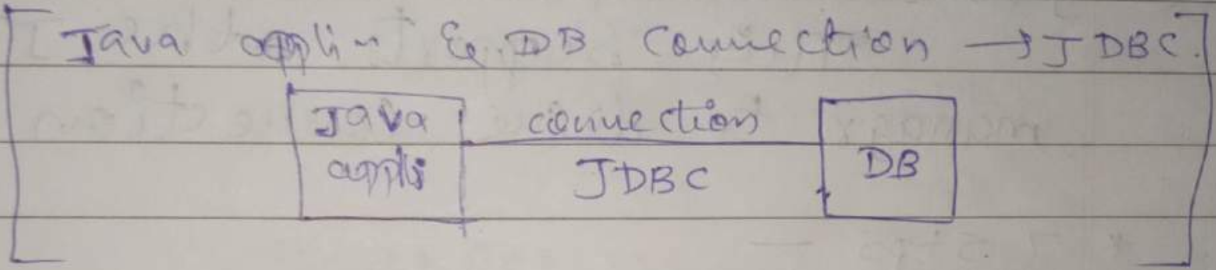
MODULE - 4

Data connectivity & Applets

01: JDBC [Java Database Connectivity]



* Java DB Connectivity commonly referred to as JDBC is a standard Java API for DB independent connectivity b/w a Java prgming lang & a wide range of DBs.

* JDBC consist of a set of Java clses, intrface & exceptions written in Java Prgming lang.

~~Java~~ JDBC connection wrk [illegible] 7 steps.

→ JDBC Architecture =

* JDBC API defines a set of Intrfaces that encapsulate major DB functionality, including running queries processing rslts & determining configuration info.

* It consist of 2 layers -

(1) JDBC API :-
This provides the applin to
JDBC managr connection.

(2) JDBC Drivr API :-
This supports the JDBC
managr to drive connection

* 7 steps —

1) Import package:- import java *;
2) Load & register drivrs :-
class.forName("com.mysql.jdbc.driver");
3) Establish connection :-
Connection.con= DriverManager.getConnection
("URL", "username", "password");
4) create statements ⟶
statements st = con.createstatement();
5) Execute query :-
Result set RS=st executeQuery
('select * from ---')
6) process result :- while (rs.next()
7) Close the connection :- st.close().

* JDBC eg :-

```
import java.sql.*
public. class JDBC Demo {
p.s.v.m (....) throws Exception {
    class.forname("oracle.jdbc.oracle
                      Driver");
    Connection con=DriverManager.get
       Connection("jdbc:oracle.
    thin: @localhost :1547x6", "scott",
                      "abc");
    Result Set rs = st.executeQuery
       ("select \* from ---");
}
```

→ 4 Components of JDBC :

1) driver manager: manage DB drivers.
driver → translator

2) Driver : Handles the commun with servr with hlp of queries
3) Statmnts : → establish connection.
3 types → ① Normal like ins | ② prepared rollno, (to entr the values) | ③ Callable. procedural lang (S). ↓ prime/not, fact

4) Queries → ① execute update() * create, ins, alt, update | ② executeQuery() retrieve [illegible] select * frm ~

CRUD — create, retrieve, update, dlt.

```
[class.forName("com.java.mysql.Driver");
con = DriverManager.getConnection("jdbc:mysql:
      @localhost", 5437, root@123");
st = con.createstatement();
]
```

→ ~~progms~~ using CRUD opr =

1) create :

~~createf~~ executeupdate

```
import java.sql.*;
public class CRUD {
    p.s.v.m (...) {
        class.forName("com.sql.jdbc.
                      driver");
        Connection con = ----
        Statement st = ----
        st.executeupdate ("create table
        Student (rollno int, name
        varchar(20), branch varchar(20)");
    st.close();
    con.close();
    }
}
```

2) insert :

```
import java.sql.*;
p.c CRUD {
    p.s.v.m (...) {
        class.forname("....");
        Connection con = ----
        Statement, st = ----
        St.execute~~[illegible]~~update(" insert into
        Student values (101, "Ansar", "BSC")");
    st.close();
    con.close();
}}
```

3) update :

```
import java.sql.*;
p.c.CRUD {
    p.s.v.m (....) {
        class.forname ("....");
        connection con = ----
        statement st = ----
        st.executeupdate ("update
        Student set branch =
        "MSC" where rollno = 101");
    st.close();
    con.close();
}}
```

4) delete :

```
import java.sql.*;
p.c.CRUD {
    p.s.v.m (...) {
        class.forname ("....");
        connection con = - - - -
        statement st = - - - -
        st.executeUpdate ("delete
         from Student where rollno=101");

    st.close() ;
    con.close();
  }
}
```

5) <u>Retrive</u> :

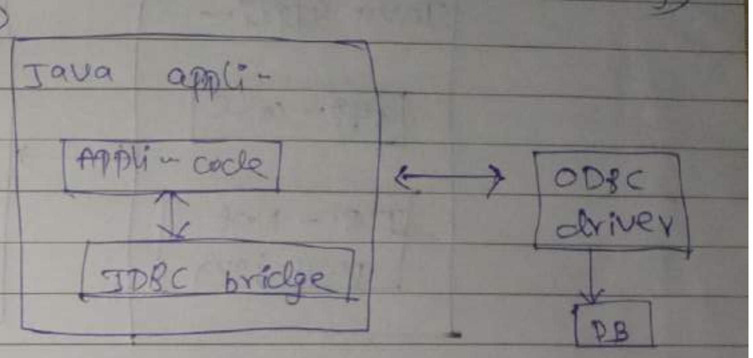
```
      - - -
      - -
    Resultset ~~rs~~ rs=st.executequery('select
              * from Student');
     st.close();
     con.close();
   }
  }
```

⇒ **JDBC drivers** : (used for connection establishment)

1) JDBC - ODBC bridge drivr.
2) JDBC - Nature API
3) JDBC - N/w protocol drivr

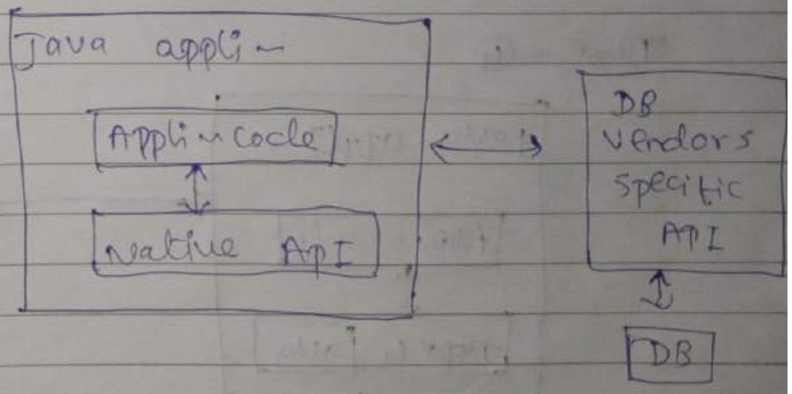
4) JDBC - universal drivers (currently we are using)

Type - ①



disadv → cannot run of all o.s. (only in windows).

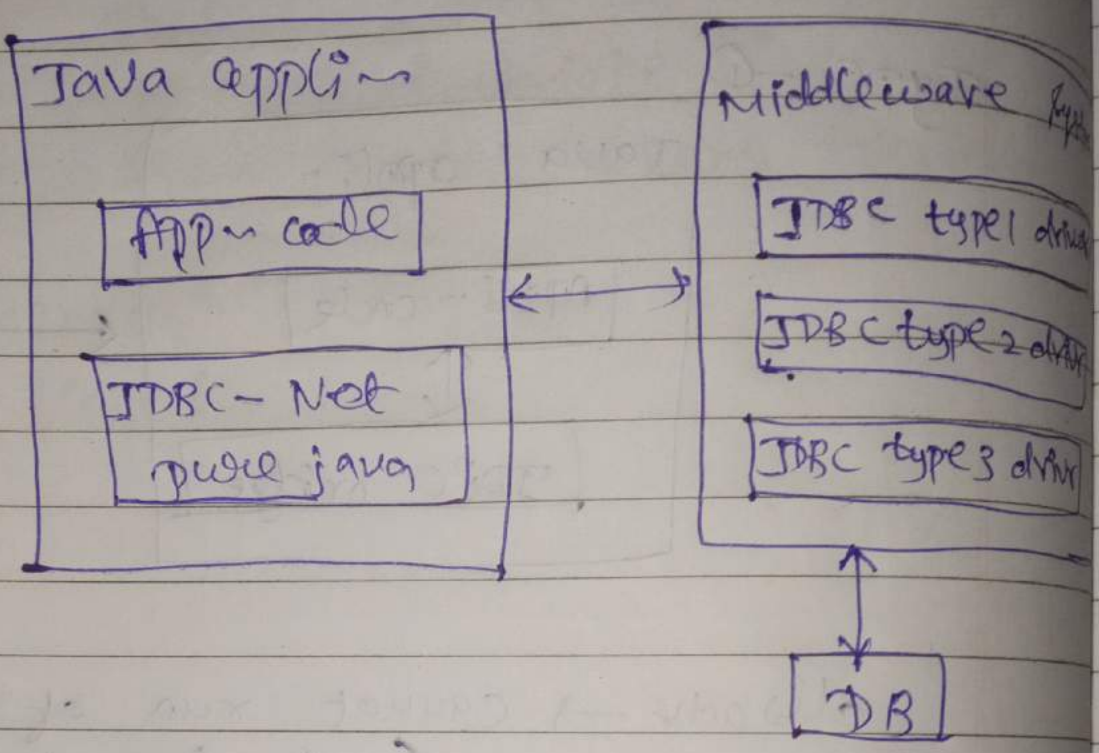
Type - ②



adv → Run on all o.s

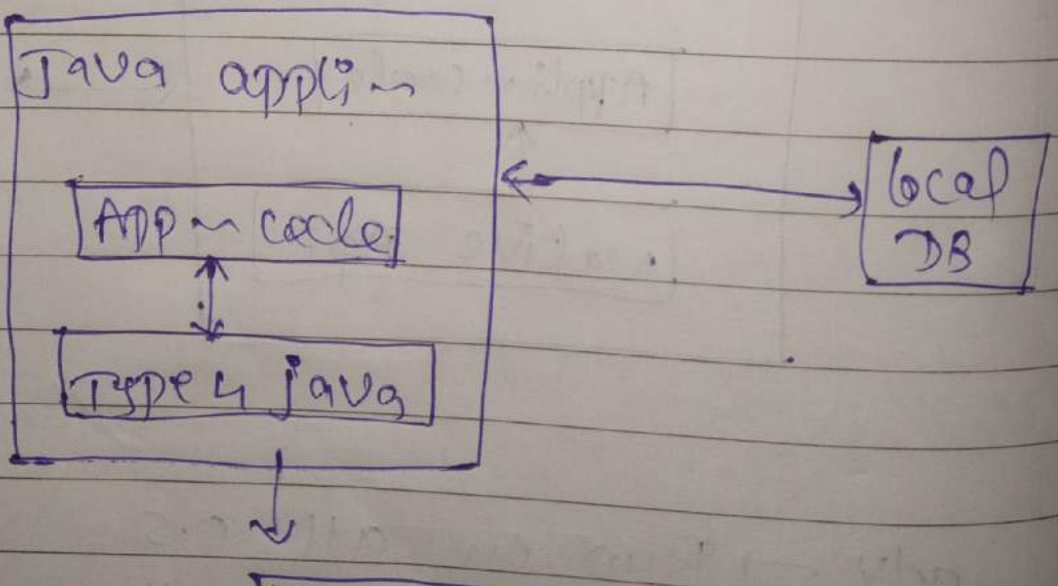
disadv → Should install in all system

Type — ③



adv → Avoid installation.

Type — ④



(direct ആയി connect ചെയ്യാം)

Result set :

```
import java.sql.*;
   public class classname {
      p.v.M (...) {
         class.forName ("com.mysql.
                  jdbc.Driver");
         connection con = Drivermanager.
            getConnection ("jdbc.mysql
               @localhost 3306/college.
                  host @ 123");
         Statement st = con.create
                  statement ();
         Resultset rs = st.execute
            ("select * from student");
         while (rs.next()) {
            S.o.pln ("r.no=" +rs.getInt(1)
               "Name=" +rs.getString(2));
         }
         st.close ();
         con.close ();
      }
   }
```

(nxt row-ലേക്ക് point ചെയ്യും)

⇒ **Applets :** (webpg-ൽ ആരെങ്കിലും content display ചെയ്യാൻ വേണ്ടി ഉപയോഗിക്കുന്ന pgm → Applets).

- Java prgm embedded in a webpg.
- used to provide features to use applin
- operate dynamic contnt (continously change ആകുന്നവ) at client side.
  ↳ eg: weathr app.
- Java applet is a tiny applin delivred to be transmitted ovr the internet to users in the form of Java byte code & executed by a java enable web browsr / in the contnt of an applet viewer.
- Applets are typically embedded inside a webpg & runs in the context of a web browser.
- It used to provide intractive features to web applin that cannot be provided by Html alone.

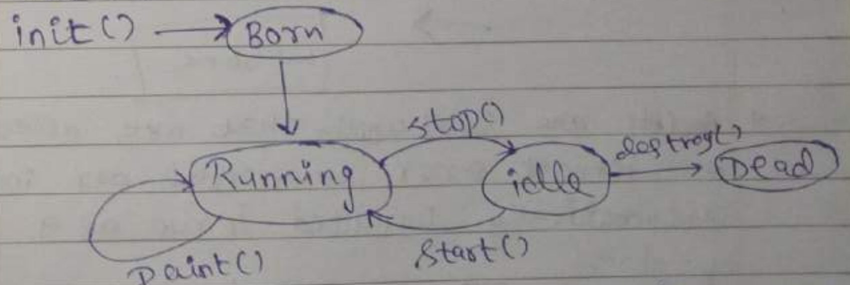
→ **Create an applets :**

1) cls extends java.applet Applet class
2) Doesn't contain main ()
3) life cycle method :
   init (), start(), paint(),
   Stop (), destroy ().
4) Does not contain sys.out.println
5) To execute an applet we use applet viewer.

✱

lifecycle		Applet
1) initialization	→	public void init()
2) Start	→	" " start()
3) paint	→	" " paint (Graphics g)
4) Stop	→	" " Stop ()
5) Destroy	→	" " destroy ()

~~init ()~~



Stop () → minimize ചെയ്യുമ്പോൾ
Start () → maximize ചെയ്യുമ്പോൾ

fig : lifecycle

Prgm :

```
import java.awt.applet.*;
import java.awt.*;
public class AppletDemo extends Applet {
        Public void init()
        {
            ~~set~~ Background(Color.black);
            set foreground(color.yellow);
        }
        public void (graphics g){
              g.draw.string("welcome",
                       100,100);
                       size
        }
  }.
```

→ welome

* Applets are small appli- that are accessed on intrnet servr, transported ovr intrnt automatically installed & run as a part of doc.

(paint() → graphics / contnt displaying)

* Requesting repaint :

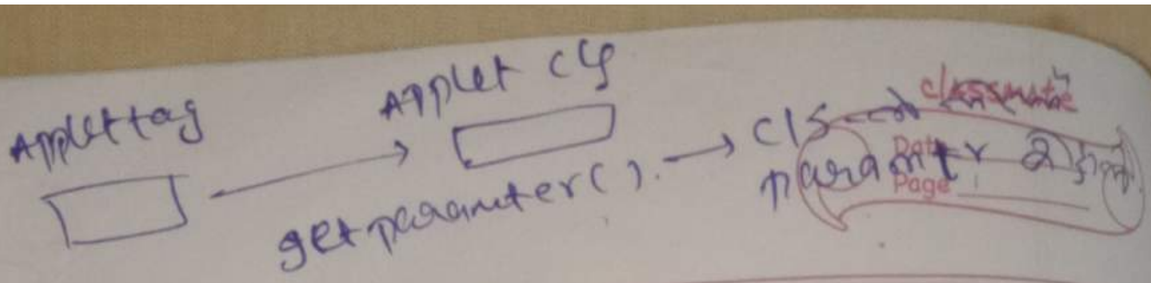
- An (A) write to its window only when its update/paint() method → by the AWT
- whnevr yr (A) needs to update the info displayed in the window it simply calls repaint method
- repaint method is defined by the AWT it causes the AWT runtime system to execute whn called (A) update method which it its default implementation → paint method.

* passing paramtrs to applet :

```
Import java.awt.*;
import java.applet.*;
/*
<applet code=" paramDemo " width=300
                          height=300>
<param name = LetterName value= welcome>
<  "     "     size value =14>
</applet>
*/
  public cls paramDemo extend Applet
  {
```

applet tag

([illegible])



```
String Lname;
  int LSize;
}
public void start () {
    Lname= getparameter ("lettername");
}
public void paint (graphics g) {
      g. drawstring ("lettername:" +
                  Lname, 0, 10);
                        (x, y)
  }
```