UNIT 4 REVISION NOTES

DOS(Disk Operating System) - Character use Interface (CUI)

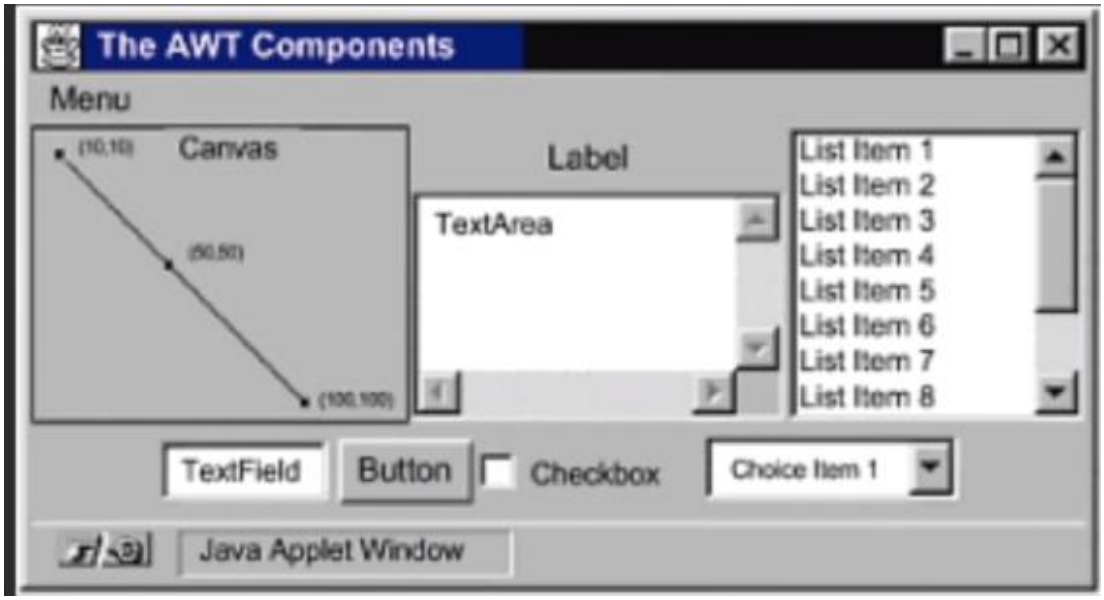
Windows Operating System and MAC - Graphical User Interface (GUI)

- **Why we need GUI ?**
  - Just to have a better look and feel of Window Operating System or any Operating that we use. Each Operating System has its own Unique GUI.

All the Programs that we practised till now and we executed the Output in the CMD is considered as CUI – Character Based Interface.

- Different Frameworks used for Developing GUI:
  - AWT - Abstract Window Interface (Outdated GUI)
  - Swings - (Outdated GUI)
  - JavaFX - Currently used GUI in Java

- **Different Components Present in AWT :**

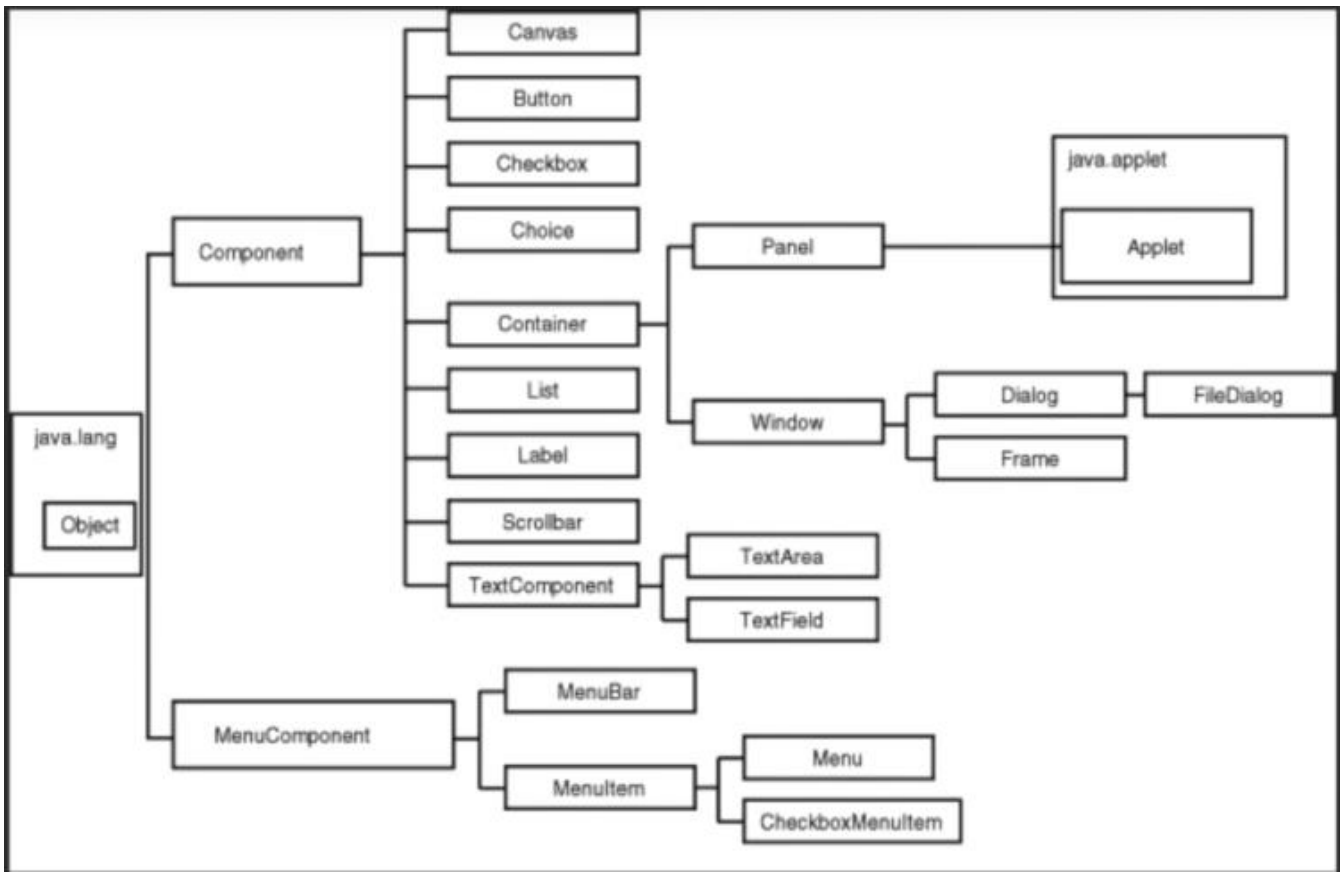


Canvas – Used for Drawing like Painters

Applet Window – for Website based Applications

Apart from this given Diagram there is also MENU and MENUITEM

- **Different Hierarchy of Java class of all these Components:**



- **For GUI AWT Programs 2 things must be Imported :**
  - Import java.awt.*;
  - Import java.awt.event.*;

- **FRAME :**

Constructor :
Frame(<“Title Name”>);

Frame f = new Frame(“New Frame”);

After Creating we need to set its Size as well as make it Visible.

f.setSize(1920,1080);

f.setVisible(true);

f.remove(Component) → used to remove a component in the Frame

- **BUTTON:**

Constructor:
Button();

Button(<“Button Name”>);

Button b = new Button(“Button”);

f.add(b); // This Functions adds the Button b into the Frame

***But This Button will cover the entire Frame. Hence we need to Set a Layout or use setBounds by using setLayout as null**

f.setLayout(new FlowLayout());

f.setLayout(new GridLayout());

f.setLayout(null);

- **TEXTFIELD:**
  Constructror:
  TextField();
  TextField(<int Size>);

  TextField tf = new TextField(20);

- **LABEL:**
  Constructor:
  Label();
  Label(<"Label Text">);

  Label lbl = new Label("Label Name");

- **Lets create a Actual Program of AWT :**

```
import java.awt.*;
import java.awt.event.*;
class MyFrame extends Frame
{
 Label l;
```

```
TextField tf;
Button b;
MyFrame()
{
 super("Frame Title");
 setLayout(new FlowLayout());
 l = new Label("Label");
 tf = new TextField(20);
 b = new Button("OK");
 add(l);
 add(tf);
 add(b);
 setSize(500,500);
 setVisible(true);
}
public static void main(String args[])
{
 new MyFrame();
}
}
```

<-Flow Layout

Now to Perform any Actions on these Components we need to know about Event Delegation Model:

- **EVENT DELEGATION MODEL**

  Every Component will have three elements:

  i) **Properties**
  ii) **Methods**
  iii) **Event** -> Components Generate Events

To Handle all these Events we need Listeners.(To Respond to all these Events).

- **For Listeners we must Implement Interfaces**

Important Event Classe and Interface

Event Classe	Description	Listener Interface
ActionEvent	generated when button is pressed, menu-item is selected, list-item is double clicked	ActionListener
MouseEvent	generated when mouse is dragged, moved,clicked,pressed or released also when the enters or exit a component	MouseListener
KeyEvent	generated when input is received from keyboard	KeyListener
ItemEvent	generated when check-box or list item is clicked	ItemListener
TextEvent	generated when value of textarea or textfield is changed	TextListener
MouseWheelEvent	generated when mouse wheel is moved	MouseWheelListener
WindowEvent	generated when window is activated, deactivated, deiconified, iconified, opened or closed	WindowListener
ComponentEvent	generated when component is hidden, moved, resized or set visible	ComponentEventListener
ContainerEvent	generated when component is added or removed from container	ContainerListener
AdjustmentEvent	generated when scroll bar is manipulated	AdjustmentListener
FocusEvent	generated when component gains or loses keyboard focus	FocusListener

- **BUTTON, ACTION EVENT and ACTION LISTENER**

  Program – Create a Label and Button whenever we click the Button the Text in the Label should be Increamented by 1.(Using Adapter Class)

```
import java.awt.*;
import java.awt.event.*;
class MyCounter extends Frame
{
 Label lbl;
 Button b;
 int i = 0;
 MyCounter()
 {
  super("Counter");
  lbl = new Label("    ");
  b = new Button("Click");
  setLayout(new FlowLayout());
  add(lbl);
  add(b);
  setSize(500,500);
  setVisible(true);
  b.addActionListener(new ActionListener()
  {
   public void actionPerformed(ActionEvent e)
   {
    i += 1;
    lbl.setText(""+i);
```

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

- **TextField**

  TextField tf = new TextField(size);

tf.setText() , tf.getText().toLowerCase()
tf.getText().toUpperCase()

tf.getText()

tf.setEchoChar("*") → For password kind of Approach

It Implements TextListener → TextEvent te

TextListener → .textValueChanged() {Should be overrided}

- **TextArea**

  TextArea ta = new TextArea(rows,columns);

  ta.getSelectedText() → Give you the Selected Text by the User
  ta.append(String) → Used to Append the text into the Text Area
  ta.insert(string,position);
  position – if we want it to be the Cursor position
  ta.getCaretPosition() → Returns Cursor position

- **List and Choice**

  Collection of Checkboxes → List
  Collection of Radio Buttons → Choice

  List l = new List(4,true);
  4->No. of Rows|| true -> Multiple Items can be Selected
  l.add("Monday");
  l.add("Tuesday");
  Choice c = new Choice();
  c.add("January");
  c.add("Febrauary");
  add(l); → Adding it to the Frame
  add(c);

  TO Handle These Event we need to implement ItemListener

  ItemStateChanged() → Overrided

  REMEMBER FOR EVERY EVENT IN ORDER TO KNOW WHICH COMPONENT IS CLICKED → Event.getSource()

  l.getSelectedItem();
  I.getSelectedItems(); → For Multiple Choices

In Some cases for List we also need to use ActionListener instead of ItemListener one such example is that we select multiple Items and we need to Display them in Text Area by clicking those Items in such cases ActionListener must be implemented.

.getItems()->List of all Items
.getSelectedIndex();
.getSelectedIndexes();
.remove(name or position);
.select();
.setMutipleMode(true or false);

FOR SWING JLIST AND JCOMBOBOX IS USED

- **ScrollBar**

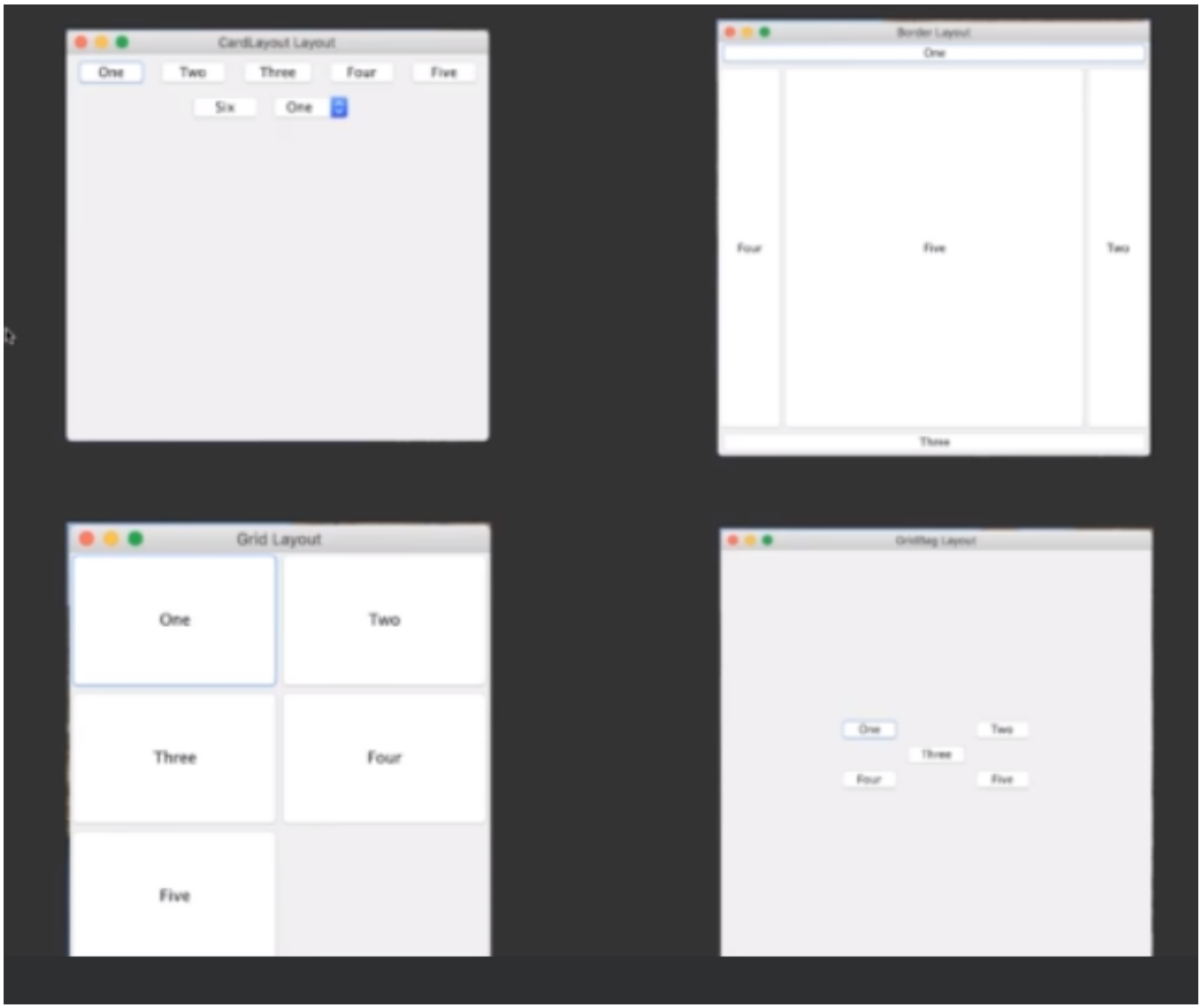
ScrollBar red = new ScrollBar();
ScrollBar green = new ScrollBar(int Orientation);
Orinetation
ScrollBar.HORIZONTAL,ScrollBar.VERTICAL
ScrollBar red = new ScrollBar(int Orientation, int current value, int visibility,int min value,int max value);

For Handling Events we need a AdjustmentListener

adjustmentValueChanged(AdjustmentEvent ae)
→Overrided

.getValue() → Returns the Value of the Scrollbar

- **Layout Managers**



Flow Layout → Line by Line Arrangement of Components

Border Layout → Regions EAST,WEST,NORTH,SOUTH,CENTRE

Grid Layout → Arrangement of Components in form of rows x columns

Card Layout → Drop Down Menu and Choosing Components

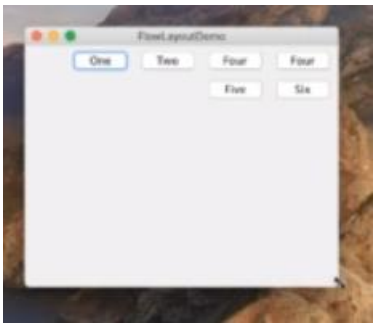
Just like Tabs in Chrome

- **Flow Layout**

Flow Layout f = new FlowLayout();

f.setAlignment(FlowLayout.RIGHT);

setLayout(f);



- **Border Layout**

Default Layout in JFrame is Border Layout

Add(b,BorderLayout.EAST);

Add(b1,BorderLayout.NORTH);

Very useful in cases of using a Panel insde a Frame

JPanel p = new Panel();

p.add(b);

p.add(b1);

add(p);

- **Grid Layout**

setLayout(new GridLayout(2,3)); → 2 rows and 3 columns

- **Key Event**

Adapter must be Used as Three Functions must be Overrided

keyPressed()

keyReleased();

keyTyped();

KeyListener must be Implemented.

.getKeyCode()

.getKeyChar();

.getKeyModifiersText();

.getKeyText()

.getWhen() → When the key is been Pressed

Virtual Keys → KeyEvent.VK_A,VK_1

- **Mouse Event**

MouseListener

mouseEntered(MouseEvent me)

mouseExited(MouseEvent me)

mouseClicked(MouseEvent me)

mousePressed(MouseEvent me)

mouseReleased(MouseEvent me)

MouseMotionListener

mouseMoved(MouseEvent me)

mouseDragged(MouseEvent me)

.getButton() → me.BUTTON1 BUTTON2 BUTTON3

.getPoint() → (x,y) Mouse Cursor

getX() → X Coordinate

getY() → Y Coordinate

- **Menu**

MenuItem,CheckBoxMenuItem → Menu → MenuBar → Frame

MenuItem open = new MenuItem(“Open“);

CheckboxMenuItem auto = new CheckBoxMenuItem(“Auto Save”);

Menu file = new Menu(“File”);

File.add(open);

File.add(auto);

MenuBar mb = new MenuBar();

Mb.add(file);

setMenuBar(mb); → Setting it to the Frame

The Events are Handled with ActionListener

- **PopupMenu**

PopupMenu mnu = new PopupMenu();

mnu.add(MenuItem);

Functions :

Mnu.show(txt,me.getX(),me.getY());

JCheckedMenuItem → for Swing

- **Checkbox**

ItemListener()
Public void ItemStateChanged(ItemEvent ie)

Ie-→ getStateChanged()

Checkbox Functions
getState() -→ Checkbox Selected or Not int SELECTED
DESELECTED
isSelected()
getLabel → name of the Checkbox
setMnemonic(KeyEvent VK_);

cbg = new CheckBoxGroup(); → To Create Radio Buttons

Cb1 – new Checkbox("CheckBox",false,cbg);
JRadio Button → only one can be Selected

Should to be Added in ButtonGroup bg = new ButtonGroup(); bg.add(r1),bg.add(r2);

getRootPane().setDefaultButton(b) → This set a Default button an Highlighted button when we click enter the button gets clicked and we can toggle through other buttons using TAB.

TO Create icon to a Button b.setIcon(new ImageIcon(“ addresswith / ”));

JFormattedTextField ->IT is used to only get Data in a particular format like only we need numbers or dates or strings we can get it using JFormattedTextField

JFormattedTextField tf = new JFormattedTextField(format);

We need to set Its columns for it to be Visible so,

Tf.setColumns(20);

Tf.setValue(new Date())→to Set Current Date →java.util.*

This format can be DateFormat,NumberFormat and so on…

For DateFormat,

Import java.text.*;

DateFormat df = new SimpleDateFormat(“dd/MMorMMMM/yyyy”);

JFormattedTextField tf = new JFormattedTextField(df);

For NumberFormat →

We need to import java.text.*;

NumberFormat nf = NumberFormat.getInstance();

NumberFormat nf = NumberFormat.getCurrencyInstance(Locale.US);

For NumberFormatter we need javax.swing.text.*;

NumberFormatter nft = new NumberFormatter(nf);

Nft.setAllowsInvalid(false); → only allows numbers to enter

We can also set Maximum minimum numbers and so on

Nft.setMaximum(1000000);

- **Split Pane**

JSplitPane sp = new JSplitPane(JSplitPane.HORIZONTAL_SPLIT,sp1,sp2);

Sp1 = new ScrollPane();

Sp2 = new ScrollPane();

Sp.setDividerLocation(200);

- **Tabbed Pane**

JTabbedPane tp = new JTabbedPane();

Tp.add("Colors",p1);

Tp.addTab("Label",p2);

- **Table Format**

DefaultTableModel model = new DefaultTableModel();

Model.addColumn(“Column Name”);

Model.addRow(Object [] {col1,col2,col3});

Model.getRowCount();

Model.getValueAt(Position,0);

Model.removeRow(Position);

Model.insertRow(Position,new Object[]{col1,col2,col3});

Model.setRowSelectionInterval(Position,0) → HighLighting Row

- **Error Messages**

JOptionPane.showMEssageDialog(jf,”Error Message”,”Error”,JOptionPane.WARNING_MESSAGE);

- **Writing and Reading a Binary File with Objects:**

FileOutputStream fout = new FileOutputStream();

ObjectOutputStream out = new ObjectOutputStream(fout);

Out.writeObject(new Student());

Out.close();fout.close();

```
FileInputStream fin = new FileInputStream();
ObjectInputStream in = new ObjectInputStream(fin);
Student s = (Student) in.readObject();→ Exception try block
```

- **Opening a File Dialog and Writing it into Frame**

```
JFileChooser jc=new JFileChooser();
int options=jc.showOpenDialog(jf);
if(options==JFileChooser.APPROVE_OPTION){
try{
        File f=jc.getSelectedFile();
        FileInputStream fis=new FileInputStream(f);
        byte b[]=new byte[(int)f.length()];
        fis.read(b);
        ta.setText(new String(b));
        jf.getContentPane().repaint();
}
```