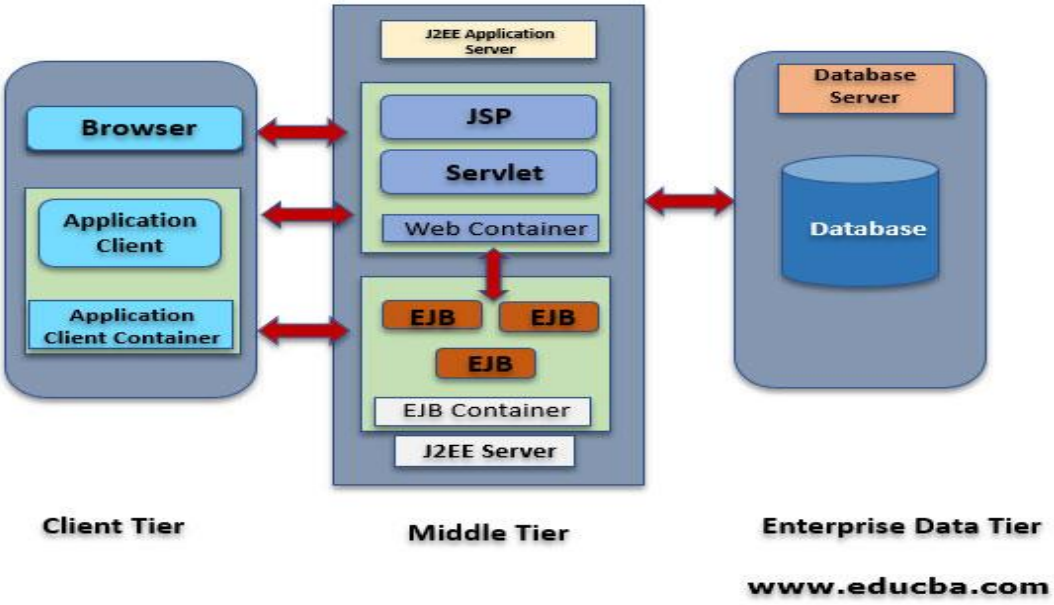
J2EE

- J2EE stands for Java2 Enterprise edition
- Java Enterprise Edition provides a large set of libraries using which we can develop enterprise applications.
- Following are the important JEE libraries (or) API'S
  - Servlet
  - JMA(Java Mail API)
  - JMS(Java Message Services)
  - JSP(Java Server pages)
  - JTA(Java Transaction Beans)
  - JPA(Java Persistence API)

Why do we need J2EE?

- Using JEE we can develop the advanced functionalities that are required for an enterprise Application.



Coupling

- Coupling is the dependency between two applications (or) two classes (or) two objects.
- We have 2 types of coupling.
  - **Tight coupling**
    - If the change in the implementation of one application if it is effecting another application that type of coupling is called tight coupling.
  - **Loose coupling**

- If the change in the implementation of one application does not effecting another application that type of coupling is called as loose coupling.

Method

- A method is a block of instructions that is used to perform a specific task.
- We have 2types of methods.
  - No argument method→syntax: methodname();
  - Parameterized method→syntax: methodname(arguments)

Parameter

- A parameter is a variable which is declared in the method declaration which is used to store the value passed by caller.

Argument

- It is the value passed for a parameter during method call.

```
public class demo1 {
	public void print(int n) {
		System.out.println(n);
	}
	public static void main(String[] args) {
		demo1 obj = new demo1();
		obj.print(10);
		obj.print(15);
	}
}
```

Note :

- In the above example, n Is the parameter of the print method where as 10 and 15 is the argument.
- If a method is accepting an argument of type interface then to invoke that method we need to pass the reference of implementation object of that interface.

Example program to understand the above note point 2

```
package practise;

public interface calci {
int add(int n1,int n2);
}
class calciImpl1 implements calci{
	@Override
	public int add(int n1, int n2) {
		return n1+n2;
	}

}
package practise;

public class clacitest {
	static public void print(calci c) {
```

```
            System.out.println(c.add(10, 21));
      }
      public static void main(String[] args) {
            print(new calciImpl1());
      }
}
```

Code modularization

- The process of dividing software into modules and submodules is called as code modularization.
- We can achieve code modularization with the help of design patterns.

Design patterns

- Design patterns are used to achieve code modularization and also for code optimization and the efficient development of the applications.
- Following are the important design patterns
    - DAO(Data Access Object)
    - MVC(Model View Controller)
    - Creational Design Pattern
    - Factory Design Pattern

Creational Design Pattern

- This design pattern is used to write object creation logic.
  Ex: Singleton design pattern

Singleton design pattern

- It is used to limit the number of objects that have to be created for a class to 1.
- We can achieve this design by making the constructor private and creating helper method to return the reference of the object.

Example program:

```
public class SingletonDesignPattern {
      private static SingletonDesignPattern d;
      private SingletonDesignPattern() {
      }
      public static SingletonDesignPattern getSingletonDesignPattern() {
            if (d == null) {
                  d = new SingletonDesignPattern();

            }
            return d;
      }
}
```

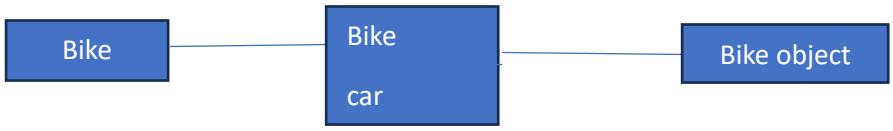
```java
package practise;
public class TestSingletonDemo {
	public static void main(String[] args) {
    System.out.println(SingletonDesignPattern.getSingletonDesignPattern());
    System.out.println(SingletonDesignPattern.getSingletonDesignPattern());
    System.out.println(SingletonDesignPattern.getSingletonDesignPattern());
	}
}
```

Factory Design Pattern

- It is a design pattern which is used to create the different objects of the same type.
- In the factory design pattern, we will create a helper method that will accept a string as an argument and return the reference of the object.
- In the factory design pattern, we have to write 3 types of logic.
  - **Implementation logic**
    - It is used to provide the implementation.
  - **Object creation logic**
    - It is used to create the objects of implementation classes
  - **Utilisation logic**
    - It is written to use the functionality of the factory design pattern.

Note :

- Without utilization logic, the other 2 types of logic does not have any use



Vehicle factory

Example program to understand factory methods

```java
public interface Vehicle {
void start();
}
-----------------------------------------------------------------
public class Bike implements Vehicle {
	@Override
	public void start() {
		System.out.println("Bike has been started ..!!");
	}
}
-----------------------------------------------------------------
public class Car implements Vehicle {
	@Override
	public void start() {
		System.out.println("Car has been started ");
```

```
	}
}
-----------------------------------------------------------------------
public class Vehicle_factory {
	public Vehicle getVehicle(String type) {
		if (type.equals("bike")) {
			return new Bike();
		} else if (type.equals("car")) {
			return new Car();
		} else {
			return null;
		}
	}
}
-----------------------------------------------------------------------


import java.util.Scanner;
public class TestVehicle {
	public static void main(String[] args) {
		Scanner sc = new Scanner(System.in);
		System.out.println("Enter the type of vehicle you want");
		String type = sc.next();
		Vehicle_factory factory = new Vehicle_factory();
		Vehicle v = factory.getVehicle(type);
		if (v != null) {
			v.start();
		} else {
			System.out.println(type + "Not present");
		}
	}
}
```

Class loading process

- The process of loading a .class file into the Java memory is called as class loading process.
- We can load a class into members in 2ways
  - By using the member of the class like variable, method (or) constructor.
  - Using the static method forName(Stringfq(N)) which belongs to java.lang.class

Example program to understand the class loading by using member of the class

```
public class A{
	static {
		System.out.println("A class is loaded into jvm memory");
	}
	public static void main(String[] args) {
		System.out.println("hello from A class");
		System.out.println(new B());
	}
}
class B{
	static {
		System.out.println("B class is loaded to jvm memory");
	}
}
```

ForName(String fqcn)

- It is a static method present in java.lang.class which is used to load a class into JVM memory.
- This method will throw classNotfoundException if the fully qualified classname is incorrect.
- As classNotfound Exception is a checked.
- Exception we must either handle or we should throw the exception ,otherwise we will get an exception .

Example program to understand classLoading process using forName(String)

```
public class smartphone {
	static {
		System.out.println("smartphone is loaded into jvm memeory");
	}

	public static void main(String[] args) {
	  System.out.println("Main starts..!!");
	  try {
			Class.forName("org.jp.practise.Simcard");
		} catch (ClassNotFoundException e) {
			System.out.println("Incorrect class name");
			e.printStackTrace();
		}
	  System.out.println("Main ends");
	}
}

class simcard{
	static {
		System.out.println("Simcard is loaded into jvm memory");
}}
```

JAR(Java Archieve)

- It is a compressed file forward which is similar to ZIP file format using which we can compress different types of files together.

  Contents of jar

  - **.Java**
    - It is used to store the java source code and having this in a jar file is optional
  - **.Class**
    - It is used to store the byte code. It is an executable file
  - **.Xml**
    - It is a the configuration file which is use to store the configuration data.
  - **.Config**
    - It is configuration file
  - **.Properties**

- It is a file where we can store the configuration data in the form of key-value pairs.

Why we need Jar file?

- We need a jar file to import the properties of an application**.**

Steps to create a Jar file in Ecllipse

- Click on file and select export option
- Select java and click on jarfile option.
- Select the files that you need to compress in the jar file.
- Browse the location where you want to save the jar file and click on finish.

Adding Jarfile into the classpath

- We can add ajar file into the classpath in 2 ways.
  - Externally
  - Internally

  Steps to add a jar file externally
  - Right click on the project and select the properties option.
  - Click on java build path
  - Click on libraries tab and open it.
  - Click on classpath and select add external jars option
  - Browse the location of the jarfile, add it click on apply and close.

  Steps to add a jar file internally
  - Right-click on the project and create a folder with the name lib
  - Copy the jar file that you need for your classpath and paste it into the lib folder.
  - Right-click on the project and click on properties.
  - Click on the java built path and open the libraries tab
  - Select classpath and click on add jars to all the jar files.

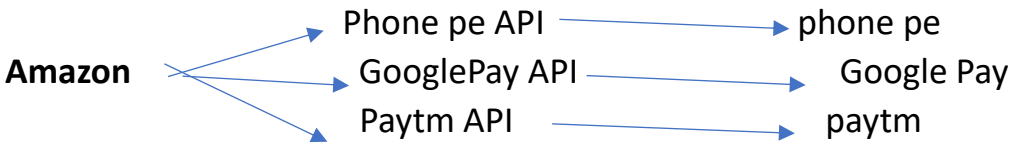
Note :

- It is always recommended to add the jar file internally rather than adding externally.

API (Application programming interface)

- API is used to achieve inter application communication using API two.
- Applications can communicate with each other.
- By using API 2 applications can interchange the information between them.

- We can also use API'S to transfer the data from one Application to another application (**DataExchange**)



JDBC (Java Database Connectivity)

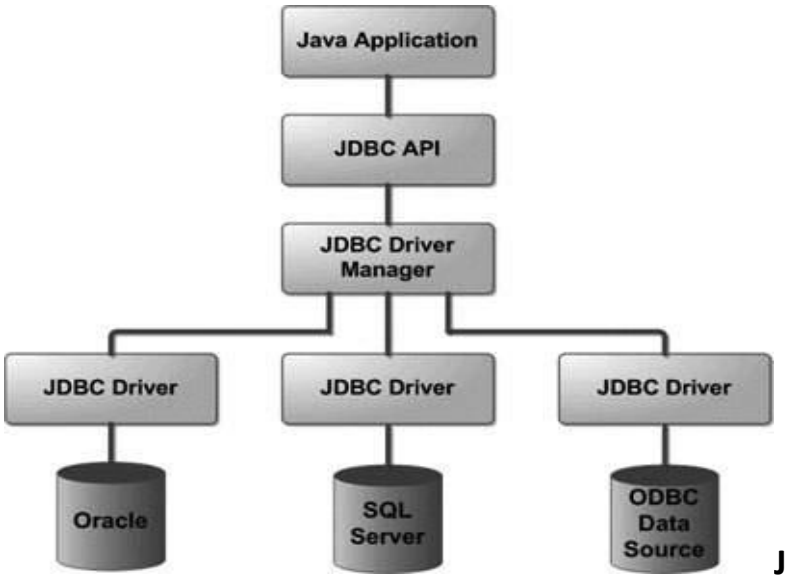
- It is a specification provided in the form of abstraction API which is used to achieve the loose coupling between Java Application and Database Server.
- JDBC is just a specification and it does not have any implementation.
- The classes and interfaces of JDBC API are present in the following packages
  - Java. sql
  - Javax. Sql

Following are the important classes and interfaces of JDBC API classes

- Java.sql.DriverManager. It is an helper class or factory class interfaces
  - **Java.sql.Connection**
  - **Java.sql.Statement**
  - **Java.sql.Prepared statement**
  - **Java.sql.Callable statement**
  - **Java.sql.ResultSet**
  - **Java.sql.Driver**

JDBC Driver (or)Driver software

- It is the implementation of JDBC API which is provided by the database vendor.
- JDBC Driver will have the implementation for all the interfaces of JDBC API



JDBC Architecture

Specifications of JDBC

- Every JDBC driver (or) Driver software most have a driver class present in it.
- The driver class must implement in java.sql.Driver interface present in the JDBC API.

Differentiate between Driver interface and Driver class

Driver interface is present in JDBC API	Driver class is present in JDBC driver(or) Driver software.
This is interface is provided is provided by sun microsystems as a part of JDBC API	This class is provided by database vendor as a part of JDBC driver
This interface is present java.sql.package	This class is the implementation of java sql.driver interface

2)Differentiate between JDBC and JDBC drivers

3)Differentiate between driver interface and Driver class and Driver software

Steps to download MYSQL Connector.java

- Open the browser and search for Mawen repository
- Open mvnrepository.com
- Search for Mysql -Connector -java and open it.
- Select the version of the Mysql driver you need and open it.
- Click on the jar and download the jar file
- Add the jar file to your classpath.

Steps to create a JDBC application

- Load and Register the driver (here driver is nothing but the driver class which is present in JDBC driver)
- Establish the connection between java application and Database server.
- Create a statement /platform
- Process the result (optional)
- Close all the costly resources.

Costly Resources

- A resource which uses the system properties in the form of stream is called as a costly resource.

Note:

- If we don't close a costly resource , it will affect the performance of the application.
- All the interfaces of JDBC API are costly resources.

Step-1 : load and Register driver class

- We can load and Register a Driver class in 2 Ways
  1. Manually
  2. By forName(String)

Code to load and register the driver class manually

```
   package org.jsp.jdbcDemo;
import java.sql.DriverManager;
import java.sql.SQLException;
import com.mysql.cj.jdbc.Driver;
public class LoadAndRegisterDriver {
	public static void main(String[] args) {

		try {
			Driver d = new Driver();
			DriverManager.registerDriver(d);
			System.out.println("Driver class loaded and registered successfully");
		} catch (SQLException e) {

			e.printStackTrace();
			System.err.println("Something went Wrong");
		}

	}
}
```

Code to load and Register the Driver class by using forName(String)

```
package org.jsp.jdbcDemo;

public class LoadAndRegisterDriver2 {
	public static void main(String[] args) {
		try {
			Class.forName("com.mysql.cj.jdbc.Driver");
			System.out.println("Driver class is loaded and registered");
		} catch (ClassNotFoundException e) {

			e.printStackTrace();
		}
	}
}
```

Java.sql.Driver.Manager

- It is an helper class belongs to JDBC API.
- It is a factory class for interface java.sql.connection
- Following are the important static methods of driver manager.
  - **i. RegisterDriver(Driver)**
    - It is used to register the Driver class with JDBC API
  - **ii. getConnection()**
    - It is a static factory method or helper method present in DriverManagerClass.
    - This method will create the implementation object of java.sql.conncection and returns the reference.
    - So the return type of this method in connection.
    - This method is overloaded and following are the overloaded methods
      - **getConnection(String url)**
      - **getConnection(String url,String username,String password)**
      - **getConnection(String url,properties info)**

java.sql.Connection

- it is an interface belongs to JDBC API
- It is used to provide a session between java application and Database Server.
- This interface Is also factory for statement, prepared statement and callable statement
- Following are the important factory methods of connection interface
  - **CreateStatement()**
  - **PrepareStatement(String)**
  - **PrepareCall(String)**

Note :

The costly resources must be closed in the finally block by using if condition to avoid NullPointerException.

<u>Step-2</u> Code to establish the connection between java application and DB server

```
package org.jsp.jdbcDemo;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
public class EstablishConnection {
	public static void main(String[] args) {
		Connection con = null;
		try {
			Class.forName("com.mysql.cj.jdbc.Driver");
			System.out.println("Driver loaded and Registered ");
			con =
DriverManager.getConnection("jdbc:mysql://localhost:3306?user=root&password=charantej@@18");
			System.out.println("Connetion is established");
 con = DriverManager.getConnection("jdbc:mysql://localhost:3306", "root", "charantej@@18");
		} catch (ClassNotFoundException | SQLException e) {
			e.printStackTrace();
		} finally {
			if (con != null) {
				try {
					con.close();
					System.out.println("connection closed");
				} catch (SQLException e) {
					e.printStackTrace();
				}
			}}}}
```

Step3:Create a statement or platform

- We need to create a statement or platform for the execution of SQl queries .
- We have the following interfaces using which can execute the Queries.
  - **Java.sql.statement**
  - **Java.sql.preparedStatement**
  - **Java.sql.CallableStatement**
- We can create implementation objects of the above interfaces by using the factory methods (or) helper methods present in connection interface.

<u>CreateStatement()</u>

- It is a factory method (or) helper method present in connection interface
- It will create the implementation object of java.sql.statement and returns the reference.
- So the return type of this method is statement.

Java.sql.statement

- It is an interface present in JDBC API
- The implementation for this interface is provided by database vendor as part ofJDBC driver.
- This interface will provide a platform for the execution of SQL queries.
- Following are the methods of statement interface using which we can execute the queries
    - **Execute(String)**
    - **ExecuteUpdate(String)**
    - **ExecuteQuery(String)**

Step-3 Code to create a statement/platform

```
package org.jsp.jdbcDemo;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

public class CreateStatement {
	public static void main(String[] args) {
		Connection con = null;
		Statement st = null;
		String url = "jdbc:mysql://localhost:3306";
		String user = "root", password = "charantej@@18";
		try {
			Class.forName("com.mysql.cj.jdbc.Driver");
			System.out.println("Connection established");
			con = DriverManager.getConnection(url, user, password);
			System.out.println("Platform created");
		} catch (ClassNotFoundException | SQLException e) {
			e.printStackTrace();
		} finally {
			if (con != null) {
				try {
					con.close();
					System.out.println("Connection closed");
				} catch (SQLException e) {
					e.printStackTrace();
				}
			}
			if (st != null) {
				try {
					st.close();
					System.out.println("Statement closed");
				} catch (SQLException e) {
					e.printStackTrace();
				}
			}}}}
```

Step-4: Execute the query

- To execute the SQL queries we have the following methods available in the statement interface.

❖ **booleanExecute(String SQl)**

- This method is used to execute all types of SQL queries.
- The return type of this method is Boolean.
- It will return true if we execute DQl queries else it will return false.
- Whenever we execute the DQL query a Resultset object will be created so it will return true, otherwise it will return false.

❖ **int executeUpdate(String DML)**

- This method is present in statement interface.
- This method is specific to execute only DML queries.
- If we execute the query other than DML it will throw SQLException.
- The return type of this method is INT.
- The value returned by this method depends upon the number of rows effected.

❖ **executeQuery(String DQL)**

- This method is present in statement interface.
- This method is specific for the execution of DQl queries
- If we execute a query other than DQL it will throw SQLException.
- The return type of this method is java.sql.Resultset.

Code to create a Database by using statement interface

```
package org.jsp.jdbcDemo;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

public class CreateDatabase {
	public static void main(String[] args) {
		String qry = "create database jdbc_demo";
	String url = "jdbc:mysql://localhost:3306", user = "root", password = "charantej@@18";
		Connection con = null;
		Statement st = null;
		try {
			Class.forName("com.mysql.cj.jdbc.Driver");
			con = DriverManager.getConnection(url, user, password);
			st = con.createStatement();
			boolean res = st.execute(qry);
			System.out.println("Database Created");
			System.out.println(res);
		} catch (ClassNotFoundException | SQLException e) {
			e.printStackTrace();
		} finally {
			if (con != null) {
				try {
					con.close();
```

```
                                System.out.println("Connection closed");
                        } catch (SQLException e) {
                                e.printStackTrace();
                        }
                }
                if (st != null) {
                        try {
                                st.close();
                                System.out.println("Statement closed");
                        } catch (SQLException e) {
                                e.printStackTrace();
                        }
                }
        }

    }
}
```

Code to create a person table using statement interface

```
package org.jsp.jdbcDemo;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

public class CreatePersonTable {
      public static void main(String[] args) {
String qry = "create table person(id int not null,name varchar(45) not null,phone bigint(20)
not null unique, password varchar(45) not null, primary key(id))";
String url = "jdbc:mysql://localhost:3306/jdbc_Demo", user = "root", password ="charantej@@18";
            Connection con = null;
            Statement st = null;
            try {
                  Class.forName("com.mysql.cj.jdbc.Driver");
                  con = DriverManager.getConnection(url, user, password);
                  st = con.createStatement();
                  boolean res = st.execute(qry);
                  System.out.println("Database Created");
                  System.out.println(res);
            } catch (ClassNotFoundException | SQLException e) {
                  e.printStackTrace();
            } finally {
                  if (con != null) {
                        try {
                              con.close();
                              System.out.println("Connection closed");
                        } catch (SQLException e) {
                              e.printStackTrace();
                        }
                  }
                  if (st != null) {
                        try {
                              st.close();
                              System.out.println("Statement closed");
                        } catch (SQLException e) {
                              e.printStackTrace();
                        }}}}}
```

Code to insert a record into person table using statement interface

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

public class _07_InsertRecordsInPerson {
	public static void main(String[] args) {
		Connection con = null;
		Statement st = null;
		//String qry = "insert into person values(1,'tej',8787837,'tej@123')";
		try {
			Class.forName("com.mysql.cj.jdbc.Driver");
			con = DriverManager.getConnection("jdbc:mysql://localhost:3306/jdbc_demo",
"root", "charantej@@18");
			st = con.createStatement();
			//st.execute(qry);
			st.execute("insert into person values(2,'jos',2165761,'jos@123')");
			System.out.println("Record is inserted in person table");
		} catch (ClassNotFoundException | SQLException e) {
			e.printStackTrace();
		} finally {
			if (con != null) {
				try {
					con.close();
				} catch (SQLException e) {
					e.printStackTrace();
				}
			}
			if (st != null) {
				try {
					st.close();
				} catch (SQLException e) {
					e.printStackTrace();
				}
			}
		}

	}
}
```

Code to update a record in the person table using statement interface

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

public class _08_UpdatePerson {
	public static void main(String[] args) {
		Connection con = null;
		Statement st = null;
		 String qry = "update person set name='priya',password='123@jos' where id=2";
		try {
			Class.forName("com.mysql.cj.jdbc.Driver");
     con = DriverManager.getConnection("jdbc:mysql://localhost:3306/jdbc_demo", "root",
"charantej@@18");
```

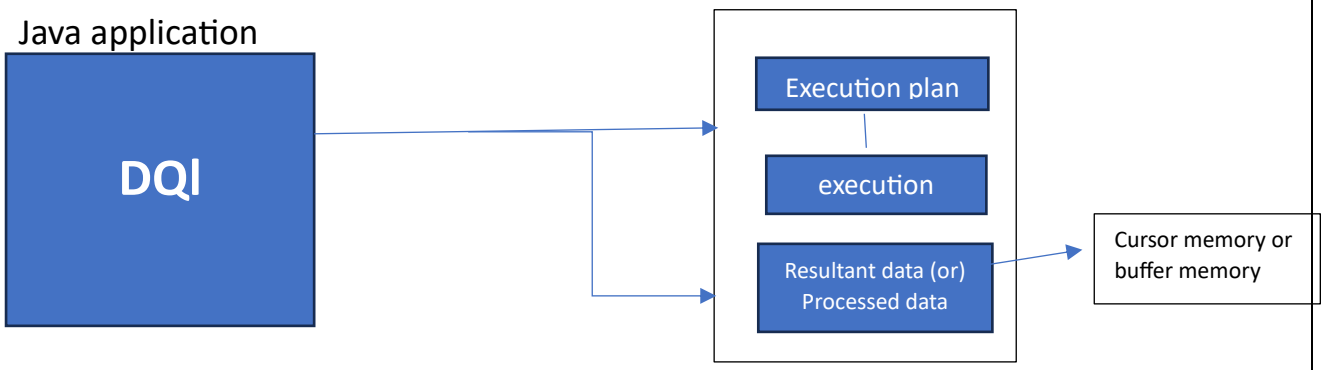
```
                st = con.createStatement();
                st.execute(qry);
                System.out.println("Record is updated in person table");
            } catch (ClassNotFoundException | SQLException e) {
                e.printStackTrace();
            } finally {
                if (con != null) {
                    try {
                        con.close();
                    } catch (SQLException e) {
                        e.printStackTrace();
                    }
                }
                if (st != null) {
                    try {
                        st.close();
                    } catch (SQLException e) {
                        e.printStackTrace();
                    }
                }
            }}}
```

Code to delete a record from the person table using statement interface

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
public class _09_DeletePerson {
    public static void main(String[] args) {
        Connection con = null;
        Statement st = null;
        String qry = "delete from  person  where id=1";
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            con = DriverManager.getConnection("jdbc:mysql://localhost:3306/jdbc_demo",
"root", "charantej@@18");
            st = con.createStatement();
            st.execute(qry);
            System.out.println("Record is deleted in person table");
        } catch (ClassNotFoundException | SQLException e) {
            e.printStackTrace();
        } finally {
            if (con != null) {
                try {
                    con.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }
            if (st != null) {
                try {
                    st.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }}}}
```

What will happen when we execute the DQL Query

Whenever we execute a DQL Query a resultant data or processed data will be created in Cursor memory or Buffer memory which is present in Database server.



- We can process data present in the cursor memory by using the reference of java.sql.ResultSet interface.
- We can create the implementation object of ResultantSet interface by using the following helper methods (or) factory methods
  - ExecuteQuery(String)
  - getResultSet()

getResultSet()

- It is a method present in statement interface
- This will create the implementation object of ResultSet interface and returns the reference whenever we execute the DQL queries ,otherwise it will return null.
- The return type of this method is ResultSet.

Java.sql.ResultSet

- It is an interface belongs JDBC API.
- The implementation of this interface is provided by database vendor as a part of JDBC Driver.
- It is used to process the resultant data that is present in the cursor memory because of the DQL query.
- Following are the important methods present in ResultSet interface.
  - **Next()**
    - This method is used to check whether any record is present in DatabaseServer (or)not.
    - The return type of this method Is Boolean and it will return true if any record is present else return false.
  - **Get***()**
    - Here *** represents datatype
    - It is a method which is used to fetch the data from ResultSet.

- It is overloaded and the returntype of the method is same as the respective datatype. The return type of getInt() is int,getString() is string, getDouble() is Double etc.
- Following are the different overloaded methods of get***()
  - ✓ **Get***(int column_index)**
  - ✓ **Get***(String column _label)**

Code to fetch All the records from person table

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
public class _10_FetchAllRecords {
	public static void main(String[] args) {
		Connection con = null;
		Statement st = null;
		ResultSet rs = null;
		try {
			Class.forName("com.mysql.cj.jdbc.Driver");
	con = DriverManager.getConnection("jdbc:mysql://localhost:3306/jdbc_demo", "root",
"charantej@@18");
			st = con.createStatement();
		       //rs=st.executeQuery("select * from person");
			boolean res = st.execute("select * from person");
			if (res) {
				rs = st.getResultSet();
			}
			while (rs.next()) {
			System.out.println("Id:" + rs.getInt(1));// rs.getInt("id")
			System.out.println("Name:" + rs.getString("name"));// rs.getString(2)
			System.out.println("phoneno:" + rs.getLong("phone"));// rs.getLong(3)
		System.out.println("Passoword:" + rs.getString(4));// rs.getString("password")
			System.out.println("------*****------");
			}
		} catch (ClassNotFoundException | SQLException e) {
			e.printStackTrace();
		} finally {
			if (con != null) {
				try {
					con.close();
				} catch (SQLException e) {
					e.printStackTrace();
				}
			}
			if (st != null) {
				try {
					st.close();
				} catch (SQLException e) {
					e.printStackTrace();
				}
			}
			if (rs != null) {
				try {
					rs.close();
```

```
                        } catch (SQLException e) {
                              e.printStackTrace();
                        }}}}}
```

Java.sql.preparedStatement

- It is an interface belongs to JDBC API.
- The implementation of prepared statement is provided by database vendor as a part of JDBC driver.
- Prepared statement is also called as precompiled statement as the query gets compiled during the creation of statement.
- By using prepared statement we will compile the query once and execute it multiple times.
- By using prepared statements we can create dynamic queries as it supports placeholder(?)

Placeholder

- Placeholder is a parameter which is used to hold the values from the user dynamically during the execution time.
- It is represented by the symbol "?".

Rules to use placeholder

- We need to assign the values for the placeholders before the execution, otherwise we will get exception.
- The number of placeholders and the values must be same else we will get the exception.
- Every placeholder will have an index which starts from 1.

Note

- We can assign the values for the placeholders by using set***(int placeholderIndex,***value) which is present in preparedstatement.
- Here *** represents datatype.
- To assign value for int we will use setInt(int,int).
- To assign value for string we will use setString(int,String).
- To assign value for double we will use setDouble(int,double)
- To assign value for long we will use setLong(int,long) etc..

Note

- The return type of set***() is void and get***() is same as the respective datatype.
- It is recommended to use preparedstatement over statement whenever we need to execute more than one query of sametype.

Program to insert multiple records using statement interface

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

public class InsertMultipleRecordsByStatement {
	public static void main(String[] args) {
		Connection con = null;
		Statement st = null;

		try {
			Class.forName("com.mysql.cj.jdbc.Driver");
	con = DriverManager.getConnection("jdbc:mysql://localhost:3306/jdbc_demo", "root",
"charantej@@18");
			st = con.createStatement();
			st.executeUpdate("insert into person values(101,'tej',378873,'tej@123')");
			st.executeUpdate("insert into person values(102,'jos',88788,'jos@123')");
		st.executeUpdate("insert into person values(103,'cherry',9908873,'cherry@123')");
			System.out.println("3 records inserted succesfully");

		} catch (ClassNotFoundException | SQLException e) {
			e.printStackTrace();
		} finally {
			if (con != null) {
				try {
					con.close();
				} catch (SQLException e) {
					e.printStackTrace();
				}
			}
			if (st != null) {
				try {
					st.close();
				} catch (SQLException e) {
					e.printStackTrace();
				}
			}
		}
	}
}
```

Program to insert multiple records by using preparedstatement interface

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.sql.Statement;

public class _12_InsertMultipleRecordsByPreparedStatement {
	public static void main(String[] args) {
		Connection con = null;
		PreparedStatement pst = null;
		try {
			Class.forName("com.mysql.cj.jdbc.Driver");
	con = DriverManager.getConnection("jdbc:mysql://localhost:3306/jdbc_demo", "root",
"charantej@@18");
			pst = con.prepareStatement("insert into person values(?,?,?,?)");
			pst.setInt(1,201);
			pst.setString(2, "virat");
			pst.setLong(3,66777);
			pst.setString(4, "RCB");
			pst.executeUpdate();
			pst.setInt(1,202);
			pst.setString(2, "dhoni");
			pst.setLong(3,89898);
			pst.setString(4, "CSK");
			pst.executeUpdate();
			pst.setInt(1,203);
			pst.setString(2, "rohith");
			pst.setLong(3,99933);
			pst.setString(4, "MI");
			pst.executeUpdate();
			System.out.println("3 records inserted succesfully");
		} catch (ClassNotFoundException | SQLException e) {
			e.printStackTrace();
		} finally {
			if (con != null) {
				try {
					con.close();
				} catch (SQLException e) {
					e.printStackTrace();
				}
			}
			if (pst != null) {
				try {
					pst.close();
				} catch (SQLException e) {
					e.printStackTrace();
				}
			}
		}
	}
}
```

Program to update a record in the person table by using preparedstatement interface

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.Scanner;

public class _13_UpdatePersonByPS {
	public static void main(String[] args) {
		Connection con = null;
		PreparedStatement pst = null;
		Scanner s = new Scanner(System.in);
		System.out.println("Enter the id to update");
		int id = s.nextInt();
		System.out.println("Enter name ,phone and password");
		String name = s.next();
		long phone = s.nextLong();
		String password = s.next();
		String update_qry = "update person set name=?,phone=?,password=? where id=?";
		try {
			Class.forName("com.mysql.cj.jdbc.Driver");
			con = DriverManager.getConnection("jdbc:mysql://localhost:3306/jdbc_demo",
"root", "charantej@@18");
			pst = con.prepareStatement(update_qry);
			pst.setString(1, name);
			pst.setLong(2, phone);
			pst.setString(3, password);
			pst.setInt(4, id);
			int r = pst.executeUpdate();
			System.out.println(r + "rows are updated");

		} catch (ClassNotFoundException | SQLException e) {
			e.printStackTrace();
		} finally {
			if (con != null) {
				try {
					con.close();
				} catch (SQLException e) {
					e.printStackTrace();
				}
			}
			if (pst != null) {
				try {
					pst.close();
				} catch (SQLException e) {
					e.printStackTrace();
				}
			}
			s.close();
		}
	}
}
```

Program to delete a record with specific id by using preparedstatement

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.Scanner;

public class _14_DeletePersonByPS {
	public static void main(String[] args) {
		Connection con = null;
		PreparedStatement pst = null;
		Scanner s = new Scanner(System.in);
		System.out.println("Enter the id to delete");
		int id = s.nextInt();
		String update_qry = "delete from person  where id=?";
		try {
			Class.forName("com.mysql.cj.jdbc.Driver");
			con = DriverManager.getConnection("jdbc:mysql://localhost:3306/jdbc_demo",
"root", "charantej@@18");
			pst = con.prepareStatement(update_qry);

			pst.setInt(1, id);
			int r = pst.executeUpdate();
			System.out.println(r + "rows are deleted");

		} catch (ClassNotFoundException | SQLException e) {
			e.printStackTrace();
		} finally {
			if (con != null) {
				try {
					con.close();
				} catch (SQLException e) {
					e.printStackTrace();
				}
			}
			if (pst != null) {
				try {
					pst.close();
				} catch (SQLException e) {
					e.printStackTrace();
				}}
			s.close();
		}}}
```

Assignment Questions

1. Delete a record from person table where name=?
2. Delete a record from person table where phoneno=?
3. Delete a record from person table where id=?and password=?
4. Delete a record from person table where phone=?and password=?
5. Delete a record from person table where name=? and password=?

1.Program to delete a record with specific name by using preparedstatment

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.Scanner;

public class _01_deleteRecordWithThisName {
public static void main(String[] args) {
	Connection con = null;
	PreparedStatement pst = null;
	Scanner s = new Scanner(System.in);
	System.out.println("Enter the name to delete");
	String name = s.next();
	String update_qry = "delete from person  where name=?";
	try {
		Class.forName("com.mysql.cj.jdbc.Driver");
		con = DriverManager.getConnection("jdbc:mysql://localhost:3306/jdbc_demo", "root",
"charantej@@18");
		pst = con.prepareStatement(update_qry);

		pst.setString(1, name);
		int r = pst.executeUpdate();
		System.out.println(r + "rows are deleted");

	} catch (ClassNotFoundException | SQLException e) {
		e.printStackTrace();
	} finally {
		if (con != null) {
			try {
				con.close();
			} catch (SQLException e) {
				e.printStackTrace();
			}
		}
		if (pst != null) {
			try {
				pst.close();
			} catch (SQLException e) {
				e.printStackTrace();
			}
		}
		s.close();
	}}}
```

2.Program to delete record with specific phoneno by using preparedstatement

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.Scanner;
public class _02_deleteRecordWithThePhoneno {
	public static void main(String[] args) {
		Connection con = null;
		PreparedStatement pst = null;
		Scanner s = new Scanner(System.in);
		System.out.println("Enter the phoneno to delete");
		Long phone = s.nextLong();
		String update_qry = "delete from person  where phone=?";
```

```
		try {
			Class.forName("com.mysql.cj.jdbc.Driver");
			con = DriverManager.getConnection("jdbc:mysql://localhost:3306/jdbc_demo",
"root", "charantej@@18");
			pst = con.prepareStatement(update_qry);

			pst.setLong(1, phone);
			int r = pst.executeUpdate();
			System.out.println(r + "rows are deleted");

		} catch (ClassNotFoundException | SQLException e) {
			e.printStackTrace();
		} finally {
			if (con != null) {
				try {
					con.close();
				} catch (SQLException e) {
					e.printStackTrace();
				}
			}
			if (pst != null) {
				try {
					pst.close();
				} catch (SQLException e) {
					e.printStackTrace();
				}
			}
			s.close();
		}}}
```

3. to delete record with specific id and password using preparedstatement

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.Scanner;

public class _03_deleteRecordwithIdAndPsswd {
	public static void main(String[] args) {
		Connection con = null;
		PreparedStatement pst = null;
		Scanner s = new Scanner(System.in);
		System.out.println("Enter the id and password to delete");
		int id = s.nextInt();
		String password = s.next();

		String update_qry = "delete from person  where id=? and password=?";
		try {
			Class.forName("com.mysql.cj.jdbc.Driver");
			con = DriverManager.getConnection("jdbc:mysql://localhost:3306/jdbc_demo",
"root", "charantej@@18");
			pst = con.prepareStatement(update_qry);

			pst.setInt(1, id);
			pst.setString(2, password);
			int r = pst.executeUpdate();
			System.out.println(r + "rows are deleted");

		} catch (ClassNotFoundException | SQLException e) {
			e.printStackTrace();
		} finally {
```

```
                if (con != null) {
                        try {
                                con.close();
                        } catch (SQLException e) {
                                e.printStackTrace();
                        }
                }
                if (pst != null) {
                        try {
                                pst.close();
                        } catch (SQLException e) {
                                e.printStackTrace();
                        }
                }
                s.close();
        }
    }
}
```

4.Program to delete record with specific phoneno and password using preparedstatement

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.Scanner;

public class _04_deleteRecordwithphonenoAndPsswd {
    public static void main(String[] args) {
        Connection con = null;
        PreparedStatement pst = null;
        Scanner s = new Scanner(System.in);
        System.out.println("Enter the phoneno and password to delete");
        Long phone = s.nextLong();
        String password = s.next();

        String update_qry = "delete from person  where phone=? and password=?";
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
        con = DriverManager.getConnection("jdbc:mysql://localhost:3306/jdbc_demo", "root",
"charantej@@18");
            pst = con.prepareStatement(update_qry);

            pst.setLong(1, phone);
            pst.setString(2, password);
            int r = pst.executeUpdate();
            System.out.println(r + "rows are deleted");

        } catch (ClassNotFoundException | SQLException e) {
            e.printStackTrace();
        } finally {
            if (con != null) {
                try {
                    con.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }
            if (pst != null) {
                try {
                    pst.close();
                } catch (SQLException e) {
```

```
                                e.printStackTrace();
                        }
                }
                s.close();}}}
```

5.Program to delete record with specified name and password using preparedstatement

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.Scanner;

public class _05_deleteRecordwithNameAndPsswd {
	public static void main(String[] args) {
		Connection con = null;
		PreparedStatement pst = null;
		Scanner s = new Scanner(System.in);
		System.out.println("Enter the name and password to delete");
		String  name = s.next();
		String password = s.next();

		String update_qry = "delete from person  where name=? and password=?";
		try {
			Class.forName("com.mysql.cj.jdbc.Driver");
			con = DriverManager.getConnection("jdbc:mysql://localhost:3306/jdbc_demo",
"root", "charantej@@18");
			pst = con.prepareStatement(update_qry);

			pst.setString(1, name);
			pst.setString(2, password);
			int r = pst.executeUpdate();
			System.out.println(r + " row is deleted");

		} catch (ClassNotFoundException | SQLException e) {
			e.printStackTrace();
		} finally {
			if (con != null) {
				try {
					con.close();
				} catch (SQLException e) {
					e.printStackTrace();
				}
			}
			if (pst != null) {
				try {
					pst.close();
				} catch (SQLException e) {
					e.printStackTrace();
				}
			}
			s.close();
		}
	}}
```

Code to fetch the record from person table where id is placeholder

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
```

```java
import java.util.Scanner;

public class _15_FetchRecordsByPS {
	public static void main(String[] args) {
		Connection con = null;
		PreparedStatement pst = null;
		ResultSet rs = null;
		String query = "select * from person where id=?";
		Scanner s = new Scanner(System.in);
		System.out.println("Enter id  to fetch :");
		int id = s.nextInt();
		try {
			Class.forName("com.mysql.cj.jdbc.Driver");
	con = DriverManager.getConnection("jdbc:mysql://localhost:3306/jdbc_demo", "root",
"charantej@@18");
			pst = con.prepareStatement(query);
			pst.setInt(1, id);
			rs=pst.executeQuery();
			while (rs.next()) {
				System.out.println("Id : " + rs.getInt(1));
				System.out.println("Name :" + rs.getString(2));
				System.out.println("Phoneno : " + rs.getLong(3));
				System.out.println("Password : " + rs.getString(4));
			}
		} catch (ClassNotFoundException | SQLException e) {

			e.printStackTrace();
		} finally {
			if (con != null) {
				try {
					con.close();
				} catch (SQLException e) {
					// TODO Auto-generated catch block
					e.printStackTrace();
				}
			}
			if (pst != null) {
				try {
					pst.close();
				} catch (SQLException e) {
					// TODO Auto-generated catch block
					e.printStackTrace();
				}
			}
			if (rs != null) {
				try {
					rs.close();
				} catch (SQLException e) {
					// TODO Auto-generated catch block
					e.printStackTrace();
				}

			}
		}}}
```

Jdbc code for login validation by phoneno and password

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Scanner;
public class _16_LoginValidationByUsingPhonenoAndPsswd {
	public static void main(String[] args) {
		Connection con = null;
		PreparedStatement pst = null;
		ResultSet rs = null;
		Scanner s = new Scanner(System.in);
		System.out.println("Enter the Phoneno and password: ");
		long phone = s.nextLong();
		String password = s.next();
		String query = "select * from person where phone=? and password=?";
		try {
			Class.forName("com.mysql.cj.jdbc.Driver");
	con = DriverManager.getConnection("jdbc:mysql://localhost:3306/jdbc_demo", "root",
"charantej@@18");
			pst = con.prepareStatement(query);
			pst.setLong(1, phone);
			pst.setString(2, password);
			rs = pst.executeQuery();
			if (rs.next()) {
				System.out.println("person Verified Successfully");
				System.out.println("Id : " + rs.getInt(1));
				System.out.println("Name :" + rs.getString(2));
				System.out.println("Phoneno : " + rs.getLong(3));
				System.out.println("Password : " + rs.getString(4));
			} else {
				System.err.println("Invalid phoneno or password");
			}
		} catch (ClassNotFoundException | SQLException e) {
			e.printStackTrace();
		} finally {
			if (con != null) {
				try {
					con.close();
				} catch (SQLException e) {
					// TODO Auto-generated catch block
					e.printStackTrace();
				}}
			if (pst != null) {
				try {
					pst.close();
				} catch (SQLException e) {
					// TODO Auto-generated catch block
					e.printStackTrace();
				}}
			if (rs != null) {
				try {
					rs.close();
				} catch (SQLException e) {
					// TODO Auto-generated catch block
					e.printStackTrace();
				}}}
		s.close();
	}}
```

Assignment questions based on fetching the data from table

1. Fetch the record from person where name=?
2. Fetch the record from person table where phoneno=?
3. Fetch names from person table
4. Fetch phone no's from person table
5. Fetch id's from person table
6. Verify the person id and password

1.Code to fetch the record from person table where name=?

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Scanner;
public class _06_fetchrecordWhereName {
	public static void main(String[] args) {
		Connection con = null;
		PreparedStatement pst = null;
		ResultSet rs = null;
		String query="select * from person where name=?";
		Scanner s = new Scanner(System.in);
		System.out.println("Enter the name to fetch the records:");
		String name = s.next();
		try {
			Class.forName("com.mysql.cj.jdbc.Driver");
	con=DriverManager.getConnection("jdbc:mysql://localhost:3306/jdbc_demo", "root",
"charantej@@18");
			pst=con.prepareStatement(query);
			pst.setString(1, name);
			rs=pst.executeQuery();
			while(rs.next()) {
				System.out.println("Id : " + rs.getInt(1));
				System.out.println("Name :" + rs.getString(2));
				System.out.println("Phoneno : " + rs.getLong(3));
				System.out.println("Password : " + rs.getString(4));
			}} catch (ClassNotFoundException|SQLException e) {
			e.printStackTrace();}
		finally {
			if (con != null) {
				try {
					con.close();
				} catch (SQLException e) {
					e.printStackTrace();
				}}
			if (pst != null) {
				try {
					pst.close();
				} catch (SQLException e) {
					e.printStackTrace();}}
			if (rs != null) {
				try {
					rs.close();
```

```
                        } catch (SQLException e) {
                                e.printStackTrace();}}}
                s.close();
        }}
```

2.Code to fetch the record from person table where name=?

```
public class _07_FetchtheRecordWherePhoneno {
        public static void main(String[] args) {
                Connection con = null;
                PreparedStatement pst = null;
                ResultSet rs = null;
                String query = "select * from person where phone=?";
                Scanner s = new Scanner(System.in);
                System.out.println("Enter the name to fetch the records:");
                long phone = s.nextLong();
                try {
                        Class.forName("com.mysql.cj.jdbc.Driver");
        con = DriverManager.getConnection("jdbc:mysql://localhost:3306/jdbc_demo", "root",
"charantej@@18");
                        pst = con.prepareStatement(query);
                        pst.setLong(1, phone);
                        rs = pst.executeQuery();
                        while (rs.next()) {
                                System.out.println("Id : " + rs.getInt(1));
                                System.out.println("Name :" + rs.getString(2));
                                System.out.println("Phoneno : " + rs.getLong(3));
                                System.out.println("Password : " + rs.getString(4));
                        }
                } catch (ClassNotFoundException | SQLException e) {
                                        e.printStackTrace();
                } finally {
                        if (con != null) {
                                try {
                                        con.close();
                                } catch (SQLException e) {
                                        e.printStackTrace();
                                }
                        }
                        if (pst != null) {
                                try {
                                        pst.close();
                                } catch (SQLException e) {
                                        e.printStackTrace();
                                }
                        }
                        if (rs != null) {
                                try {
                                        rs.close();
                                } catch (SQLException e) {
                                        e.printStackTrace();
                                }
                        }
                }
                s.close();
        }
}
```

3.Code to fetch the names from person table

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Scanner;

public class _08_fetchTheNames {
	public static void main(String[] args) {
		Connection con = null;
		PreparedStatement pst = null;
		ResultSet rs = null;
		String query = "select name from person";
		try {
			Class.forName("com.mysql.cj.jdbc.Driver");
		con = DriverManager.getConnection("jdbc:mysql://localhost:3306/jdbc_demo", "root",
"charantej@@18");
			pst = con.prepareStatement(query);
			rs = pst.executeQuery();
			while (rs.next()) {
				System.out.println("Name :" + rs.getString(1));
			}
		} catch (ClassNotFoundException | SQLException e) {
			e.printStackTrace();
		} finally {
			if (con != null) {
				try {
					con.close();
				} catch (SQLException e) {
					e.printStackTrace();
				}
			}
			if (pst != null) {
				try {
					pst.close();
				} catch (SQLException e) {
					e.printStackTrace();
				}}
			if (rs != null) {
				try {
					rs.close();
				} catch (SQLException e) {
					e.printStackTrace();
				}}}}
```

4.Code to fetch the phone no's from person table

```
public class _09_FetchThePhonenos {
	public static void main(String[] args) {
		Connection con = null;
		PreparedStatement pst = null;
		ResultSet rs = null;
		String query = "select phone from person";
		try {
			Class.forName("com.mysql.cj.jdbc.Driver");
			con = DriverManager.getConnection("jdbc:mysql://localhost:3306/jdbc_demo",
"root", "charantej@@18");
```

```
                pst = con.prepareStatement(query);
                rs = pst.executeQuery();
                while (rs.next()) {
                        System.out.println("phoneno :" + rs.getLong(1));
                }
        } catch (ClassNotFoundException | SQLException e) {
                e.printStackTrace();
        } finally {
                if (con != null) {
                        try {
                                con.close();
                        } catch (SQLException e) {
                                e.printStackTrace();
                        }
                }
                if (pst != null) {
                        try {
                                pst.close();
                        } catch (SQLException e) {
                                e.printStackTrace();
                        }
                }
                if (rs != null) {
                        try {
                                rs.close();
                        } catch (SQLException e) {
                                e.printStackTrace();
                        }
                }}}}
```

5.Code to fetch the id's from person table

```
public class _10_FetchTheIds {
        public static void main(String[] args) {
                Connection con = null;
                PreparedStatement pst = null;
                ResultSet rs = null;
                String query = "select id from person";
                try {
                        Class.forName("com.mysql.cj.jdbc.Driver");
                        con = DriverManager.getConnection("jdbc:mysql://localhost:3306/jdbc_demo",
"root", "charantej@@18");
                        pst = con.prepareStatement(query);
                        rs = pst.executeQuery();
                        while (rs.next()) {
                                System.out.println("Idno's :" + rs.getInt(1));
                        }
                } catch (ClassNotFoundException | SQLException e) {
                        e.printStackTrace();
                } finally {
                        if (con != null) {
                                try {
                                        con.close();
                                } catch (SQLException e) {
                                        e.printStackTrace();
                                }
                        }
                        if (pst != null) {
                                try {
                                        pst.close();
                                } catch (SQLException e) {
                                        e.printStackTrace();
```

```
                        }}
                  if (rs != null) {
                        try {
                              rs.close();
                        } catch (SQLException e) {
                              e.printStackTrace();
}     }}}}
```

6.Code to Verify the person id and password

```
public class _11_VerifyThePersonIdAndPassword {
      public static void main(String[] args) {
            Connection con = null;
            PreparedStatement pst = null;
            ResultSet rs = null;
            Scanner s = new Scanner(System.in);
            System.out.println("Enter the id and password: ");
            long id = s.nextInt();
            String password = s.next();
            String query = "select * from person where id=? and password=?";
            try {
                  Class.forName("com.mysql.cj.jdbc.Driver");
                  con = DriverManager.getConnection("jdbc:mysql://localhost:3306/jdbc_demo",
"root", "charantej@@18");
                  pst = con.prepareStatement(query);
                  pst.setLong(1, id);
                  pst.setString(2, password);
                  rs = pst.executeQuery();
                  if (rs.next()) {
                        System.out.println("person Verified Successfully");
                        System.out.println("Id : " + rs.getInt(1));
                        System.out.println("Name :" + rs.getString(2));
                        System.out.println("Phoneno : " + rs.getLong(3));
                        System.out.println("Password : " + rs.getString(4));
                  } else {
                        System.err.println("Invalid phoneno or password");
                  }

            } catch (ClassNotFoundException | SQLException e) {

                  e.printStackTrace();
            } finally {
                  if (con != null) {
                        try {
                              con.close();
                        } catch (SQLException e) {
                              // TODO Auto-generated catch block
                              e.printStackTrace();
                        }
                  }
                  if (pst != null) {
                        try {
                              pst.close();
                        } catch (SQLException e) {
                              // TODO Auto-generated catch block
                              e.printStackTrace();
                        }
                  }
                  if (rs != null) {
                        try {
                              rs.close();
                        } catch (SQLException e) {
```

```
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                            }

                        }
                    }
                    s.close();
                }}
```

Stored procedures

- A group of SQL statements that perform a particular task.
- Normally created by the DBA.
- Created in a SQL language supported by the native database.
- Can have any combination of input, output and input/output parameters.

Using Callable Statement

- To call stored procedures from java →the JDBC API provides the callable statement

- **CallableStatement myCall=myConn.prepareCall("{call some_stored_proc()}");**
- **MyCall.execute();**

Parameters

- The JDBC API supports different parameters.
  - IN(default)
  - INOUT
  - OUT
- The stored procedure can also return result sets.
- Following code examples will show how to register parameter types and values.

IN Parameters

- **Our DBA created a stored procedure**
  - **Developed for MySQL**

```
PROCEDURE `increase_salaries_for_department`(
          IN the_department VARCHAR(64), IN increase_amount DECIMAL(10,2))

BEGIN
     UPDATE employees SET salary= salary + increase_amount
     WHERE department=the_department;
END
```

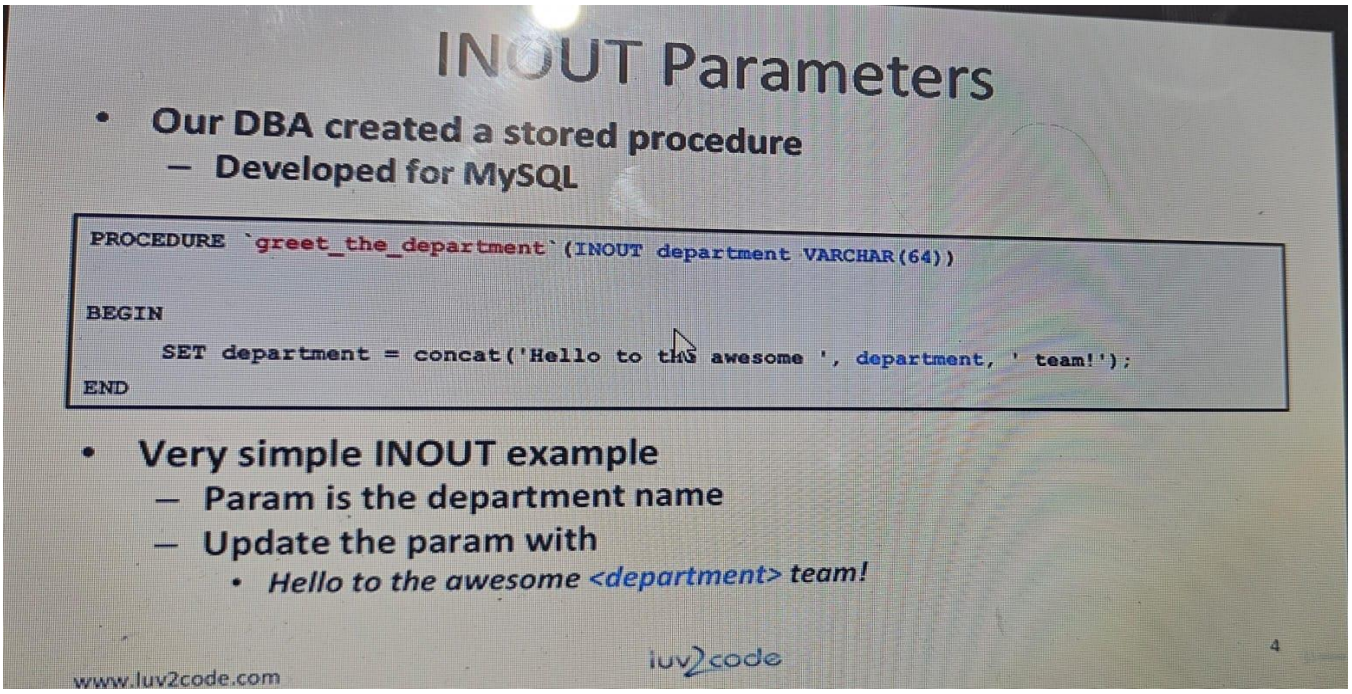
- **Increase salary for everyone in a department ☺**
  - **First param is the department name**
  - **Second param is the increase amount**

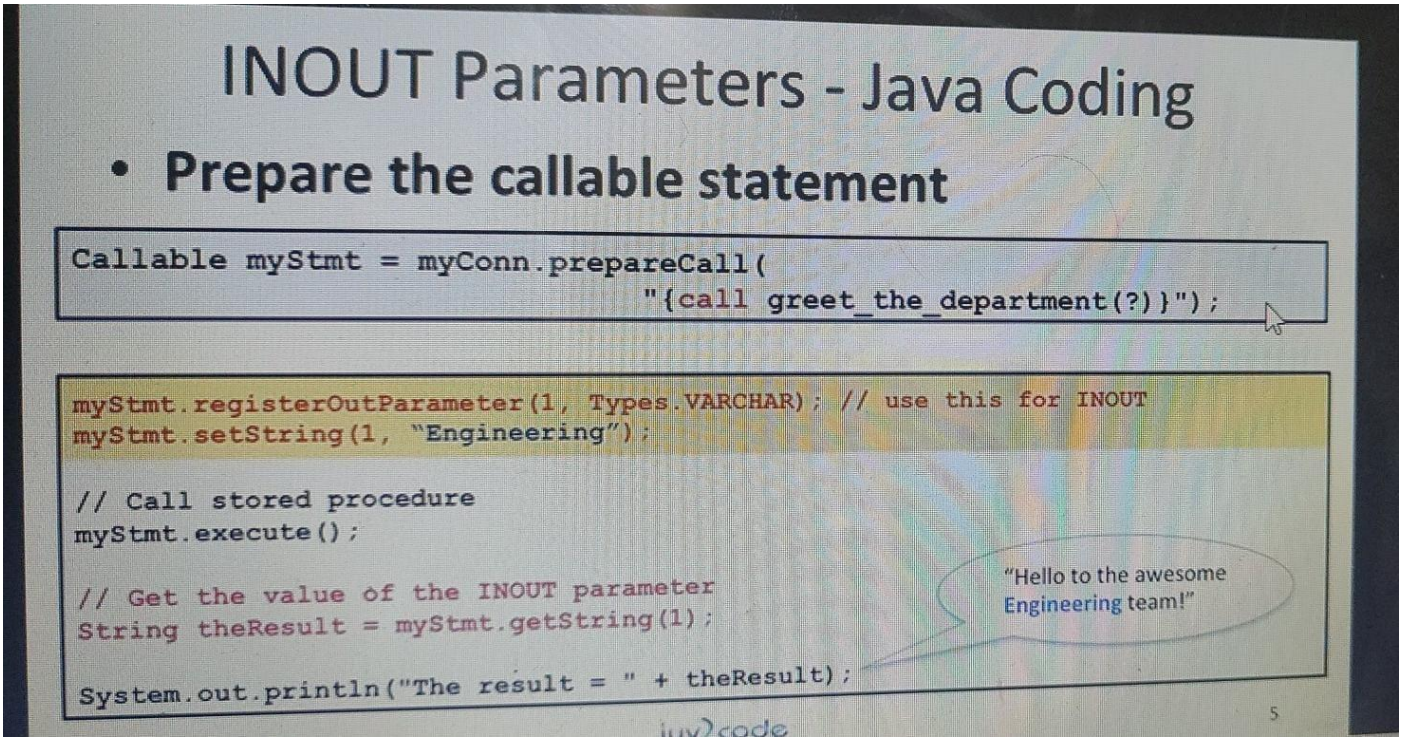
IN Parameters - Java Coding

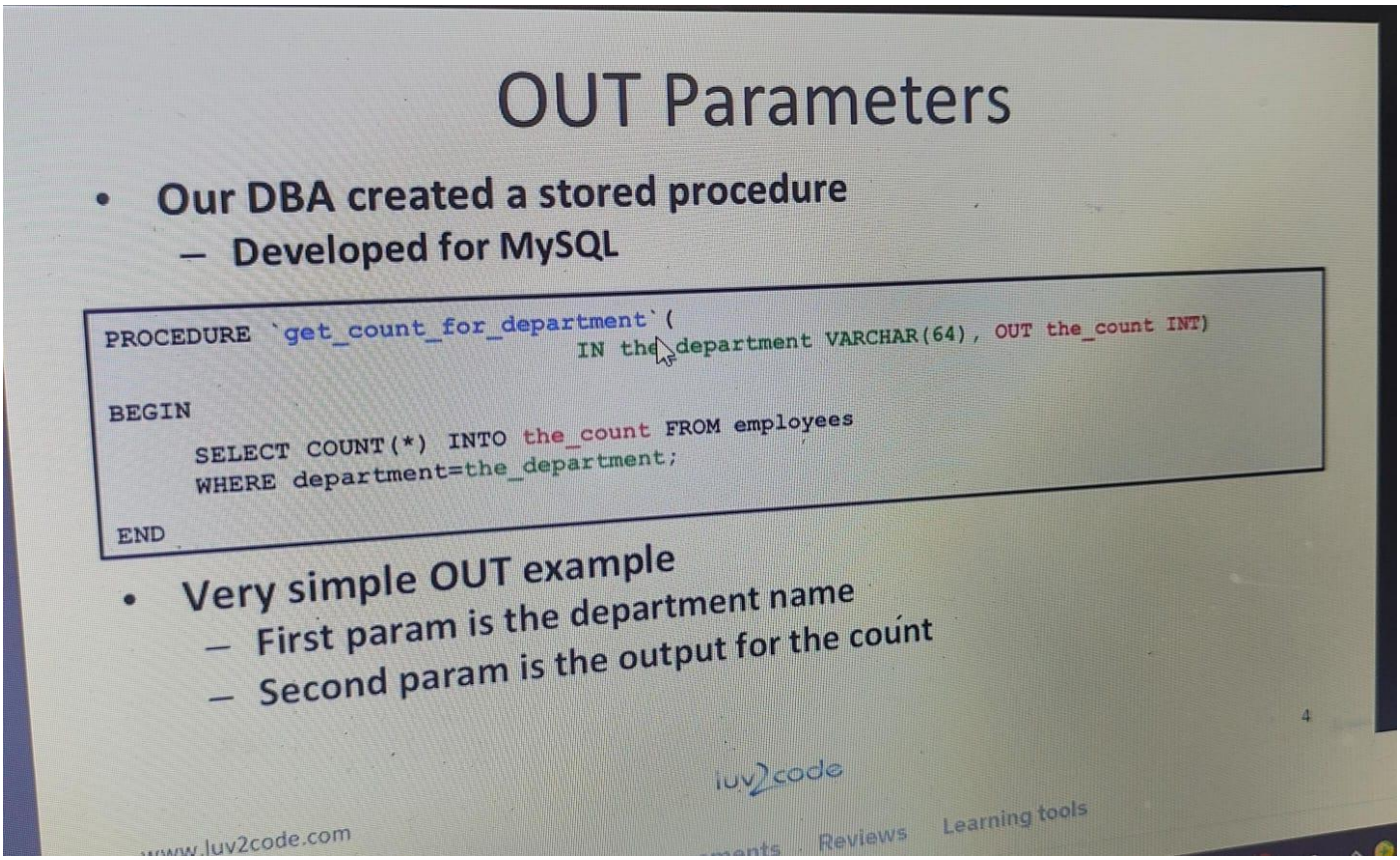
- **Prepare the callable statement**

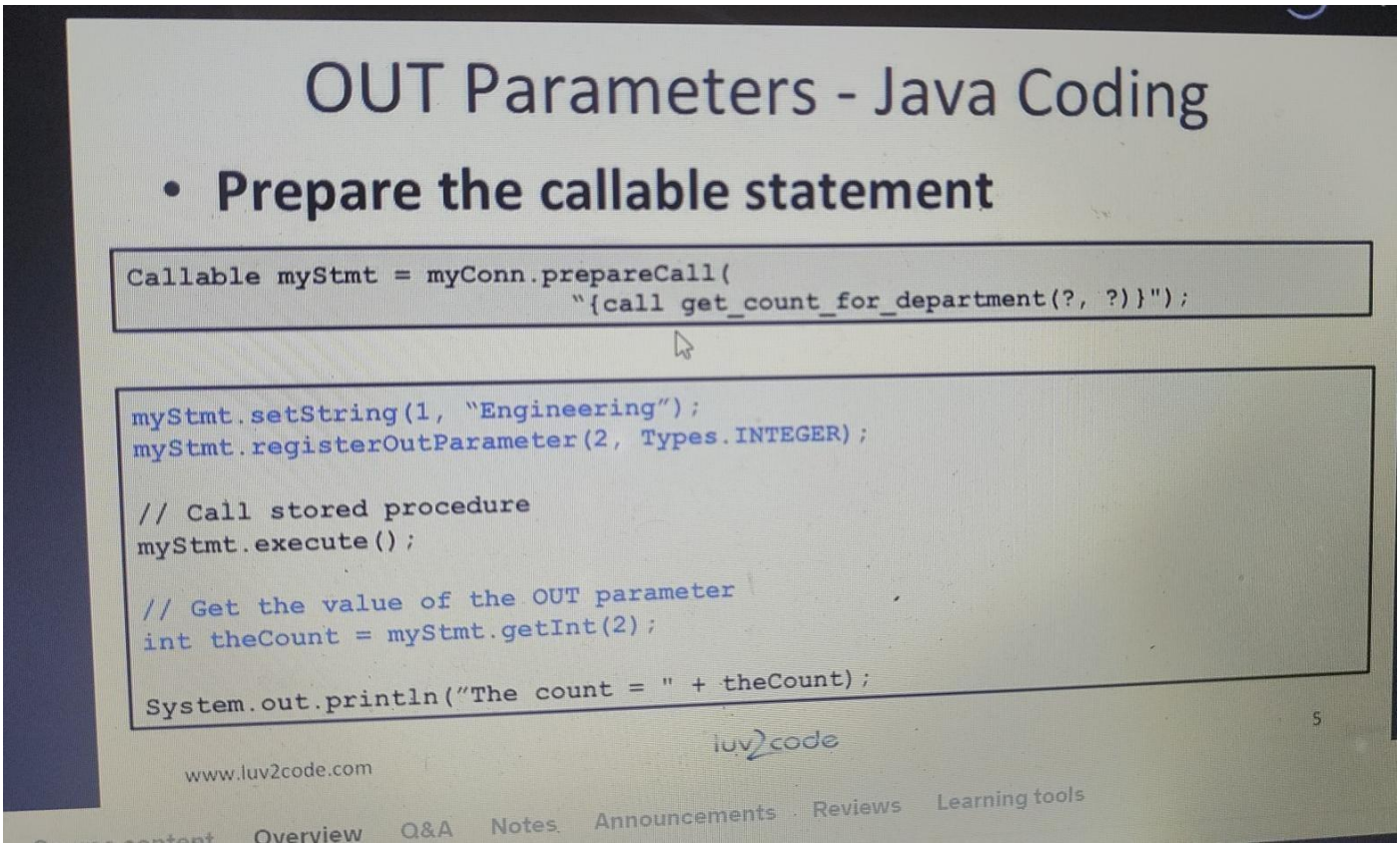
```
Callable myStmt = myConn.prepareCall(
          "{call increase_salaries_for_department(?, ?)}");
```

```
myStmt.setString(1, "Engineering");
myStmt.setDouble(2, 10000);

// now execute the statement
myStmt.execute();
```


INOUT Parameters
• Our DBA created a stored procedure
– Developed for MySQL
PROCEDURE `greet_the_department`(INOUT department VARCHAR(64))
BEGIN
SET department = concat('Hello to the awesome ', department, ' team!');
END
• Very simple INOUT example
– Param is the department name
– Update the param with
• Hello to the awesome <department> team!
www.luv2code.com
luv2code
4


INOUT Parameters - Java Coding
• Prepare the callable statement
Callable myStmt = myConn.prepareCall(
"{call greet_the_department(?)}");
myStmt.registerOutParameter(1, Types.VARCHAR); // use this for INOUT
myStmt.setString(1, "Engineering");
// Call stored procedure
myStmt.execute();
// Get the value of the INOUT parameter
String theResult = myStmt.getString(1);
System.out.println("The result = " + theResult);
"Hello to the awesome Engineering team!"
luv2code
5


OUT Parameters
Our DBA created a stored procedure
Developed for MySQL
PROCEDURE `get_count_for_department`(
IN the_department VARCHAR(64), OUT the_count INT)
BEGIN
SELECT COUNT(*) INTO the_count FROM employees
WHERE department=the_department;
END
Very simple OUT example
First param is the department name
Second param is the output for the count
4
luv2code
www.luv2code.com
Reviews
Learning tools


OUT Parameters - Java Coding
Prepare the callable statement
Callable myStmt = myConn.prepareCall(
"{call get_count_for_department(?, ?)}");
myStmt.setString(1, "Engineering");
myStmt.registerOutParameter(2, Types.INTEGER);
// Call stored procedure
myStmt.execute();
// Get the value of the OUT parameter
int theCount = myStmt.getInt(2);
System.out.println("The count = " + theCount);
5
luv2code
www.luv2code.com
Overview
Q&A
Notes
Announcements
Reviews
Learning tools

Stored Procedure – Return Result Set

- **Our DBA created a stored procedure**
  - **Developed for MySQL**

```
PROCEDURE `get_employees_for_department`(IN the_department VARCHAR(64))

BEGIN
    SELECT * from employees where department=the_department;
END
```

- **Stored procedure returns a result set / cursor**
  - **First param is the department name**

Result Set - Java Coding

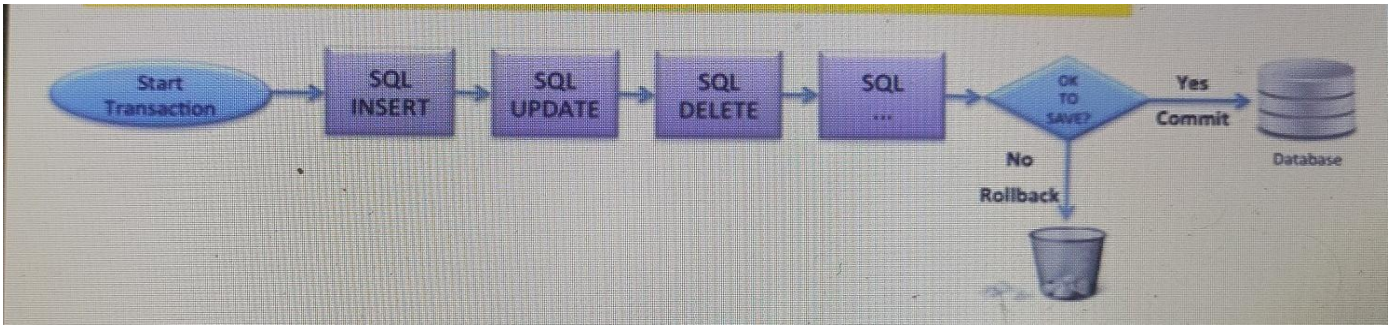
- **Prepare the callable statement**

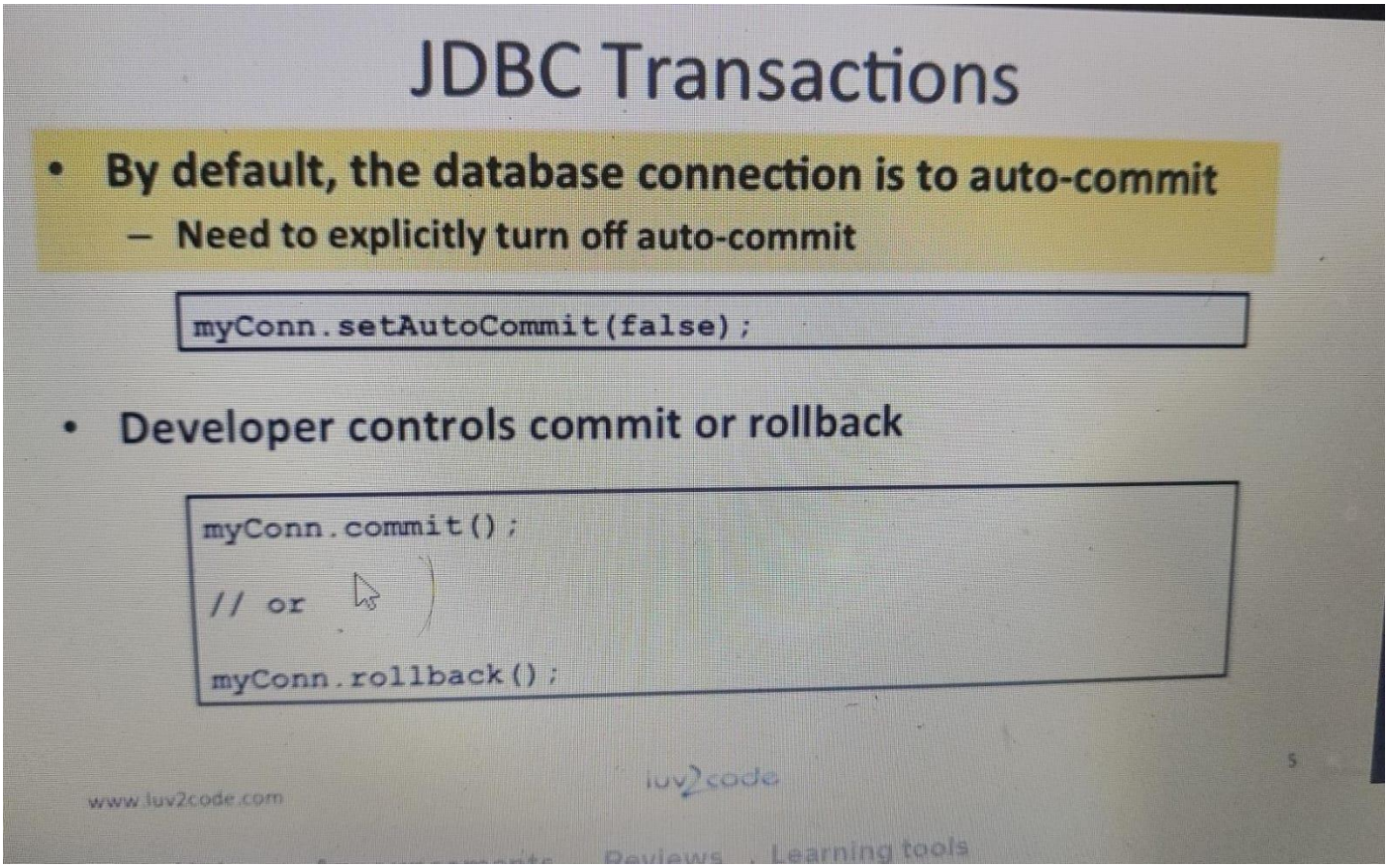
```
Callable myStmt = myConn.prepareCall(
                    "{call get_employees_for_department (?)}");
```

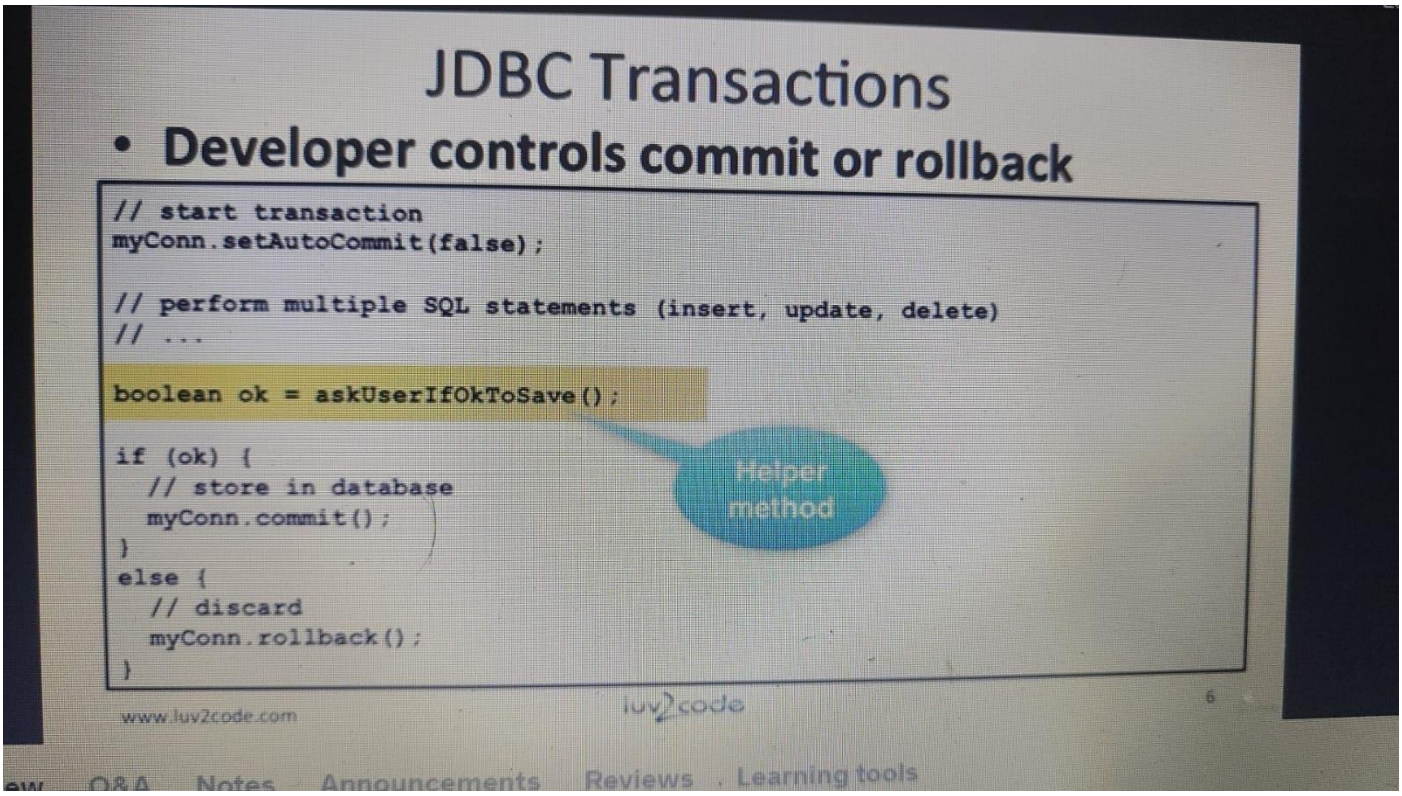
```
myStmt.setString(1, "Engineering");

// Call stored procedure
myStmt.execute();

// Get the result set
myRs = myStmt.getResultSet();

// Display the result set
...
```

Transactions

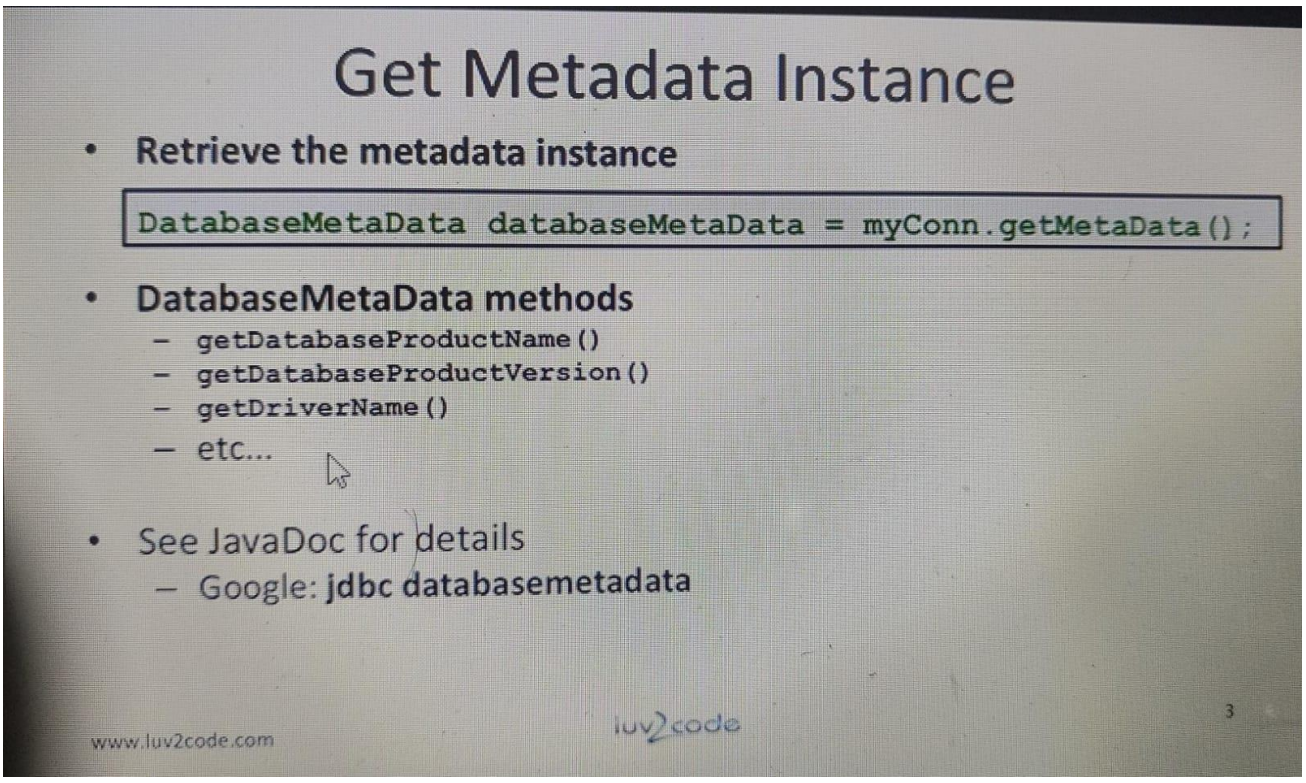
- A transaction is a unit of work
- One or more SQL statements executed together .
  - Either all of the statements are executed→**commit**
  - Or none of the statements are executed →**Rollback**







Accessing database metadata



Database scheme

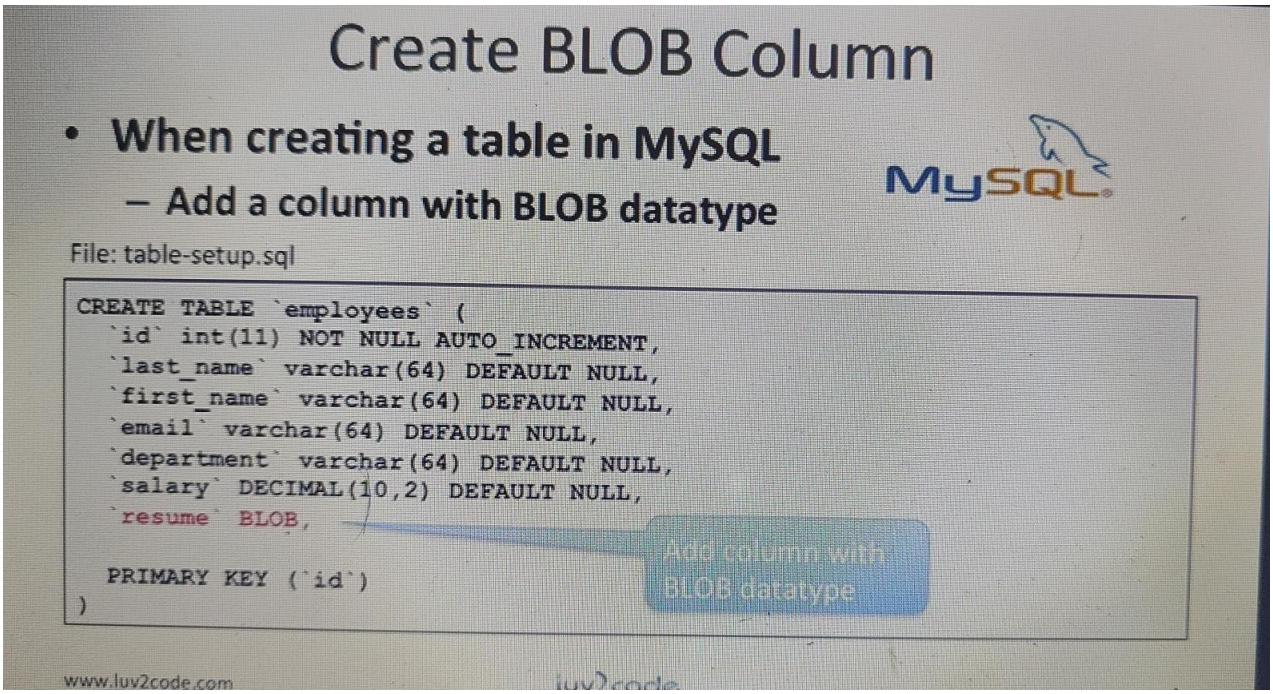
- Use Databasemetadata to get db schema
- List of tables
- List of columns

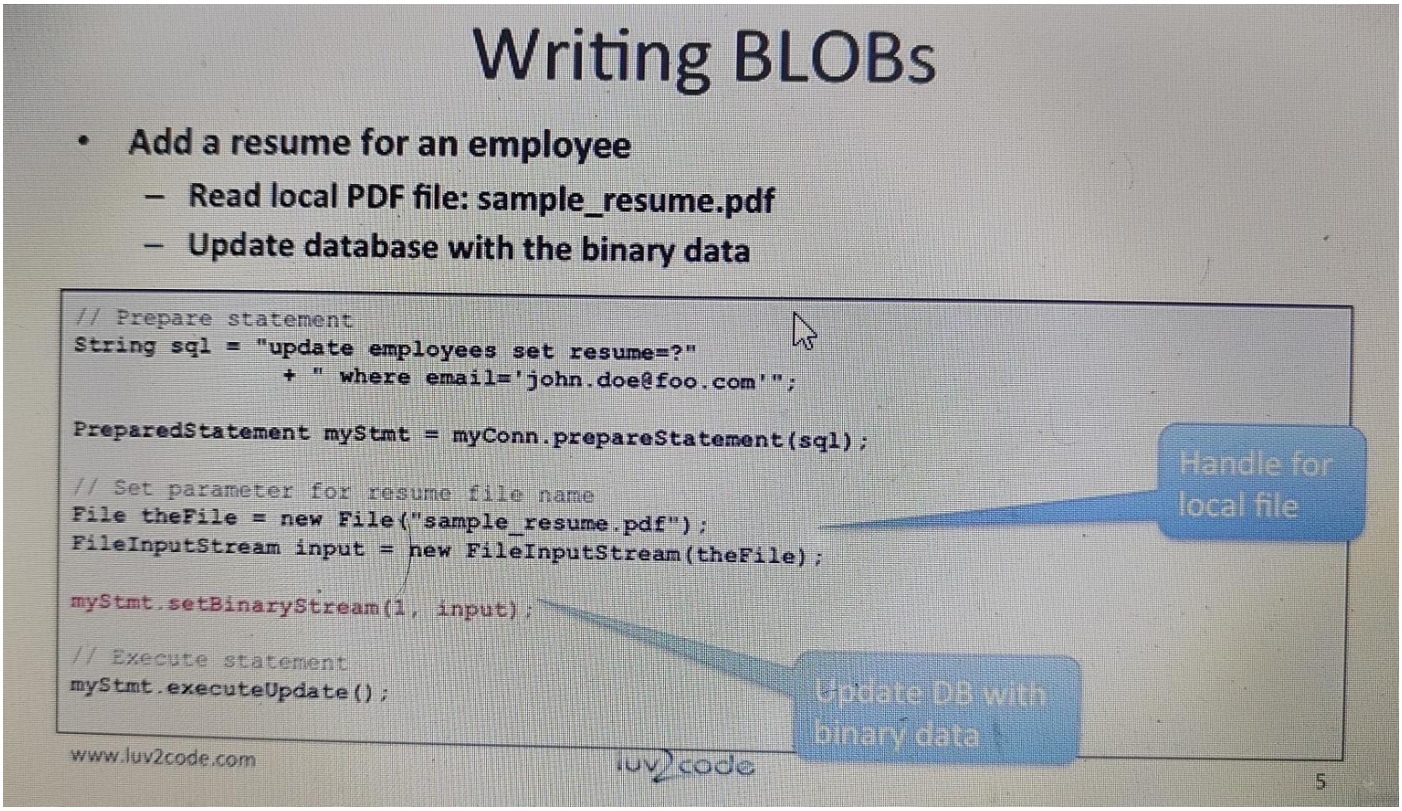
What is BLOB?

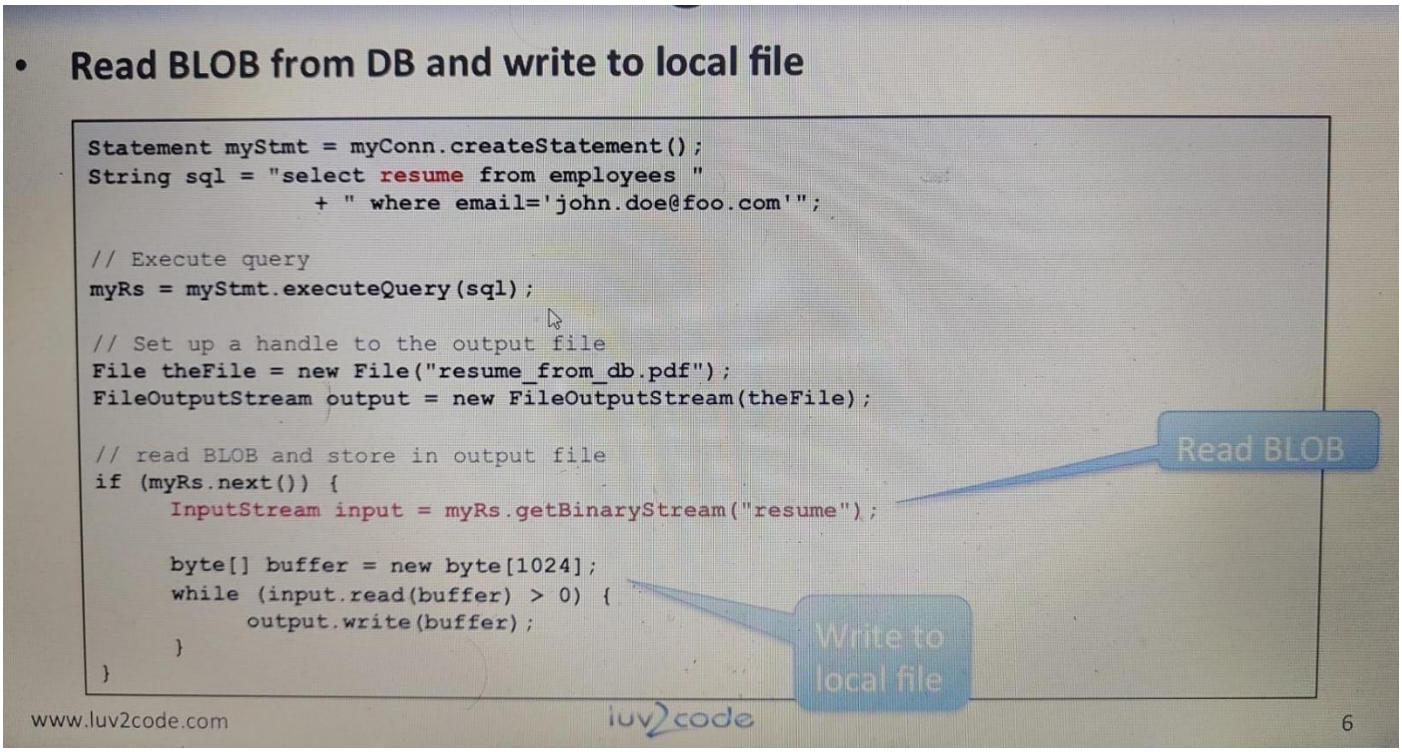
- A BLOB (binary large object) is a collection of binary data stored as a single entity in database.
- BLOB's are typically documents, images ,audio or other binary objects.

NOTE:

Database support for BLOB's is not Universal.



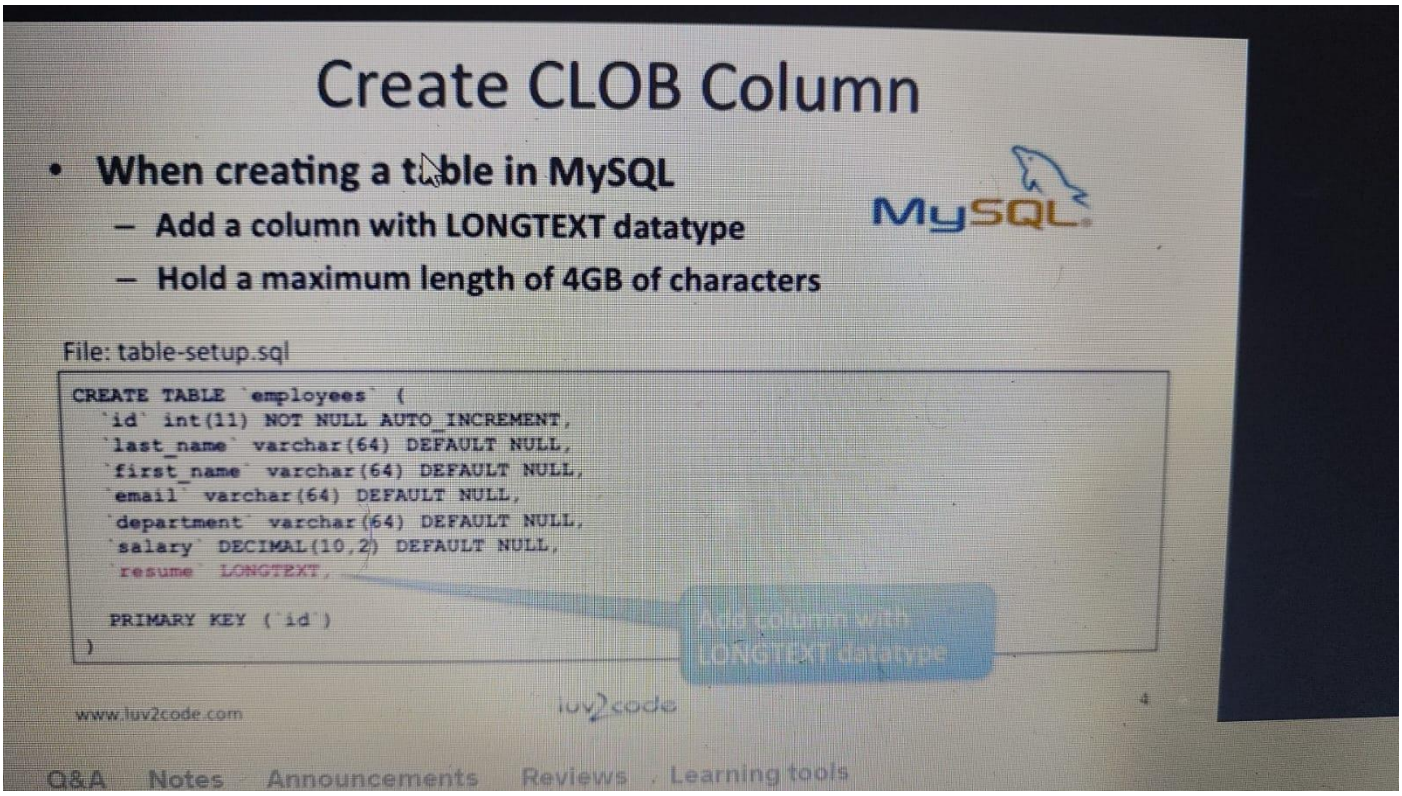


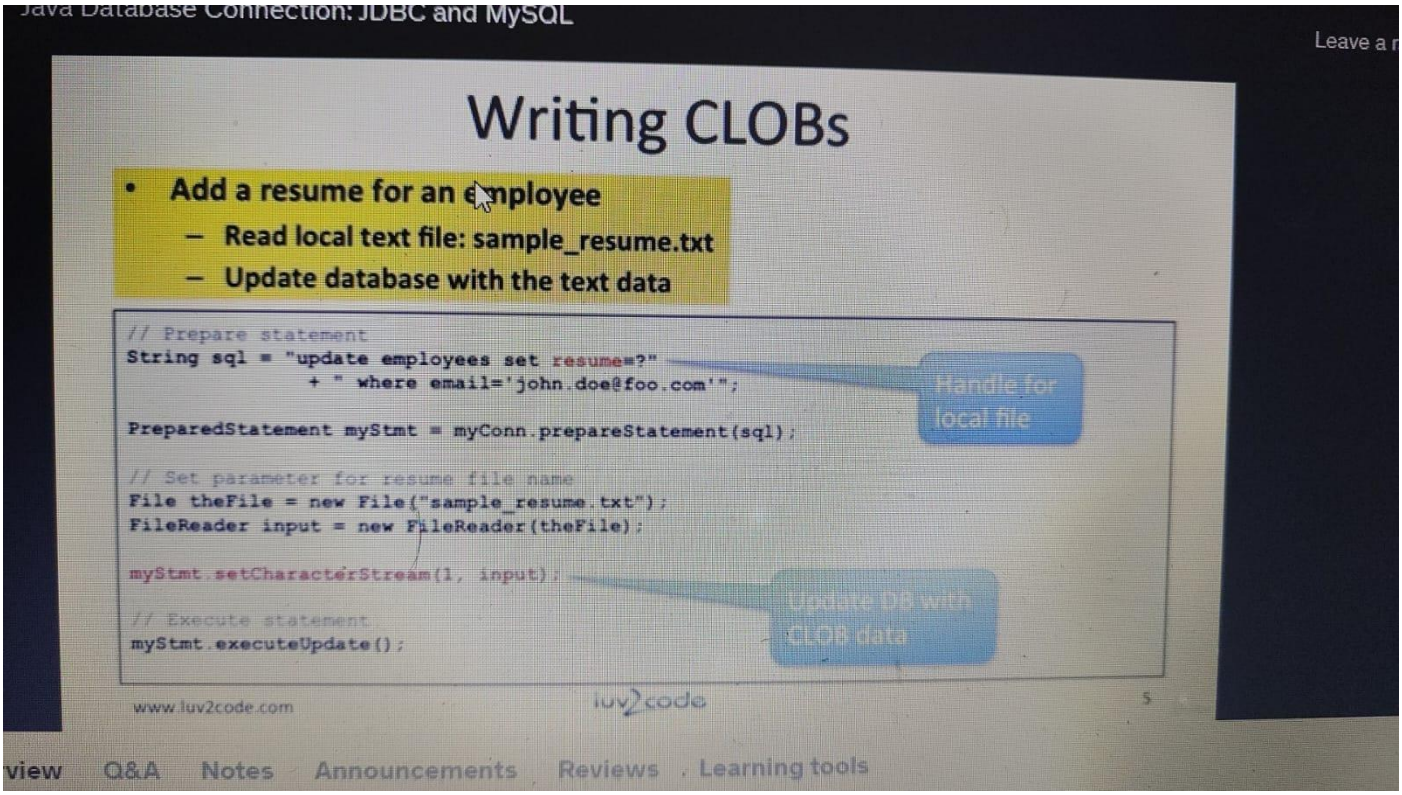


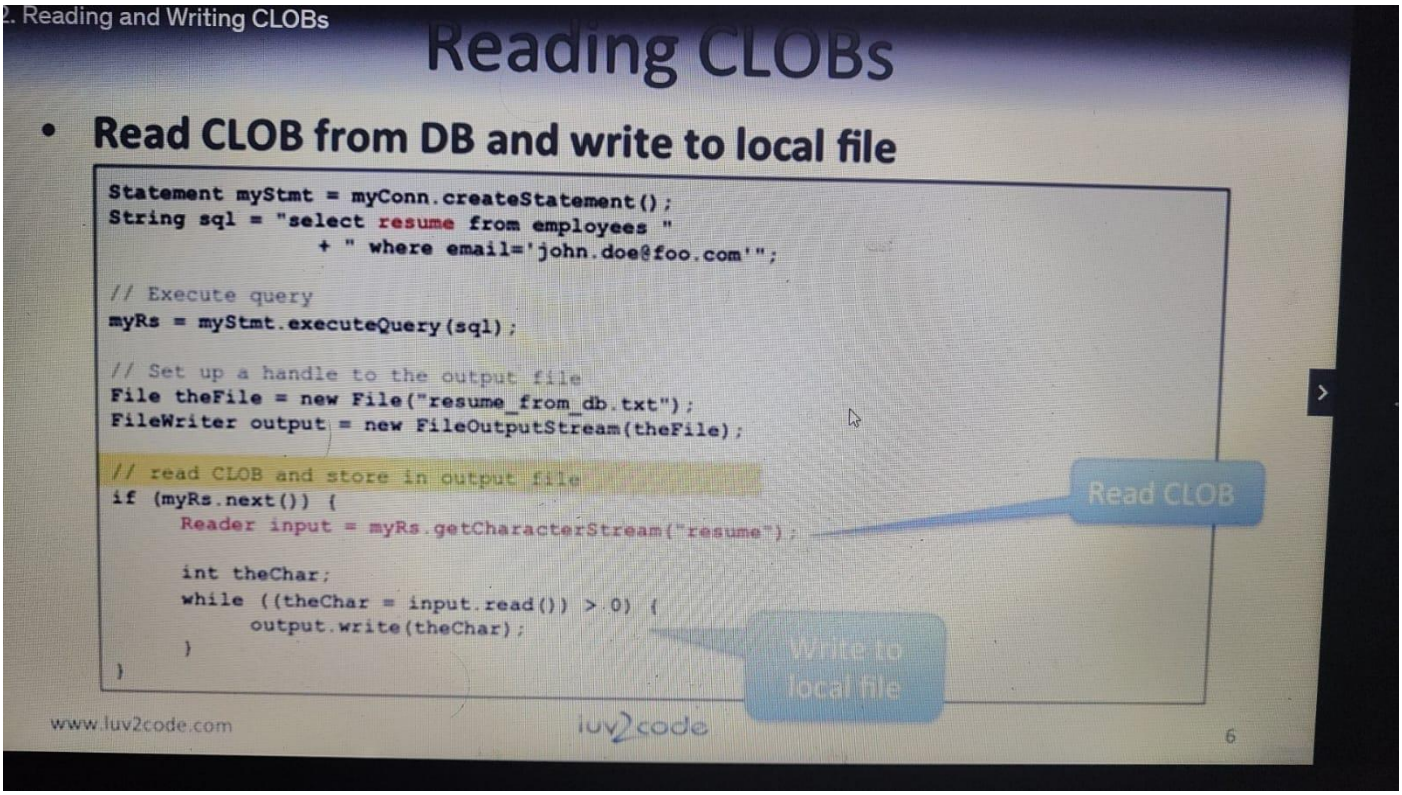
What is CLOB?

- A CLOB (character large object) is a collection of character data stored as a single entity a database .
- CLOB's are typically used to store large text documents(plain text of XML)

NOTE Database support for CLOB's is not Uniiversal




Java Database Connection: JDBC and MySQL
Leave a r
Writing CLOBs
• Add a resume for an employee
– Read local text file: sample_resume.txt
– Update database with the text data
// Prepare statement
String sql = "update employees set resume=?"
+ " where email='john.doe@foo.com'";
PreparedStatement myStmt = myConn.prepareStatement(sql);
// Set parameter for resume file name
File theFile = new File("sample_resume.txt");
FileReader input = new FileReader(theFile);
myStmt.setCharacterStream(1, input);
// Execute statement
myStmt.executeUpdate();
Handle for local file
Update DB with CLOB data
www.luv2code.com
luv2code
5
view Q&A Notes Announcements Reviews Learning tools


2. Reading and Writing CLOBs
Reading CLOBs
• Read CLOB from DB and write to local file
Statement myStmt = myConn.createStatement();
String sql = "select resume from employees "
+ " where email='john.doe@foo.com'";
// Execute query
myRs = myStmt.executeQuery(sql);
// Set up a handle to the output file
File theFile = new File("resume_from_db.txt");
FileWriter output = new FileOutputStream(theFile);
// read CLOB and store in output file
if (myRs.next()) {
Reader input = myRs.getCharacterStream("resume");
int theChar;
while ((theChar = input.read()) > 0) {
output.write(theChar);
}
}
Read CLOB
Write to local file
www.luv2code.com
luv2code
6

Project on Employee Management Application using JDBC

```
package com.employeeApp;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DbConnection {
	static Connection con;
	public static Connection createDBConnection() {
		String url = "jdbc:mysql://localhost:3306/employee_db";
		String user = "root";
		String psswd = "charantej@@18";
		try {
			Class.forName("com.mysql.cj.jdbc.Driver");
			con = DriverManager.getConnection(url, user, psswd);
		} catch (ClassNotFoundException | SQLException e) {

			e.printStackTrace();
		}
		return con;
	}
}
```

Employee.java

```
package com.employeeApp;

public class Employee {
	private int id;
	private String name;
	private double salary;
	private int age;
	private String desg;
	public Employee() {
		super();
		this.id = id;
		this.name = name;
		this.salary = salary;
		this.age = age;
		this.desg = desg;
	}
	@Override
	public String toString() {
		return "Employee [id=" + id + ", name=" + name + ", salary=" + salary + ", age=" +
age + ", desg=" + desg + "]";
	}
	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public double getSalary() {
```

```
		return salary;
	}
	public void setSalary(double salary) {
		this.salary = salary;
	}
	public int getAge() {
		return age;
	}
	public void setAge(int age) {
		this.age = age;
	}
	public String getDesg() {
		return desg;
	}
	public void setDesg(String desg) {
		this.desg = desg;
	}
}
```

Employee Interface

```
package com.employeeApp;

public interface EmployeeInterface {

	public void createEmployee(Employee emp);

	public void DisplayAllEmployee();

	public void showEmployeeBasedOnID(int id);

	public void updateEmployee(int id, String name);

	public void deleteEmployee(int id);
}
```

EmployeeImplementation

```
package com.employeeApp;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Scanner;

public class EmployeeImpl implements EmployeeInterface {
	Connection con = null;
	PreparedStatement pst = null;
	ResultSet rs = null;

	@Override
	public void createEmployee(Employee emp) {
		String query = "insert into emp values(?,?,?,?,?)";
		try {
			con = DbConnection.createDBConnection();
			pst = con.prepareStatement(query);
			pst.setInt(1, emp.getId());
			pst.setString(2, emp.getName());
			pst.setDouble(3, emp.getSalary());
			pst.setInt(4, emp.getAge());
			pst.setString(5, emp.getDesg());
```

```
                    pst.executeUpdate();
                    System.out.println("Employee data inserted Successfully");
              } catch (SQLException e) {
                    e.printStackTrace();
              }
       }

       @Override
       public void DisplayAllEmployee() {
              String query = "select * from emp";
              System.out.println("Employee Details :");
              System.out.println("-----------------------------------------");
              try {
                    con = DbConnection.createDBConnection();
                    pst = con.prepareStatement(query);
                    rs = pst.executeQuery();
                    while (rs.next()) {
                          System.out.println("Employeeid :" + rs.getInt(1) + "\nEmployeeName
:" + rs.getString(2) + "\nSalary :"
                                      + rs.getDouble(3) + " \nAge : " + rs.getInt(4) +
"\nDesignation :" + rs.getString(5));
                          System.out.println("--------------------");
                    }
              } catch (Exception e) {
                    e.printStackTrace();
              }
       }

       @Override
       public void showEmployeeBasedOnID(int id) {
              String query = "select * from emp where eid=?";
              try {
                    con = DbConnection.createDBConnection();
                    pst = con.prepareStatement(query);
                    pst.setInt(1, id);
                    rs = pst.executeQuery();
                    if (rs.next()) {
                          System.out.println("EmpId :" + rs.getInt(1) + "\nEmpName :" +
rs.getString(2) + "\nEmpSalary :"
                                      + rs.getDouble(3) + "\nAge :" + rs.getInt(4) +
"\nDesignation :" + rs.getString(5));
                    } else {
                          System.err.println("The specific id  is not present in the table");
                    }
              } catch (Exception e) {
                    e.printStackTrace();
              }
       }

       @Override
       public void updateEmployee(int id, String name) {
              String query = "update emp set ename=? where eid=?";
              try {
                    con = DbConnection.createDBConnection();
                    pst = con.prepareStatement(query);
                    pst.setString(1, name);
                    pst.setInt(2, id);
                    int r = pst.executeUpdate();
                    if (r != 0) {
                          System.out.println("Employee Details updated successfully !!");
                    } else {
```

```
                        System.err.println("The id or name is not present in the table");
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        }

        @Override
        public void deleteEmployee(int id) {
            String query = "delete from emp where eid=?";
            try {
                con = DbConnection.createDBConnection();
                pst = con.prepareStatement(query);
                pst.setInt(1, id);
                int r = pst.executeUpdate();
                if (r != 0) {
                    System.out.println(id + "Record of Employee table Deleted
Successfully!!! ");
                } else {
                    System.err.println("The record is not present in the table");
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
}
```

Employee test.java

```
package com.employeeApp;

import java.util.Scanner;

public class EmpTestApp {
    public static void main(String[] args) {
        String name;
        int id = 0;

        EmployeeInterface dao = new EmployeeImpl();
        System.out.println("Welcome to Employee management application");

        Scanner sc = new Scanner(System.in);
        do {
            System.out.println();
            System.out.println("---------menu----------");
            System.out.println("1. Add Employee\n" + "2. Show All Employee\n" + "3.
Show Employee based on id \n"
                    + "4. Update the employee\n" + "5. Delete the employee\n");

            System.out.println("Enter Choice: ");
            int ch = sc.nextInt();
            switch (ch) {
            case 1:
                Employee emp = new Employee();
                System.out.println("Enter ID : ");
                id = sc.nextInt();
                System.out.println("Enter name ");
                name = sc.next();
                System.out.println("Enter Salary ");
                double salary1 = sc.nextDouble();
                System.out.println("Enter age");
```

```
                        int age1 = sc.nextInt();
                        System.out.println("Enter Designation ");
                        String desg1=sc.next();
                        emp.setId(id);
                        emp.setName(name);
                        emp.setSalary(salary1);
                        emp.setAge(age1);
                        emp.setDesg(desg1);
                        dao.createEmployee(emp);
                        break;
                case 2:
                        dao.DisplayAllEmployee();
                        break;
                case 3:
                        System.out.println("Enter id to show the details ");
                        int empid = sc.nextInt();
                        dao.showEmployeeBasedOnID(empid);
                        break;
                case 4:
                        System.out.println("Enter id to update the details");
                        int empid1 = sc.nextInt();
                        System.out.println("Enter the new name");
                        name = sc.next();
                        dao.updateEmployee(empid1, name);
                        break;
                case 5:
                        System.out.println("Enter the id to delete");
                        id = sc.nextInt();
                        dao.deleteEmployee(id);
                        break;
                case 6:
                        System.out.println("Thank you for using our Application !!!");
                        System.exit(0);
                default:
                        System.out.println("Enter valid choice !");
                        break;
                }

        } while (true);

    }
}
```