Shri Shankaracharya Institute of Professional Management & Technology, Raipur

August -2022- Class Test-2

Date: 05/08/2022

Student Name: V. Om Sai Nageshwar Sharma

Roll No.: 3 0 3 3 0 2 2 2 0 0 2 0

Enrollment No.: B J 4 5 9 9

Course: B.Tech **Semester:** 4th

Branch: C S E.

Subject Name: JAVA (OOPs)

Subject Code: B 0 2 2 4 1 4 (0 2 2)

Mobile No.: 8602727389

Email id: nageshwar.sharma@ssipmt.com

Signature:

Q1)

Ans

JDBC Driver is a software component that enables java application to interact with the database. There are 4 types of JDBC drivers:

1.) JDBC - ODBC bridge driver
2.) Native - API driver (partially java driver).
3.) Network Protocol driver (partially fully Java driver).
4.) Thin driver (full java driver).

1.) JDBC - ODBC bridge driver:
The JDBC - ODBC bridge driver uses ODBC driver to connect to the database. The JDBC - ODBC bridge driver converts JDBC method calls into the ODBC function calls. This is now discouraged because of thin driver.

2.) Native - API driver: The Native API driver uses the client-side libraries of the database. The driver converts JDBC method calls into native calls of the database API. It is not written entirely in JAVA.

3.) Network Protocol driver:

The Network Protocol driver uses middleware (application server) that converts JDBC calls directly or indirectly into the vendor-specific database protocol. It is fully written in java.

4.) Thin Driver:

The thin driver converts JDBC calls directly into the vendor-specific database protocol. That is why it is known as thin driver. It is fully written in Java language.

Java Database Connectivity with 5 steps:

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

→ Register the Driver class.
→ Create connection.
→ Create statement.
→ Executes quries.
→ close connection.

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.

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

2.) Create the connection object:

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

Syntax of getConnection() method:

1) public static Connection getConnection(String url) throws SQLException.

2) pubic static Connection getConnection(String url, string name, string password) throws SQLException.

Eg.

```
connection con = DriverManager.getConnection(
    "jdbc:oracle:thin:@localhost:1521:xe","system",
        "password");
```

3.) 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

Eg: Statement stmt = con.createStatement();

4.) 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.

5.) 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: public void close() throws SQLExcept Exception.

Eg: con.close();

Q27>

Ans

Lock is an interface available in the java.util.concurrent.locks package. Java lock acts as thread synchronization mechanisms that are similar to the synchronized blocks. After some time, a new locking mechanism was introduced. It is very flexible and provides more options in comparision to the sychronized block.

The lock() method:

The lock() method is one of the most important methods of the lock interface. It is used for acquiring the lock. For thread becomes disabled when the lock is not available. The lock() method is public method that returns void.

The tryLock() method:

~~It is~~ This method is mainly used at the time of invocation for acquiring the lock. It returns the lock immediately with the Boolean value true when the lock is available. It returns the Boolean value false when the lock is not available.

The tryLock() method:

The tryLock() method is mainly used at the time of invocation for acquiring the lock. It returns the lock immediatdy with the Boolean value true when the lock is available. It returns the Boolean value false when the lock is not available.

The tryLock(long time, TimeUnit unit) method

It is another variation of the tryLock() method which is used for acquiring the lock when:

In the given waiting time, the lock will be free.

The current thread will not be interrupted.

The unlock() method:

The unlock() method is another most common method which is used for releasing the lock. The unlock() method is a public method that returns nothing and takes no parameter.

The newCondition() method :

The newCondition() method is used for getting a new Condition instance that is bound to this Lock instance.

The lock must be held by the current thread before waiting on condition.

Priority of a Thread (Thread Priority):

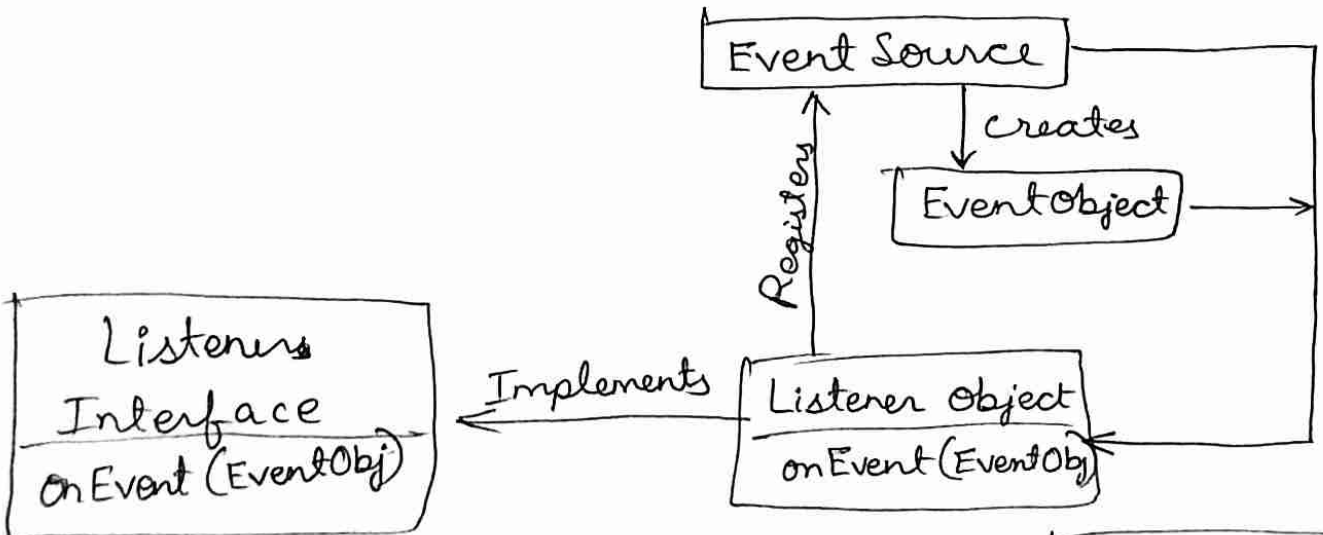
Each thread has a priority. Priorities are represented by a number between 1 and 10. In most cases, the thread scheduler schedules the threads ~~the thread~~ according to their priority (known as preemptive scheduling). But it is not guaranteed because it depends ~~not~~ on JVM specification that which scheduling it chooses. Note that not only JVM a java programmer can also assign the priorities of a thread explicitly in a java program.

Q3)

Ans

The Delegation Event model is defined to handle events in GUI stands for Graphical User Interface, where a user graphically / visually interacts with the system.

The GUI programming is inherently event driven; whenever a user initiates an activity such as a mouse activity, clicks, scrolling, etc., each is known as an event that is mapped to a code to respond to functionality to the user. This is known as event handling.



Events:

The Events are the objects that define state change in a source. An event can be generated as a reaction of a user while interacting with GUI elements. Some of the event generation activities are moving the mouse pointer, clicking on a button, pressing the keyboard key, selecting an item from the list, and so on. We can also consider many other user operations as events.

Event Sources:

A source is an object that causes and generates event. It generates an event when the internal state of the object is changed. The sources are allowed to generate several different types of events.

Event Listeners:

An event listener is an object that is invoked when an event triggers. The listeners require two things; first, it must be registered with a source; however, it can be registered with several resources to recieve notification about the events. Second, it must implement the methods to recieve and process the recieved notifications.

Q4)

Ans

Socket	Server Socket
1.) Client socket would initiate connection and only listen to response from server.	1.) Server Socket would always listen and only speak as a response to client.
2.) Client sockets is an endpoint for communication between two machines.	2.) Server socket waits for requests to come in over the network. It performs some operation based on that request, and then possibly returns a result to the requester.
3.) Establish themselves with connect().	3.) Establish themselves with Listen().
4.) It is placed in do client side, which sends request to server side socket (server socket) and wait for the response from sever.	4.) It is placed in server side, which sends request to client side socket (socket) and wait for the response from client.

5.) Socket s = new Socket ("Localhost", 1111)

5.) ServerSocket ss = new ServerSocket (1111)

Program to show the client server communication :-

File : MyServer.java

```
import java.io.*;
import java.net.*;
public class MyServer{
public static void main (string[] args) {
 try{
 ServerSocket ss = new ServerSocket (6666);
 Socket s = ss.accept();
 DataInputStream dis = new DataInputStream
                       (s.getInputStream());
 string str = (string) dis.readOTP();
 system.out.println("message = " + str);
 ss.close();
 }
 catch(exception e)
 {system.out.println(e);}
 }
}
```

File : MyClient.java

```
import java.io.*;
import java.net.*;
public class MyClient {
  public static void main(String[] args) {
    try {
      Socket s = new Socket("localhost", 6666);
      DataOutputStream dout = new DataOutputStream(s.getOutputStream());
      dout.writeUTF("Hello Server");
      dout.flush();
      dout.close();
      s.close();
    }
    catch (Exception e) {System.out.println(e);}
  }
}
```

To execute this program open two command prompts and execute each program each prompt and a message will be displayed on the server console.

Q5>

Ans>

The jar (java Archive) tool of JDK provides the facility to create the executable jar ~~fat~~ file. An executable jar file calls the main method of the class if you double click it.

To create the executable jar file, you need to create .mf file, also known as manifest file.

To create manifest file, you ~~need~~ need to write Main class, then colon, then space, then classname then enter.

For example:

```
Main-class: First
~~mit~~ myfile.mf.
```

As you can see, the mf file starts with MainClass colon space class name. Here, class name is First.

Creating executable jar file using jar tool:

The jar tool provides many switches, some of them are as follows:-

1.) -c creates new archive file.

2.) -v generates verbose output. It displays the included or extract resource on the standard output.

3.) -m includes manifest information from the given mf file.

4.) -f specifies the archive file name.

5.) -x extracts file from the archive file.

Now, let's write the code to generated the executable jar using mf file.

You need to write jar & then swiches then mf-file then jar-~~file~~ file then .classfile as given below:

jar -cvmf myfile.mf myjar.jar First.class

~~It is sh~~ Now it will create the executable jar file. If you double click on it, it will call the main method of the First class.

```
import javax.swing.*;
public class First {
  First(){
    JFrame f=new JFrame();

    JButton b=new JButton("click");
    b.setBounds(130,100,100,40);
    f.add(b);

    f.setSize(300,400);
    f.setLayout(null);
    f.setVisible(true);
   f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  }
  public static void main(String[] args){
      new First();
    }
}
```