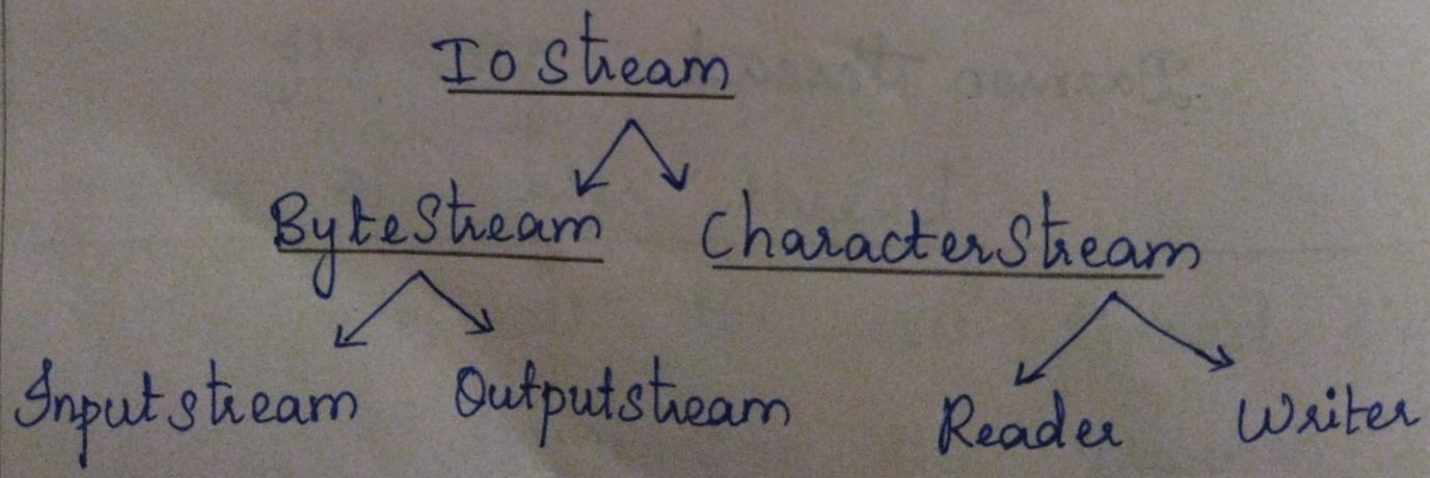
10/2023.

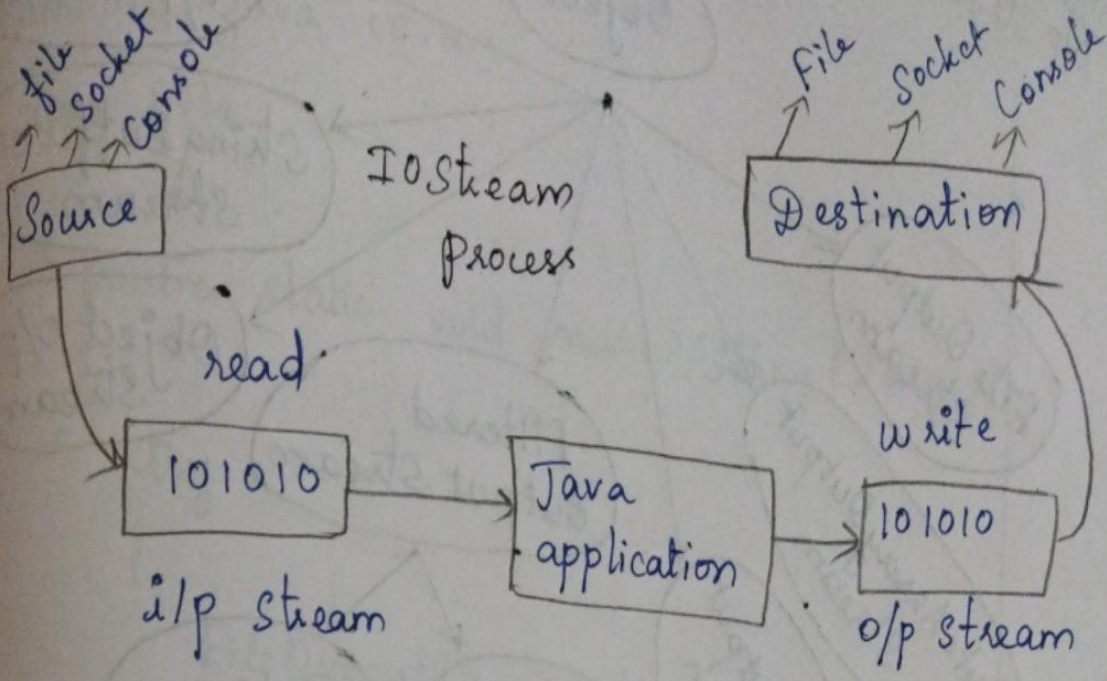
Java I/O.

Package: import java.io.*;

- Stream : Sequence of data.
- A Stream can be of i/p or o/p.



I/O Process.



3- Standard streams :-

⇒ System.out ⇒ O/p stream.

⇒ System.in ⇒ I/p stream.

⇒ System.err ⇒ Error stream.

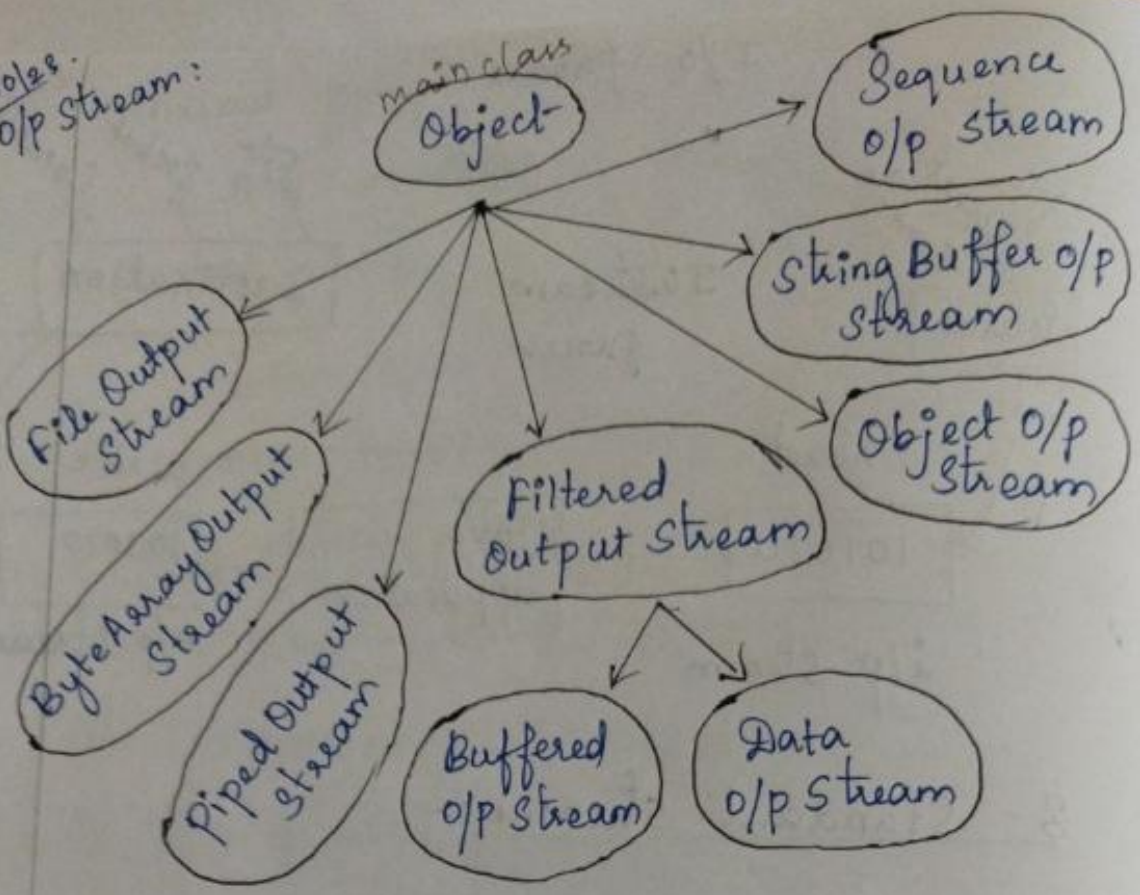
Output Stream.

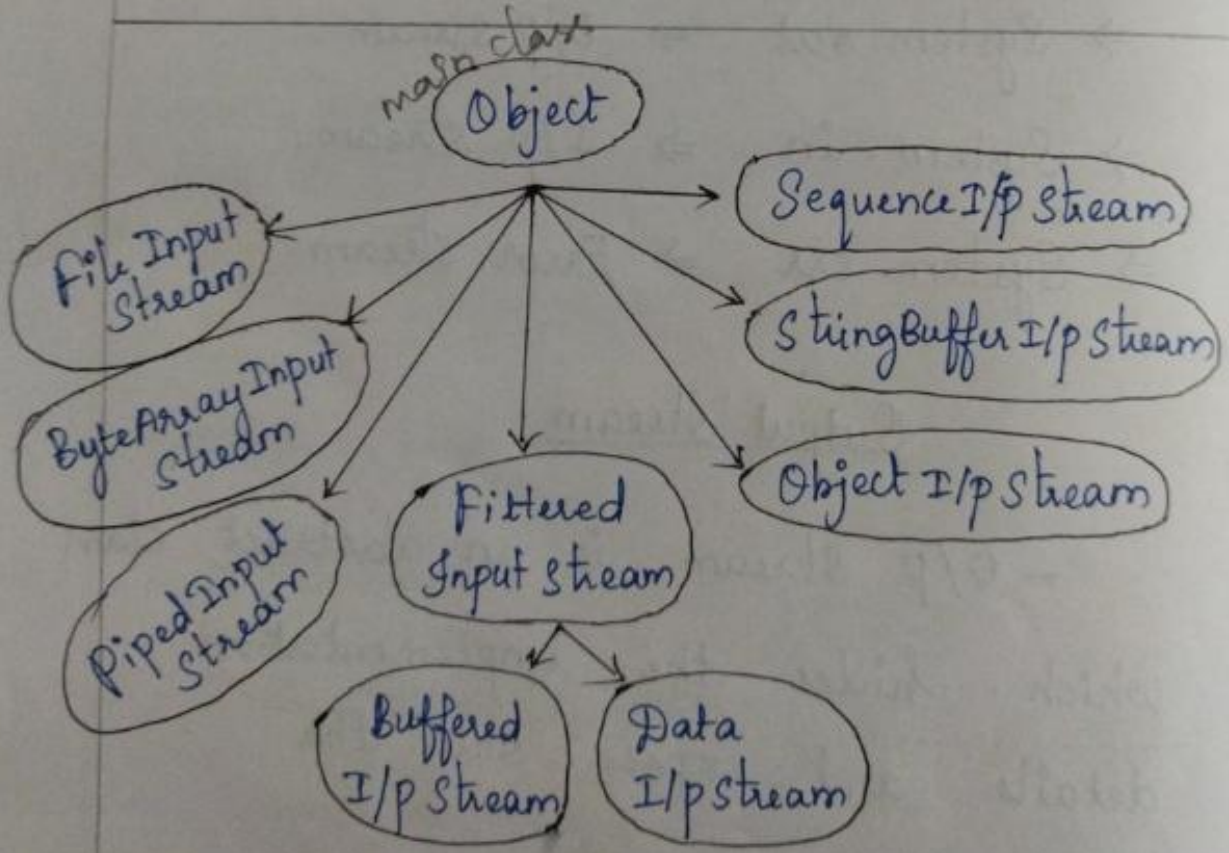
- O/p Stream is an abstract class which hides the implementation details and expose only the functionality.

- Object cannot be instantiated.

21/10/23:

O/p Stream:





Input Stream

File O/P Stream:

```
import java.io.*;
class File
{
  public static void main (Sting[] args)
  {
    try
    {
      FileOutputStream fin = new FileOutput
      char s;          Stream("D:\\test.txt");
      s = fin.write(65);
      System.out.println(s);
      fin.close();
    }catch ( Exception e)
     {
       System.out.println(e);
     }
  }
}
```

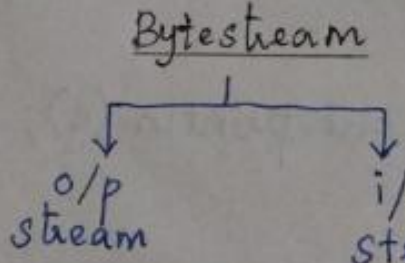
File I/P Stream:

```
import java.io.*;
class File {
    public static void main (Sting[] args)
    {
```

21/10/2023.

```
try
{
  FileInputStream fout = new
        FileInputStream ("D:\\test.txt");
  char s;
  fout.write(s); fout.read(s);
  System.out.println(s);
  fout.close();
} catch (Exception e)
  {
    System.out.println(se);
  }
}
}
```

26/10/2023.



Output Stream :

Methods available in output stream are :

1) write()
2) write(byte b[])
3) write(byte b[], int n, int m)
4) close()
5) flush() - clear the o/p stream.

Input stream :

1) read ()
2) read (byte b[])
3) read (byte b[], int n, int m).
4) close ()
5) reset ()
6) available ()
7) skip (n).

Writing a file by getting input from the user using o/p stream.

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

class filedemo
{
  public static void main (String[] args)
                        throws IoException
  {
     String str;
     Scanner s = new Scanner (System.in);
     str = s.nextLine();
     byte b[] = str.getBytes();
     System.out.println (b.length);
     FileOutputStream fos = new FileOutputStream
                        ("D:\\sample test.txt");
```

26/10/23.

```
        for (int i=0; i< b.length; i++)
        {
            fos.write (b[i]);
        }
        fos.close();
    }
}
```

O/P :-

Sample.txt
Welcome to CSE

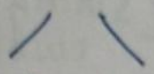
Reading a file "Sample.txt" and printing the sting in the monitor using I/P stream.

```
import java.io.*;
class filedemo
{
  public static void main (String[] args)
                throws IOException
  {
     int c;
     FileInputStream fis = new FileInputStream
                ("D:\\Sample.txt");
     while ((c = fis.read()) != -1) → EOF
     {
        System.out.println ((char) c);
     }
```

typecasting

```
        fis.close();
    }
}
```

Character Stream :-



Reader Writer.

Reader :

Methods available are :

1. File Reader
2. Filter Reader
3. Buffered Reader
4. Char Array Reader
5. Piped Reader
6. String Reader
7. Printer Reader
8. Input Stream Reader.

Writer :

Methods available are :

1. File Writer
2. Filter Writer
3. Buffered Writer
4. Char Array Writer
5. Piped Writer
6. String Writer
7. Printer Writer.
8. Output Stream Writer.

Write a file by getting input from the user using file Writer().

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

class fileDemo
{
  Public static void main(String[] args)
            throws IoException
  {
     string str;
     Scanner s = new Scanner(System.in);
     str = s.nextLine();
     byte b[] = str.getBytes();
     System.out.println(b.length());
     FileWriter fos = new FileWriter("D:
                  \\Sample.txt");
     for(int i=0; i< b.length; i++)
     {
        fos.write(b[i]);
     }
     fos.close();
  }
}
```

UNIT - 4.

EVENT HANDLING.

GUI - Graphical User Interface.

- It is a type of user interface that allows user to interact with electronic devices, such as computers, smartphones & various software apps., through graphical elements like icons, buttons, windows & menus as opposed to text-based interfaces, (status bar, title bar, task bar etc.)

- These elements are used to expose the user Interface.

2m Packages for creating GUI :-

(JFC - Java Foundation Class).

- The swing is a part of JFC & provides a rich set of GUI libraries.

(AWT - Abstract Window Tool) kit.

- AWT is the original GUI library in java.

(JavaFX - Java FunctionExtended)

- JavafX is a modern rich client Platform providing a wide range of

02/11/2023

features of interactive GUI

Java Swings :-

- Java Swing is a part of JFC that is used to create window based application.

- import javax.swing.*;

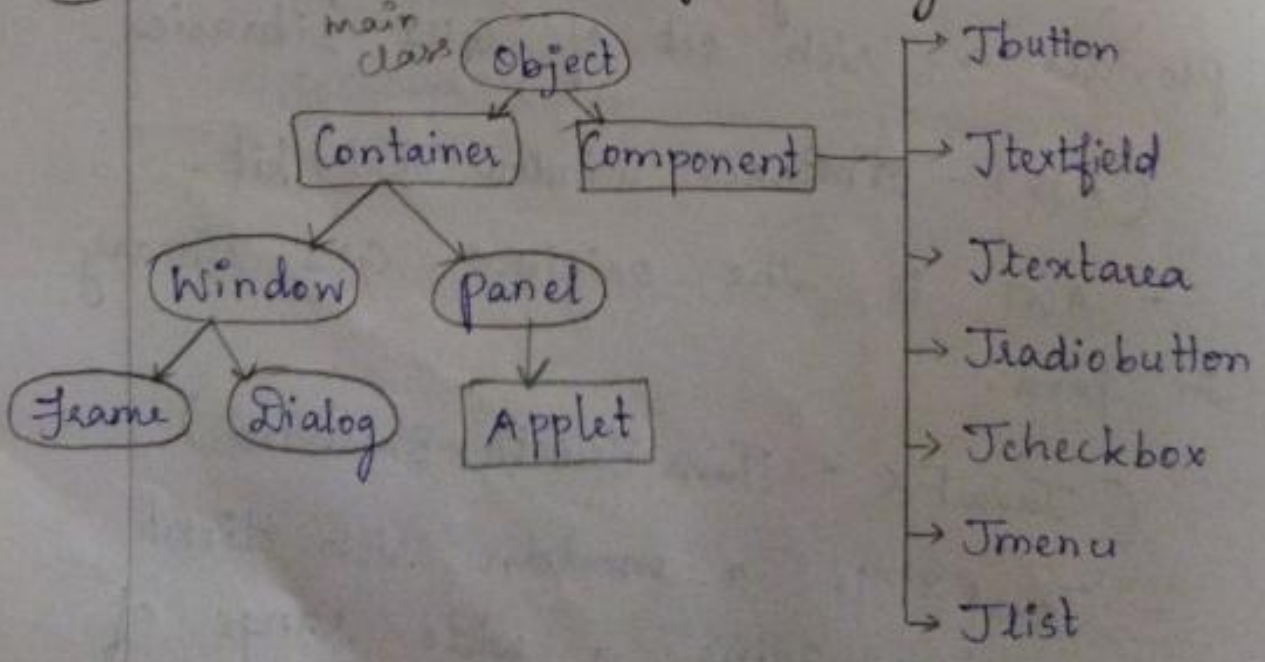
- Javaswing is platform independent.

- The package used for swing is javax.swing.*

- The Javax.swing package provides classes for javaswing api's such as Jbutton, Jtextfield, Jtextarea, Jradiobutton, Jcheckbox, Jmenu, Jlist etc.

2m

Java Swing Hierarchy



Commonly used methods in Component class :-

1. public void add(Component c)
2. public void setsize (int width, int height)
3. public void setVisible (boolean b)
4. public void setLayout (Layout_Manager m).

JFrames :

- The package used for Jframe is import javax.swing.JFrame ;

- JFrame works like a main window where the components can be added to create GUI .

JFrame Constructors :-

1. JFrame ( ) - It constructs a new frame it initially invisible.

2. JFrame (String title) - It creates a new with a specified title for the frame.

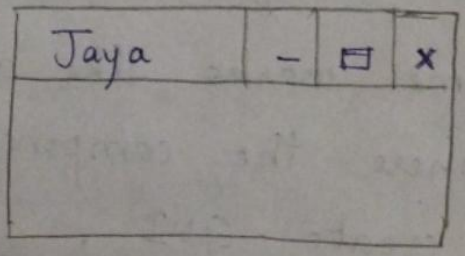
Create the object of frame class with the size (400, 300).

Program:

02/11/2023.

```
import javax.swing.JFrame;
public class Frame1
{
  public static void main(String[] args)
  {
     JFrame fram = new JFrame("Jaya");
     Fram.setsize(400, 300);
     Fram.setvisible(true);
  }
}
```

O/p :-



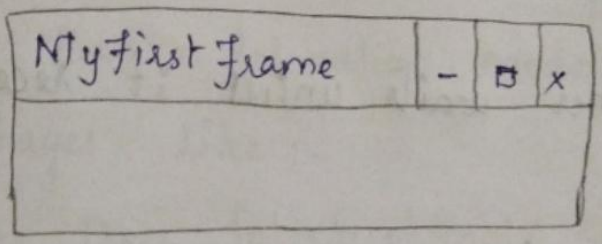
03/11/23. Sample program (2) :-

To create frame

```
import javax.swing.*;
class Frame1 implements JFrame
{
   JFrame t;
   Frame1()
   {
      setsize(100, 200);
      setTitle("My First Frame");
      setVisible(true);
      setLayout(null);
   }
```

```
}
class Demo
{
  public static void main (String[] args)
  {
    new Frame1();
  }
}
```

O/P:-



13m

Event Delegation Model :

The Delegation model is fundamental to java programming because it is used to create event. The modern approach for handling the events is based on the delegation model.

Event :

- Event is defined as change in the state of an object. (ie) it describes the change of event in the source state.
- Eg : Clicking a button, entering a character to keyboard, clicking the mouse, etc.

03/11/23

Source :

- Source is defined as the event which occurs on that object.

Listener :

- It is also known as Event handler.
- Listener is responsible for generating the response.
- Listener waits untill it receives an Event.

Types of Events :

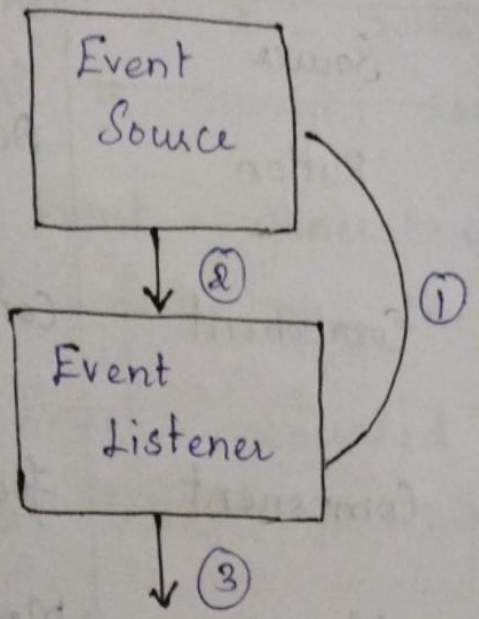
Two types of events :

* Foreground event - the event which requires direct interaction with the users. eg: Clicking the mouse, Pressing the button, Selecting the icon in the menu.

* Background event - in which there is no user interaction. Eg: Hardware failure, Software failure, OS Corruption, etc.

Event Delegation Model :

① - Source triggers the Listener.
② - It fires the event object.
③ - Reacts to the event.



The Event delegation model has advantages like :-

i) The User Interface logic is completely separated from the logic that generates the event.

ii) The User Interface element is able to delegate the processing of an event to separate the piece of code.

The Event delegation model is a 2-step process.

Step 1 : Implement the appropriate interface in the listener.

Step 2 : Implement code to register the listener as a recepient of the event.

05/11/2023.

	Event	Source	Listener
1.	Action Event	Button	Action Listener
2.	Component event	Component	Component listener
3.	Focus event	Component	Focus listener.
4.	Mouse event	Mouse	Mouse listener.
5.	Key event	Keyboard	Key listener.
6.	Text Event	Text Component	Text listener.
7.	Window event	Window	Window listener.
8.	Adjustment Item event	Radio button, Checkbox	Item listener.

04/11/23.

(7) AWT Event Classes:

1. Action Event - generated when the button is pressed.
2. Adjustment Event - generated when the scroll bar is manipulated.
3. Container Event - generated when the component is added / removed in the container.

4. Component Event - generated when the component is moved, resized, hidden, etc.

5. Focus Event - generated when the component focus on the keyboard.

6. Input Event - generated when the event is based on input.

7. Text Event - generated when there is a change in the text field or text area.

8. Mouse Event - generated when the mouse is clicked, dragged, moved, etc.

9. Mouse motion Event - " .

10. Key Event - generated when the input is received from the keyboard.

11. Window Event - generated when the window is opened, closed, minimized, maximized, etc.

Program to create Button:

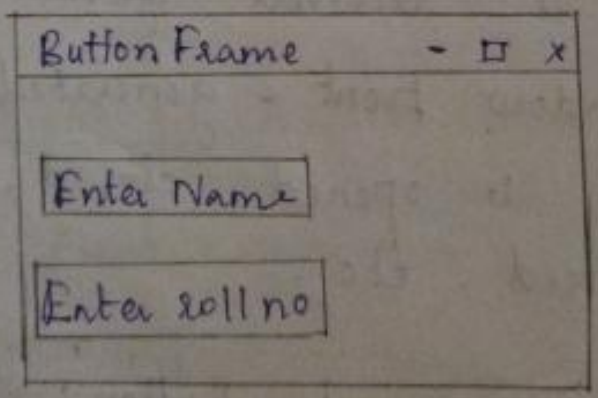
```
import javax.swing.*;
import java.awt.event.*;
class Button1
{
    public static void main(String[] args)
    {
```

09/11/22

```
JFrame JF1 = new JFrame("Button
                         Frame");

JButton B1, B2;
B1 = new JButton ("Enter name");
B2 = new JButton ("Enter rollno");
B1.setBounds (100, 200, 50, 50);
B2.setBounds (150, 250, 50, 50);
JF1.add (B1);
JF1.add (B2);
JF1.setsize (100, 200);
JF1.SetVisible (true);
JF1.Set Layout (null);
  }
}
```

O/P:



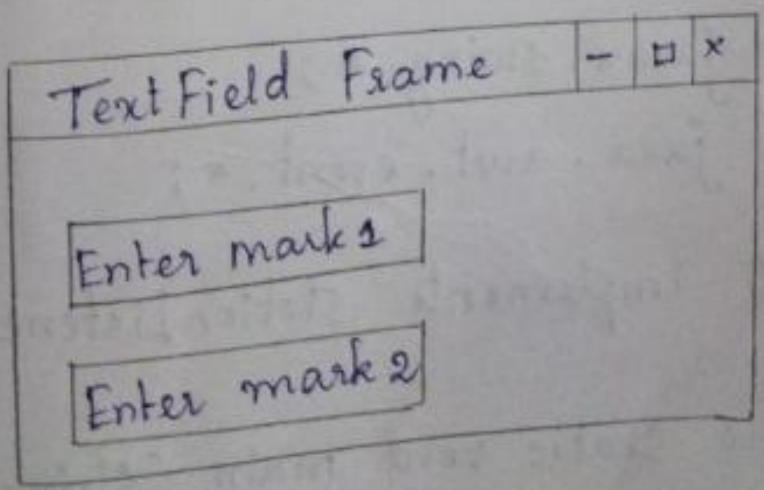
JText Field :

Sample program :

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

```
class TextField2
{
  public static void main (String[] args)
  {
    JFrame JF1 = new JFrame ("TextField
                              Frame");
    JTextField T1, T2;
    T1 = new JTextField ("Enter mark1");
    T2 = new JTextField ("Enter mark2");
    T1.SetBounds (100, 200, 50, 50);
    T2.SetBounds (150, 250, 50, 50);
    JF1.add (T1);
    JF1.add (T2);
    JF1.setSize (100, 200);
    JF1.setVisible (true);
    JF1.setLayout (null);
  }
}
```

O/P:



04/11/23

Adopter Event :

- Adopters are the abstract classes for receiving various events. There are 5 different types of adopter events.

1. Focus Adopter event - for receiving focus events.
2. Mouse Adopter event - for receiving mouse events.
3. Key Adopter event - for receiving key events.
4. Mouse motion Adopter - for receiving mouse motion events.
5. Window Adopter - for receiving window events.

Action Listener Interface:

Program :

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

class A implements ActionListener
{
    public static void main (String[] args)
    {
        JFrame JF1 = new JFrame ("Calculator");
```

```
JTextField T1, T2, T3;
JButton B1, B2;
T1 = new JTextField ("Enter value 1");
T2 = new JTextField ("Enter value 2");
T3 = new JTextField ("Result");
B1 = new JButton ("+");
B2 = new JButton ("-");
T1. setBounds (50, 55, 50, 50);
T2. setBounds (60, 65, 50, 50);
T3. setBounds (70, 75, 50, 50);
B1. setBounds (80, 85, 40, 40);
B2. setBounds (90, 95, 40, 40);
JF1. add (T1);
JF1. add (T2);
JF1. add (T3);
JF1. add (B1);
JF1. add (B2);
```

Database Connectivity.

JDBC - Java Data Base Connectivity.

ODBC - Open " " " .

* JDBC is nothing but Java DataBase Connectivity which is an advancement of ODBC.

* Steps for connectivity between java Program and Database :-

1. Import the packages.
2. Load the drivers using the forName() method.
3. Register the drivers using Driver Manager.
4. Establish a connection using the connection class objects.
5. Create a statement. (SQL).
6. Execute the Query.
7. close the connection.

1. Import the packages :

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

04/11/25.

2. Load the drivers :

- In order to begin with, we need to load the driver before using it in the program.

3. Register with Driver manager :

- The Driver class is loaded using the class.forName() method. The Syntax is

class.forName("oracle:jdbc.driver.Oracle Driver");

4. Establish Connection :

- Driver manager is a java inbuilt class with a static member register. Syntax :

DriverManager.registerDriver(new oracle: jdbc.driver.OracleDriver);

- Establish a connection using the connection class object. Syntax :

Connection con = DriverManager.getConnection (url, user, password);

* Url denotes Uniform resource locator.
* User denotes the username from which

your SQL command can be accessed.
* pwd denotes the password from which the SQL command can be accessed.

5. Create a Statement:

Once a connection is established, we can interact with a Database. Use the following jdbc statement. Syntax :-

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

6. Execute the Query:

Now comes the important part (ie). Executing the Query. The Query here is SQL Query.

The Execute Query method is used to execute the query of retrieving the values from the database.

07/11/2023.

UNIT - V.

Generics and Collections.

- The term Generic means parameterized types.

- The Generics are important because they enable to create classes, interfaces and methods in which the type of data is specified as a parameter.

- A class, method or interface that operates on a parameterized type is called as Generic class or Generic method.

Syntax: class Gen <T>
        T a,b;

Example Program :-

```
class Gen <T> {
  T ob;
  Gen (T o) {
    ob = o;
  }
  T getob(){
```

```
        return ob;
    }
    void ShowType()
    {
        System.out.println("Type of T is"
            + ob.getclass().getName());
    }
}
class GenDemo {
    public static void main(String[] args)
    {
        Gen<Integer> iOb;
        iOb = new Gen<Integer>(88);
        iOb.showType();
        int v = iOb.getOb();
        System.out.println("value "+v);
        System.out.println();
        Gen<String> strOb = new Gen<String>
                                ("Generics Test");
        strOb.showType();
        String str = strOb.getOb();
        System.out.println("value "+str);
    }
}
```

O/P :-

```
Type of T is java.lang.Integer
value 88
Type of T is java.lang.String
value Generics Test
```

07/11/22

A Generic class with 2 type parameters :-

- We can declare more than one type parameter with the generic type.
- To specify 2 or more type parameters simply use a comma separated list.

Syntax:
```
class Gen < T, V > {
    T a;
    V b;
}
```

Example program :

```
class TwoGen < T, V >
{
  T ob1;
  V ob2;
  TwoGen (T o1, V o2)
  {
     ob1 = o1;
     ob2 = o2;
  }
  void showTypes (){
   System.out.println ("Type of T is "+ ob1.
                        getclass().getName());
   System.out.println ("Type of V is " + ob2.
                        getclass().getName());
  }
  T getOb1 (){
     return ob1;
```

```
    }
    V getOb2() {
        return ob2;
    }
}
class SimpGen {
    public static void main(String[] args) {
        TwoGen<Integer, String> tgobj = new
      ↳ TwoGen<Integer, String>(88, "Generics");
        tgobj.showTypes();
        int v = tgObj.getOb1();
        S.O.P.("Value : "+v);
        String str = tgObj.getOb2();
        S.O.P.("Value: " +str);
    }
}
```

O/P :-

Type of T is java.lang.Integer

Type of V is java.lang.String

Value : 88

Value : Generics.

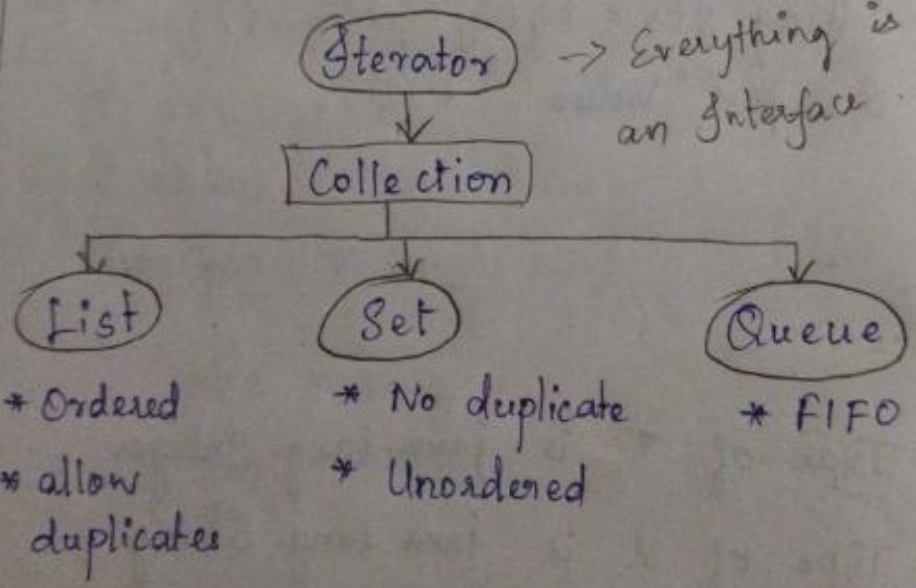
09/11/23.

COLLECTION. (Framework/API).

- Collection is a framework which provides readymade architecture like template to store and manipulate a group of objects as a single unit.
- Operations include Insertion, Deletion & Sorting.

Advantages:

- provides Dynamic Arrays.
- Can also implement trees & linked lists



→ Everything is an Interface

② Package : import java.util.Collection;
(or)
import java.util.*;

Example program:-

① · import java.util.*;

```
public class Kakashi{
    public static void main (String[] args) {
        Collection values = new ArrayList();
        values.add (10);
        values.add (20);
        values.add ("Hello");
        System.out.println (values);
    }
}
```

(2)

```
import java.util.*;
public class Kakashi {
    public static void main (String[] args){
        Collection <Integer> values = new ArrayList
                                        <> ();
        values.add (10);
        values.add (20);
        values.add ("Hello");
        for (Object a : value) {
            S.O.P (a);
        }
    }
}
```

O/P:

11/18.

Accessing a Collection :-

1. Iterator.
2. ListIterator.
3. Foreach loop.
4. for loop.

Program (Access Using Iterator).

```
import java.util.*;
class Demo {
    public static void main (String[] args) {
        ArrayList <String> ar = new ArrayList
                                <>();
        ar.add("ab");
        ar.add("bc");
        ar.add("cd");
        ar.add("de");
        Iterator it = ar.iterator();
       //while (it.hasNext())
        {
            System.out.println(it.next() + " ");
        }//
    }
}
    //Accessing with for loop.
... // for (String str:ar) {
        System.out.println (str + " ");
    } //
```

o/p : ab bc cd de