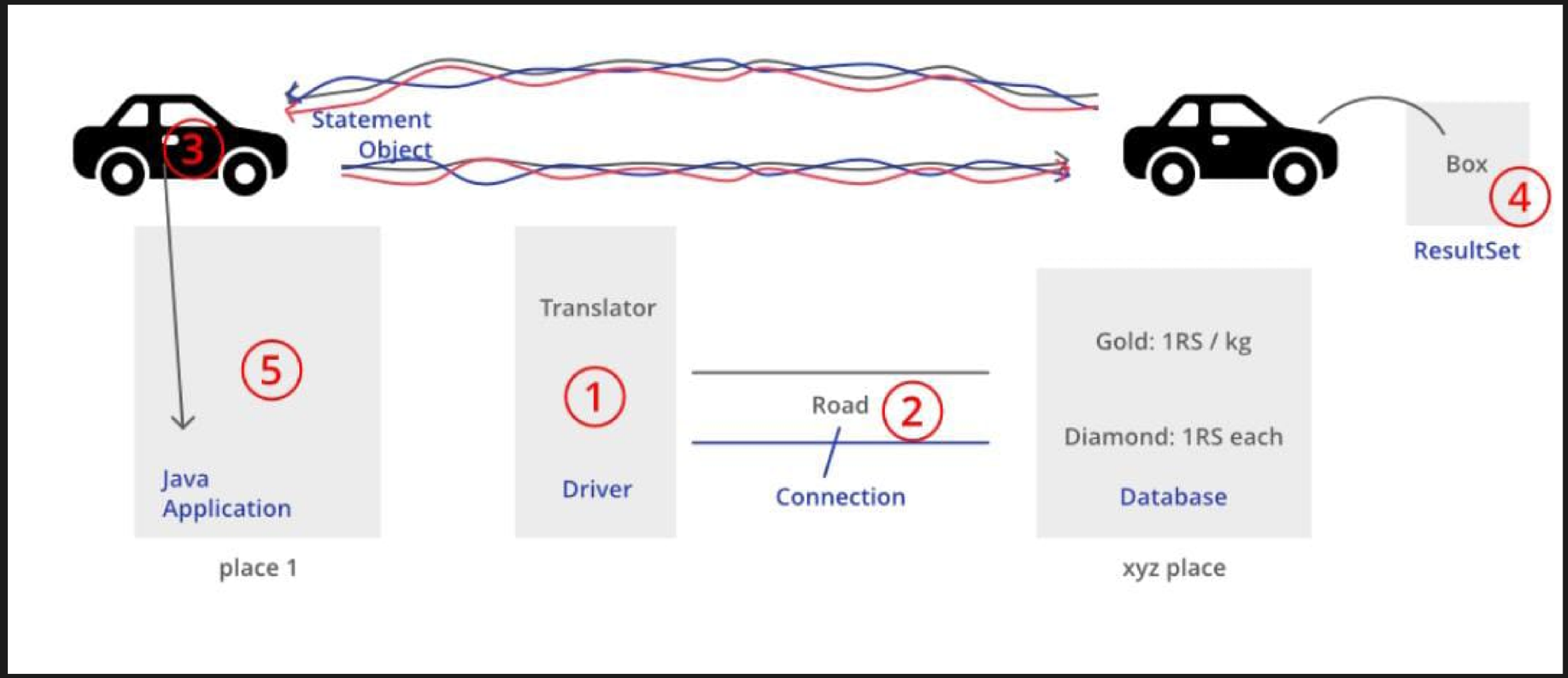
JAVA PROGRAMMING

ESTABLISHING CONNECTIONS IN JDBC

KIRUTHIGA.K - 20384115

ASWATHI - 20384107

Working of JDBC co-relating with real-time



Import the Packages 01

Loading the drivers 02

Establish connection using Connection object 03

Creating a statement 04

Execute the query 05

Closing the connection 06

STEPS TO ESTABLISH CONNECTION

JAVA PROGRAMMING

IMPORTING JAVA SQL PACKAGES

WHAT IS A PACKAGE?

- Namespace that organizes a set of related classes and interfaces
- Provide access protection and prevent naming conflicts
- There are 2 types of packages
    - User-defined package
    - Built-in package

WHAT IS THE JAVA.SQL PACKAGE?

- Include classes and interface to perform operations such as creating and executing SQL queries
- Contains the entire JDBC API that sends SQL statements to relational databases and retrieves the results
- Provides the API for accessing and processing data stored in a database

SOME OF THE CLASSES AND INTERFACES IN THE PACKAGE

Driver

- Provides the API for registering and connecting drivers based on JDC technology
- Generally used only by the DriverManager class

DriverManager

- Provides basic service for managing a set of JDBC drivers
- Acts as an interface between the user and drivers
- Maintains a list of Driver classes that have registered themselves by calling registerDriver method

DriverPropertyInfo

- Keeps an account for managing the various properties of JDBC drivers which are required for making a secure connection
- Provides properties for a JDBC driver

Connection

- Provides methods for creating statements and managing connections and their properties

SOME OF THE CLASSES AND INTERFACES IN THE PACKAGE (contd.)

Statement

- Used for executing a static SQL statement and returning the results it produces
- Extension of Wrapper and AutoCloseable interfaces

PreparedStatement

- Subinterface of Statement interface
- Used to execute a parameterized query
- Performance of the application will be faster since a query is compiled only once

CallableStatement

- Used to execute SQL stored procedures
- Used to call database stored procedures

ResultSet

- Provides methods for retrieving and manipulating the results of executed queries
- ResultSet objects can have different functionalities and characteristics

Importing the packages

Driver

import java.sql.driver

DriverManager

import java.sql.DriverManager

DriverPropertyInfo

import java.sql.DriverPropertyInfo

Connection

import java.sql.Connection

Importing the packages (contd.)

Statement

import java.sql.statement

PreparedStatement

import java.sql.PreparedStatement

CallableStatement

import java.sql.CallableStatement

ResultSet

import java.sql.ResultSet

JAVA PROGRAMMING

LOADING & REGISTERING THE DRIVERS

WHAT IS A DRIVER?

- Software component enabling a Java application to interact with a database
- Convert requests from java programs to a protocol that the database can understand
- To connect to a database in java you must use a driver made specifically for that database

Loading & Registering

- First we need to load the driver or register it before using it
- Registration is to be done once in your program
- There are 2 ways to register drivers

WAYS TO REGISTER A DRIVER

METHOD I

Class.forName()

- Here, we load the driver’s class file into memory at the runtime
- No need of using new to create objects
- Preferable because it allows you to make the driver registration configurable and portable
- Example:

  Class.forName(“oracle.jdbc.driver.OracleDriver");:

WAYS TO REGISTER A DRIVER

METHOD II

DriverManager.registerDriver()

- DriverManager is a Java inbuilt class with a static member register
- Here we call the constructor of the driver class at compile time
- Example:

```
Driver myDriver = new oracle.jdbc.driver.OracleDriver();

DriverManager.registerDriver(myDriver);
```

HOW TO ESTABLISH CONNECTION?

- First, establish a connection with the data source you want to use
- A data source can be a database, a legacy file system, or some other source of data with a corresponding JDBC driver
- The connection is represented by a Connection object

CONNECTION OBJECT

Connection con = DriverManager.getConnection(url,user,password)

con

- It is a reference to the Connection interface

url

- Uniform Resource Locator which is created as a string
- String url = “ jdbc:oracle:thin:@localhost:1521:xe”
  - oracle - database
  - thin - driver
  - @localhost - IP address
  - 1521 - port number
  - xe - service provider

user

- Username from which your SQL command prompt can be accessed

password

- Password from which the SQL command prompt can be accessed

JAVA PROGRAMMING

CREATING A STATEMENT

HOW TO CREATE A STATEMENT?

- Once a connection is established you can interact with the database

- The JDBCStatement, CallableStatement, and PreparedStatement interfaces define the methods that enable you to send SQL commands and receive data from your database

- Syntax:

  Statement st = con.createStatement();

JAVA PROGRAMMING

EXECUTING THE QUERY

HOW TO EXECUTE A QUERY?

- The query here is an SQL Query
- The executeQuery() of the Statement interface is used to execute queries of retrieving values from the database
- This method returns the object of ResultSet that can be used to get all the records of a table

HOW TO EXECUTE A QUERY? (CONTD.)

EXAMPLE:

```
int m = st.executeUpdate(sql);
if (m==1)
    System.out.println("inserted successfully : "+sql);
else
    System.out.println("insertion failed");
```

JAVA PROGRAMMING

CLOSING THE CONNECTIONS

HOW TO CLOSE A CONNECTION?

- By closing the connection, objects of Statement and ResultSet will be closed automatically

- The close() method of the Connection interface is used to close the connection

- EXAMPLE:
    - **con.close();**

Java Program to Establish Connection in JDBC

```
// Importing database
import java.sql.*;

// Importing required classes
import java.util.*;

// Main class
class Main {

    // Main driver method
    public static void main(String a[])

    {

        //Creating the connection using Oracle DB
        String url = "jdbc:oracle:thin:@localhost:1521:xe";

        // Username and password to access DB
        // Custom initialization
        String user = "system";
        String pass = "12345";
```

Java Program to Establish Connection in JDBC (contd.)

```
  // Entering the data
Scanner k = new Scanner(System.in);
  System.out.println("enter name");
String name = k.next();

  System.out.println("enter roll no");
  int roll = k.nextInt();
  System.out.println("enter class");
  String cls = k.next();

  // Inserting data using SQL query
String sql = "insert into student1 values('" + name
              + "'," + roll + ",'" + cls + "')";

  // Connection class object
Connection con = null;
```

Java Program to Establish Connection in JDBC (contd.)

```
        // Try block to check for exceptions
      try {

          // Registering drivers
          DriverManager.registerDriver(new oracle.jdbc.OracleDriver());

          // Reference to connection interface
        con = DriverManager.getConnection(url, user, pass);

        // Creating a statement
        Statement st = con.createStatement();
```

Java Program to Establish Connection in JDBC (contd.)

```
            // Executing query
            int m = st.executeUpdate(sql);
            if (m == 1)
                System.out.println("inserted successfully : " + sql);
            else
                System.out.println("insertion failed");

            // Closing the connections
            con.close();
            }

            // Catch block to handle exceptions
        catch (Exception ex) {

                // Display message when exceptions occurs
            System.err.println(ex);
          }
      }
}
```

REFERENCES

javaguides.net

javatpoint.com

theserverside.com

geeksforgeeks.org

careerride.com

JAVA PROGRAMMING

THANK YOU

KIRUTHIGA.K - 20384115

ASWATHI.R - 20384107