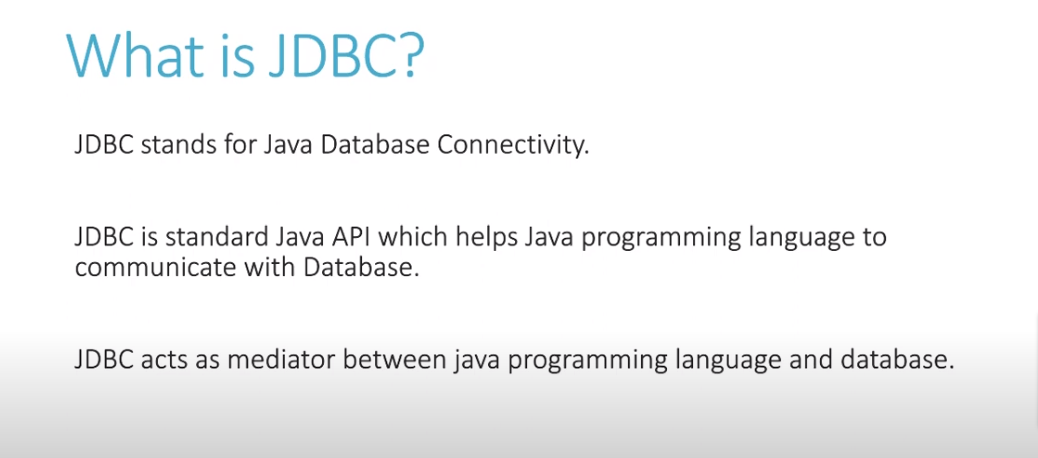
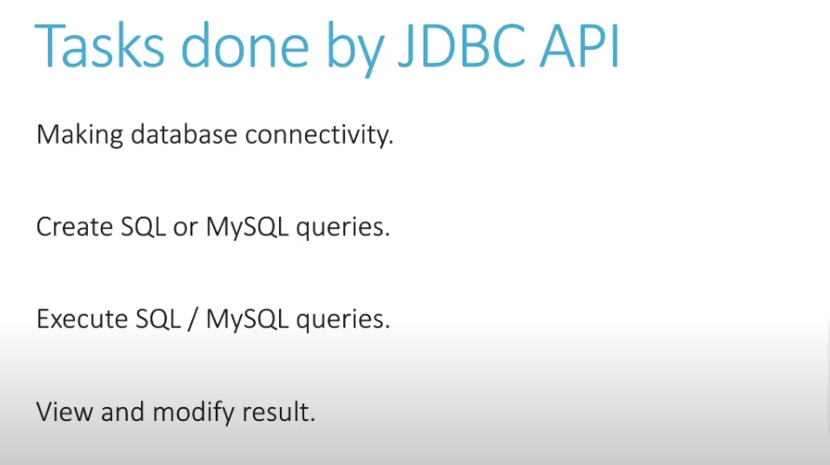
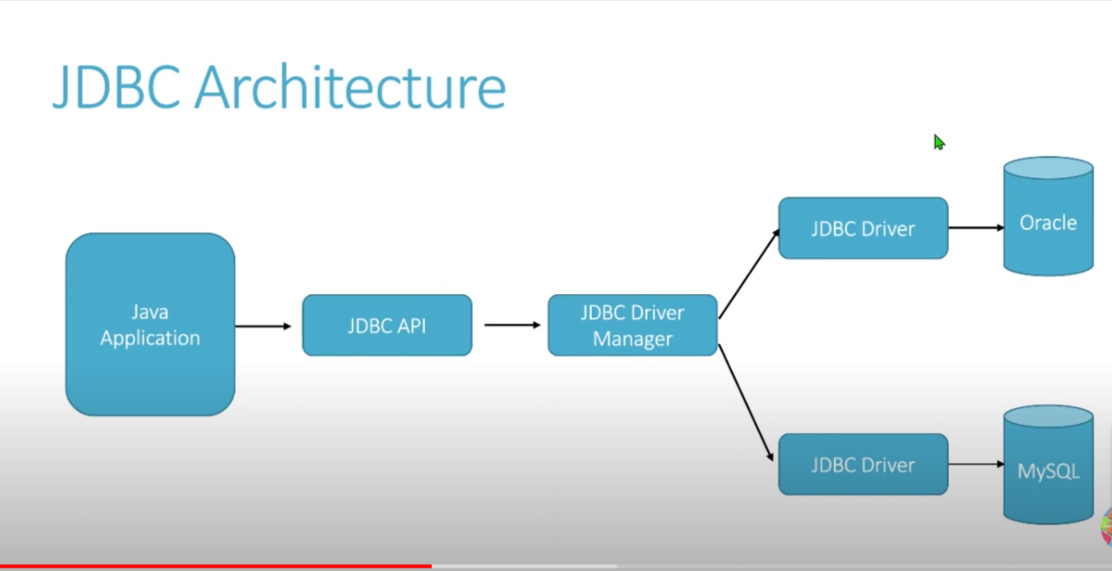
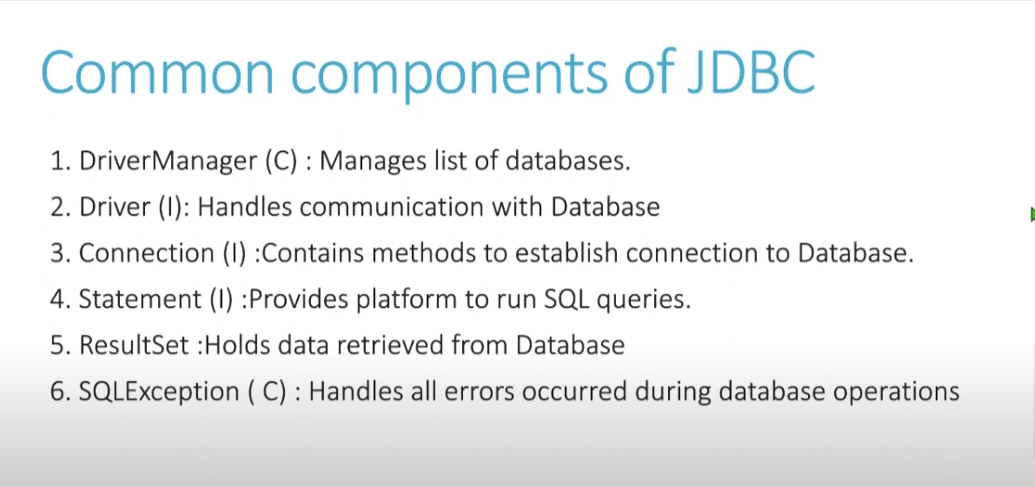
JDBC











🡺Config jdbc

* To perform operation we have 3 type of statement :

i)preparedStatement

ii)Statement -> createStatement

iii)callableStatement

Inserting data in database :

package jdbc.student;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.SQLException;

import java.util.Scanner;

public class StudentDatabase {

//creating connection

private static Connection *connection*=null;

private static Scanner *sc*=new Scanner(System.***in***);

public static void main(String[] args) {

StudentDatabase St=new StudentDatabase();

try {

Class.*forName*("com.mysql.cj.jdbc.Driver");

// to find the port MYSQL->show global variables like 'PORT';

String dbURL="jdbc:mysql://localhost:3306/jdbcdatabase";

String username="root";

String password="Ajaj13994010#";

*connection*= DriverManager.*getConnection*(dbURL,username,password);

System.***out***.println("Enter choice");

System.***out***.println("1.Inset record");

int choice=Integer.*parseInt*(*sc*.nextLine());

switch (choice) {

case 1:

St.insertRecord();

break;

default:

break;

}

} catch (Exception e) {

throw new RuntimeException(e);

// to get exact message : e.getMessage();

}

}

private void insertRecord() throws SQLException {

String SQLquery="INSERT INTO student(name,percentage,address) values(?,?,?)";

PreparedStatement ps=*connection*.prepareStatement(SQLquery);

//inserting dynamically

ps.setString(1, "Aman");

ps.setFloat(2, (float) 90.6);

ps.setString(3, "Bihar");

int rows=ps.executeUpdate();

if(rows>0) {

System.***out***.println("Record inserted sucessfully.");

}

}

}

* Read the data form mysql

private void getRecord() throws SQLException {

String SQLquery="SELECT \* FROM student where roll\_no=10";

Statement st= *connection*.createStatement();

ResultSet result= st.executeQuery(SQLquery);

if(!result.next()) {

System.***out***.println("No Record present !");

}

else {

while(result.next()) {

int roll\_no=result.getInt("roll\_no");

String name=result.getString("name");

double percentage=result.getDouble("percentage");

String address=result.getString("address");

System.***out***.println(roll\_no+" "+name+" "+percentage+" "+address);

}

}

}

Get record using callableStatment :

1. Create a procedure

A) DELIMITER @

B) DROP PROCEDURE IF EXISTS GET\_ALL

-> @

C) CREATE PROCEDURE GET\_ALL()

-> BEGIN

-> SELECT \* FROM STUDENT;

-> END

-> @

D) DELIMITER ;

E)TO CHECK -> CALL GET\_ALL();

🡺for creating procedure with parameter -> CREATE PROCEDURE PROCEDRE\_NAME(table\_column\_name data\_type)

private void callabeStatmentGetRecord() throws SQLException {

CallableStatement cbst= *connection*.prepareCall("{ call GET\_ALL() }");

ResultSet result= cbst.executeQuery();

while(result.next()) {

int roll\_no=result.getInt("roll\_no");

String name=result.getString("name");

double percentage=result.getDouble("percentage");

String address=result.getString("address");

System.***out***.println(roll\_no+" "+name+" "+percentage+" "+address);

}

}

🡪Updat data

private void updateRecord() throws SQLException {

// checking whether result is present in table or not

System.***out***.println("Enter the roll no :");

int Roll\_no=Integer.*parseInt*(*sc*.nextLine());

String SQLquery="SELECT \* FROM student WHERE roll\_no=+"+Roll\_no;

Statement st= *connection*.createStatement();

ResultSet result= st.executeQuery(SQLquery);

if(!result.next()) {

System.***out***.println("No such student exist. Please Enter the correct Roll no !");

}

else {

int roll\_no=result.getInt("roll\_no");

String name=result.getString("name");

double percentage=result.getDouble("percentage");

String address=result.getString("address");

System.***out***.println("Roll\_no is :"+roll\_no);

System.***out***.println("Name is :"+name);

System.***out***.println("Percentage is :"+percentage);

System.***out***.println("Adress is :"+address);

System.***out***.println("Enter your choice you want to update -");

System.***out***.println("1.Update Name");

System.***out***.println("2.Update percentage");

System.***out***.println("3.Update Adress");

int choice=Integer.*parseInt*(*sc*.nextLine());

String updateQuery="UPDATE student set ";

switch (choice) {

case 1:

System.***out***.println("Enter the new name : ");

updateQuery=updateQuery+"name=? WHERE roll\_no="+Roll\_no;

PreparedStatement ps =*connection*.prepareStatement(updateQuery);

ps.setString(1, *sc*.nextLine());

int row=ps.executeUpdate();

if(row>0) {

System.***out***.println("Name has been updated successfully");

}

break;

default:

break;

}

}

}

* Delete Record :

private void deleteRecord() throws SQLException {

String SQLquery="DELETE FROM student where roll\_no=+"+Integer.*parseInt*(*sc*.nextLine());

Statement st= *connection*.createStatement();

int row=st.executeUpdate(SQLquery);

if(row>0) {

System.***out***.println("Record has been sucessfully deleted !");

}

else {

System.***out***.println("Somthing went wrong");

}

}

Transction management ->

Note : transction management is required when we perform multiple query .

ii)without transiction management what we noticed that if the query which fails that not excuted but the remaining excuted.

iii)with the transcition management what we can achive is that if any of the query failed then no of the other query will going to excute.

private void transctionManagement() throws SQLException {

*connection*.setAutoCommit(false);

String quer1="INSERT INTO student(name,percentage,address) VALUES('saddam',89,'Bihar')";

String quer2="INSER INTO student(name,percentage,address) VALUES('anurag',89,'Bihar')";

PreparedStatement st= *connection*.prepareStatement(quer1);

int row1=st.executeUpdate();

st=*connection*.prepareStatement(quer2);

int row2=st.executeUpdate();

if(row1>0 && row2>0) {

*connection*.commit();

}

}

Batch processing :

By using batch processing we can excute multiple query in single shot .

private void BatchProcessing() throws SQLException {

*connection*.setAutoCommit(false);

String quer1="INSERT INTO student(name,percentage,address) VALUES('saddam',89,'Bihar')";

String quer2="INSER INTO student(name,percentage,address) VALUES('anurag',89,'Bihar')";

PreparedStatement st= *connection*.prepareStatement(quer1);

st.addBatch(quer1);

st.addBatch(quer2);

int [] arr=st.executeBatch();

for(int i:arr) {

if(i>0) {

continue;

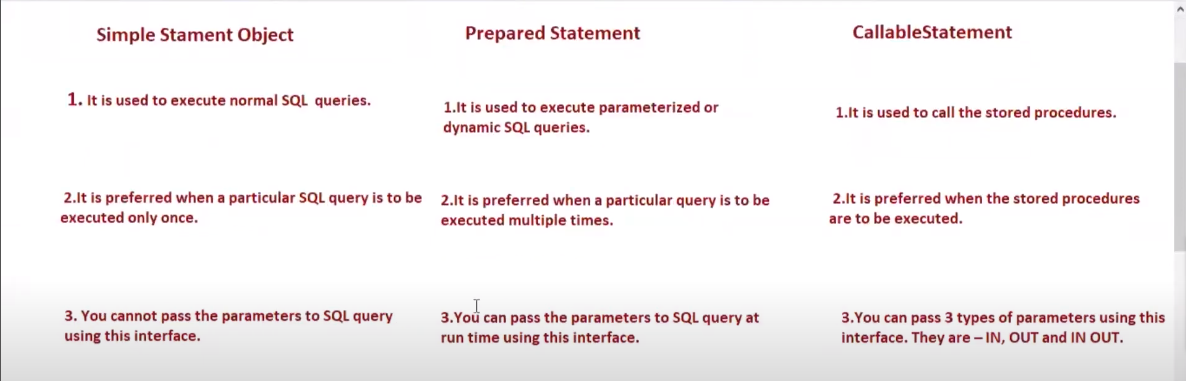
}

*connection*.rollback();

}

*connection*.commit();

* Diffrence between statement , prepared statement and callable statement object interface in jdbc :



Uploading Image :

1.creating table for uploading image

CREATE TABELE img\_table{

-> img\_id int auto\_increment primary key,

-> img\_data LONGBLOB NOT NULL,

-> upload\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

-> )

-> ;

1.uploading image

private void uploadimg(){

String imgpath="C:\\Users\\91930\\Downloads\\1.png";

try {

// interface for converting IMG into binary form

FileInputStream fileInputStream=new FileInputStream(imgpath);

byte [] imgData=new byte[fileInputStream.available()]; // Available will going

fileInputStream.read(imgData);

String SQLquery="INSERT INTO img\_table(img\_data) VALUES(?)";

PreparedStatement preparedStatement= *connection*.prepareStatement(SQLquery);

preparedStatement.setBytes(1,imgData);

int row=preparedStatement.executeUpdate();

if(row>0) {

System.***out***.println("Data has been inserted Sucessfully !");

}

} catch (SQLException e) {

// **TODO**: handle exception

throw new RuntimeException(e.getMessage());

}catch (FileNotFoundException e) {

throw new RuntimeException(e.getMessage());

}catch (IOException e) {

throw new RuntimeException(e.getMessage());

}

}

3.Reading image

private void ReadImageData() {

try {

String folder\_path="C:\\Users\\91930\\Downloads\\";

String SQLquery="SELECT img\_data from img\_table where img\_id=1";

Statement statment= *connection*.createStatement();

ResultSet result=statment.executeQuery(SQLquery);

if(result.next()) {

String img\_path=folder\_path+"extractedImg.jpg"; // give the path and name of the image

OutputStream outPutStream=new FileOutputStream(img\_path);

byte[] imgdata=result.getBytes("img\_data");

outPutStream.write(imgdata);

System.***out***.println("Image has been fetched sucessfuly");

}

} catch (SQLException e) {

// **TODO**: handle exception

throw new RuntimeException(e.getMessage());

}

catch (FileNotFoundException e) {

// **TODO**: handle exception

throw new RuntimeException(e.getMessage());

}

catch (IOException e) {

// **TODO**: handle exception

throw new RuntimeException(e.getMessage());

}

}