**Module 5 Question Bank Answers**

**Object Oriented Programming Using Java (21CIC34)**

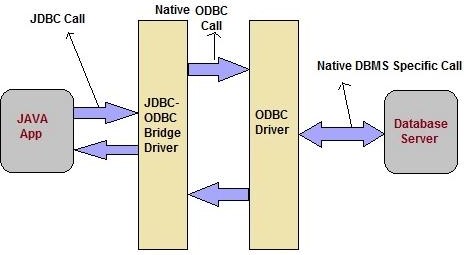
**Semester:3rd Session: July-Dec 2022**

1. **Explain the different JDBC drivers. (CO6)**

Ans:

JDBC Driver is required to process SQL requests and generate result. The following are the different types of driver available in JDBC.

* Type-1 Driver or JDBC-ODBC bridge
* **Type-2 Driver** or **Native API Partly Java Driver**
* Type-3 Driver or Network Protocol Driver
* **Type-4 Driver** or **Thin Driver**

Type-1 Driver or JDBC-ODBC bridge

**Type-1 Driver** act as a bridge between JDBC and other database connectivity mechanism(ODBC). This driver converts JDBC calls into ODBC calls and redirects the request to the ODBC driver.

**Advantages**

* Easy to use
* Allow easy connectivity to all database supported by the ODBC Driver.

**Disadvantages**

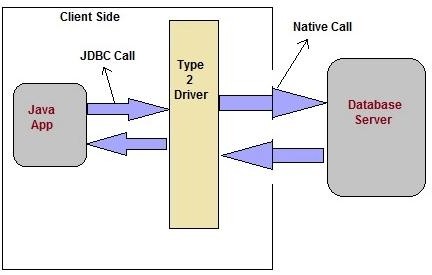
* Since the Bridge driver is not written fully in Java, Type 1 drivers are not portable
* A performance issue is seen as a JDBC call goes through the bridge to the ODBC driver, then to the database, and this applies even in the reverse process. They are the slowest of all driver types.
* The client system requires the ODBC Installation to use the driver.
* Not good for the Web.

Type-2 Driver or Native API Partly Java Driver

This type of driver make use of Java Native Interface(JNI) call on database specific native client API. These native client API are usually written in C and C++. Will be working with Database software’s like Oracle or SQL.

**Advantages**

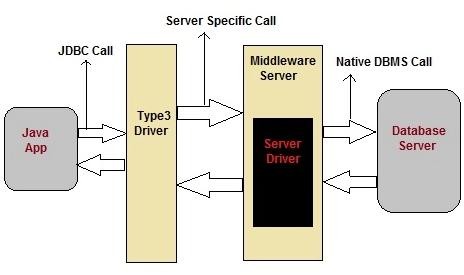
* faster as compared to **Type-1 Driver**
* Contains additional features.



**Disadvantages**

* Native API must be installed in the Client System and hence type 2 drivers cannot be used for the Internet.
* Like Type 1 drivers, it’s not written in Java Language which forms a portability issue.
* If we change the Database we have to change the native api as it is specific to a database

Type-3 Driver or Network Protocol Driver

This driver translate the JDBC calls into a database server independent and Middleware server-specific. Middleware server further translate JDBC calls into database specific calls.

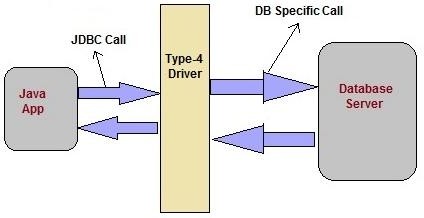
**Advantages**

* This driver is server-based, so there is no need for any vendor database library to be present on client machines.
* This driver is fully written in Java and hence Portable. It is suitable for the web.
* This driver is very flexible allows access to multiple databases using one driver.
* They are the most efficient amongst all driver types

**Disadvantage**

* It requires another server application to install and maintain.

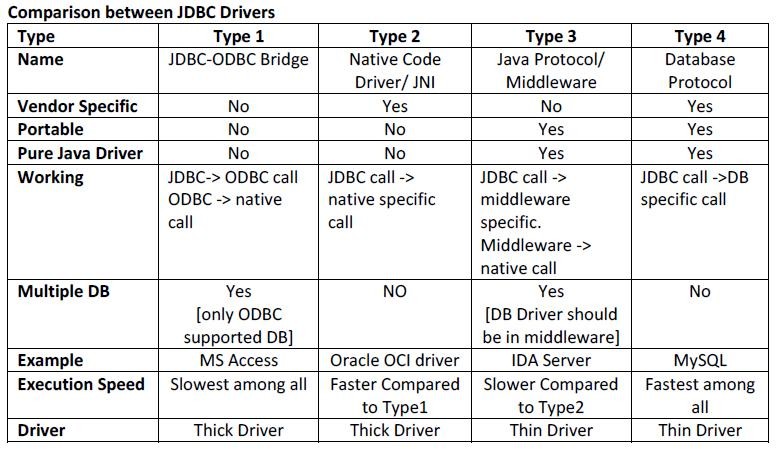
Type-4 Driver or Thin Driver

This is Driver called Pure Java Driver because. This driver interact directly with database. It does not require any native database library, that is why it is also known as Thin Driver.

**Advantages**

* The major benefit of using a type 4 jdbc drivers are that they are completely written in Java to achieve platform independence
* It is most suitable for the web.
* You don’t need to install special software on the client or server.
* Further, these drivers can be downloaded dynamically

**Disadvantage**

* Slow due to increase number of network call.

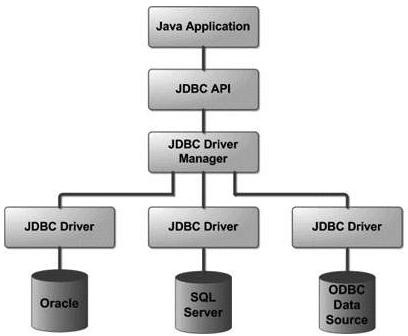
1. **Give an overview of JDBC process.(CO6)**

Ans: The JDBC API supports both two-tier and three-tier processing models for database access but in general, JDBC Architecture consists of two layers −

* **JDBC API:** This provides the application-to-JDBC Manager connection.
* **JDBC Driver API:** This supports the JDBC Manager-to-Driver Connection.

The JDBC API uses a driver manager and database-specific drivers to provide transparent connectivity to heterogeneous databases.

The JDBC driver manager ensures that the correct driver is used to access each data source. Following is the architectural diagram, which shows the location of the driver manager with respect to the JDBC drivers and the Java application –



Common JDBC Components

The JDBC API provides the following interfaces and classes −

* **DriverManager:** This class manages a list of database drivers. Matches connection requests from the java application with the proper database driver using communication sub protocol. The first driver that recognizes a certain subprotocol under JDBC will be used to establish a database Connection.
* **Driver:** This interface handles the communications with the database server. You will interact directly with Driver objects very rarely. Instead, you use DriverManager objects, which manages objects of this type.
* **Connection:** This interface with all methods for contacting a database. The connection object represents communication context, i.e., all communication with database is through connection object only.
* **Statement:** We use objects created from this interface to submit the SQL statements to the database. Some derived interfaces accept parameters in addition to executing stored procedures.
* **ResultSet:** These objects hold data retrieved from a database after you execute an SQL query using Statement objects. It acts as an iterator to allow you to move through its data.
* **SQLException:** This class handles any errors that occur in a database application.

1. **Discuss the different types of Swing buttons. (CO6)**

# Ans: Java JButton

The JButton class is used to create a labeled button that has platform independent implementation. The application result in some action when the button is pushed. It inherits AbstractButton class.

## JButton class declaration

Let's see the declaration for javax.swing.JButton class.

1. **public class** JButton **extends** AbstractButton **implements** Accessible

Commonly used Constructors:

|  |  |
| --- | --- |
| **Constructor** | **Description** |
| JButton() | It creates a button with no text and icon. |
| JButton(String s) | It creates a button with the specified text. |
| JButton(Icon i) | It creates a button with the specified icon object. |

Commonly used Methods of AbstractButton class:

|  |  |
| --- | --- |
| **Methods** | **Description** |
| void setText(String s) | It is used to set specified text on button |
| String getText() | It is used to return the text of the button. |
| void setEnabled(boolean b) | It is used to enable or disable the button. |
| void setIcon(Icon b) | It is used to set the specified Icon on the button. |
| Icon getIcon() | It is used to get the Icon of the button. |
| void setMnemonic(int a) | It is used to set the mnemonic on the button. |
| void addActionListener(ActionListener  a) | It is used to add the [action listener](https://www.javatpoint.com/java-actionlistener) to this object. |

# Java JRadioButton

The JRadioButton class is used to create a radio button. It is used to choose one option from multiple options. It is widely used in exam systems or quiz.

It should be added in ButtonGroup to select one radio button only.

## JRadioButton class declaration

Let's see the declaration for javax.swing.JRadioButton class.

1. **public class** JRadioButton **extends** JToggleButton **implements** Accessible

Commonly used Constructors:

|  |  |
| --- | --- |
| **Constructor** | **Description** |
| JRadioButton() | Creates an unselected radio button with no text. |
| JRadioButton(String s) | Creates an unselected radio button with specified text. |
| JRadioButton(String s, boolean selected) | Creates a radio button with the specified text and selected status. |

Commonly used Methods:

|  |  |
| --- | --- |
| **Methods** | **Description** |
| void setText(String s) | It is used to set specified text on button. |
| String getText() | It is used to return the text of the button. |
| void setEnabled(boolean b) | It is used to enable or disable the button. |
| void setIcon(Icon b) | It is used to set the specified Icon on the button. |
| Icon getIcon() | It is used to get the Icon of the button. |
| void setMnemonic(int a) | It is used to set the mnemonic on the button. |
| void addActionListener(ActionListener a) | It is used to add the action listener to this object. |

# Java JCheckBox

The JCheckBox class is used to create a checkbox. It is used to turn an option on (true) or off (false). Clicking on a CheckBox changes its state from "on" to "off" or from "off" to "on ".It inherits [JToggleButton](https://www.javatpoint.com/java-jtogglebutton) class.

## JCheckBox class declaration

Let's see the declaration for javax.swing.JCheckBox class.

1. **public class** JCheckBox **extends** JToggleButton **implements** Accessible

Commonly used Constructors:

|  |  |
| --- | --- |
| **Constructor** | **Description** |
| JJCheckBox() | Creates an initially unselected check box button with no text, no icon. |
| JChechBox(String s) | Creates an initially unselected check box with text. |
| JCheckBox(String text, boolean selected) | Creates a check box with text and specifies whether or not it is initially selected. |
| JCheckBox(Action a) | Creates a check box where properties are taken from the Action supplied. |

Commonly used Methods:

|  |  |
| --- | --- |
| **Methods** | **Description** |
| AccessibleContext getAccessibleContext() | It is used to get the AccessibleContext associated with this JCheckBox. |
| protected String paramString() | It returns a [string](https://www.javatpoint.com/java-string) representation of this JCheckBox. |

1. Explain the different steps in JDBC with code snippets. (CO6)

Ans: Java Database Connectivity with 5 Steps / Overview of the JDBC Process

There are 5 steps to connect any java application with the database using JDBC. These steps are as follows:

1. Register the driver class

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

Syntax of forName() method

**public static void** forName(String className)**throws** ClassNotFoundException

Example to register the MySqlDriver class Class.*forName*("com.mysql.jdbc.Driver");

1. **Create the connection object**

The **getConnection()** method of DriverManager class is used to establish connection with the database.

Syntax of getConnection() method

**public static** Connection getConnection(String url,String name,String password)

**throws** SQLException

Example to establish connection with the MYSql database

Connection con = DriverManager.*getConnection*("jdbc:mysql://localhost:3308/java","root","");

1. Create the Statement object

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

Syntax of createStatement() method

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

Example to create the statement object

Statement stmt=con.createStatement();

1. Execute the query

The executeQuery() method of Statement interface is used to execute queries to the database. This method returns the object of ResultSet that can be used to get all the records of a table.

Syntax of executeQuery() method

**public** ResultSet executeQuery(String sql)**throws** SQLException

Example to execute query

ResultSet rs=stmt.executeQuery("select \* from author");

**while**(rs.next()){

System.out.println(rs.getString(1)+" "+rs.getString(2) +" "+rs.getInt(3));

}

1. Close the connection object

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

Syntax of close() method

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

Example to close connection

con.close();

1. **What are swing?Create swing application that contains JLabel,JButton, JTextField and implement action listener to perform action.(CO6)**

Ans: Java Swing l is a part of Java Foundation Classes (JFC) that is *used to create window-based applications*. It is built on the top of AWT (Abstract Windowing Toolkit) API and entirely written in java.

Unlike AWT, Java Swing provides platform-independent and lightweight components.

The javax.swing package provides classes for java swing API such as JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser etc.

import java.awt.event.\*;

import javax.swing.\*;

public class FirstSwingExample{

    public static void main(String[] args) {

        JFrame f=new JFrame();//creating instance of JFrame

        JLabel lb=new JLabel();

        lb.setText("Basic Swing Application");

        lb.setBounds(130, 150, 200, 50);

        JTextField tf=new JTextField();

        tf.setBounds(130, 200, 200, 50);

        JButton b=new JButton("click");//creating instance of JButton

        b.setBounds(130,300,100, 40);//x axis, y axis, width, height

        b.addActionListener(new ActionListener(){

        public void actionPerformed(ActionEvent e){

        tf.setText("Listner in Action..");

        }

    });

        f.add(b);//adding button in JFrame

        f.add(lb);

        f.add(tf);

        f.setSize(400,500);//400 width and 500 height

        f.setLayout(null);//using no layout managers

        f.setVisible(true);//making the frame visible

        }

}

1. **Create a Frame with a button having caption PRESS ME, once the button is clicked, the caption changes to PRESSED. (CO6)**

**Ans:**

import java.awt.event.\*;

import javax.swing.\*;

public class ButtonDemo {

    public static void main(String args[]){

    JFrame f=new JFrame();

    JButton btn=new JButton("PRESS ME");

    btn.setBounds(120, 200, 150, 50);

    btn.addActionListener(new ActionListener(){

        public void actionPerformed(ActionEvent e){

        btn.setText("PRESSED..");

        }

    });

    f.add(btn);

    f.setSize(400,500);//400 width and 500 height

    f.setLayout(null);//using no layout managers

    f.setVisible(true);//making the frame visible

    }

}

1. **Create a Frame that has two buttons named alpha and beta. When either of the buttons pressed, it should display “Alpha is pressed” or : Beta is pressed” (CO6)**

Ans:

import java.awt.event.\*;

import javax.swing.\*;

public class ButtonDemo {

    public static void main(String args[]){

    JFrame f=new JFrame();

    JLabel lb=new JLabel("Changes on click");

    lb.setBounds(120, 150, 150, 50);

    JButton abtn=new JButton("Alpha Button");

    abtn.setBounds(50, 200, 100, 50);

    abtn.addActionListener(new ActionListener(){

        public void actionPerformed(ActionEvent e){

        lb.setText("Alpha is Pressed..");

        }

    });

    JButton bbtn=new JButton("Beta Button");

    bbtn.setBounds(250, 200, 100, 50);

    bbtn.addActionListener(new ActionListener(){

        public void actionPerformed(ActionEvent e){

        lb.setText("Beta is Pressed..");

        }

    });

    f.add(lb);

    f.add(abtn);

    f.add(bbtn);

    f.setSize(400,500);//400 width and 500 height

    f.setLayout(null);//using no layout managers

    f.setVisible(true);//making the frame visible

    }

}

1. **Create a Frame with three text fields and a button. Input integer values in first two Text Fields. On Click of add button, the sum should be displayed in the third Text Field. (CO6)**

Ans:

import java.awt.event.\*;

import javax.swing.\*;

public class SimpleAddDemo {

    public static void main(String args[]){

        JFrame f=new JFrame();

        JTextField tf1=new JTextField();

        tf1.setBounds(100, 100, 80, 30);

        JTextField tf2=new JTextField();

        tf2.setBounds(100, 150, 80, 30);

        JTextField res=new JTextField();

        res.setBounds(100, 280, 80, 30);

        JButton btn = new JButton("Add");

        btn.setBounds(100,200,80,50);

        btn.addActionListener(new ActionListener(){

            public void actionPerformed(ActionEvent ev){

                    int a = Integer.parseInt(tf1.getText());

                    int b = Integer.parseInt(tf2.getText());

                    int c = a + b;

                    res.setText(String.valueOf(c));

            }

        });

f.add(tf1);

f.add(btn);

f.add(tf2);

f.add(res);

    f.setSize(400,500);//400 width and 500 height

    f.setLayout(null);//using no layout managers

    f.setVisible(true);

    }

}

Recordings:

**Swings:** https://zoom.us/rec/share/i9Qc0E4N3\_EFCWaaIgRjZYUvFzcZAlWW-yBf2SFSnAvt5g9eDwBgbP600jm9NCep.BGfDhq9FsBsBFhBz?startTime=1671759617000

Passcode: J+7sL6%T

**JDBC:**

https://zoom.us/rec/share/bNOvpcF2o9xcqXmS4AD46WLg-XX60ajqBy5hX6\_cxte2g76toOnPR-Pnf8fi9gea.DyjpZtUFhK-sHy3T?startTime=1671776997000

Passcode: S@bG4LJs