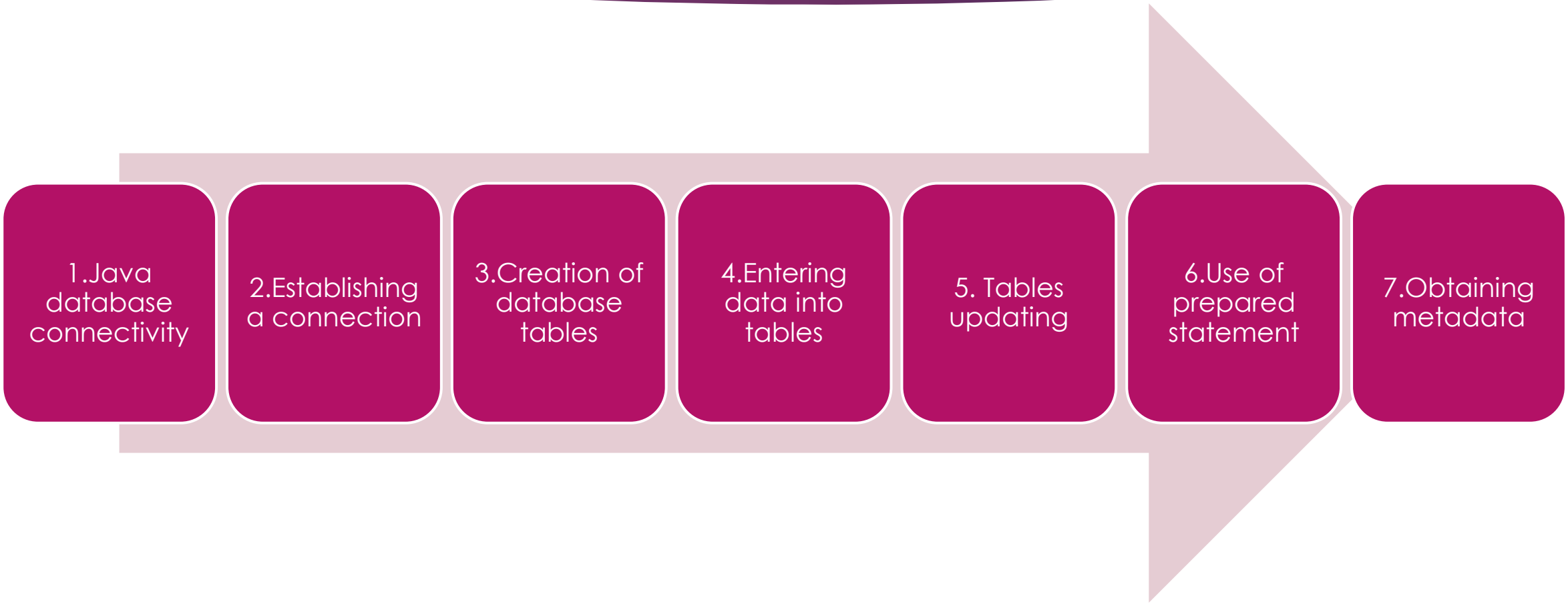
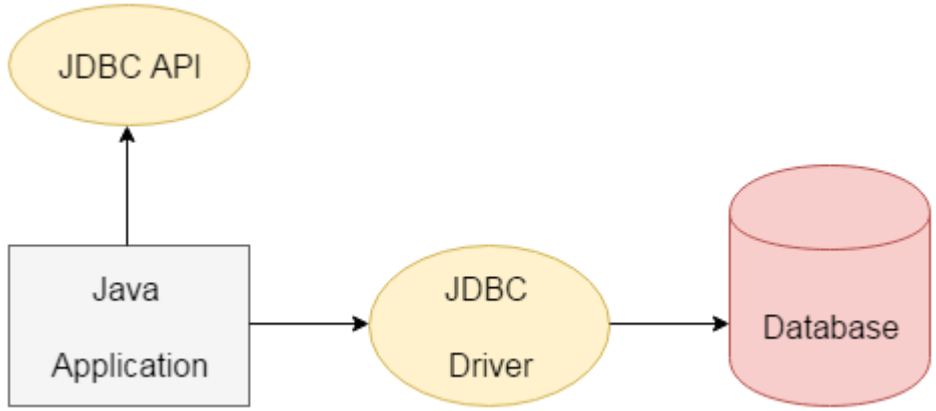
JAVA DATABASE CONNECTIVITY



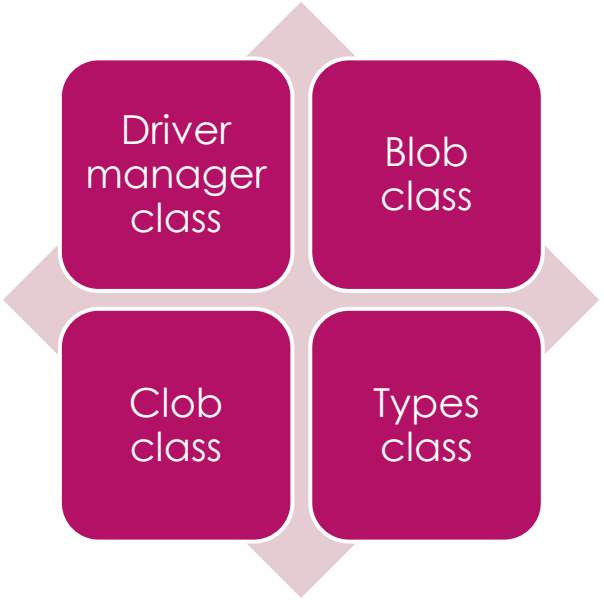
JAVA DATABASE CONNECTIVITY

- JDBC stands for Java Database Connectivity. JDBC is a Java API to connect and execute the query with the database.
- It is a part of Java SE (Java Standard Edition). JDBC API uses JDBC drivers to connect with the database. There are four types of JDBC drivers:
- JDBC-ODBC Bridge Driver , Native Driver ,Network protocol , and Thin Driver
- The java.sql, package contains classes and interfaces for JDBC API.



CLASS & INTERFACE USED IN JDBC

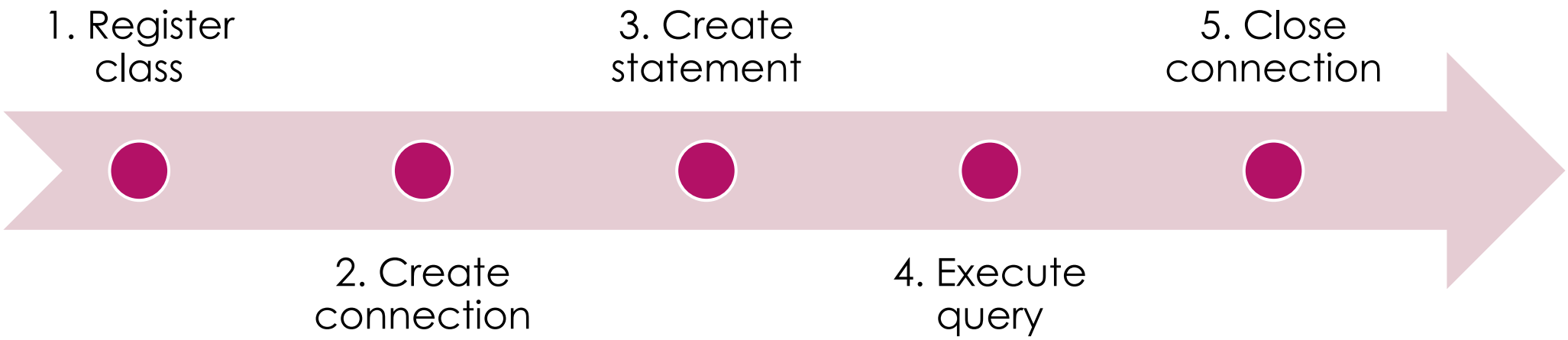
A list of popular classes of JDBC are given below



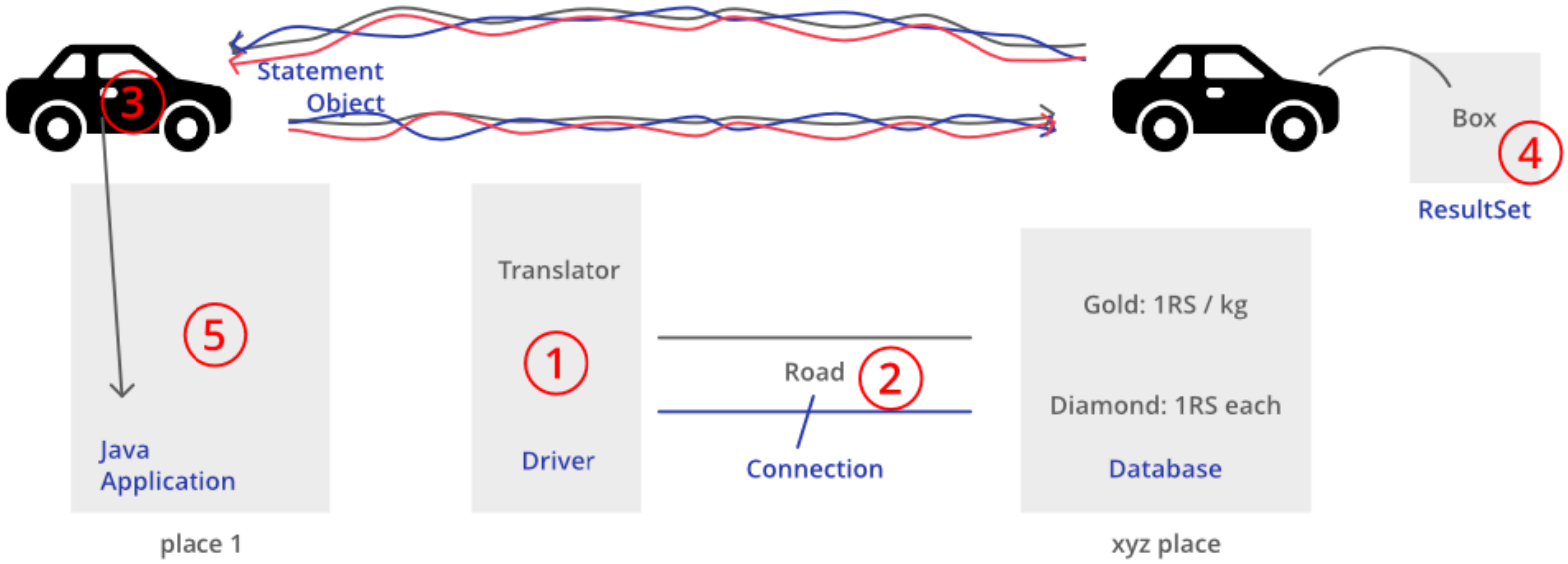
A list of popular interface of JDBC are given below

- Driver interface
- Connection interface
- Statement interface
- Prepared Statement interface
- Callable Statement interface
- Result Set interface
- Result Set Meta Data interface

Java database connectivity steps:



Steps are:


Statement
Object
3
Box
4
ResultSet
5
Java
Application
place 1
Translator
1
Driver
Road
2
Connection
Gold: 1RS / kg
Diamond: 1RS each
Database
xyz place

1.Register the driver class

Definition:

- The **forName()** method of Class is used to register the driver class. This method is used dynamically used for the driver class

Syntax:

- **public static void** forName(String className)**throws** ClassNotFoundExcepion

For example

- **Class.forName("oracle.jdbc.driver.OracleDriver");**

2.Create the connection object

Definition

- The get Connection() method of Driver Manager class is used to establish connection with the database.

Syntax:

1) **public static** Connection get Connection(String url l)**throws** SQL Exception

- 2) **public static** Connection get Connection(String url, String name , String password) **throws** SQL Exception

Example

- Connection con=Driver Manager .get Connection(" jdbc :oracle :thin :@localhost:1521:xe","system","password");

3.Create the statement object

Definition:

- The **createStatement()** method of Connection interface is used to create statement. The object of statement is responsible **to execute queries with the database.**

Syntax;

- **public** Statement createStatement()**throws** SQLException

for example:

- Statement stmt=con.createStatement();

4.Execute the query

Definition:

- The execute Query() method of Statement interface **is used to execute queries to the database**. This method returns the object of **Result Set** that can be used to get all the records of a table.

Syntax:

- **public** Result Set execute Query(String sql)**throws** SQLException

For example:

```
ResultSet rs=stmt.executeQuery("select * from emp");
```

5.Close the connection object

Definition:

- By closing connection object statement and Result Set will be closed automatically. The close() method of Connection interface is used to close the connection.

Syntax:

- **public void** close()**throws** SQLException

For example:

- con.close();

```
// java create a database connection  and produce output
import java.sql.*;
1.class MysqlCon{
2.public static void main(String args[]){
3.try{
4.Class. forName("com.mysql.jdbc.Driver");
5.Connection con=DriveManager.getConnection(
6."jdbc:mysql://localhost:3306/sonoo","root","root");
7.//here sonoo is database name, root is username and password
8.Statement stmt=con.createStatement();
9.ResultSet rs=stmt.executeQuery("select * from emp");
10.while(rs.next())
11.System.out.println(rs.getInt(1)+"  "+rs.getString(2)+"  "+rs.getString(3));
12.con.close();
13.}catch(Exception e){ System.out.println(e);}
14.}
15.}
```

Example program for insert,update,delete,select in jdbc

```
import java.sql.*;
class EmployeeRecord
{
    public static final String DBURL = "jdbc:oracle:thin:@localhost:1521:XE";
    public static final String DBUSER = "local";
    public static final String DBPASS = "test";
    public static void main(String args[])
    {
        try
        {
            //Loading the driver
            Class.forName("oracle.jdbc.driver.OracleDriver");
            //Create the connection object
            Connection con = DriverManager.getConnection(DBURL, DBUSER,
DBPASS);
```

INSERT INTO RECORD

INSERT THE RECORD

- String sql = "INSERT INTO emp (emp_id, empname, email, city) VALUES (?, ?, ?, ?)";
  PreparedStatement statement = con.prepareStatement(sql);
  statement.setInt(1, 159);
  statement.setString(2, “sharifasulthana");
  statement.setString(3, “Ksharifasulthana24@gmail.com");
  statement.setString(4, “salem");

  int **rowsInserted =statement.executeUpdate();**
  if (rowsInserted > 0)
  {
  System.out.println("A new employee was inserted successfully!\n");}

OUTPUT

```
C:\>java JDBCExample Inserting records into the table...
Inserted records into the table... C:\>
```

Update into record using JDBC

- **//Update the record**

```
String sql2 = "Update Emp set email = ? where empname = ?";
PreparedStatement pstmt = con.prepareStatement(sql2);
pstmt.setString(1, "Jaya@gmail.com");
pstmt.setString(2, "Jaya");
int rowUpdate = pstmt.executeUpdate();
if (rowUpdate > 0)
{
    System.out.println("\nRecord updated successfully!!\n");
}
```

Display the record

```
{String sql = "UPDATE Registration " + "SET age =
30 WHERE id in (100, 101)";
 stmt.executeUpdate(sql); ResultSet rs =
stmt.executeQuery(QUERY);
while(rs.next())
{ //Display values
System.out.print("ID: " + rs.getInt("id"));
 System.out.print(", Age: " + rs.getInt("age"));
 System.out.print(", First: " +
rs.getString("first"));
 System.out.println(", Last" +
rs.getString("last"));
 }
```

```
C:\>java JDBCExample ID:
100, Age: 30, First: Zara, Last: Ali ID:
101, Age: 30, First: Mahnaz
, Last: Fatma ID: 102, Age: 30, First: Zaid, Last: Khan
 ID: 103, Age: 28, First: Sumit, Last: Mittal
 C:\>
```

Use of PreparedStatement

Definiton:

- **Prepared Statement** is a pre-compiled SQL statement. It is a **sub-interface** of the statement in java. It has valuable features which are in addition to that of objects in a statement. The object of the Prepared Statement has the feature of **executing parameterized queries** instead of hard coding.

For example:

- String sql="insert into emp values(?,?,?)";

Methods of PreparedStatement

Following are the methods of Prepared Statement in java :

setInt(int, int) : it is used for setting the value of integer at a given index in the parameter.

setString(int, string) : it is used to set the value of a string at a specified index given in the parameter.

setFloat(int, float) : It is used to set a float value at a specified index

setDouble(int, double) : It is used to set a double value at a specified value.

executeUpdate() : It is used to create drop, update, insert and delete etc. The return type is int.

executeQuery() : It is used for returning instance of ResultSet when a query is selected

Obtaining metadata

- Metadata in Java, defined as the data about the data, is called "Metadata".
- Metadata is also said to be documentation about the information required by the users. This is one of the essential aspects in the case of data warehousing.
- *DatabaseMetaData* interface provides methods to get metadata of a database such as a **database product name, database product version, driver name, name of a total number of tables, a name of a total number of views etc.**

Methods of Metadata

public String getDriverName()**throws SQLException:** it returns the name of the JDBC driver.

public String getDriverVersion**()throws SQLException:** it returns the version number of the JDBC driver.

public String getUserName()**throws SQLException:** it returns the username of the database.

public String getDatabaseProductName**()throws SQLException:** it returns the product name of the database.

public String getDatabaseProductVersion**()throws SQLException:** it returns the product version of the database.

public ResultSet getTables(String catalog, String schemaPattern, String tableNamePattern, String[] types)throws SQLException: it returns the description of the tables of the specified catalog.

1.For example”

```
2.Import java.sql.*;
3.class Dbms{
4.public static void main(String args[]){
5.try{
6.Class.forName("oracle.jdbc.driver.OracleDriver");
7.
8.Connection con=DriverManager.getConnection(
9."jdbc:oracle:thin:@localhost:1521:xe","system","oracle");
10.DatabaseMetaData dbmd=con.getMetaData();
11.
12.System.out.println("Driver Name: "+dbmd.getDriverName());
13.System.out.println("Driver Version: "+dbmd.getDriverVersion());
14.System.out.println("UserName: "+dbmd.getUserName());
15.System.out.println("Database Product Name: "+dbmd.getDatabaseProductName());
16.System.out.println("Database Product Version: "+dbmd.getDatabaseProductVersion());
17.
18.con.close();
19.}catch(Exception e){ System.out.println(e);}
20.}  }
```

OUTPUT; Driver name: oracle jdbc name

Driver Version: 10.2.0.1.0XE
Product Name: Oracle Database
Product Version
: Oracle Database 10g Express Edition Release 10.2.0.1.0 - Production


Thank You