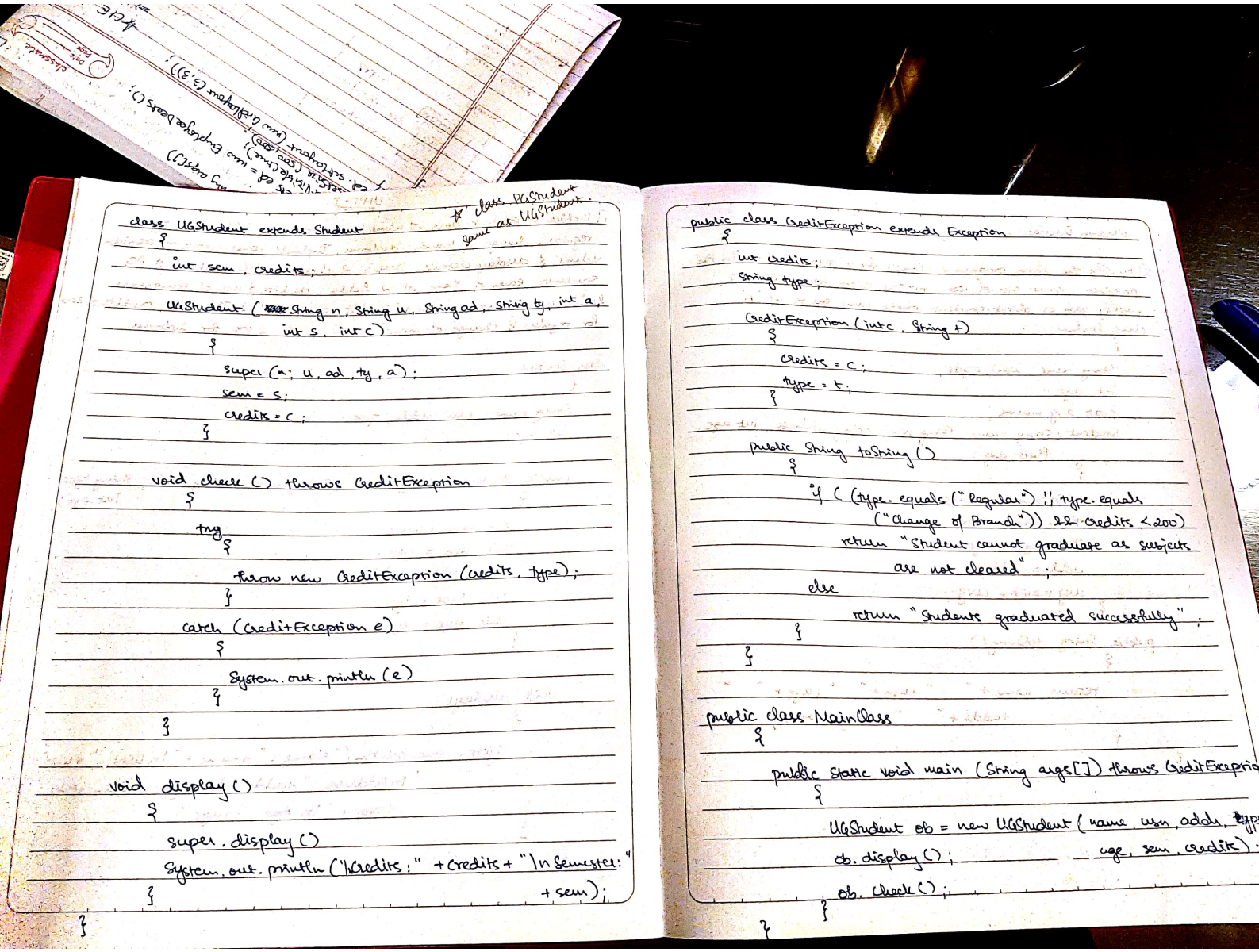
UNIT-I

1] Define a student class w/ name, usn, age, address & type - regular, change of branch, diploma. Include constructors to assign values & display details. Define 2 classes : UG student & PG student, each of them w/ 2 fields : credits & no. of semesters. Raise a UDException called 'Credit exception' : If no. of credits ≤ 200 for regular & change-of-branch students & < 150 for diploma.

```
class Student
  {
  String name, usn, addr, type;
  int age;

  public Student (String name, String usn, String addr, String type,
                  int age),
      {
        this.name = name;
        this.usn = usn;
        this.addr = addr;
        this.type = type;
        this.age = age;
      }

  void display()
      {
        System.out.println("Name :" +name+ "\n USN :" +usn+
                "\n Address :" +addr+ "\n Type of Semester:"
                +type+ "\n Age:" +age);
      }
}
```



```
class UGStudent extends Student
{
    int sem, credits;

    UGStudent (String n, String u, String ad, String ty, int a,
               int s, int c)
    {
        super (n, u, ad, ty, a);
        sem = s;
        credits = c;
    }

    void check () throws CreditException
    {
        try
        {
            throw new CreditException (credits, type);
        }
        catch (CreditException e)
        {
            System.out.println (e)
        }
    }

    void display ()
    {
        super.display ()
        System.out.println ("Credits:" + credits + "In Semester:"
                            + sem);
    }
}
```

```
public class CreditException extends Exception
{
    int credits;
    String type;

    CreditException (int c, String t)
    {
        credits = c;
        type = t;
    }

    public String toString ()
    {
        if ((type.equals ("Regular") || type.equals
            ("Change of Branch")) && credits < 200)
            return "Student cannot graduate as subjects
                    are not cleared";
        else
            return "Students graduated successfully";
    }
}

public class MainClass
{
    public static void main (String args[]) throws CreditExcepti
    {
        UGStudent ob = new UGStudent (name, usn, addr, ty
        ob.display ();                  age, sem, credits);
        ob.check ();
    }
}
```

I Collection Interface :

Q] Modify the given program : Check for a given student in the Arraylist, print size of list, retrieve object at location 2, modify usn of object student whose name starts w/ 'A'.

```
class Student
{
    String name, usn, addr;
    int age;
    float avg_marks;
    Student (String name, String usn, String address, int age,
             float avg)
    {
        this.name = name
        this.usn = usn
        this.age = age
        addr = address
        avg_marks = avg;
    }
public String toString()
    {
        return name + "  " +usn + "  " +age + "  "
               +addr + "  " + avg_marks;
    }
}
```

```
class MainClass
{
    public static void main (String args[])
    {
        ArrayList <Student> arrst = new ArrayList <Student> ();
        arrst.add (new Student ("ABC", "1MS17IS041",
                "Bangalore", 18, 89.68));
        Student s1 = new Student ("PQR", "1MS18IM038", "Mumbai",
                19, 80.83);
        arrst.add (s1);
        for (Student s: arrst)
        {
            System.out.println(s);
        }
    }
}
```

```
Student s2 = new Student ("ABCD", "1MS18CS091", "Chennai", 18,
arrst.add(s2);                                          82.75);
int length_arrlist = arrst.size();
System.out.println(arrst.get(2));
for (Student s: arrst)
{
    if (s.name.charAt[0] == 'A')
    {
        int in = arrst.indexOf(s);
        s.usn = "1MS19041";
        Student s3 = arrst.set(in, s);      // replaces the whole object.
    }                  ↑ old value    ↳ returns old value
}
```

Q] Print the students from index location 1 to 5.

a)
```
for (Student s : arrst)
 {
  int in = arrst.indexOf(s)
  if (in >=1 && in<=5)
     System.out.println(s);
  if (in>5)
     break;
 }
```

OR

b)
```
ArrayList <Student> stdlist = arrst.sublist (1, 6);
```
(6 → 6-1 = 5)

Syntax: `List <E> ls = <List l>.sublist (begin index, end index);`

Q] Remove Student object having usn 1MS18CS038

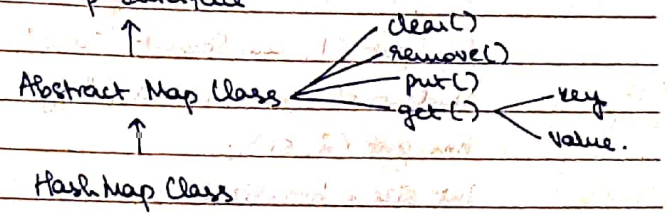
```
for (Student s : arrst)
 {
  if (s.usn.equals ("1MS18CS038"))
   {
    int in = arrst.indexOf(s);
    Student s3 = s;
   }
 }
arrst.remove(s3);
```

★ Iterator:

```
ArrayList <Student> al = new ArrayList <Student> ();
Iterator <Student> it = al.Iterator();
while (it.hasNext())
 {
  Student s1 = it.next();
  System.out.println (s1.name);
  System.out.println (s1.age);
 }
```

II Map Interface:

Maps are key value pairs.



Q] class Student
```
 {
  String name, usn;
  int age;
 }
```

```
Student (String name, String usn, int age)
{
this.name = name;
this.usn = usn;
this.age = age;
}
public String toString()
{
return name+"\n"+usn+"\n"+age;
}
}

Class MapClass
{
public static void main (String args[])
{
```

Create a collection ⇒ `HashMap <Integer, Student hm = new HashMap <Integer, Student>();`

Adding objects ⇒
```
hm.put (1, new Student ("abc", "15041", 20));
Student s1 = new Student ("efg", "18039", 19));
hm.put (2, s1);
```

Obtain size ⇒ `int size = hm.size();`

Obtain obj from collection ⇒ `Student s2 = hm.get (2);`

Converting Map to Set ⇒
```
Set <Map.Entry <Integer, Student>> set1 = hm.entrySet();
for (Map.Entry <Integer, Student> set2 : set1)
{
System.out.println (set2.getKey());
System.out.println (set2.getValue());
}
```

OR

```
for ( - - set2 : set1)
{
S.O.P (set2.getValue().name);
S.O.P (set2.getValue().display());
}
```

* getKey() & getValue() are methods of Map.Entry NOT HashMap.

Note: Iterator cannot be used to iterate through maps; only for each can be used.

1) Modify USN of student w/ ID = 1

```
Student s1 = hm.get(1);
s1.usn = "1MS17" + s1.usn;
hm.put (1, s1);
```

2) Remove obj w/ value ID = 10

```
Student s1 = hm.get(10);
hm.remove (s1);
```

⇒ `boolean isEmpty();`

⇒ `Set <K> keySet()` → set of keys.

`HashSet s1 = hm.keySet();`

`s1 = {1, 2, 3};`

⇒ `boolean containsKey (Object k)`

`boolean v1 = hm.containsKey (3);`

⇒ `boolean containsValue (Object v)`

`boolean v2 = hm.containsValue (s1);`

4/2/20

Hashmap Constructors

```
Hashmap<Integer, Double> hm = new Hashmap<Integer, Double>();
Hashmap<Integer, Double> hm2 = new Hashmap<Integer, Double>();

hm.put(1, new Double(23.55));
hm.put(2, new Double(128.56));
hm.put(3, 168.36);

Hashmap<Integer, Double> hm1 = new Hashmap<Integer, Double>(hm);
                                //Second constructor

hm2.putAll(hm);

Hashmap<Integer, Double> hm3 = new Hashmap<Integer, Double>(10);
                                //Third constructor

Hashmap<Integer, Double> hm4 = new Hashmap<Integer, Double>(16, 0.25);
```

(* as soon as 75% completes, size changes)

Incompatible

ClassCastException

size

fill ratio

EXCEPTIONS

1. ClassCastException
2. NullPointerException - placing a null value
3. UnsupportedOperationException - replace / place a change / update in an unmodifiable hashmap
4. IllegalArgumentException - invalid argument into a hashmap

Create a collection - Employee having unique emp id, name, designation, qualification, address & salary.

a) If emp id is 0010, modify the designation.

b) Obtain emp ids & an array list of employees from the map

c) Based on qualification of employee, do the following:

i) If qualification = high, tax to be paid = 30% of salary.

ii) If " = medium, " " = 20% of salary

iii) If " = low, " " = 10% of salary.

- Place tax details of employees in another map.

d) Print only those employee's details who have emp id divisible by 3.

e) For a given emp id remove details from hashmap.

f) Replace employee details whose emp name is "Diya".

```
class Employee
{
    String name, desg;
    String qual, addr;
    double sal;

    Employee(String n, String d, String q, String a, double sal)
    {
        name = n;
        desg = d;
        qual = q;
        addr = a;
        this.sal = sal;
    }
```

```
public String toString()
    {
    return name + "  " + desg + "  " + qual + "  "
        + addr + "  " + sal;
    }
}

class EmployeeDeets
{
public static void main (String [] args)
    }
    HashMap <Integer, Employee> hmemp = new HashMap
                                <Integer, Employee> ();
    Employee emp1 = new Employee ("Diya", "PAR", "High",
                                "Street 101", 164785);
    Employee emp2 = new Employee ("Deeksha", "XYZ", "Medium",
                                "Street 501", 155432);
    Employee emp3 = new Employee ("Yash", "ABC", "Low",
                                "Street 102", 164785);
    hmemp.put (10, emp1);
    hmemp.put (20, emp2);
    hmemp.put (30, emp3);

(a) if (hmemp.containsKey (10))
    {
    Employee e1 = hmemp.get (10);
    e1.desg = "modify";
    hmemp.put (10, e1);
    }
```

set - has no duplicate elements list - duplicate elements map - key value pair

```
Set <Integer> ids = hmemp.keySet ();   // places keys in a set
ArrayList <Employee> empA = hmemp.values ();

Collection <Employee> V = hmemp.values ();   // place value into an
(b) ArrayList <Employee> emv = new ArrayList      Arraylist Collection
                                <Employee> (V);

(c) HashMap <Integer, Double> taxhm = new HashMap <Integer, Double> ();
    → double taxPay = 0.;
    Set <Map.Entry <Integer, Employee>> set = hmemp.entrySet();
    for (Map.Entry <Integer, Employee> s : set)
        {
        int key = s.getKey ();   // getKey() - func of Map.Entry
        Employee emp = s.getValue ();   // getValue() - " "
        if (emp.qual.equals ("high"))
            { taxPay = 0.3 * emp.sal;
            }
        else if (emp.qual.equals ("medium"))
            { taxPay = 0.2 * emp.sal;
            }
        else
            taxPay = 0.1 * emp.sal;
        taxhm.put (s.getKey (), taxPay);

(d) (e) if ((s.getKey() % 3) == 0)
        {
        System.out.println (s.getKey());
        System.out.println (s.getValue());
        }
    }
```

(e)
```
int empid = 20;
Employee old = hmEmp.remove(20);
```

(f)
```
for(Map.Entry<Integer, Employee> set1 : set)
{
    Employee emp = set1.getValue();
    emp.name.replace("Diya");
}
```

Arrays Class

An array class provides various methods that are useful when working w/ arrays. These methods bridge the gap between collections & arrays.

```
eg: int intar = new int[10];
    // read elements.
    Arrays.sort(intar); // to sort elements in array intar.
int loc = Arrays.binarySearch(intar, 16); // returns index of key searched
int[] cop = Arrays.copyTo(intar, 9); // copies first 9 elements of intar to cop
int[] cop1 = Arrays.copyOfRange(intar, 3, 8); // copies 3rd to 8th elements of
                                              intar to cop1
boolean b1 = Arrays.equals(intar, cop1); // Compares arrays.
```

Q] Create an Employee class, Sort it based on (a) Salaries & then on (b) name

```
class Employee
{
    String name;
    double sal;
    int eid;

    Employee(String n, double s, int id)
    {
        name = n;
        sal = s;
        eid = id;
    }

    public String toString()
    {
        return name + "\n" + sal + "\n" + eid;
    }
}

(a) class EmpCompare implements Comparator<Employee>
{
    public int compare(Employee e1, Employee e2)
    {
        return (int)(e1.sal - e2.sal);
    }
}

class MainClass
{
    public static void main(String args[])
    {
        Employee[] emp = new Employee[10];
        emp[0] = new Employee("ABC", 23456, 10);
```

```
emp[1] = new Employee ("DEF", 78910, 12);
emp[2] = new Employee ("ADE", 43217, 11);
Arrays.sort(emp, new Empl
}
}
// if comparable is used, then CompareTo must be overridden
```

(b)
```
class EmpName implements Comparator<Employee>
{
public int compare(Employee e1, Employee e2)
{
return (e1.name.compareTo(e2.name));
}
}
```

```
class MainClass
{
psvm()
{
Employee[] emp = new Employee[10];
for (Employee e : emp)
{
S.O.P (e);
}
}
}
```

* Descending order

```
Comparator<Employee> reverseemp = Collections.reverseOrder
Arrays.sort(emp, (new Employee.compar
(emp, reverseemp);
```

UNIT - II

Swings

Swing is a java foundation class (JFC) library & an extension of abstract window toolkit (AWT)

The JFC contains:

i) Swing - which is a large UI component package
ii) Cut & paste - clipboard support
iii) Accessibility features - aims at developing GUI's for users of disabilities
iv) Desktop colour features
v) Java 2D - provides improved colour, image & text support.

Swings key features:

(1) Light weight components:
- For a component to qualify as light weight, it must not depend on any non-java system classes.
- Swing components have their own view supported by java look & feel classes.

(2) Pluggable look & feel:
- This feature enables the user to switch the look & feel of swing components w/o restarting an application
- i.e., look & feel of components remain the same across all platforms wherever the program runs.

11/2/20

★ Difference b/w AWT & Swings:

Simple swing components have the following capability that are not available for AWT components:

a) Swing buttons & labels can display images in addition to text.

b) Borders around most swing components can be changed.

c) Swing components need not be rectangular in shape.

d) Assistive technologies such as screen readers can easily get information from swing components.

Imp points:

① Package used to write swing applications ⇒ javax.swing.*;
② To change border ⇒ import javax.swing.border;
③ for events ⇒ import javax.swing.event;
④ for pluggable look & feel ⇒ import javax.swing.plaf;

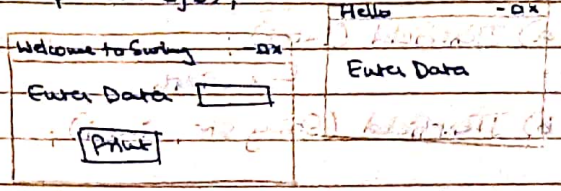
★ Swing Components:

JFrame	JList
JPanel	JMenu
JLabel	JTextField
JButton	JPassword
JCheckBox	JTextArea
JOptionPane	JRootPane
JCheckBoxMenu	JToolBar
JRadioButton	JScrollBar
JButtonGroup	JComboBox

★ Steps to create a control Pane for Java V6 & below:

① Create a frame obj
JFrame frm = new JFrame ("Hello");
② Create a handle to its contentPane
Container cp = frm.getContentPane();
③ Add objects to this container.
cp.add (some component);
↳ eg: JLabel jl = new JLabel ("Enter Data");
cp.add (jl);

Q] Create a frame:




```
import javax.swing.*;
class FirstSwing
{
    psvm (String args[])
    {
        JFrame frm = new JFrame ("Welcome to Swing");
        JLabel jl = new JLabel ("Enter Data");
        JTextfield jtxt = new JTextfield (20);
        JButton jbn = new JButton ("Print");
        //Add all to frame in order
        frm.add (jl);
        frm.add (jtxt);
        frm.add (jbn);
        frm.setSize (500, 800);
        frm.setLayout (new GridLayout (4,8));
        frm.setVisible (true);
    }
}
```

★ Constructors of Swing Components:

① JLabel Constructors: (LEFT, RIGHT, CENTER → constants of class called SwingConstants (int values))

a) JLabel (String str, int i);

b) JLabel (String str, Icon ic);
↳ Icon ic = new Icon ("1.jpg");

c) JLabel (Icon ic);

② JTextfield Constructors:

a) JTextfield (int i);
↳ width

b) JTextfield (String str, int i);

③ JPasswordfield Constructors:

a) JPasswordfield (int i);
↳ width

b) JPasswordfield (String str, int i);
↳ default password

④ JButton Constructors:

a) JButton (String str);

b) JButton (String str, Icon ic);

c) JButton (Icon ic);

⑤ JTextArea Constructors:

a) JTextArea (int row, int col);

b) JTextArea (String str, int row, int col);

⑥ JCheckBox Constructors: (JRadio Button uses same constructors).

a) JCheckBox (String str);

b) JCheckBox (boolean);
↳ TRUE / FALSE. → default - unchecked.

c) JCheckBox (Icon);

d) JCheckBox (Icon, boolean);

e) JCheckBox (String, Icon);

f) JCheckBox (String, Icon, boolean);

⑧ JComboBox Constructors:

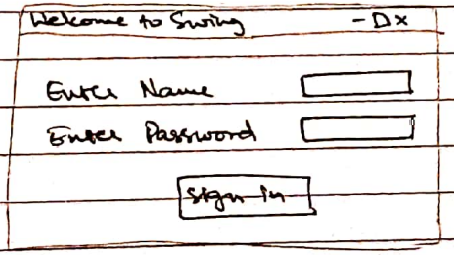
a) JComboBox ();

b) JComboBox (Object [ ]);
↳ eg: String [ ] st = {"item1", "item2"};
JComboBox cbox = new JComboBox (st);

★ Radio Buttons must be placed in a Button Group & the Button Group must be placed to the frame.

★ Multiple panels can be added to a frame.
Multiple frames cannot be added to a container.

Q] Create the frame shown below: using panels.



```
import javax.swing.*;
class SecondSwing
{
  psvm (String args[])
  {
    JFrame frm = new JFrame ("Welcome to Swing");
    JPanel jpan1 = new JPanel();
    JLabel jl1 = new JLabel ("Enter Name");
    JTextField jtxt = new JTextField (20);
    jpan1.add (jl1);
    jpan1.add (jtxt);
    frm.add (jpan1);
    JPanel jpan2 = new JPanel();
    JLabel jl2 = new JLabel ("Enter Password");
    JPasswordField jpass = new JPasswordField (15);
    jpass.setEchoChar ('#'); // whatever is typed, displayed as #
    jpan2.add (jl2);
    jpan2.add (jpass);
    frm.add (jpan2);
    JButton jb = new JButton ("Sign In");
    frm.add (jb);
  }
}
```

12/2/20

Q] Create a swing application to allow the user to enter a user id & password on the click of a button. Check for validity.

```
import javax.swing.*;
import javax.event.*;   // to perform at click of button import event package
public class LogicClass extends JFrame implements ActionListener
{
  JTextField login_id;
  JPasswordField passwd;
  JButton check;
  LoginClass
  {
    super ("Login Page");
    JLabel jusname = new JLabel ("UserName:");
    JLabel jpass = new JLabel ("Password:");
    login_id = new JTextField (20);
    passwd = new JPasswordField (20);
    check = new JButton ("Submit");
    login_id.setMnemonic ('a');   // Shortcut key
    check.setMnemonic ('?');
    login_id.setToolTipText ("Enter username");
    passwd.setToolTipText ("Enter valid value");
    add (jusname);
    add (login_id);
    add (jpass);
    add (passwd);
    add (check);
    check.addActionListener (this);
  }
```

any listener event has this syntax → ___Performed(___Event)

```
public void actionPerformed (ActionEvent)
  {
②{ if (evt.getSource() == check)   // used when multiple JButton objs present.
    {
     if (login_id.getText().equals ("Hello123"))
      {
       if (passwd.getText().equals ("123heLLo"))
         System.out.println ("Save Password?");
       else
         System.out.println ("Enter valid password");
      }
     else
       System.out.println ("Enter valid username");
    }
  }
public static void main (String args[])
  {
   LoginClass log = new LoginClass();
   log.setVisible (true);
   log.setSize(500, 800);
   log.setLayout (new GridLayout (7,5));
  }
}
```

① * Instead of "implements ActionListener" we can do like

```
check.addActionListener(new ActionListener() {
  public void actionPerformed (ActionEvent) { }
});
```

→ don't need to check source ② not regd.

* Alignments are constants. ← int.

* Methods of Swing components:

① JLabel Class:

① void setHorizontalAlignment (int);
↳ LEFT, RIGHT, CENTER, TRAILING, LEADING

② void setVerticalAlignment (int);
↳ LEFT, RIGHT, CENTER

③ int getHorizontalAlignment ();

④ int getVerticalAlignment ();

eg: JLabel jl = new JLabel ("Enter Data", SwingConstant.LEFT);
jl.setHorizontalAlignment (SwingConstant.TRAILING);

⑤ void setIconTextGap (int)
↳ no. of pixels space b/w icon & text.

⑥ int getIconTextGap ();

⑦ void setIcon (Icon img);

⑧ Icon getIcon ();

⑨ String getText ();

⑩ void setText (String str);

eg: String s1 = jl1.getText();
jl1.setText ("Place Data");

② JTextfield Class:

① String getText ();
eg: JTextfield jtxt = new JTextfield (20);
String st = jtxt.getText();

② void setText (String str);

③ void setEditable (boolean); // if true (default), data can be placed in text box.
↳ TRUE / FALSE

④ boolean getEditable();

⑤ void setColumns (int); // used to increase/decrease size while working

⑥ int getColumns();

⑦ void setHorizontalAlignment (int) → default.
↳ LEFT, CENTER, RIGHT

⑧ int getHorizontalAlignment();

③ JPasswordField Class:

Uses: all ⑧ as JTextField class methods +

eg: JPasswordField pwd = new JPasswordField(20);

⑨ void setEchoChar (char);
↳ can be any char → '#' ; * → default

⑩ char getEchoChar();

④ JTextArea Class:

① append (String str);

② insert (String str, int i); → row no.

③ void setRow (int r);

④ int getRow();

⑤ void setColumn (int c);

⑥ int getColumn();

⑦ void setFont (Font);
↳ font is an obj → has a separate class.

⑤ JRadioButton & JCheckBox Classes:

① String getLabel();

② void setLabel (String);

③ boolean getState();

④ void setState (boolean);
↳ TRUE / FALSE

* Diff b/w Radio Button & CheckBox →

a) - Only 1 RB can be selected in a group.
- multiple CBs can be checked

b) - RBs must be placed in a button group.
- CBs need not be placed in a button group.

c) - RBs in a button group as they cannot be added directly to a panel or frame. Button group is added separately to a panel & panel added to frame.
- CBs can be added to the panel directly separately & then panel added to frame.

Eg: JRadioButton r1 = new JRadioButton ("Option 1", true);
" r2 = " ("Option 2");
" r3 = " ("Option 3");
ButtonGroup bg = new ButtonGroup(); // Event (AWT) ∴ not JButtonGp
bg.add(r1); bg.add(r2); bg.add(r3);
JPanel jp1 = new JPanel();
jp1.add(r1); or jp1.add(bg);
JFrame jf = new JFrame();
jf.add(jp1);

④ JComboBox Class:

① void addItem (Obj);
② void insertItem (Obj, int);
   eg: String st = jcb.getItemAt(2);

Eg: eg: JComboBox jcb = new JComboBox();
jcb.addItem("Driving");
jcb.addItem("Music");
("Dance");
("Riding");

③ Object getItemAt(int);
④ Object getSelectedItem();
⑤ void removeAllItems();
⑥ void removeItemAt(int);
⑦ int getItemCount();

Event Handling:

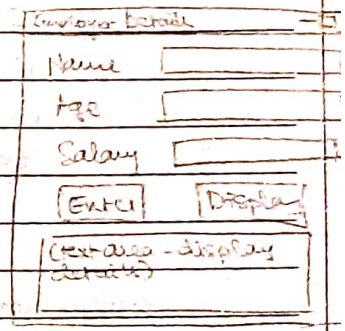
	Event Class	Event Method	Listener Class	Listener Event
①	Action Event (for buttons)	public Object getSource()	ActionListener	public Object actionPerformed (ActionEvent)
②	Item Event (Lists, ComboBox, choiceBox, checkBox)	public Object getItem() public Object getItemSelect() public void getStateChange()	ItemListener	public void itemStateChanged (ItemEvent)

⑤ FocusEvent
⑥ MouseEvent
⑤ AdjustmentEvent (scrollbar)
⑥ KeyEvent (keypress, keyrelease, etc)

Steps in Event Handling:

① Class implements the listener class
② Swing component has to be added with an event handler
③ Define the listener event which performs the required action.

Q] Write a java program using swings to enter details such as name, age & salary. Include button 'Enter' to insert the entered details into a collection. On clicking the 'display' button, the employee details should be displayed in a text area. Also perform validation on age & salary.




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

public class Employee1 extends JFrame
                    implements ActionListener
{
    JTextField cname, cage, csal;
    JButton enter, disp;
    JTextArea txtA;
```

```
EmployeeP()
{
super ("Employee Details");
JLabel name = new JLabel ("Employee name");
JLabel age = new JLabel ("Age");
JLabel sal = new JLabel ("Salary");
cname = new JTextfield (20);
cage = new JTextfield (4);
csal = new JTextfield (20);
enter = new JButton ("Enter");
disp = new JButton ("Display");
txtA = new JTextArea (10,20);
add (name); add (cname);
add (age); add (cage);
add (sal); add (csal);
add (enter); add (disp);
add (txtA);
enter.addActionListener (this);
disp.addActionListener (this);
}

public void actionPerformed (ActionEvent evt)
{
if (evt.getSource() == enter)
{
try
{
int a = Integer.parseInt (cage.getText());
double s = Double.parseFloat (csal.getText());
String n = cname.getText();
if (age <18)
cage.requestFocus();
else
{
Employee emp = new Employee (n,a,s)
emplist.add (emp);
}
}
catch (NumberFormatExcepion e)
{
txtA.setText ("Invalid age or salary");
cage.requestFocus();
}
}
else if (evt.getSource() == disp)
{
for (Employee e : emplist)
txtA.append (e + "\n");
}
}

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

```
public class Employee
 {
   String name;
   int age;
   double sal;
   Employee (String n, int a, double s);
     {
       name = n;  age = a;  sal = s;
     }
   public String toString()
     {
       return name + "\n" + age + "\n" + sal;
     }
 }
```

★ Swing Basic Containers :

★ Root Pane :

① Content Pane
② Glass Pane – multi layers
③ Menu Bar
④ Layered Pane.

★ Layers :

① DEFAULT_LAYER → Standard bottom most layer where most of the contents are placed.
② PALETTE_LAYER → Useful for adding floating toolbars, palettes.
③ MODAL_LAYER → Modal dialog boxes
④ POPUP_LAYER → Displays above modal dialog box
⑤ DRAG_LAYER → When a component is dragged assigning it to drag layer, make sure it is positioned over all the other components in a container.

★ Using Dialogs :-
- Modal
- Non-modal
  ↳ (no need to close)

① JDialog()
② JDialog (Frame parent)
③ JDialog (Frame parent, String title)
④ JDialog (Frame parent, boolean modal)
  ↳ TRUE → modal / FALSE → non-modal
⑤ JDialog (Frame parent, boolean modal, String title)

★ JOptionPane Class :

⇒ 4 types of modal Dialog Boxes :
① ConfirmDialog
② InputDialog
③ MessageDialog
④ OptionDialog.

⇒ JOptionPane methods :-
  ↳ of InputDialog / Message / Option

① a) int showConfirmDialog (Frame parent, String title)
   eg: int response = JOptionPane.showConfirmDialog(null, "This is a test confirm");

* JOptionPane.YES = 1
.NO = 2
.CANCEL = 3.

eg: if (response == JOption

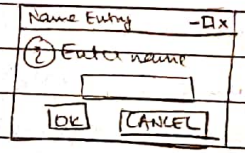
5) int showConfirmDialog (Frame parent, String title, String window_value, int x, int y, → type of message, data to be in title bar);

JOptionPane.YES_NO_CANCEL_OPTION
. YES_CANCEL_OPTION
. NO_CANCEL_OPTION
. NO_OPTION

⇒ 5 types of messages (int):
ⓐ PLAIN_MESSAGE
ⓑ INFORMATION_MESSAGE (i)
ⓒ WARNING_MESSAGE ⚠
ⓓ ERROR_MESSAGE ⊗
ⓔ QUESTION_MESSAGE (?) (default ← InputDialogBox).

② String showInputDialog (Frame parent, String title, String window_value, int y);

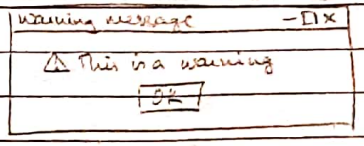
eg: String name = JOptionPane.showInputDialog (null, "Enter name", "Name Entry", JOptionPane.INFORMATION_MESSAGE);



19/02/20.

③ void showMessageDialog (Component, String);

eg: JOptionPane.showMessageDialog (null, "This is a warning", "warning message", JOptionPane.WARNING_MESSAGE);



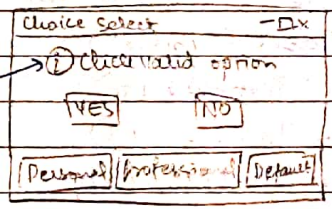
④ int showOptionDialog (Frame parent, String title, String window_value, int x, int y, Icon obj, array of objs);

eg: int n1 = JOptionPane.showOptionDialog (null, "Click Valid Option", "Choice select", ⓐ 0, JOptionPane.INFORMATION_MESSAGE, ⓓ null, ⓑ choices, ⓒ choices[2]);

a) constants of JOptionPane:
- 0 → no button → default
- 1 → YES_NO_OPTION
- 2 → YES_NO_CANCEL_OPTION

b) String choices = {"Personal", "Professional", "Default"}

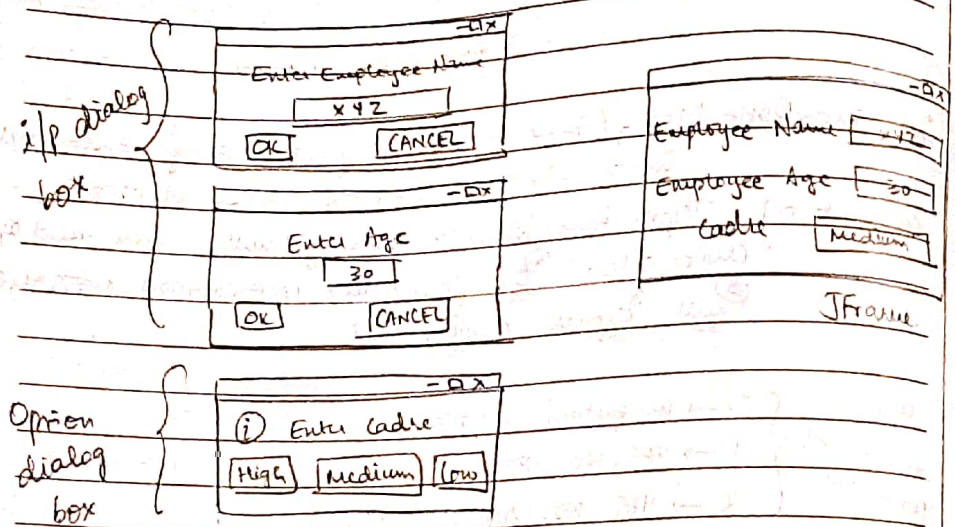
c) choices[2] will be selected on the dialog box as a default.



an icon will display instead of (i).

* size can be set for dialog box

Q] Write a Java application using swings that accepts Employee name, age & a dialog box that identifies cadre of an employee where cadre can be high, medium or low. The information accepted from each of the dialog boxes are displayed in text fields.




```
public class EmpDeets extends JFrame
  {
    JLabel jl1 = new JLabel ("Employee Name");
    JLabel jl2 = new JLabel ("Employee Age");
    JLabel jl3 = new JLabel ("Cadre");
    JTextField jt1 = new JTextField (20);
    JTextField jt2 = new JTextField (3);
    JTextField jt3 = new JTextfield (8);

    EmpDeets ()
     {
      super ("Employee Details");
      String s1 = JOptionPane.showInputDialog(null," Enter
                                    Employee Name");
      jt1.setText(s1);
      String s2 = JOptionPane.showInputDialog(null,"Enter Age");
      jt2.setText(s2);
      String cadre[] = {"High","Medium","Low"};
      int i = JOptionPane.showOptionDialog(null,"Enter
              Cadre","Cadre",0,JOptionPane.INFORMATION_
              MESSAGE,null,cadre,cadre[2]);
      jt3.setText(cadre[i]);
      add(jl1); add(jt1);
      add(jl2); add(jt2);
      add(jl3); add(jt3);
     }

    public static void main (String[] args)
     {
      EmpDeets ed = new EmpDeets();
      ed.setVisible(true);
      ed.setSize(500,500);
      ed.setLayout(new GridLayout(3,3));
     }
  }
```

⟹ CIE I