How to make Daemon thread

22.1 → daemon thread test

k2.setDaemon(True);

Advance Java

16:46

JDK

JRE → installed JRE.

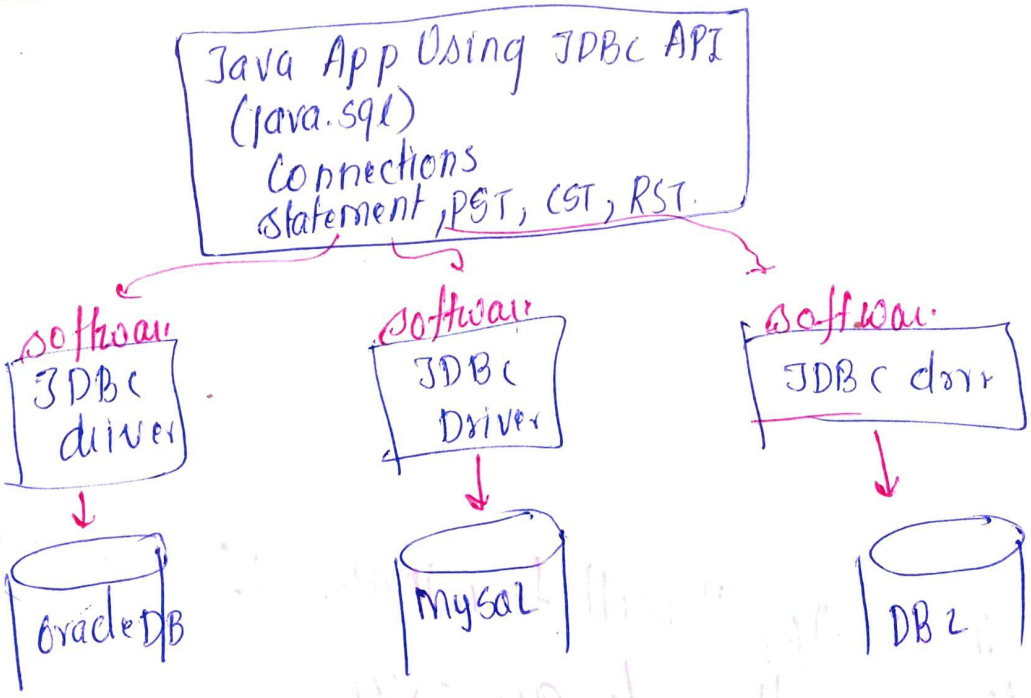
JDBC → Java database Connectivity.

What is it ? → API → Java.sql → to allow prog...
to connect to DB, CRUD, close connection.

Why JDBC?

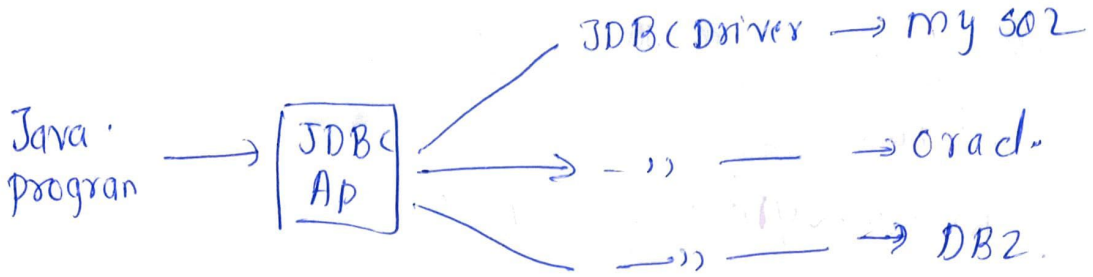
① platform independances → Java
→ allows platform independant apps (database) + DB vendor independant.
→ JDBC allows only Partial Independance
(*) → some part may vary from DBT to DBT.

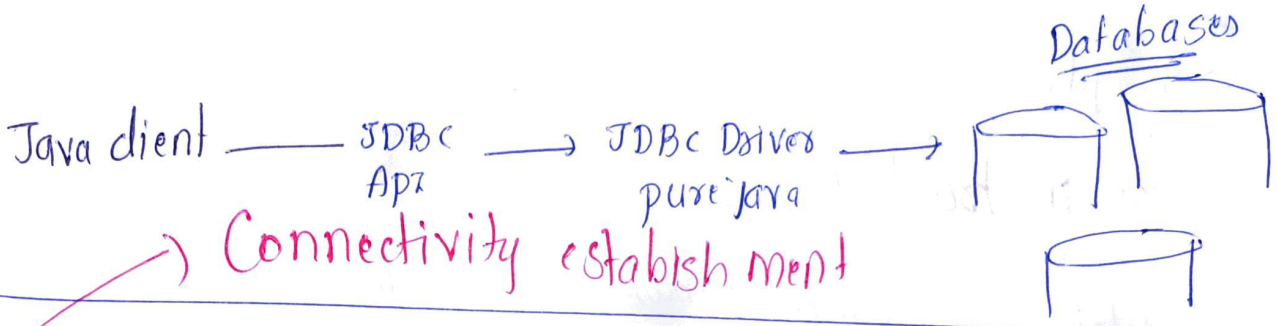
JDBC Driver → guarntees platform independance



How are JDBC are supplied?
in form of JARS.

JDBC driver → platform dependent
JDBC App → independent
When change Database → JDBC driver change.





Create java project → Next → libraries → classpath
→ add external jar 8.0.20 →

⊗ com.mysql.cj.jdbc → Driver.class

Load Type IV JDBC driver in JVM's methodarea Testing.

Api → java.lang.Class<T>
→ forName(FQCName) (Qualified)
Method.

SRC → tester → testconnection → load to MArea

```
main(){
//load JDBC driver in JVM memory (methodarea).
try {
Class.forName("com.mysql.cj.jdbc.Driver");
} catch ClassNotFoundExcepn.
```

optional in standalone but needed on Web server

22. Javadocs → class DriverManager

The basic service provi

→ getConnection (url, user, password);

To know user & password :- → syntax of JDBC URL

status

JDBC URL → Syntax → jdbc:subprotocol:subname

e.g jdbc:mysql://

17:37

DB related instruction

③ create database → create dabas sunbeam21;

use sunbeam21;

→

String Url → "jdbc.mysql://localhost:3306/sunbeam21?

→ use SSL=false&allowPublicKeyRetrieval=true";

main() {

string url =

establis

→ Connection establishment

getconnection (string

java.sql.

try (Connection Cn= DriverManager Connection interface

.getConnection( ); auto closable

{

syso ("connected to DB " +Cn);

}

ⓐ seperating Connections in JDBC utils

Day 1.1. → src → utils → DBUtils.java

```
public class DBUtils {
import java.sql.*  private static Connection cn;
```

(b/c we require only one connection)

```
//add static method to retur "SINGLETON"
   = single in entire java App) to call.

public static Connection fetchConnectn()
{
    if ( cn==null) {
        cn = DriverManager.getConnection (url, "root", "manag..
    {
    return cn;
}
```

Tester → testDButils →

```
main() {
try (Connection cn=fetchConnection()) {
    sysO("connected to dB"+cn);
```

ⓐ

My_emp → empid, name, addr, salary, deptid, join_date.

Query holder — java.sql.connection.

public Statement

Java docs → CreatStatement() throws SQL Exception

Returns → empty wrapper.

ⓐ Display on Java

222 Tester → testStatement

@ 1.1 main() {

try(Connection cn = fetchConnctn();) ⊗

Javadocs → interface Statement → AutoClosable

Statement

⊗ create empty statement → try

Statement st = cn.createStatement() { ⊗

Query Execution

executeQuery()
DQL(select)

executeUpda
DDL → create tab
DML → in/u/Del

execute
Stored
procedur

Javadoc → ExecuteQuery() →

ResultSet executeQuery(String sql)

↳ select x from emp;)

Collection of result

Interface ResultSet → AutoClosable →

→ table of data representing a database result set

Cursor of ExecuteQuery :-

cursor → positioned befor first row.

next → more cursor to next row

↳ can use while loop.

No Semicoln

⊗ add resultset autoclosable)

ResultSet rst = st.execatQuery("select * from myemp");

// rst is pointing to before first row
while(rst.next()){
//read & display;

Rows & colums are counted from 1

Result set processing

How to read Columns ?

public Type getType(int colPos), ,, → (String colName);

from Result set not DB

Generic SQL Type → independant of DB

Char/varchar/varchar → String
number → int/long.
number(m,n) → float/double
date → java.sql.Date
time → Time
timestamp → Timestamp.
clob → java.sql.clob → Character large ob.
blob → java.sql.Blob, Binary large Obj

Conversion syntax

while(empid ← rst.getInt(1)
Read name → rst.getString(2)
add → rst.getString(3)
deptid → rst.getString(4)
joindat → rst.getDate(5)

while(rst.next())
System.out.printf("Empid %d Name %s Address %s
%1.2f. Dept %s joined on %s (

224/ (, rst.getInt(1), rst.(getString(2), rst. . . . —
(3), (4) rst.getDate()).
. }

(*) → tuesday → Web server basic → (2) ☆☆
What is HTTP → (1)
Tcpip primer

15/11/2021 AS: 17 important

(*) interface remain same → } only implementat^n changes
(*) specs remain same → } (oracle/mysql)

JDBC Driver Type 1

Disadvantage: platform independance

Different types of JDBC Driver → 15/11/2021 → 8:31

Object

(1) display all emp emp details
sql = select * from my-emp

Steps: 0: add JDBC driver in run time class path (i.e add ext.jar)
1: DButil
1.1: load JDBC driver class in method area
class.forName ("com.mysql.cj.jdbc.Driver")
1.2: Get DB connect
Connection cn = DriverManager.getConnection("jdbc:mysql://localhost:3306/sunbeam21?"useSSL...", name, pw)

if we call connection ten times it will create 10 connection ☆☆☆☆

2. Create statement:

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

(3) Query : execution

```
select * from
ResultSet rst = st.executeQuery(sql);
```

4) RST => in memory view of selected rows and columns.

Cursor :—→ before first row

next() —→ return boolean —→ True → if next row available
—→ false → if not row available.

Row no start with "1"

(5) Read from cursor

```
while(rst.next()){
rst.getType ⟹ JDBC data type (generic SQL type):
DB independant.
```

(6) Autoclosable → rst —→ st —→ cn —→

Objective → Display details (id, name, salary, joindate) of all emps from a specific dept joined b/w start date & end date

sql = select empid, name, salary, join.date from my-emp where deptid =
and join.date b/w ( )

Javadocs → PreparedStatement

(X) It represents pre parsed or prepared

(X) cn.prepareStatement() —→ populated → holding Query.

(X) prepareStatement() —→ Support "IN" params

22L Image :-

Use Case → limitation of create statement

→ public statement createStatement()
used when we want parameterless Queries
→ non-repeatative
→ public PreparedStatement → Adv of prepared statement
used when we want parameterized Queries
→ prepared cnc → parsing & compilation → 1'
eg dynamic Queries
→ Repeative statemes used for

SQL Injection

(x) is a 10 Top 10 security attack
(x) statement → can be injected easily prone to attacks
(x) prepared → will not be prone to attacks

Prepared statement : compiled and stored → Counter of Rows

? → placeholder (In parameter counted from 1 onwards from left to right

→ I/p from user and access database.

→ Code 2.1 copy DB utils from 1.1 / Tester for Prepared Statement

```
main(){
    String sql = "select * param → Prepared statement
    try( Scanner ()
        Connection cn = fetchConnection();
        PreparedStatement pst = cn.preparedStatement(sql); → X
        // RST → executeQuery X
```

Befor executing query parameters post → in param → ? from left to right.

set in params.

public void setType(int paramPosition, Type val)

Type → JDBC

```
*
{
  syso("Enter dept_id, start date, end date")
  pst.setString(1, sc.next());
  pst.setDate(2, Date.valueOf(sc.next()));
  pst.setDate(2, Date.valueOf(sc.next()));
  pst.setDate(3, Date.valueOf(sc.next()));
```

set in parameters

setDate → Javadocs

String → java.sql Date

valueOf(String s)

yyyy-[m]m-[d]d

Query execution

execution method

Public ResultSet() → prepared statement

public ResultSet(String sql); → Statement

```
try(ResultSet rst = pst.executeQuery())
{ while(rst.next()){
    syso("Emp id %d Name %s salary %.", rst.getInt(1),
    rst.getString(2), rst.getDate(3), rst.getDouble());
```

(count column as per result set)

String sql = "select empid, name, sal,) Date (?) and join dat b/w (?) and (?)

UI and DBT are expected seperate

Layered architecture

228 Relath Database → Tables, rows, columns, primary key
No sql database

DESC my_emp → will be represented as class in java

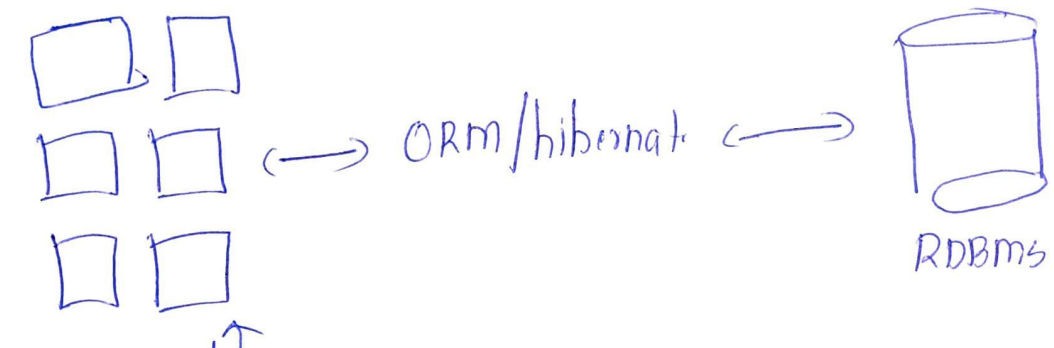
Table columns → non static / non-transient data members

Table Rows → emp class instance

Transform Table → Java

Object to relational Matching (ORM)

ORM overview



Object world	Relath DB
pojo classes	DB Tables
pojo properties	Table column
pojos	Table rows
unique id property	primary key

pojo → plane old java object
Vanila flavoured (no flavour)

⑧ Ordinary object (plane object)

⑧ no business logic; just plane.

Objective: Display (id, n, s, joindate) of all specific emps (above question)

Tester

DAO layer (data access object layer)
DAO i/f → to declare crud requirement
JDBC based class imp
① public Package class.
② state
③ constr

how DAO ret result to Tester?
ResultSet ✗
Collection of pojo

"POJO" → pojo/Entity/model/Domain classes → object view of data
pojo → DB table, Pojo property → Table Colo
Pojo row

DB Layer

DB/Tier/Layer (ERS layer)
Tables/rows/column

→ layered architecture

Code → copy utils → 2.0.2
SRC → pojos → Employee → Creating pojo

class employee {
// private int empId;
private String name;
private String address;
private Double salary;
private string deptId;
private Date joinDate;

Table ~~class~~ → class
Colum → attribute

public Employee(int empId name, salary join Date)
{

}

Getters & setters

}

230 Data access Layer

DAO layer

Naming IEmployeeDao

SRC → interface → DAO

{ add method declaration for getting selected employee.

① Layer → Database (my_emp)

② POJO → Employee

③ Heart of JDBC DAO →

3.1 → Interface

{

1) Getting selected method name.

for methods of DAO

** List<emp> getSelectedEmps(String dept, String strDate, String endDate) throws. {

DTO → data transfer object

Impl class → dao → employeeDaoImpl

class {

@ Override

getSelectedEmployee() {

State

private Connection cn;

private PreparedStatement pstl;

Add Constructor (one time implementation)

// Constructor() {

① get db connection

cn = fetchConnection();

sql = "select empid, name salary ....."

Prepared pst1 = cn.preparedstatm(sql)

Syso (DAO created)

{

(X) clean up method → closing all conection (Database

```
public void cleanupp(){
  if(pst1!=null)
     pst.close();
  if(cn!=null)
     cn.close();
  syso(employ dao clean up);
{
```

→ closing resources cleanup().

Boiler plate = repeatative cod

@ get selected em

① set placeholders → ?

```
pst1.setString(1,dept);
pst.setDate(2,Date.valueOf(strtDate));
pst.setDate(3,Date.valueOf(endDate));
```

→ parameter positio

1 2 3

? ? ?

Execute Query

Result set → method local variabl.

```
try(ResultSet rst=pst1.executeQuery()){
  while(rst.next()){
    Employee e1
    emps.add((new Emp(rst.getInt(1),rst.getString(2),
              rst.getDouble(3),rst.getDate(4)))
  {
  return emps;
}
```

adding to collection

Result Set View

Topmost layer Tester (User interface) TestLayeredApp

```
main(){
  try(scanner()){
  Create Dao instance
  EmployeeDaoImpl dao=new EmployeeDaoImpl();
```

232

Client request servicing phase

Syso(Enter dept start date, end date);

dao.getselectedEmploy(sc.next(), sc.next(), sc.next()).forEach(System.out::println);

Cleanup();

TestLayer AppCrud :→ copied earlier tester

{ boolean exit = false;

while(!exit){

Syso(1. ~~~ 2. ~~~ 3. ~~~ 4 ~~~);

try{

switch(sc.nextInt()){

case 1 :- Get selected emp;

Insertion emp in Database :

3 → no change in db table

3.2 → pojo → add required arg constructor (all args expt empid)

3.3 → DAO →

DTO → Data transfer object (pojo in tester) ← instance / object.

String addEmpDetails(Employee e) throws ;{

↖ add to interface.

Method implementation in DAO

⊗ private PreparedStatement pst1, pst2;

pst2 = Cn.

sql = "insert into my_emp values(default, ?, ? ? ? ?)" ← empid

Dao over

close pst2;

@ public String addEmpDetails(Employee newEmp) throws 233
SQLException {

pst2.setString(1, newEmp.getName());
.setString(2, .getAddress());
.setDouble(3, .getSalary());
(.getDeptId());
(.joinDate());

Query execution → executeUpdate();

DDL

int executeUpdate(String sql);

if no row update = 0 ← int executeUpdate();

if

if(updateCount == 1)
(return "Emp added");

Add case 2 :- Syso(Enter employee detail);
Syso(dao.addEmpDetails(new Employee(sc.next(), sc.next(), ...)));

Update empdetails

4.1) table → No chang
pojo → no chang → plane object of Employ

DAO if String → String updateEmpDetails(int empId, String newDept, Double salinc); throws SQLException.

→ Implemental method → private pst3;

pst3 = cn.prepareStatement("update my_emp set deptId=?
salary=salary+? where
empId=?");

close pst3

234 @Override

```
public EmployeeDaoImp
    pst3.setString(1, newdept);
    pst3.setDouble(2, salinc);
    pst3.setInt(3, empid);
```

Execute Update (DML)

```
if (updateCount == 1)
    return "Emp details updated";
return "failed to update";
```

Case 3 → to update

```
sysout(Enter emp_id, newdept, sal inc)
sysout(dao.updateEmpDetails(sc.nextInt(), sc.next(), ...));
```

---

⑤ Remove Employee

DAO Interface → string deleteEmp(Employee deleteEmp){

```
pst4 = cn.prepareStatement("delete from my_emp where
                            empid = ?");
```

@ → ? position

```
pst4.setInt(1, empid)
```

execute Update:

```
int updateCount = pst4.executeUpdate();
if (uc == 1)
    return "deleted";
return failed
```

---

case 4 → Delete added

Case study:

Transfer fund → stored procedure

```
Delimiter $$
create procedure transfer_funds (
    in sid int,
    in did int,
    in amt double,
    out sbal double,
    out dbal doubl
)
begin
```

Inpu

Stored procedure ↑ the speed of operation

Diff in proced & function

return valu

procedure vs function

- Stored procedure does not return anything

Callable Statement

Statement → pst → CST → CST hierarchy.

- used to execute procedure & function
- To pass in/out, inout params.

2.0.2 → pojos → Account

```
class Account{
    private int accNo;
    private String customerName;
    private String acType;
    private double balance;

    construct();
    toString();
}
```

Add new interface..

```
Dao → IAccountDao{
    String transferFunds(int sid, int destinId, double account)

Dao → Implementation class .. IAccountDao(
    private Connection cn;
    private CallableStatement cst1;
```

236 default constructor

```
Account ( ) {
    cn = fetchConnection ( );
```

Interface Connection
- cn.~~statement~~ createStatement
- cn.prepareStatement
- cn.prepareCall

→ creating statement methods of Statement

Callable → Statement → Returns o/p if fuction

"?" for procedure

{ ? = call <procedure-name>

call <procedure-name> → used for procedure

String ProcInvocation = "{ call <transfer_funds(?,?,?,?)}"

cst1 = cn.prepareCall(procInvoc);

Sysout(acct dao created);

? → Input, Result store

Assignment

solve → Tester ---> DAO (i/f, implclass) ---> DButils ---> POJO / Entity / Model / Controller / View

1. POJO : user, props, def constr
2. DButils
3. DAO i/f
   User AuthenticateUser (String email, String pwd)

DAO impl class

State Cn, pst1;

Constructor :- get the connection from DB utils,
pst1 = select 2 in params.

clean up :- close db resource

CRUD :→ set in params, pst1.executeQuery - - -→ RST.
If (rst.next()) - - -→ Valid login → return new user.
return null.

---

② Change password

I/p → email, password, new password
O/p A message string.

Topics page :-

③ Get All available topic names.
step1 :- Table
2. pojo : Topic
3 DAO (new) I/f, ITopicDao :-
List<String> / List<Topics> getAllTopics()
4) Impl → pst1 : select query

④ Get All tutorial titles under selected topic, ordered in desc manner
of Vlss
I/p → topic name.
e.g = Join query.
① DB Tables → tutorials
② POJO :- tutorial
③ properties : 6
④ DAO : I/f → List<String> getTutorialsByTopicName(String topicNa
⑤ Imp class → select on join on topics & tutorials

234

5 Get specific tutorial details :-

Content varchar[5000].

i/p tutorial id

~~Note: a~~

How to get Seives

show.

Rituals

Registering is Must of o/p

How to add Tomcat Server → Readme

16/11/2021 → Stored procedure Continuation

→ public void cleanUp(){

if (rst != Null)

rst.close();

if (cn != null)

cn.close();

check for out parameters → Register Out Parameter

public void registerOutParameter(int paramPos, int JDBCType);

Informing JVM about JDBC datatype of

DB —o/p param→ JDBC → Java(JVM)

Inform JVM of JDBC Type

Java docs → collableStatement

JDBC Types → java.sql.types.

Double → '8'

Registering param → 4 → (4, Types.Double);
(5, Types.Double);

register out param

cstl.registerOutParameter(4, Types.Double);
cstl.registerOutParameter(5, Types.Double);

IN — set value.
Out → Register its JDBC type with JVM.
} → IN/out/InOut parameters

IN/Out → Both

When to use IN/out/InOut:

IN → @CRUD methods (Service)

Execute Stored procedure.

cstl.execute();

Read Result from Out Params :-

⑥ Retrive the result stored in out param

API of CST →

public Type getType(int ...) throws SQLExecutn:
(Parameter Index)

return "updated src balance" + cstl.getDouble(4)
(Out parameter Pos)
+ "dest balance" + cstl.getDouble(5);