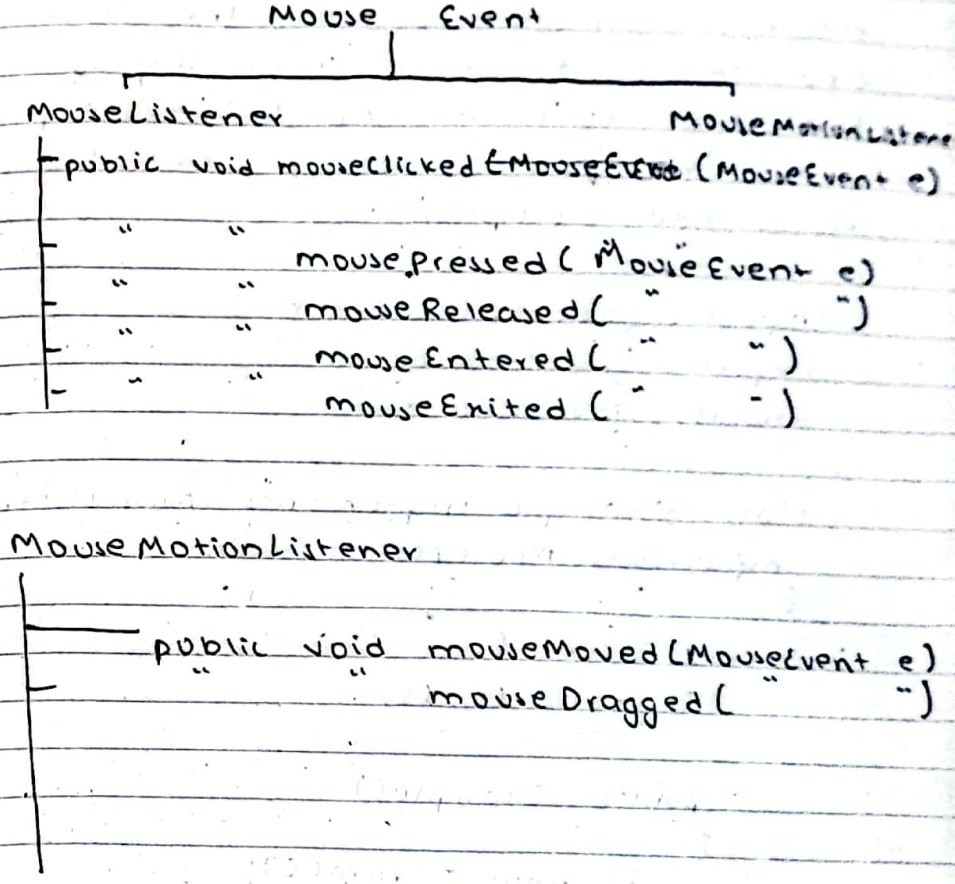
```
import javax.swing.*;
import java.awt.event.*;
import java.awt;

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);
        setLayout(FlowLayout());
        setVisible(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(
            }
            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 (")

(Label)

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 void mouseDragged(MouseEvent e){}
    public void mousePressed(MouseEvent e){}
    public void mouseReleased(MouseEvent e){}
}
}
```

```
public static void main(String [Jargs)
{
    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.

08/04 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.

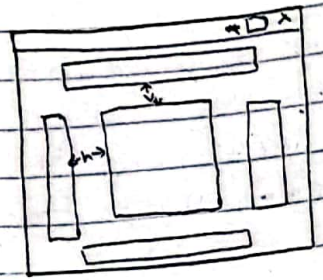
1) BorderLayout

→ 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.

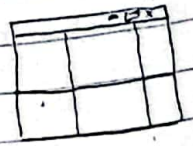
```
import java.awt.*;
import javax.swing.*;
class Example
{
    public Example()
    {
        JFrame f= new JFrame();
        f.setSize(500, 600);

        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, 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

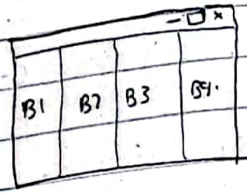
→ It is used to place the components in specified rows and columns.

Constructors

→ new GridLayout(int rows, int columns)

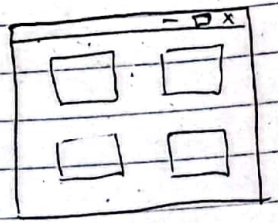


→ new GridLayout()



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
    ~~new~~ 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_gap);

fields

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

FlowLayout is the default 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.LEFT));
    }
    public static void main(String []args)
    {
        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

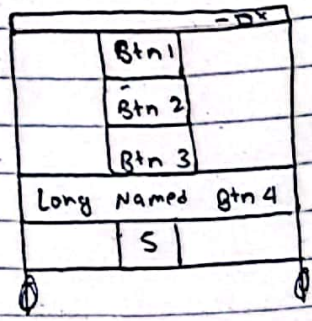
→ It places the components in the same manner the words are placed as in sentence as per the orientation property of the container.

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

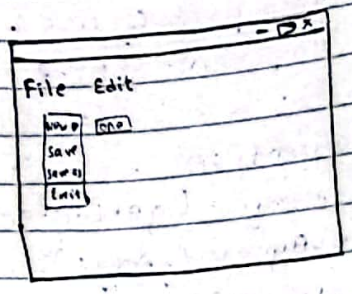
2081/08/10 Sunday



⇒

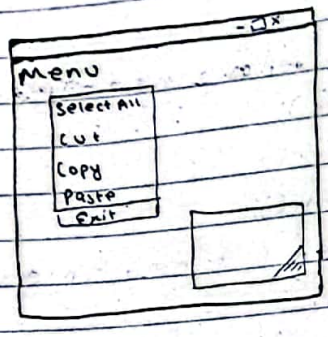
```
import javax.swing.*;
import java.awt.*;
class Box extends JFrame
{
    public Box()
    {
        JButton one = new JButton("Btn 1");
        "       two = "      "  ("Btn 2");
        "       three = "    "  ("Btn 3");
        "       four = "     "  ("Long
                    Named Btn 4");
 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

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

        ~~setLayout(B.getContentPan~~
        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 with a menu having select All, cut, copy, paste, exit as item 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 two items, and a label. The menu should appear wherever 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 e)
{
    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)
2. " " windowClosing(")
3. public void windowClosed(WindowEvent e)
4. public void windowActivated(WindowEvent e)
5. public void windowDeactivated(WindowEvent e)
6. public void windowIconified(WindowEvent e)
7. 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.

=>

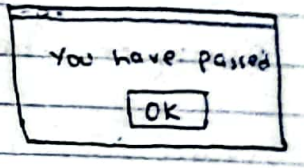
```
import java.awt.*;
import java.awt.event.*;

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. It is used to display dialogBox or popup alert.

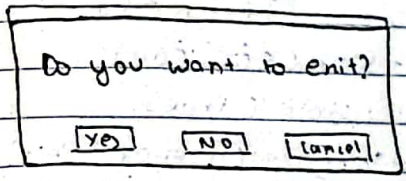
MessageDialog



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

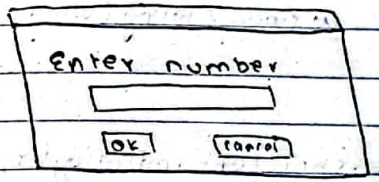
Confirmation Dialog

(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,

Q) 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 WindowAdapter
{
    JFrame f;

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

    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 []args)
    {
        new DialogEx();
    }
}
```

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 ti;
    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,20);
    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~~ 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~~ 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

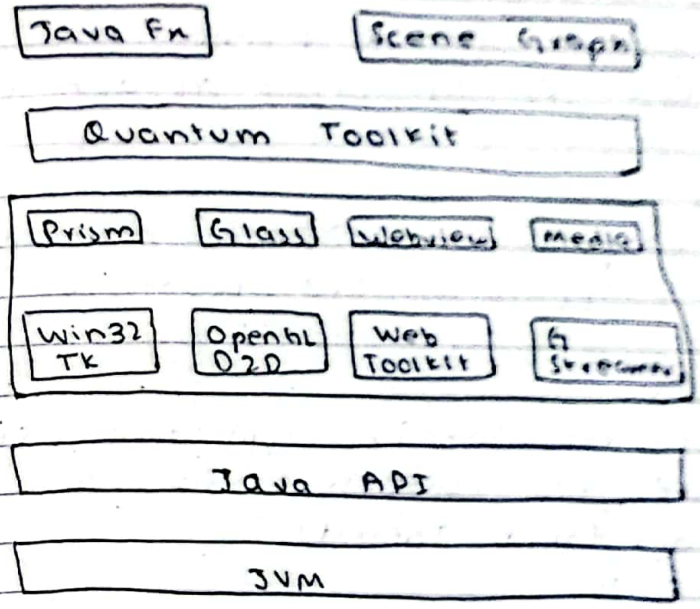
Storing Interoperability

- ~~It~~ Swing application can be converted into Javafx application and Javafx elements 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 hierarchical ~~tree~~ tree structure of nodes that represents all the visual elements of UI.

→ It is also responsible 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 GStreamer 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);
    }
}
```

08/16

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, 400);
      S.setScene(scene);
      S.show();

      upper.setOnAction(new EventHandler<Action-
                        -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 Exception
    {
        Label l1 = new Label();
        Label l2 = new Label();

        HBox root = new HBox()
        root.getChildren().addAll(l1, l2);
        Scene scene = new Scene(root, 400,300
        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

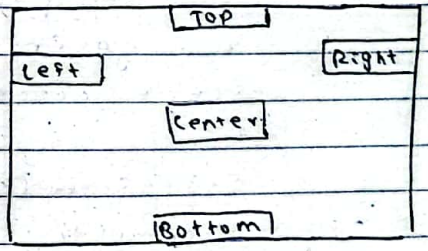
This layout allows the nodes to be placed in five different regions ie. top, bottom, left, right, center.

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:



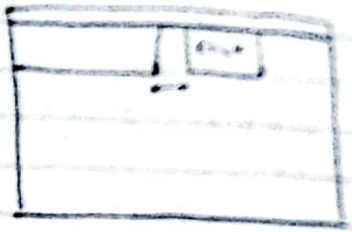
2) HBox

- 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);
    }
}
```



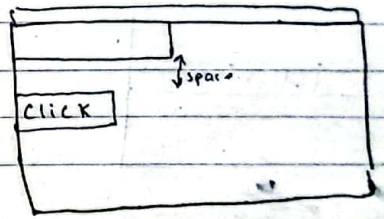
3) VBox

- 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
}
```

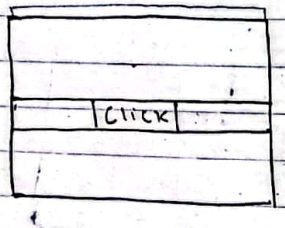
output



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, 500,600);
        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.

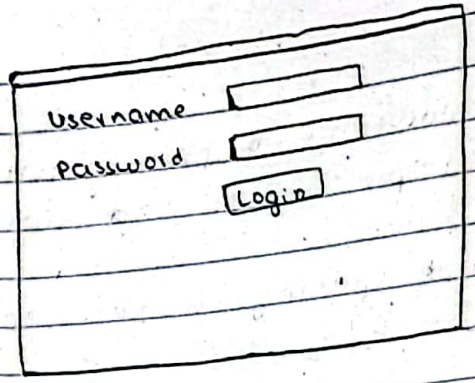
Methods

- 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.

d)




```
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, login);

        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 environment / applications
ii) It partially supports MVC pattern.	ii) It completely supports MVC pattern.
iii) It has more components.	iii) It has components that are highly customizable
iv) It doesn't support hardware accelerated graphics.	iv) It supports high performance hardware accelerated graphics pipeline
v) It has limited support for audio and video media.	v) It uses open source GStreamer media engine to render audio and video multimedia
vi) It is straight forward and easy to use.	vi) It has steeper learning curve.

ODBC: Open Database

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

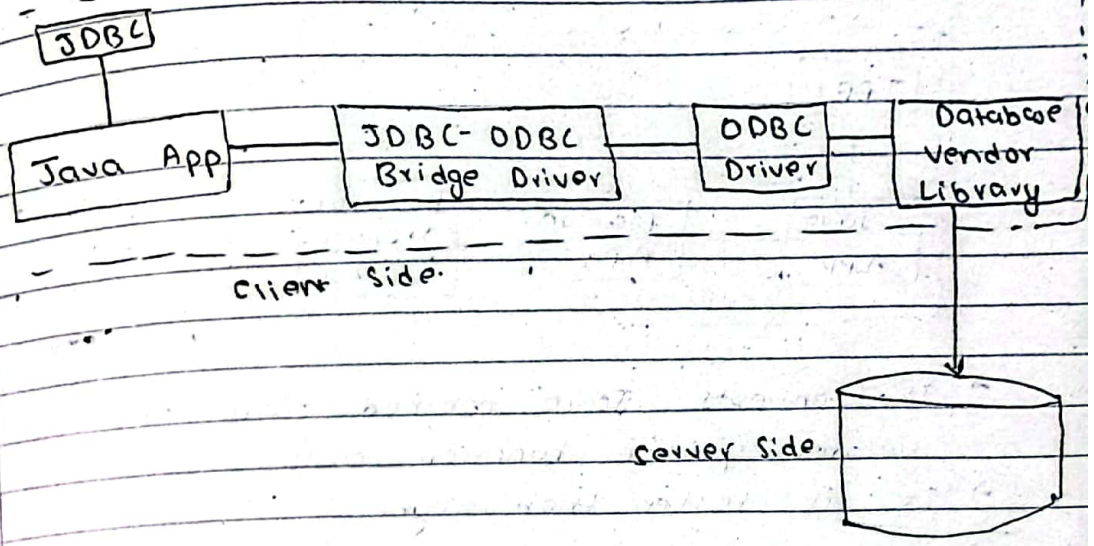
It is a software that enables java application to communicate with vendor database.

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

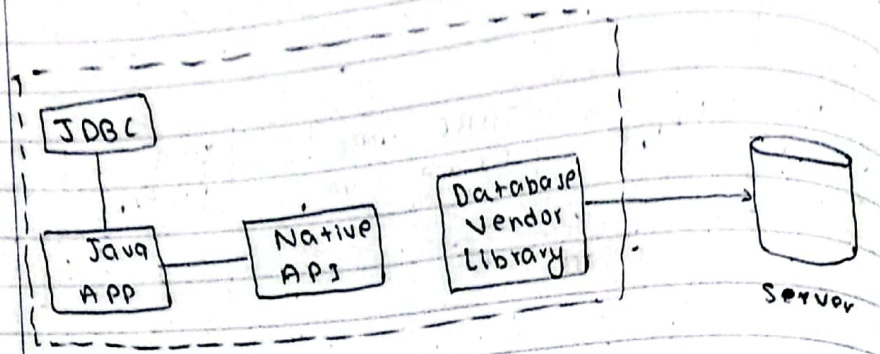
→ It is easy to use.
→ It can connect to any database

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

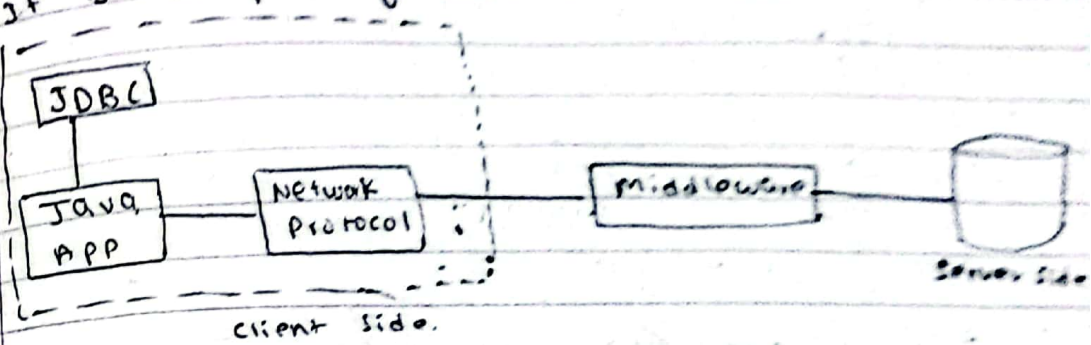
→ No need to install Bridge driver.

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.

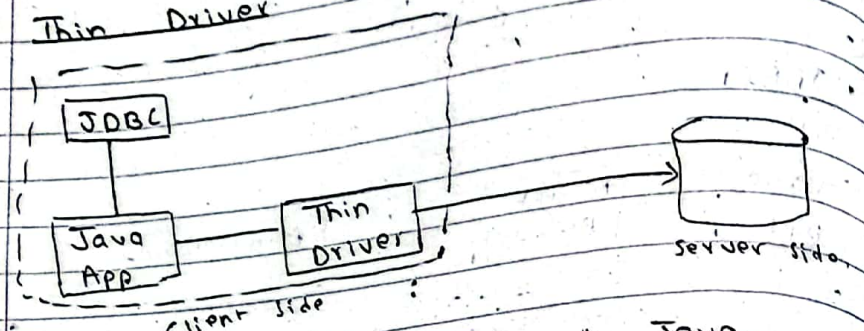
Disadvantage

→ 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://localhost
                          :3306/ncit";

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

        Connection cn = DriverManager.getConnection
                        (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(1)
        + "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 []args)
    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 statement st. ~~Connection cn~~).

```
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

Q) 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();
}
}
```

Q) 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();
    }
}
```

08/21 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:3306/
                      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);
```

OR

```
while(!(scan.next().toLowe
```

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

Q) 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/ncct",
          "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 SQLExcep
    {   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 SQLException
{
    ps = cn.prepareStatement("Delete From
         student where roll = ?");

    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();
    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 SQLException

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

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

b) public static synchronized deregisterDriver (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 associated 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.

g. public static void setLoginTimeout(int sec)

→ 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 SQLExceptio

→ This method creates an object of statement 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 SQLExceptior.

→ 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 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 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 SQLException

c) public void setInt(int paramIndex, int value)

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

d) public void setString(int paramIndex, String value)

→ 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 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

Q. 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.Driver");
      JdbcRowSet rs = RowSetProvider.
            newFactory().createJdbcRowSet();
      rs.setUrl("jdbc:mysql://localhost:3306/ncet");
      rs.setUsername("root");
      rs.setPassword("abc");
      rs.setCommand("Select * from student");
      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 allows provides flexibility to use features related to database that are not available through standard Jdbc methods and properties

Syntax:

{ key 'parameters' }

example for keys:

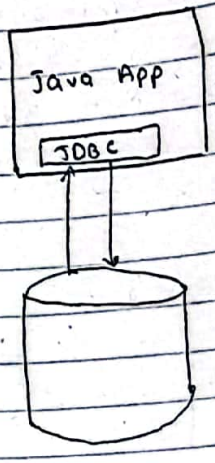
d (date)

t: (time)

ts (timestamp)

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

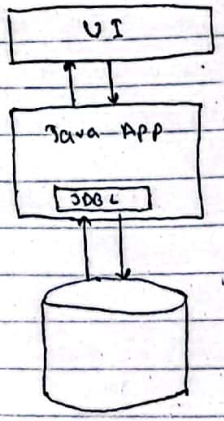
Jdbc Architecture



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.

It is a java application that allows communication between two or more java application running on different JREs.

Components:

IP Address

→ It is a numerical value that is assigned to a computer in a network.
→ It is used for Network Interface Identification and location addressing.
→ IP address is of 2 types:
IPv4 and IPv6.

IPv4

→ It is 32 bits long and is divided into 4 octates.

IPv6

→ It is 128 bits long

Port

→ It is a logical construct
→ It is a number that uniquely identifyies a process running in computer.
→ It is 16 bits long unsigned integer. ranging from 0 to 65535

Types of Ports

1) Well known ports (0-1023)
→ Port numbers running from 0-1023 is well known ports.
→ These are assigned to commonly used applications. e.g. (HTTP: 80, HTTPS: 443, SMTP: 25,

2) Registered Ports (1024-49151)
→ These numbers are assigned to vendor applications.
→ e.g. (MySQL: 3306, Oracle: 1521)

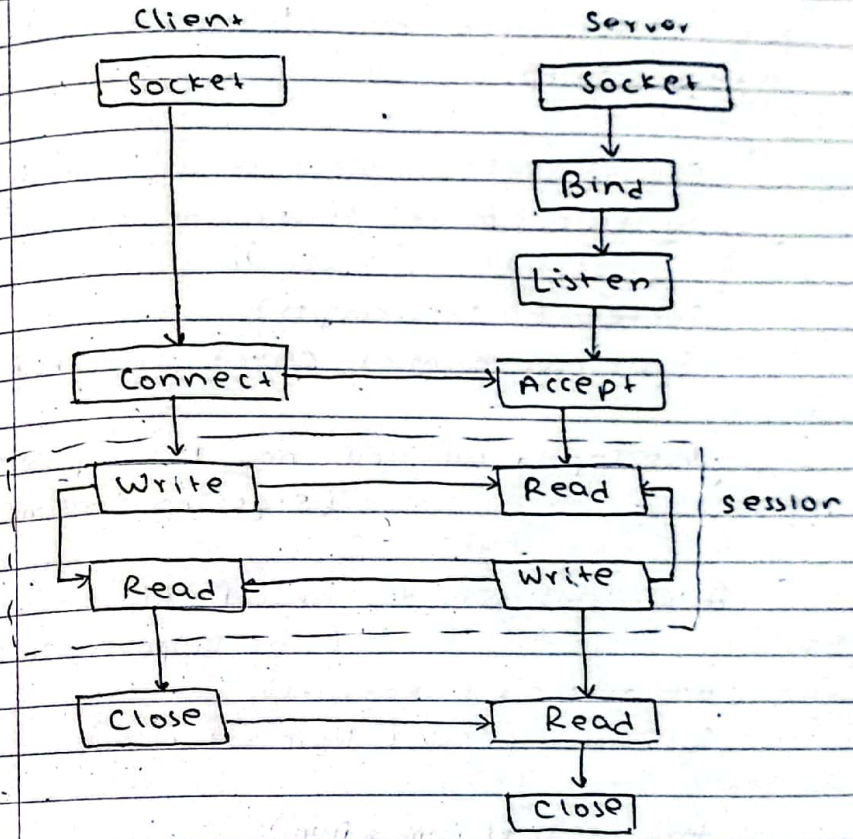
3) Dynamic Ports (49152-65535)
→ Ephemeral
→ Used by applications which runs for short period of time.

Socket
- It is defined as end point of two way application.
→ combination of IP address and port number

Protocol
→ It is defined as the set of rules followed by receiver and transmitter in data communication.

→ Protocol can be either connection oriented (TCP) or connectionless (UDP).

TCP Architecture



Q) Create a TCP application where client sends a number (integer) and server responds by sending its square.

⇒

```
import java.net.*;
import java.io.*;
class Server
{
    public static void main(String [] args)
    throws Exception
    {
        ServerSocket ss = new ServerSocket(6000);
        System.out.println("Server running on
                            6000");
        Socket s = ss.accept();
        System.out.println("Client connected");

        DataInputStream dis = new DataInputStream
                              (s.getInputStream());

        DataOutputStream dos = new DataOutputStream
                               (s.getOutputStream());
        int num = dis.readInt();
        System.out.println("Num = " + num);

        dos.writeInt(num * num);

        dos.close();
        dis.close();
        s.close();
        ss.close();
    }
}
```

Client side

```
import java.net.*;
import java.io.*;
import java.util.Scanner;

class Client
{
    public static void main(String [] args)
    throws Exception
    {
        Scanner scan = new Scanner(System.in);
        System.out.println("Enter num:");
        int num = scan.nextInt();

        Socket s = new Socket("localhost", 6000);

        DataInputStream dis = new DataInput-
                              Stream(s.getInputStream());
        DataOutputStream dos = new DataOutput-
                               Stream(s.getOutputStream());
        dos.writeInt(num);
        int square = dis.readInt();
```

```
        System.out.println("Square :"+square);

        dos.close();
        dis.close();
        s.close();
    }
}
```

Q) Create a TCP architecture where client sends a string and server echos the same string in uppercase.

⇒

```
import java.net.*;
import java.io.*;
class Server
{
    public static void main(String [] args)
    throws Exception
    {
        ServerSocket ss = new ServerSocket
                          (5000);
        System.out.println("Server is running
            on port 5000");

        Socket s = ~~new~~ ss.accept();
        System.out.println("Client connected");

        DataInputStream dis = new DataInputStr-
                              eam(s.getInputStream());
        DataOutputStream dos = new DataOutputStr-
                               eam(s.getOutputStream());
        String st= dis.readUTF();
        System.out.println("String :"+st);
        dos.writeUTF(st.toUpperCase());

        dos.close();
        dis.close();
        s.close();
        ss.close();
    }
}
```

Client side

```
import java.net.*; import java.util.Scanner;
import java.io.*;
class Client
{
    public static void main(String [] args)
    throws Exception
    {
        Scanner scan = new Scanner(System.in);
        ~~String st = new Scanner();~~
        System.out.println("Enter string:");
        String st = ~~new~~ scan.next();
```

```
Socket s = new Socket("localhost", 5000);

DataInputStream dis = new DataInputStre-
                      -am(s.getInputStream());
DataOutputStream dos = new DataOutputStream
                       (s.getOutputStream());

dos.writeUTF(st);
String str = dis.readUTF();

System.out.println("Uppercase =" +str);

dos.close();
dis.close();
s.close();

    }
}
```

09/05 Friday

Q) Create a TCP client server application where client sends string and server replies whether the string is palindrome or not.

```
import java.net.*;
import java.io.*;

class Server
{
    public static void main(String []args)
    throws Exception
    {
        ServerSocket ss = new ServerSocket(5000);
        System.out.println("Server running on 5000");

        Socket s = ss.accept();
        System.out.println("Client connected");
    DataInputStream dis = new DataInput(tre-
                          -am(s.getInputStream());

    DataOutputStream dos = new DataOutputStream
                           (s.getOutputStream());

    String str = dis.readUTF();
    String ans = isPalindrome()? "Palindrome":
                 "Not palindrome";
    dos.writeUTF(ans);
    dos.close();
```

s.getPort();

```
        dis.close();
        s.close();
        ss.close();
    }
    public static boolean isPalindrome(String str)
    {
        String rev = "";
        for(int i = str.length()-1 ; i >=0 ; i--)
        {
            rev = rev + str.charAt(i);
        }
        if( rev.equalsIgnoreCase(str))
        {
            return true;
        }
        else
        {
            return false;
        }
    }
}
```

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

        Socket s = new Socket("localhost",3000);

        DataInputStream dis = new DataInputStre-
                -am( s.getInputStream);
        DataOutputStream dos = new DataOutputStre-
                -eam (s.getOutputStream);

        System.out.println("Enter a string:");
        String msg = scan.next();
        dos.writeUTF(msg);
        dis.writeUTF(msg);
        String res = dis.readUTF();

        System.out.println("msg +" is " + res);

        dos.close();
        dis.close();
        s.close();
    }
}
```

Q) Create TCP client server app. where client sends string and server says palindrome or not. The communication continues until the client says 'bye'.

=>

Client side

```
import java.util.*;
import java.io.*;
import java.net.*;

class Client
{
    public static void main(String []args)
    throws exception
    {
        Scanner scan = new Scanner(System.in);

        Socket s = new Socket("localhost", 5000);

    DataInputStream dis = new DataInputStream
                (s.getInputStream());
    DataOutputStream dos = new DataOutputStream
                (s.getOutputStream());

    while (true)
    {
        System.out.println("Enter string: ");
        String msg = scan.next();
        dos.writeUTF(msg);

        if(msg.equalsIgnoreCase("Bye"))
        {
            break;
        }

        String res = dis.readUTF();

        System.out.println(msg + " is " + res);
    }
    dos.close();
    dis.close();
    s.close();
}
}
```

Server side

```
import java.io.*;
import java.net.*;

class Server
{
    public static void main(String []args)
    throws Exception
    {
        ServerSocket ss = new ServerSocket
                (5000);
```

```
System.out.println("Server running");

Socket s = ss.accept();
System.out.println("Client connected");

DataInputStream dis = new DataInput-
        Stream(s.getInputStream());
DataOutputStream dos = new DataOutput-
        -Stream(s.getOutputStream());

while(true)
{
    String msg = dis.readUTF();
    if(msg.equalsIgnoreCase("Bye"))
    {
      break;
    }

    String rep = isPalindrome(msg)? "Palind-
            -rome" : "Not Palindrome";

    dos.writeUTF(rep);

  }
  dos.close();
  dis.close();
  s.close();
  ss.close();
 }
}
```

Thursday

UDP (User Datagram Protocol)

Q Create a UDP application where client sends a string and server responds by echoing in uppercase

⇒

```
import java.net.*;

class Client
{
    public static void main(String []args)
    throws Exception
    {
        DatagramSocket client = new Datagram
                        Socket();
        String msg = "Hello";
        byte [] snBuffer = msg.getBytes();
        int serverPort = 6000;
        InetAddress serverIp = InetAddress.getLo-
                        -calHost();
        DatagramPacket snPacket = new Data-
                -gramPacket(snBuffer,
            snBuffer.length, serverIp, serverP
            -ort);

        client.send(snPacket);
```

```
// Receiving
byte [] rxBuffer = new byte [1024];

DatagramPacket rxPacket = new Datagram-
                -Packet( rxBuffer, rxBuffer.length);
client.receive( rxPacket);

String reply = new String( rx.getData());
System.out.println(reply);

   client.close();
  }
}


//Server Code:
import java.net.*;

class Server
{
   public static void main (String [] args)
   throws Exception
   {
      DatagramSocket server = new
            DatagramSocket (6000);

      byte [] rxBuffer = new byte [1024];

DatagramPacket rxPacket = new DatagramPacke-
                -er( rxBuffer, rxBuffer.length);

      server.receive(rxPacket);
      String msg = new String(rxPacket
      String reply = msg.to getData());

      String reply = msg.toUpperCase();

// Sending
      byte [] snBuffer = reply.getBytes();

      int clientPort = rxPacket.getPort();
   InetAddress clientIp = rxPacket.getAddress();

   DatagramPacket snPacket = new
            DatagramPacket(snBuffer,
               snBuffer.length, clientIp,
                  clientPort);
    server.send(snPacket);

     server.close();
   }
}
```

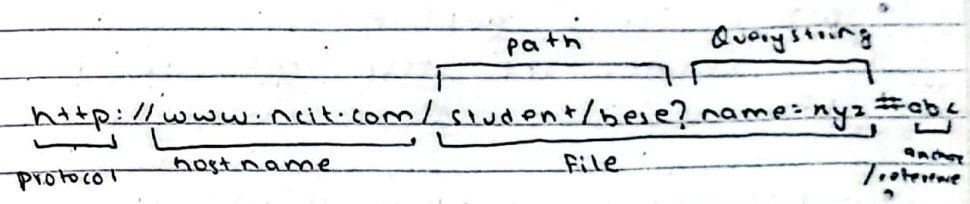
TCP	UDP
1) Transmission Control Protocol	1) User Datagram Protocol.
2) It is connection oriented protocol.	2) It is connectionless protocol.
3) It is reliable; it gurantees the delivery of data.	3) It is not reliable; it doesn't gurantee of delivery of data.
4) It has the provision of retransmitting in case of data loss.	4) It doesn't retransmit data.
5) Acknowledge segment is present.	5) It doesn't have acknowledge segment.
6) It has variable header length of 20 to 60 bytes.	6) It has header length of 8 bytes.
7) It has extensive error detection and error correction mechanism.	7) It only implements basic error detection mechanism.
8) It is slower.	8) It is faster.
9) It doesn't support broadcasting.	9) It supports broadcasting
10) It is used by HTTP, HTTPs, SMTP, etc.	10) It is used by DNS, VoIP, Live telecast, etc.

2081/09/19
Friday

URL

→ It specifies the location of resource in a computer network.

→ In Java, URL is a class which is a part of java.net package that allows us to work with URLs.



Methods

- public String getProtocol()

→ It returns the protocol defined in the URL.

- public String getHost().

→ It returns the hostname of the URL.

- public int getPort()

→ It returns the port number specified in URL.

→ If the port no. is not specified, it returns -1.

- public int getDefaultPort()
→ It returns the default port no. based on the protocol. eg: for Http. returns 80

- public String getPath()
→ It returns the path specified in the URL.
→ If path is not specified it returns null.

- public String getFile()
→ It returns the file specified in the URL.

- public String getAuthority()
→ It returns the authority of URL
(combination of hostname and port number)
e.g. localhost:3000.

- public String getQuery()
→ It returns the query string of URL.

- public String getRef()
→ It returns the reference / anchor (after #)

have to import java.io.*;
so simply (throws Exception)

Q. WAP in Java to show URL processing

→

```
import java.net.*;
class Example
{
    public static void main(String [] args)
    throws MalformedURLException
    { URL url = new URL("http://ncit.com
                /student?q=abc#smth");
      System.out.println("Protocol: " + url.getProtocol());
      System.out.println("Host : " + url.getHost());
      System.out.println("Port: " + url.getPort());
      System.out.println("Default Port: " + url.getDe-
                -faultPort());

      System.out.println("Authority: " + url.getAutho-
                -rity());
      System.out.println("Path:" + url.getPath());
      System.out.println("Query:" + url.getQuery());

      System.out.println("File:" + url.getFile());
      System.out.println("Ref:" + url.getRef());

    }
}
```

http
ncit.com
-1
443
ncit.com
/student
q=abc
/student?q=abc
smth

URLConnection class

This class provides us method that allows us to read or write from the resource specified by URL.

Q. WAP to display HTML code of "https://ncit.edu.np".

```
import java.io.*;
import java.net.*;
class Example
{
    public static void main(String [] args)
    throws Exception
    {
        URL url = new URL("https://ncit.edu.np");

        URLConnection cn = url.openConnection();

        InputStream in = new InputStream(cn.
                         getInputStream());

        int value;
        while ((value = in.read()) != -1)
        {
            System.out.print((char) value);
        }
        in.close();
        cn.close();
    }
}
```

2081/09/25
Thursday

InetAddress

→ It is a part of java.net package and it allows us to work with IP addresses.

→ It has two sub-classes:
1) Inet4Address and 2) Inet6Address to work with IPv4 and IPv6 Addresses.

Q) WAP to print IP Address of ncit.edu.np

```
import java.net.*;
class Example
{
    public static void main(String [] args)
    throws Exception
    {
        InetAddress inet = InetAddress.getByName
                           ("www.ncit.edu.np");
        System.out.println("host:" + inet.getHostName());
        System.out.println("IP Address:" + inet.getHo-
                           -stAddress());
    }
}
```

Q) WAP to all the IP Address associated with netflix.com. Also print the IP Address of localhost.

```
import java.net.*;
class Example
{
    public static void main(String [] args)
    throws Exception
    {
        InetAddress [] inet = InetAddress.
                    getAllByName("www.netflix.com");
        for(InetAddress i:inet)
        {
            System.out.println(i.getHostAddress());
        }

        //Localhost
        InetAddress local = InetAddress.getLocal
                            Host();
        System.out.println("Name :" + local.getHost.
                            -Name());
        System.out.println("IP :" + local.getHostAddress());

    }
}
```

RMI (Remote Method Invocation)

→ It is a java based technology that is used to create distributed application.

→ It allows an object on a client side to invoke methods of remote object (server)

Communication through RMI (in) takes place through stub and skeleton.

Stub

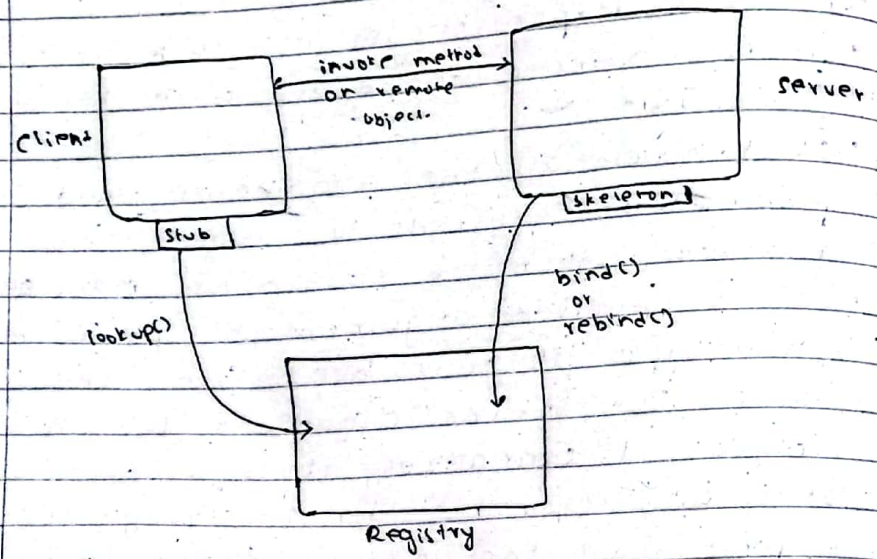
It is an object at the client side that acts as an gateway for the client. All the requests from client are routed through stub.

→ The main responsibilities of stub are:

- initiate ~~remote~~ connection with remote JVM
- It bundles all the arguments and parameters required to invoke remote method which is known as marshalling.
- If a required argument of primitive data type, it adds header in the data. But if the argument is an object it searializes it.
- It is also responsible for receiving the response from the invoked method. (remote)

Skeleton

→ Skeleton is an object that acts a gateway for server side.
→ All the incoming request are routed through it.
→ When the skeleton receives an incoming request it does the following:
- it reads the parameter for the remote method.
- it invokes the ~~obj~~ method on the remote object.
- It transmits or writes data back to the client.



Registry

→ Registry is a namespace
→ All the objects from the server are stored in this namespace
→ Every object in the registry must be bound to a unique name which is known as binding name
→ Client can locate an object in registry using lookup() method.

Imp

Q Create an RMI application where client invokes a remote method that returns the sum of two integers.

Steps

- Create a common interface that defines the methods.
- In server side provide the implementation of interface
- Bind the object in the registry.
- In client lookup for the object and invoke the method.

=>

```
// Common Interface
import java.rmi.*;
interface Maths extends Remote
{
    int sum(int x, int y) throws
        RemoteException;
}
```

```
import java.rmi.*;
import java.rmi.server.*;
class RemoteMaths extends UnicastRemoteOb-
-ject implements Maths
{
    public RemoteMaths throws RemoteException
    {
        super();
    }
    @Override
    public int sum(int x, int y)
    {
        return x+y;
    }
}
```

```
start rmiregistry 5000

//server
import java.rmi.*;
import java.rmi.registry.*;

class Server
{
    public static void main(String []args)
    throws Exception
    {
        Maths obj = new RemoteMaths()

        Naming.rebind("rmi://localhost:5000/smth",
                                    obj);
        System.out.println("Server running");
    }
}
```

```
// Client
import java.rmi.*;

class Client
{
    public static void main(String []args)
    throws Exception
    {
        Maths stub = (Maths) Naming.lookup
                        ("rmi://localhost:5000/smth");
        int ans = stub.sum(9,10);
        System.out.println(ans);
    }
}
```

2081/09/26
Friday

Q) Create an RMI application where client can invoke following methods:
(i) returns string in uppercase
(ii) returns length of string.
(iii) returns "hello world".

=>

```
// Common interface
import java.rmi.*;
interface Common extends Remote
{
    String strUp(String str) throws
    RemoteException;
    int strLength(String str) throws
    RemoteException;
    String sayHello() throws
    RemoteException;
}

// Implementation
import java.rmi.*;
import java.rmi.server.*;

class RemoteCommon implements
Common extends UnicastRemoteObject
{
    public RemoteCommon()
    {
        super();
    }

    public String strUp(String str)
    {
        return str.toUpperCase();
    }

    public int strLength(String str)
    {
        return str.length();
    }

    public String sayHello()
    {
        return "Hello World";
    }
}

// Server side.

import java.rmi.*;
import java.rmi.registry.*;
class Server
{
    public static void main(String []args)
    throws Exception
    {
        Common obj = new RemoteCommon();
        Naming.rebind("rmi://localhost:5000/
                       sth", obj);
        System.out.println("Server is running");
    }
}
```

```
// Client side
   import java.rmi.*;
class Client
{
   public static void main (String [] args)
   throws Exception
   {
      Common stub = (Common). Naming.
                    lookup("rmi://localhost:5000/sts");
      System.out.println("Upper = " +
                         stub.strUp("rajan"));
      System.out.println("Length = " +
                         stub.strLength("Rajan"));
      System.out.println("Say " + stub.say-
                         -Hello());

   }
}
```

CORBA (Common Object Request Broker Application)

→ It is a middleware that provides mechanism to create distributed application.
→ It is developed and managed by OMG (Open Management Group).
→ stub & skeleton
→ It is platform & language independent
→ It provides IDL (Interface Definition Language) to provide implementation of common interface.

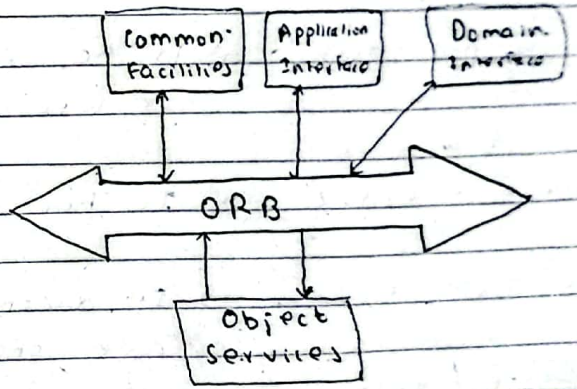


fig: OMG reference model.

1) Object Services
These are Domain independent interfaces that are used by distributed object programs, such as; the naming service allows client to find object based on name

P2P

Ans

vi) The trading services - Allows client to find object based on properties.

Comparision Between RMI and CORBA:

RMI	CORBA
i) It is language dependent i.e it can be implemented only using java.	i) It is language independent.
ii) It has simple architecture.	ii) It has complex architecture.
iii) It is server-centric.	iii) It has P2P architecture.
iv) It allows JVM to download libraries from remote JVM.	iv) It doesn't support code sharing.
v) It supports automatic Garbage collection.	v) It doesn't have Garbage collector.
vi) Interface must be defined using Java.	vi) common interface is written in IDL.
vii) It doesn't have any sp. Security mechanism.	vii) It has sp. security mechanism.
viii) JVM is responsible for locating an object.	viii) Object Adapters are used for locating objects.
ix) It uses JRMP (Java Remote Method Protocal).	ix) It uses Internet Inter-ORB Protocol.

Java Mail API

⇒ It is platform independent as well as protocol independent java framework that allows us to write, compose, read electronic mail.

⇒ All the classes of this interface are in javax.mail and javax.mail.activation package.

⇒ Protocols for Email.

i) SMTP (Simple Mail Transfer Protocol)
It provides mechanism to deliver email.

2081/10/03
Thursday

Servlet / JSP

Q) Create a web application using servlet where user sends a number and servlet should responds whether it is even or odd.

⇒

index.html

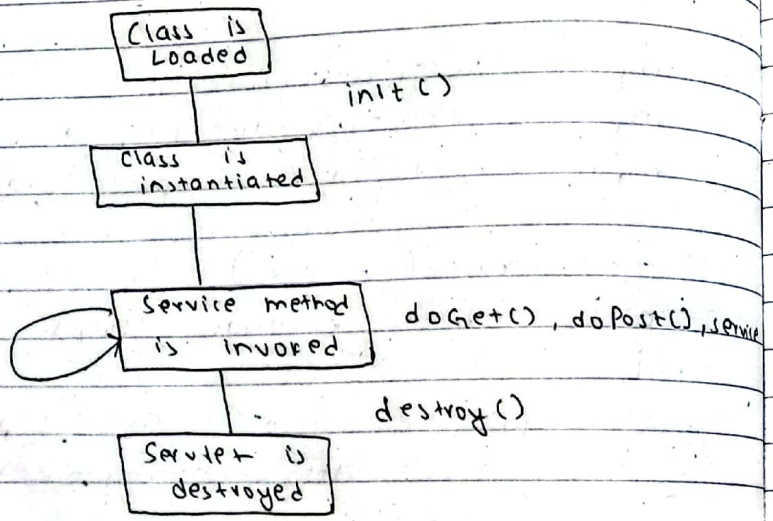
```
<html>
<body>
<form action="check">
    Enter num: <input type="text" name
                    ="num">
    <br>
    <input type="submit" value="check">
</form>
</body>
</html>
```

```
import jakarta.servlet.*;
import jakarta.servlet.http.*;
import java.io.*; import jakarta.servlet.annotation.*;

@WebServlet("/check")
class Check extends HttpServlet
{
    protected void doGet(HttpServletRequest req,
            HttpServletResponse res) throws
```

```
ServletException, IOException
{
    int n = Integer.parseInt(req.getParameter("num"));

    String ans;
    if (n%2==0)
    { ans = "Even"; }
    else
    { ans = "Odd"; }
    PrintWriter out = res.getWriter();
    out.print("<p>" + ans + "</p>");
    }
}
```

Servlet

Servlet is a java technology that is used to create dynamic web application.



@ Create a servlet that displays total number of visitors. (hit counter)

```
import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
import java.io.*;
@WebServlet("/")
class Counter extends HttpServlet
{
    private int count;
    public void init()
    {
        count = 0;
    }
    protected void doGet(HttpServletRequest req,
        HttpServletResponse res) throws
        IOException, ServletException
    {
        res.setContentType("text/html");
        count ++;

        PrintWriter out = res.getWriter();
        out.println("<b> Visit: " + count +
                    "</b>");
    }
    public void destroy()
    { }
}
```

Servlet runs inside a container in a web server and it handles requests and response.

HTTP Request

1) GET Request

This request is used to fetch a single resource or group of resource from a server.

A successful get request is responded with ~~200~~ 2XX statuscode.

We can also send data using GET request but such data are visible in URL as a parameter or query string.

2) POST Request

This request is used to send data to the server. It accepts request body. i.e data are send through request body.

→ It is used to create resource in the backend.

3) DELETE Request

→ It is used to remove resource or group of resource from backend.

4) PUT Request

→ It is used to update an existing resource.

→ It completely replaces the existing resource with request body.

5) PATCH Request

- It is also used to update an existing resource.

→ It accepts request body and partially replaces the existing resource.

Beside these, there are other http (verbs) requests such as: HEAD, TRACE.

HTTP response code

1) 1xx (Informational Response)
Response code starting from 1xx is sent when server is in processing state.
e.g 100 (Continue)
It means client should continue the request.
- 101 (Switching Protocol)

2) 2xx (Success Response)
When a user request is successful the response is accompained by 2xx.
e.g. 200 (OK): The GET req. is successful.
201 (Created): POST req is successful and new resource is created.

3) 3xx (Redirection Message)
It indicates redirection.
e.g. 300 (Multiple Choice)
→ indicates that request has more than one possible response.

301 (Moved Permanently)
→ it means the URL of the requested resource has been changed permanently.

→ 307 (Tempovary Redirection)
→ 308 (Permanent Redirection)

4) 4xx (Client Error Response)
→ indicates there has been error while processing the req. and the error is due to client.
e.g. 400 (Bad Request)
→ It means a server cannot or will not process the req. due to client error.

401 (Unauthorized)
→ The resource requested by the user is protected and client doesn't have required authorization

402 (Payment Required)
403 (Forbidden)
404 (Page Not Found)

5) 5xx (Server Error Response)
→ indicates error due to server's inability to process request.

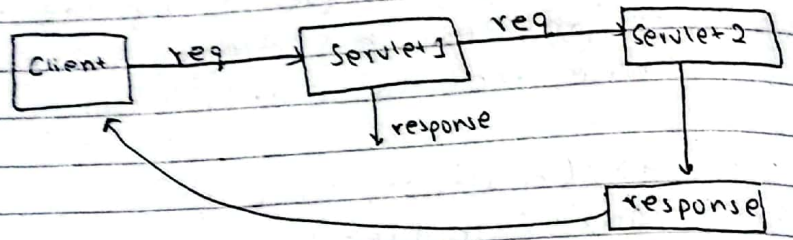
500 (Internal Server Error)
504 (

RequestDispatcher

It is an interface that allows us to dispatch a request to another resource which may be a servlet, JSP or HTML

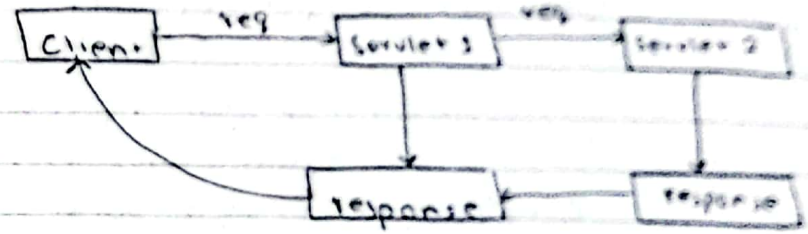
Methods

1) public void forward(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException



Using this method, a request can be forwarded into another method but the response of initial servlet is ignored. and only the response of last servlet is sent to the client

2) public void include(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException



This method also allows us to dispatch request from one resource to another but unlike the forward method it also allows us to include response of ~~servlet~~ One servlet with another

```
<html>
<body>
    <Form action:"/register" method:"POST">
    Username: <input type:"text" name
                :"username"> <br>
    Password: <input type="password"
                name :"pwd"> <br>
    <input type:"submit" value:"Signup">
    </Form>
</body>
</html>

//Database Conn & query
import java.sql.*;
class Database
{
    public static boolean Insert (String
            uname, string pwd) throws
    Exception
    {
        Class.forName ("com.msql.cj.jdbc.Driver");
        Connection cn = DriverManager.getConnection
            ("jdbc:mysql://localhost:3306/
                ncit", "root", "");

        PreparedStatement ps = cn.prepareStatement(
            "Select * from student where
                uname = ?");
        ps.setString (1, uname);

        ResultSet rs = ps.executeQuery();
        if (rs.next())
        {   return false;
        }

        Prepo
        ps = cn.prepareStatement("Insert into
            student values (?, ?)");
        int
        ps.setString (1, uname);
        ps.setString (2, pwd);

        int a = ps.executeUpdate();
        if(i==1)
        {
            return true;
        }
        return false;
    }
}
```

```
import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
import java.io.*;

@WebServlet("/register")
class Register extends HttpServlet
{
    protected void doPost(HttpServletRequest
            req, HttpServletResponse res)
    throws ServletException, IOException
    {
        res.setContentType("text/html");
        String uname = req.getParameter("username");
        String pwd = req.getParameter("pwd");

        PrintWriter out = res.getWriter();

        try
        {
            if( Database.insert(uname, pwd))
            {
                RequestDispatcher rd = req.getRequest
                                Dispatcher("/dashboard");
                rd.forward(req, res);
            }
            else
            {
                out.print("<p style='color:red;'>
                        User already exists</p>");

                RequestDispatcher rd = req.getRequestDis-
                        -patcher("/index.html");
                rd.include(req, res);
            }
        } catch(Exception e)
        {
        }
    }
}
// Dashboard


@webservlet("/dashboard");
class Dashboard extends HttpServlet
{
    protected void service(HttpServletRequest
            req, HttpServletResponse res)
    throws ServletException, IOException
    {
        res.setContentType("text/html");
        PrintWriter out = res.getWriter();
        out.print("<p> Welcome User </p>");
    }
}
```

2081/10/07

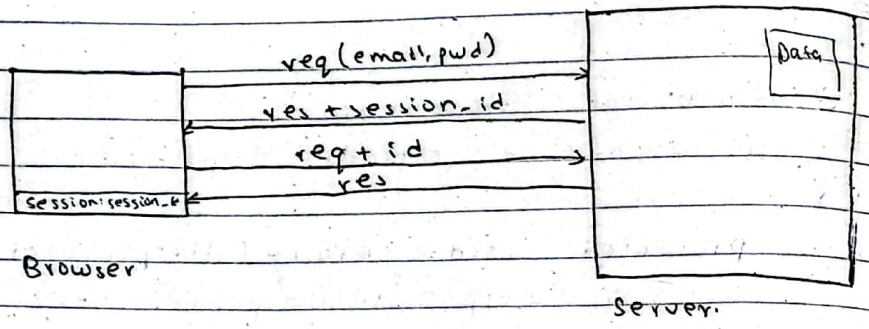
Session

→ Session is a session management technique used by web application where user data are stored in server.

→ When a session is created for user, unique session id is generated which is sent to the client through response.

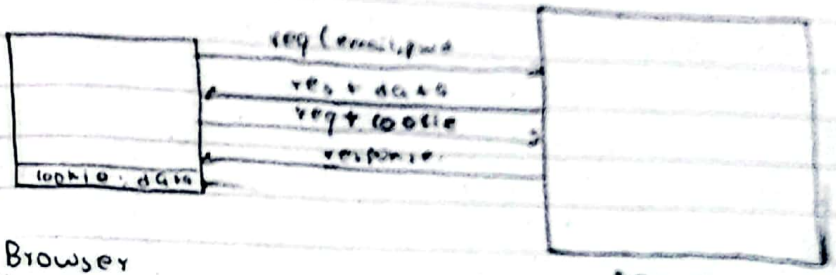
→ This session id is stored in session cookie in the user's device.

→ Every successive request is attached with session id which helps server to track the user's session.



Cookie

→ Cookie is a small text file that is stored in user's browser or device which helps to track user's activity in a web application.



Session	Cookie
1) Data are stored in server side.	1) Data are stored in client side.
2) It can store any kind of data.	2) It can only store text data
3) It is safer.	3) It can be easily tempered, relatively unsafe.
4) Theoritically there is no limit to session storage capacity. But due to limitation of scripting language, the max. session storage is 128 MB.	4) It's max. storage limit is 4KB
5) Session storage is destroyed when user logs out or closes the browser or tab.	5) Cookie is destroyed by the time specified during while making the cookie.
6) Sensitive data is stored in session.	6) Less sensitive data is stored in cookie.

Q) Create a servlet application using session that displays the date and time of first visit, last visit and the no total visits for each user.

```
import java.util.*;
import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
import java.io.*;

@WebServlet("/")
public class Visit extends HttpServlet
{
    protected void doGet( HttpServletRequest req,
        HttpServletResponse res) throws IOException,
        ServletException
    {
        res.setContentType("text/html");
        Http Session ssn = req.getSession();
        // Creates new session if session doesn't already exist
        int count;
        Date creation = new Date(ssn.getCreation
                        Time());
        Date lastAccess = new Date(ssn.get
                        LastAccessedTime() );
        IF( ssn.isNew())
        {   count = 1;
            ssn.setAttribute("visit", count);
        }
```

```
else
{
    count = (Integer). ssn.getAttribute("visit");
    count++;
    ssn.setAttribute("visit", count);
}
PrintWriter out = res.getWriter();
out.println("<p> First visit" + (creation +
                "</p>");
out.print("<p> Last visit" + lastAccess +
                "</p>");
out.print("<p> Total visit" + count + "</p>");
}
}
```

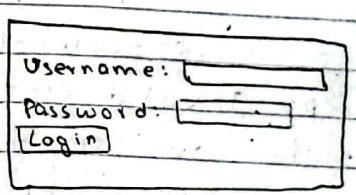
2081/10/10
Thursday

Q) Create a login application using servlet

→ If user is authenticated, store the username in session and redirect the user to dashboard. In dashboard display the username in dashboard and also display a logout button in dashboard.

→ Destroy the session when user logout

→ Also prevent the user from directly visiting dashboard without authentication.



For auth

```
import
.....

@WebServlet("/auth")
public class Auth extends HttpServlet
{

    pub
    protected void doPost(HttpServletRequest req,
                HttpServletResponse res) throws
    ServletException, IOException
    {
        res.setContentType("text/html");
        String uname = req.getParameter("uname");
        String pwd = req.getParameter("pwd");
```

```
if( uname.equals("abc") && pwd.equals("123"))
{
    HttpSession ssn = req.getSession();
    ssn.setAttribute("user", uname);
    res.sendRedirect("/dashboard");
}
else
{
    PrintWriter out = res.getWriter();
    out.print("<p> Invalid </p>");
    RequestDispatcher rd = req.getRequest
                .Dispatcher("/index.html");
    rd.include(req, res);
}
}
}
```

// Dashboard.

```
import
@WebServlet("/dashboard")
public class Dashboard extends HttpServlet
{
    protected void service( HttpServletRequest req,
        HttpServletResponse res) throws IOException,
        ServletException
    {
        HttpSession ssn = req.getSession(false);
        PrintWriter out = res.getWriter();
        if (ssn==null)
        {
            RequestDispatcher rd = req.getRequest
                    Dispatcher("/index.html");
            out.print("<p> Auth required </p>");
            rd.include(req, res);
        }
        out.println("<p> Hello " + ssn.getAttribute
                ("user") + "</p>");

        out.println("<Form action='logout'>
                <input type='submit' value=
                'logout'>
                </Form>");
    }
}
```

// Logout

```
import
@WebServlet("/logout")
public class Logout extends HttpServlet
{
    protected void service( HttpServletRequest req,
        HttpServletResponse res) throws IOException,
        ServletException
    {
        HttpSession ssn= req.getSession();
```

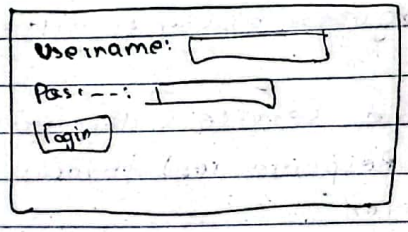
```
PrintWriter out = res.getWriter();
ssn.invalidate(); // destroy session.

out.print("<p> You have logged out
          </p>");

    }
}
```

Cookie

→ It is also a technique used for session management.

→ A cookie is a small text file that is stored in a user's browser or device.

Q) Login Using Cookie (Similar as prev. program)




```
// Auth.
Import
:

@WebServlet ("/auth")
Public class Auth extends HttpServlet.
{
    protected void service( ..... ) throws .....
    {
        String user = req.getParameter("uname");
        String pwd = req.                ("pwd");
        if( user.equals ("abc") && pwd.equals("xyz"))
        {
            Cookie ck = new Cookie("user", user);
            res.addCookie (ck);
            res.sendRedirect ("/dashboard");
        }
        else
        {
            same as अजगरि को program
        }
    }
}
```

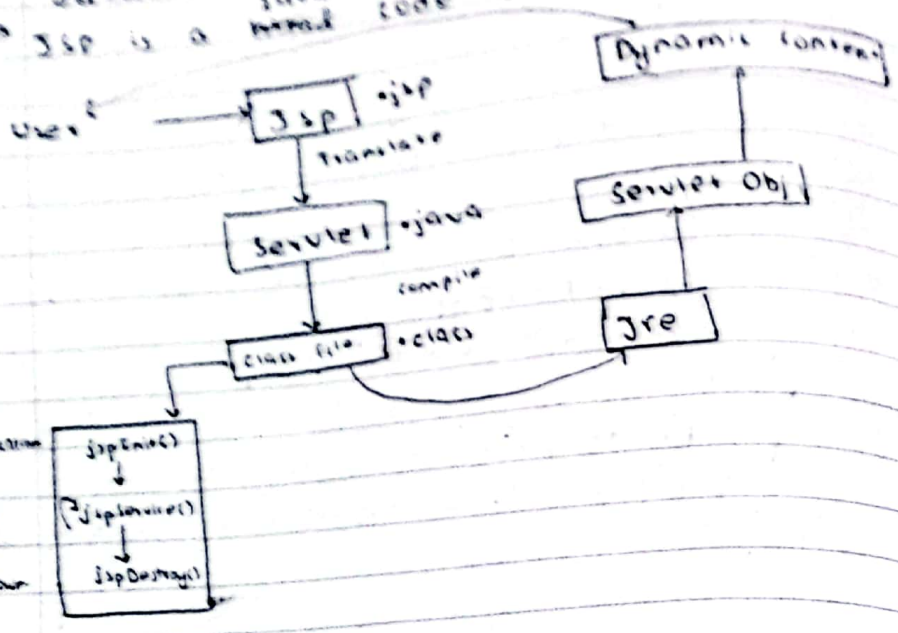
```
}
// Dashboard        ["user":abc]
import
:
@WebServlet("/dashboard")
public class Dashboard extends HttpServlet
{
    protected void service(---) throws ----
    {
        Cookie [] ck = req.getCookies();
        String name = ck[0].getValue();

        if(name==NULL || name="")
        {
            //Redirection code same as 3rd file
        }
        Print out = res.getWriter();
        outprint("<p> Hello" + name + "</p>");
        out.print("<form action='logout'>
                    <input type='submit' value=
                        'logout'>
                    </form>");

    }
}
```

```
// logout
import
:
@WebServlet("/logout")
public class Logout extends HttpServlet
{
    protected void service(...) throws ...
    {
        Cookie ck = new Cookie("user", "");
        ck.setMaxAge(0);
        res.addCookie(ck);

        PrintWriter out = res.getWriter();
        out.print("<p> You have logged out </p>");
    }
}
```

JSP

→ Java server page and new name is Jakarta server page

→ JSP is a normal code embedded in html



JSP Tags

1) Scriplet Tags <% %>
2) <%= %> Expression
3) <%! %> Declaration
4) <%@ %> Directive

Implicit objects

- request
- response
- out
- session.

6) Using JSP display the following
   1) Date and Time of first access
   2) Date and Time of last visit
   3) Total no of visit by each user (Using cookie)

index.jsp

```
<html>
<head> <title> JSP Session </title>
<body>
    <% if (session.isNew()){
        session.setAttribute("visit", 1);
      }else{
        int count = (Integer) session.getAttribute("visit");
        count++;
        session.setAttribute("visit", count);
      } %>
    <%@ page import="java.util.*" %>
    <%
        Date first = new Date(session.getCreationTime());
        Date last = new Date(session.getLastAccessedTime());
    %>
    <p> First visit = <%= first %> </p>
    <p> Last access = <%= last %> </p>
    <p> Visits = <% session.getAttribute("visit"); %> count %> </p>
</body>
</html>
```

Difference between Servlet and JSP

Servlet	JSP
i) It consists of pure java code only.	i) It is java code embedded in HTML.
ii) Servlet is faster.	ii) It is slower since it is translated into servlet.
iii) We can override service().	iii) We cannot override service() method.
iv) Session management is not initialized by default.	iv) Session management is created beforehand (by default).
v) There is no provision for tags.	v) JSP has scriplet, declaration, expression and directive tags.
vi) It is ideal for web application that needs extensive data processing.	vi) It is not ideal for web application that needs extensive data processing.
vii) In MVC pattern, servlet acts as controller.	vii) In MVC pattern, JSP acts as view.
viii) There are no implicit objects in-servlet.	viii) JSP has 9 implicit objects that can be used without declaration e.g. request, response, etc.
ix) any changes requires recompilation.	ix) Changes in html does require recompilation

Chapter: 7

Design Pattern:

→ It's a proven solution to common problems encountered during software development.
→ It enables developers to write more robust, maintainable, and scable software system.

→ characteristics:

ir Reusability:
→ Pattern can be applied to different projects and problems, which saves time and cost.

iir Standardization: They provide a ~~stan~~ shared language and understand among developers helping in communication and collaboration.

riir Efficiency: By using design patterns, developers can avoid the problems that might be encountered ~~so it~~.

ivr Flexiblity: Patterns are abstract solutions and templates that can be adapted to fit different situations.

→ Types:

i7 creational design pattern:
→ It focus on object creation and problems related to that.
→ They help v. system making the system independent of how objects are created, composed and represented.
→ eg: singleton pattern, factory, and abstract factory.

Behavioral
ii> ~~Singleton~~ pattern {<

ii> Behavioral pattern:
→ These are concerned with behavior of objects.
→ They don't just describe pattern of object and classes, but also pattern of communication between them.
→ ~~Eg~~: { Adapter method design pattern,
Fasad design pattern,
decorator design pattern. }
→ eg: Chain of responsibility.

iii> Structural design pattern:
→ It solves problems related to how ~~the~~ classes or objects to form large structure, which are more efficient.
→ types } eg: { [?

(I)r creational design pattern:
i7 singleton:
→ Singleton is used to guarantee that an object is only one and is available globally in the system.

→ eg: class Singleton {
private final static singleton instance = new singleton();
eager instantiation
private singleton() {
}

does it do?

```
public static Singleton getInstance() {
        return instance;
    }
}

Class Test{
public static void main(String[] args){
        Singleton obj1 = Singleton.getInstance();
        Singleton obj2 = Singleton.getInstance();
        if(obj1.hashCode() = obj2.hashCode()){
                System.out.println("Both are same");
        }
    }
}
```

⇒ eager instantiation / lazy instantiation.

Single Pattern:
→ It ensures that a class can have only single instance.
→ It creates an object that's used by all other program.
→ Advantages:
ii It saves memory, because only 1 object is created.

→ Usages:
ii It's used in multithreaded and db applications.
→ It's used in caching, login apps etc.
, cont

→ Steps to create to singleton:
ii It should have a static member, which contains the an instance of singleton class.
ii It should have private constructor which prevents creation of non instance from outside.
iii It should have static method to that returns an instance.

Factory Pattern (factory method pattern):
→ eg: An app has a provision of creating different Objects with different shapes. Implement factory method pattern where user can create different instances of these objects through a common factory method.

```
public interface Shape{
        void draw();
}

class Square implements Shape{
    public void draw(){
        System.out.println("I'm Square.");
    }}

class Rectangle implements Shape{
    public void draw(){
        System.out.println("I'm circle");
    }}
```

```
class Circle implements Shape {
    // System.out.println(" I'm Circle");
    public void draw() {
        System.out.println("I'm Circle.");
    }
}

class ShapeFactory {
    public static Shape createShape(String type) {
        switch(type){
            case "square":
                return new Square();
// break; is not needed because we're returning
// above.
            case "Rectangle":
                return new Rectangle();
            case "circle":
                return new Circle();
            default:
                return null;
        }
    }
}

class FactoryExample {
    public static void main(String[] args) {
        ShapeFactory fac = new ShapeFactory();
        Shape circle = factory.createShape("circle");
        circle.draw();
        Shape rec = fac.createShape("Rectangle");
        rec.draw();
        Shape sq = fac.createShape("square");
        sq.draw();
    }
}
```

(Qn) A ride sharing app provides services to book bike, car, cycle. Implement a method that displays a total fare of each type of ride, depending on the ride distance, using a factory design pattern.

=)

```
public Interface Book {
    void ride();
}

class Bike implements Book {
    public void ride() {
        System.out.println(" Bike booked. cost
            is Rs. 1000.");
    }
}

class Car implements Book {
    public void ride() {
        System.out.println(" Car booked. cost is
            Rs. 5000.");
    }
}

class Cycle implements IBook {
    public void ride() {
        System.out.println(" Cycle Booked. cost is
            Rs. 10000."); }
```

```
Class BookFactory {
    public Book createBook (String type) {
    // switch (type) {
    // case "car"
        if (type == "car") {
            return new Car();
        }
        else if (type == "Bike") {
            return new Bike();
        }
        else {
            return new Cycle();
        }
    }

Class Factory {
    public static void main (String[] args) {
    // BookBach
        BookFactory fac = new BookFactory();
        Book bike = fac.createBook ("Bike");
        bike.ride();
        Book car = fac.createBook ("car");
        car.ride();
        Book cycle = fac.createBook("cycle");
        cycle.ride();
    }
}
```

-Note: ride() ma: get distance:

-> eg: public void ride(int distance) {
    sy.o.p.("Booked bike. cost is:"+(distance*3));

Factory Pattern:

-> It defines an interface or abstract class for creating an object but let the subclasses to decide which class to instantiate. ie sub-classes are responsible for creating object of class.

-> It's also known as

-> Uses:

i) When a system needs to be independent of how its object are created, composed, and represented.

ii) When the family of related objects has to be used together.

iii) When we want to provide library of objects or that doesn't show implementation and only reveals ~~int~~ interfaces.

iv) When system needs to configured with one of multiple family of objects.

-> Advantages:

i) It isolates the code from interface.