**Name:** Syed Asad Abrar

**Roll No:** L16-4292

**Subject:** Advanced Programming

**Section:** A

**Homework 2**

**Q1)** According to the Javadoc, database metadata is “Comprehensive information about the database as a whole”. DatabaseMetaData is an interface which has methods which can be used to get information about a DBMS in combination with the driver based on JDBC that is used with it.

//written inside main function

try{

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

Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:db","system","oracle");

DatabaseMetaData metadata = con.getMetaData();

System.out.println("Driver Name: "+ metadata.getDriverName());

System.out.println("Driver Version: "+ metadata.getDriverVersion());

System.out.println("UserName: "+ metadata.getUserName());

System.out.println("Database Product Version: "+ metadata.getDatabaseProductVersion());

con.close();

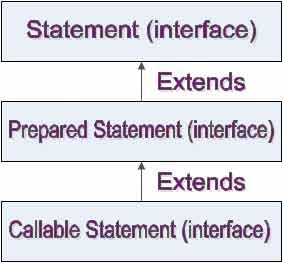
}

catch(Exception e){

System.out.println(e);

}

**Q2)**



There are 3 types of statements:

* **Statement**, useful when using static SQL statements.
* **PreparedStatement**, useful when using SQL statements with parameters (using Statement may cause SQL injection). They can be reused and are faster than a regular statement, they use native java types instead of strings (like in a statement).
* **CallableStatement**, used when executing SQL stored procedures.

**Q3)** It is faster than a Statement object because they are sent to the database and compiled or prepared beforehand. The variables are rebound every time the same statement is to be executed, prepared only once.

**Q4)** FlowLayout

**Q5)**

|  |  |
| --- | --- |
| **Container** | **Layout Manager** |
| JApplet | BorderLayout (on its content pane) |
| JBox | BoxLayout |
| JDialog | BorderLayout (on its content pane) |
| JFrame | BorderLayout (on its content pane) |
| JPanel | FlowLayout |
| JWindow | BorderLayout (on its content pane) |

**Q6)** panel.getContentPane().setLayout(new GridLayout(int rows , int columns)); //considering panel’s name is panel

**Q7)** The process of event handling is:

* When an event is triggered, the JAVA runtime first determines its source and type.
* If a listener for this type of event is registered with the source, an event object is created.
* For each listener to this type of an event, the JAVA runtime invokes the appropriate event handling method to the listener and passes the event object as the parameter.

**Q8)** Java adapter classes provide the default implementation of listener interfaces. If those are inherited:

* Implementing all the methods of an interface involves a lot of work, which would be saved.
* If you are interested in only using some methods of the interface, only few methods must be overridden.

**Q9)** RMI is a Java-specific technology. CORBA has implementations for many languages. You can use CORBA to share objects between programs written in different languages (e.g. C++ and Java).

CORBA uses IDL (Interface Definition Language) to separate interface from implementation. RMI just uses Java interfaces.

Because CORBA is not tied to a particular language, the data types do not always map exactly to the types used by your programming language (e.g. a long in IDL is an int in Java).

RMI programs can download new classes from remote JVMs. CORBA doesn't have this code sharing mechanism.

RMI can be configured to operate over IIOP (the protocol used by CORBA).

**Q10)** Skeleton hides the communication details away from the developer. It is used to communicate with stub and exists on the server side. It has access to the real remote objects, delegates the stub’s request to them and returns response to stub.

**Q11)** Using sockets, we have to handle the ports being used and also have to specify protocol. Sockets work in terms of bytes i.e. we can only send strings which are marshaled first and then sent over the network. RMI hides network-specific code and it works in terms of objects and method calls (can send and receive complete objects and remotely execute methods as if they were on client machine), so no need of handling byte conversion. Hence, RMI works better in some cases as it is beginner friendly.

**Q12)** A remote object is an object which resides on another machine (in the case of java, another JVM). UnicastRemoteObject is used to provide remote behavior by exporting a remote object with Java Remote Method Protocol (JRMP) and obtaining a stub that communicates to the remote object. If it is not extended, the implementation class must then assume the responsibility for the correct semantics of the hashCode, equals, and toString methods inherited from the Object class, so that they behave appropriately for remote objects.

**Q13)** We can use RMI for this purpose, we can do this using the following process:

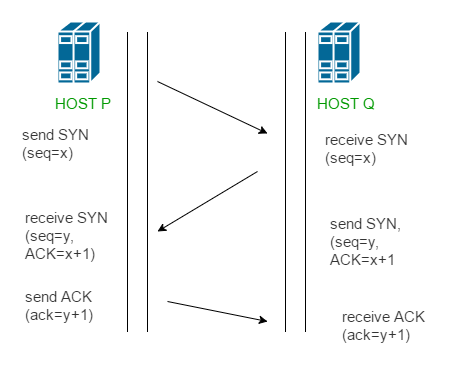
1. Create the remote interface
2. Provide the implementation of the remote interface
3. Compile the implementation class and create the stub and skeleton objects using the rmic tool
4. Start the registry service by rmiregistry tool
5. Create and start the remote application
6. Create and start the client application

When client application invokes a method, the server binds the registry, performs the necessary work after calling the method locally and returns the result over the network.

**Q14)** A three-way handshake is a method used in a TCP/IP network to create a connection between a local host/client and server. TCP/IP protocol is preferred.

It is a three-step process:

1. In the first step, client wants to establish a connection with server, so it sends a segment with SYN (Synchronize Sequence Number) which informs server that client is likely to start communication and with what sequence number it starts segments with.
2. Server responds to the client request with SYN-ACK signal bits set. Acknowledgement (ACK) signifies the response of segment it received and SYN signifies with what sequence number it is likely to start the segments with.
3. In the final part client acknowledges the response of server and they both establish a reliable connection with which they will start the actual data transfer.



**Q15)** UDP is more economical as datagrams make economical use of network bandwidth (up to 3 times the efficiency of TCP/IP for small messages). No, TCP cannot be used for multicasting and broadcasting; its main purpose is to communicate between exactly two endpoints.