B. Tech.

ECS 305

(SEM. III) ODD SEMESTER THEORY EXAMINATION, 2010-11

OBJECT ORIENTED SYSTEMS

Time: 3 *Hours* *Total Marks:* 100

Note: *Attempt all questions*

PART - I

1. Attempt any four parts of the following: (5 × 4 = 20)

Q. 1 (*a*) Describe object oriented modeling. How it affects the software development?

Ans. Object-Oriented Modeling or ODM, (Object Oriented Programming-OOP) is a modeling paradigm mainly used in computer programming. Prior to the rise of OOM, the dominant paradigm was procedural programming, which emphasized the use of discreet reusable code blocks that could stand on their own, take variables, perform a function on them, and return values. The object-Oriented paradigm assists the programmer to address the complexity of a problem domain by considering the problem not as a set of functions that can be performed but primarily as a set of related, interacting Objects. The modeling task then is specifying, for a specific context, those objects (or the Class the objects belong to), their respective set of Properties and Methods, shared by all Objects embers of the class. As an example, in a model of a Payroll System, a Company is an object. An Employee is another object. Employment is a Relationship or Association. An Employee Class (or Object for simplicity) has Attributes like Name, Birthday, etc. The association itself may be considered as an object having attributes or qualifiers like position, etc. An employee method may be promote, raise etc.

Q. 1 (*b*) What do you mean by encapsulation? Explain.

Ans. The wrapping up of data and member functions into a single unit is called encapsulation. Using the method of encapsulation, the programmer cannot directly access the data. Data is only accessible through the functions present inside the class. Data encapsulation led to the important concept of data hiding. Data hiding is the implementation details of a class that are hidden from the user. The concept of restricted access led programmer to write specialized functions or methods for performing the operations on hidden members of the class. Like class, because class have data variable and methods.

```
class A
{
int a = 5;
void sum()
{System.out.println("hi";
}
public static void main (String args [])
{
A l = new A();//.sum();}
}
```

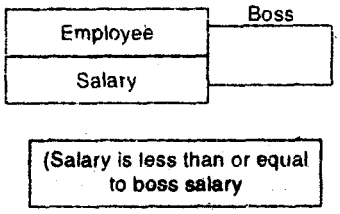
Q. 1 (*c*) Differentiate generalization and aggregation with example.

Ans. Please see Q.1(e) of 2005-06

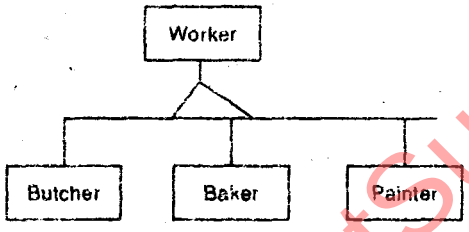
Q. 1 (*d*) Explain constraints and abstraction by taking a suitable example.

Ans. Constraints: They are functional relationships between entities of an object model. The term entity includes objects, classes, attributes, links, and associations. A constraint restricts the values that entries can assume. Example includes number of

employees, salary can exceed the salary of employees boss.

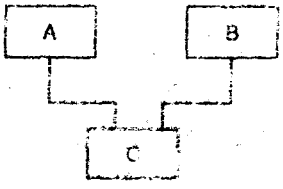


Abstraction: Abstraction is a concept of hiding unnecessary things of the system and exposing the desired functionality of the system. In object modelling abstraction is achieved by abstract classes which are class that has no direct instances but whose descendent classes have direct instances. For example there is a class worker which is an abstract class. The classes Butcher, Baker, painter are the classes called the concrete classes will inherit the abstract class properties as their direct instance will be used.



Q. 1 (*e*) What do you mean by multiple inheritance? Explain with an example.

Ans. The mechanism of inheriting the features of more than one base class into a single class is known as multiple inheritance. Java does not support multiple inheritance but the multiple inheritance can be achieved by using the interface. In Java multiple inheritance can be achieved through use of interfaces by implementing more than one interfaces in a class.




```
Interface A
{
Void sum();
}
Interface B
{
Void sum 1();
}
Class C implements A,B
{
Void sum ()
{
System.out.println ("hi");
}
Void sum 1()
{
System.out.println ("hello");
}
Public static void main (String args [])
{
C I = new C();
I.sum ();
I.sum 1();
}
}
```

Q. 1 (*f*) What do you mean by link and association? Explain with example.

Ans. Please see Q.1(*a*), 2009-10.

2. Attempt any four parts of the following: **(5 × 4 = 20)**

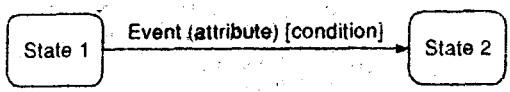
Q. 2 (*a*) What do you mean by event trace diagram? Explain.

Ans. Please see Q.2(*b*) of 2005-06.

Q. 2 (*b*) What do you mean by condition in a state diagram? Explain with suitable example.

Ans. A condition is a Boolean function of object values. For example, when we say the temperature was below freezing point from Nov. to March. Here the condition to be temperature below freezing point. It has duration. A state can be defined in terms of a condition; conversely being in a state is a condition. Conditions can be used as guards on transitions. "A guarded transition fires when its events occur but only if the condition is true." For example, let us say a person goes out in the morning (event), if the temperature is below freezing (condition), then put on your gloves (next state). A

guarded condition on a transition is shown as a Boolean expression in brackets following event name.



Q. 2 (*c*) What is the role of operations in state diagram? Differentiate between activity and action.

Ans. State diagrams could be of little use if they just described the patterns of the events. A behaviour description of an object must specify what the object does in response to events. Operations attached to states/transitions are performed in response to corresponding states or events.

Difference between activity and action

1. An activity is behaviour that can be executed in response to an event. Actions can also represent internal control operations, such as setting attributes or generating other events.
2. An activity can be performed upon a transition, upon the entry to or exit from a state or upon some event within a state. Action takes place due to event.
3. An *activity* is an operation that takes time to complete. Where as an action is an instantaneous operation.
5. Activities include continuous as well as sequential events action is instantaneous.

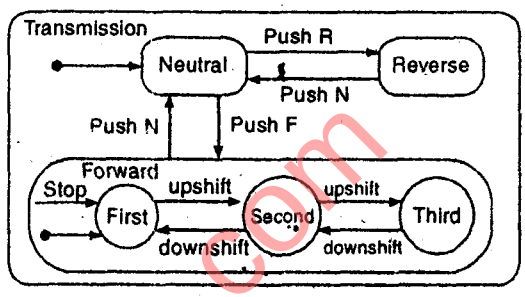
Q. 2 (*d*) How concurrency within state of single object be represented?

Ans. Sometimes an object must perform two or more activities concurrently. Both activities must be completed before the object can progress to its next state. The target state occurs after both events happen in any order. Concurrency within a single state arises when the object can be partitioned into subsets of attributes or links, each of which has its own sub diagram. Concurrency within a single composite state is shown by:

- Partitioning the composite state into sub diagrams with dotted line.
- The name of the overall composite state can be written in a separate region of the box, separated by a solid line from the current subdiagram.

Q. 2 (*e*) Explain state generalization by taking suitable example.

Ans. State generalization: A nested state diagram is actually a form of generalization on states. Generalization is the "OR relationship". An object in a state in the high level diagram must be in exactly one state in the nested diagram.



The states in a nested diagram are all refinements of the state in the high level diagram.

The transitions of a superstate are inherited by each of its substates. Selecting "*N*" in any forward gear causes a transition to neutral. Within state **Forward** substrate **First** is the default initial state (shown by an unlabeled transition).

The transition on event stop from the forward contour to state **First** represents a transition inherited by all three substates. In many forward gear, stopping the car causes a transition to **First**.

Q. 2 (*f*) What is synchronization of concurrent activities? Give example in your explanation.

Ans. Sometimes an object must perform two or more activities concurrently. The internal steps of the activity are not synchronized, but both the activities must be completed before the object can progress to its next state. For example, consider a cash dispensing machine that dispenses cash and returns the users card at the end of a transition. The machine must not reset itself until the user takes both the cash and the card, but user may take them in any order or simultaneously. The order in which they are taken are irrelevant, only the fact that all the activities must perform before the state changes. This is an example of splitting control into concurrent activities and later merging control.

3. Attempt any two parts of the following: **(10 × 2 = 20)**

Q. 3 (*a*) Write short notes on the following:

(*i*) SA/SD

(*ii*) JSD

Ans. (*i*) SA/SD: An SA/SD design has a clearly-defined system boundary, across which the software procedures must communicate with the real world. The structure of a SA/SD design is derived in part from the system boundary, so it can be difficult to extend a SA/SD design to a new boundary. It is much easier to extend an object-oriented design: one merely add objects and relationhip near the boundary to represent objects that existed previously only in the outside world. An object oriented design is more resilient to change and more extensible.

The direct analogy between objects in an object-oriented and the objects in the problem domain results in systems that are easier to understand. This makes the design more initiative and simplifies traceability between requirement and software code. It also makes a design more coherent to persons who are not part of the original design team.

In SA/SD the decomposition of a process into subprocesses is somewhat arbitrary. Different people will produce different decompositions. In object-oriented deign the decomposition is based on obejcts in the problem domain, so developers of different programs in the same domain tend to discover similar objects. This increases reusability of components from one project to the next. An object-oriented approach better integrates databases with programming code. One uniform paradigm, the object, can model both database and programming structure. Research on object-oriented databases may further improve this situation. In contrast a procedural design approach is inherently awkward at dealing with databases. It is difficult to merge programming code organized about functions with a database organized about data. There are many reasons why data flow approaches are in such wide use. Programmers have tended to think in terms of functions, so data flow based methodologies have been easier to learn. Another reason is historical; SA/SD was one of the first well-thought-out formal approaches to software and system development.

(*ii*) JSD: Jackson Structured Development (JSD) is another mature methodology, which has a different style than SA/SD or OMT. The JSD methodology was developed by Michael Jackson and is especially popular in Europe. JSD does not distinguish between analysis and design and instead lumps both phases together as specification. JSD divides sytem development into two stages: specification, then implementation. JSD first determines the "what" and then the "how". JSD is intended especially for applications in which timing is important.

A JSD model begins consideration of the real world. The purpose of a system is to provide functionality, but Jackson feels that one must first consider how this functionality fits in with the real world. A JSD model describes the real world in terms of entities, actions, and ordering of actions. Entities usually appear as nouns in requirement statements and actions appear as verbs.

JSD software development consists of six sequential steps: entity action step, entity structure step, initial model step, function step, system timing step, and implementation step.

During the entity action step the software developer lists entities and actions for part of the real world. The purpose of the overall system guides the choice of entities and actions. The input to the entity action step is the requirements statement; the output is a list of entities and actions.

Actions occur in the real world and are not an artifact of the system. Actions take place at a point in time, are atomic, and not decomposable. The entity structure step partially orders the actions of each entity by time.

The initial model step states how the real world connects to the abstract model. JSD suports state-vector and data stream connection.

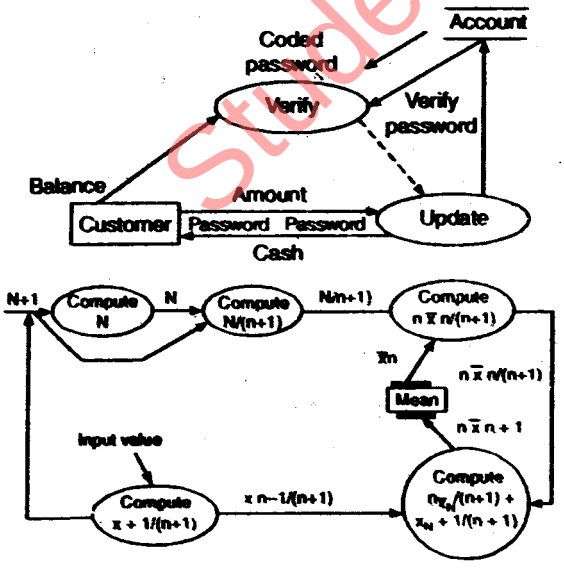
The function step uses pseudo code to state outputs of actions. At the end of this step the developer has a complete specification of the required system.

The system timing step considers how much the model is permitted to lag the real world. For the most part, the result of the timing step is a set of informal note on performance constraints. The designer explicitly makes performance trade-offs during the system timing step.

The implementation step focusses on the problems of process scheduling and allocates processors to proceses. The number of processes may be different from the number of processors.

Q.3. (*b*) Define data flow and control flow. Prepare a data flow diagram for computing the mean of a sequence of input values. A separate control input is provided to reset the computation. Each time a new value is input, the mean of all values input since the last reset command should be output. Since, you have no way of knowing how many values will be processed between resets, the amount of data storage that you use should not depend on the number of input values. Detail your diagram down to the level of multiplications, divisions, and additions.

Ans. Data flow: A data shows the functional relationships of the values computed by a system, including the input values, output values, and internal data stores. It is represented by Data flow diagram (DFD). DFD is a graph showing the flow of data values from their sources in objects through processes that transform them to their destinations in other objects. A DFD contains processes that transform data, data flows that move data, actor objects that produce and consume data, and data store objects that store data passively. The DFD shows the sequence of transformations performed, as well as the external values and objects that affect the computation.



Control flow: A control flow is a Boolean value that affects whether a process is evaluated. The control flow is not an input value to the process itself. A control flow is shown by a dotted line from a process producing a Boolean value to the process being controlled.

Q. 3 (*c*) What is OMT methodology? Discuss the phases of OMT methodolocy.

Ans. OMT combines three views of modeling system. The object model represents the static, structural data aspect of a system. The dynamic model represents the temporal, behavioural, "control" aspects of a system. The functional model represents the transformational, "function" aspects of a system. A typical software procedure incorporates all three aspects.

Data structure (Object model): Model-describes the static structure of the objects in a sytem and their relationships. This model mainly contains object diagrams.

Sequences operation in time (Dynamic model): describes the aspects of a system that change over time. This model mainly contains state diagrams.

Transform values (functional model): describes the data value transformations within a system. This model contains the data flow diagrams.

Phases of OMT methodology

The OMT methodology has the following stages:

1. Analysis: An analysis model is a concise, precise abstraction of what the desired system must do, not how it will be done. It should not contain any implementation details. The objects in the model should be application domain concepts and not the computer implementation concepts.

2. System design: The designer makes high level decisions about the overall architecture. In system design, the target system is organized a various subsystem based on both the analysis structure and the proposed architecture.

3. Object design: The deigner builds a design model based on the analysis model but containing implementation details. The focus of object design is the data structures and algorithms needed to implement each cycle.

4. Implementation: The object classes and relationships developed during object design are finally translated into a particular programming language, database, or hardware implementation. During implementation, it is important to follow good software engineering practice so that the system can remain the traceability, flexibility and extensibility.

4. Attempt any two parts of the following: **(10 × 2 = 20)**

Q. 4 (*a*) Why is Java known as platform neutral language? Write a program in Java with class Rectangle with the data fields width, length, area and colour. The length, width and area of double type and colour is of string type. The methods are set_length ( ), set_width ( ), set_colour ( ) and find_area ( ). Create two objects of Rectangle and compare their area and colour. If area and colour both are the same for the objects then display "Matching Rectangles", otherwise display "Non-matching Rectangles".

Ans. Java is Platform Neutral: Java is compiled into a general bytecode. It is interpreted by the Java Virtual Machine. Because the code is not compiled into something specific to one platform, it can be interpreted in the same way by multiple JVMs. For example, a user with a Windows JVM can run a program in the same way as a user with a Mac OS JVM can run a program. Therefore, Java as a language has no affiliation with any particular platform and is platform independent.

Program

```
class Rectangle {
private double width;
        private double length;
        private double area;
        private String colour;
        Rectangle (double width, double length,
        String colour)
        {
        this.width = width;
        this.length = length;
        this.colour = colour;
}
        public doubleget_width(){
        return width;'
}
        public double get_length () {
        return length;
        }
        public double find_Area() {
        area = width*length;
        return area:
        }
        public void set_colour(String colour) {
        this.colour = colour;
        }
public static void main (String args[]){
Rectangle r1 = new Rectangle (10, 20, "red");
Rectangle r2 = new Rectangle (10, 20, "blue");
if (r1.find_area() = = r2.find_area()) &&(r1.colour)
equals (r2.colour))
System.out.println("Matching Rectangle");
System.out.println("Non_Matching Rectangle");
}
```

Q. 5 (*a*) What are threads? Briefly explain the difference between a thread and a process.

Ans. Threads: The thread is the smallest unit of dispatchable code. This means that a single program can perform two or more tasks simultaneously. For instance, a text editor can format text at the same time that it is printing, as long as these two actions are being performed by two separate threads. Thus, process-based multitasking deals with the "big picture" and thread-based multitasking handles the details. Multitasking threads require less overheads than multitasking processes.

Difference between thread and process

- Processes are heavyweight tasks that require their own separate address spaces. Threads on the other hand, are lightweight. They share the same address space and cooperatively share the same heavyweight process.
- Interprocess communication is expensive and limited. Context switching from one process to another is also costly. Interthread communication is inexpensive and context switching from one

thread to the next is low cost. While Java programs make use of process-based multitasking environments.

- Process based multitasking is not under the control of Java. However, multithreaded multitasking is.
- Multithreading enables you to write very efficient programs that make maximum use of the CPU, because idle time can be kept to a minimum. This is especially important for the interactive, networked environment in which Java operates, because idle time is common. In a traditional, single-threaded environment, your program has to wait for each of these tasks to finish before it can proceed to the next one—even though the CPU is sitting idle most of the time. Multithreading lets you gain access to this idle time and puts it to good use.

Q.4. (*b*) (*ii*) Develop an applet that receives three numeric values as input from the user and then displays the largest of the three on the screen.

Ans. Applet for three number showing largest number

```
/* <applet code = "MaxOf3No" height = 150 width
= 400> </applet> */
import java.awt.*;
import java.applet.*;
publicclass MaxOf3No extends Applet
{
    TextField T1, T2, T3;

    publicvoid init() {
        T1 = new TextField(10);
        T2 = new TextField(10);
        T3 = new TextField(10);

        add(T1);
        add(T2);
        add(T3);

        T1.setText("0");
        T2.setText("0");
        T3.setText("0");
    }
    publicvoid paint(Graphics g){
        int a, b, c, result;
        String str;
        g.drawString ("Enter value to Check the
        Maximum of 3", 10, 50);
    str=T1.getText();
    a=Integer.parseInt(str);
    str=T2.getText();
    b=Integer.parseInt(str);
    str=T3.getText();
    c=Integer.parseInt(str);

    g.setColor(Colour.blue);
    if (a > b) ;
        if (a > c)
            result = a;
        else
            result = c;
    }
    else {
        if (b > c)
            result=b;
        else
            result=c ;
    }
    g.drawString("Maximum of 3 No is" +result,
    10, 70);
    showStatus ("MAXIMUM OF 3
    NUMBERS");
    }
    public boolean action (Event e, Object o){
        repaint();
        returntrue;
    }
}
```

Q.4. (*c*) Explain how a Java GUI component handles the event with an example. Write a program in Java which creates three buttons labelled "Yes", "No" and "Undecided" Each time one is pressed, a message is displayed that reports which button has been pressed.

Ans. We consider the class "Event", which are all derived from java.util.EventObject. Examples of events are "button pushed", "key pushed", "mouse moved". Some subclasses of Event are **ActionEvents** and **WindowEvents**. Following steps are followed for event handling

(1) Implement the listener interface using any reasonable class. In our example, the class will be an extension of JPanel. To implement the ActionListener, you need to supply the method **actionPerformed(ActionEvent)**. The class for implementing Actionlistener here is "MyPanel":

```
public class MyPanel extends JPanel
   implements ActionListener {
public void actionPerformed(ActionEvent e){
   // reaction to button click goes here
      ...
} // actionPerformed
} // class MyPanel
```

(2) Create a listener object:

```
Listener lis = new MyPanel();
```

(3 register this object with the event detector.

```
button.addActionListener(lis);
```

Program

```
import java.awt.BorderLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.JPanel;

    public class TestEvent {
    public static void main(String[] args) {
        final JFrame frame = new JFrame();
        JButton btnyes = new JButton("YES");
        btnyes.addActionListener(
             new ActionListener(){
             public void actionPerformed(ActionEvent e) {
             JOptionPane.showMessageDialog(frame,"You've clicked YES
button");
             }
        });

    JButton btnno = new JButton("NO");
    btnno.addActionListener(
```

```
            new ActionListener(){
            public void actionPerformed(ActionEvent e) {
            JOptionPane.showMessageDialog(frame,"You've clicked NO button");
            }
            });
            JButton btnun= new JButton("Undecided");
            btnun.addActionListener(
                 new ActionListener(){
            public void actionPerformed(ActionEvent e) {
            JOptionPane.showMessageDialog(frame,"You've clicked Undecided
button");
            }
            });
            // Add buttons to a panel
            JPanel buttonPanel = new JPanel( );
            buttonPanel.add(btnyes);
            buttonPanel.add(btnno);
            buttonPanel.add(btnun);

            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            frame.setSize(300, 200);
            frame.getContentPane( ).add(buttonPanel,BorderLayout.SOUTH);
            frame.setVisible(true);
            }

    }
```

5. Attempt any two parts of the following: **(10 × 2 = 20)**

Q. 5 (*a*) (*i*) Describe the different types of JDBC drives with examples.

Ans. JDBC drivers are divided into four types or levels. The different types of **JDBC** drivers are:

Type 1: JDBC-ODBC Bridge driver (Bridge)

Type 2: Native-API/partly Java driver (Native)

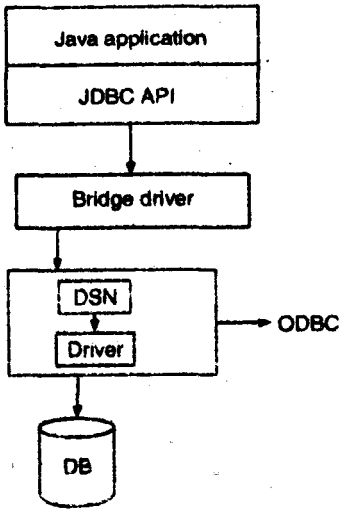
Type 3: AllJava/Net-protocol driver (Middleware)

Type 4: All Java/Native-protocol driver (Pure)

Four types of JDBC drivers are elaborated in detail as shown below:

Type 1 JDBC Driver

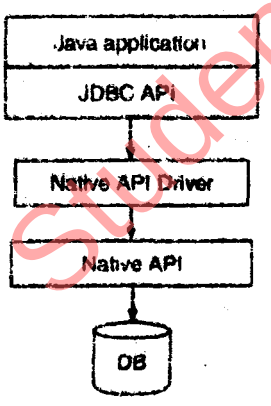
JDBC-ODBC Bridge driver : The type 1driver translates all JDBC calls into ODBC calls and sends them to the ODBC driver. ODBC is a generic API. The JDBC-ODBC Bridge driver is recommended only for experimental use or when no other alternative is available.



Type 1: JDBC-ODBC Bridge

Type 2 JDBC Driver

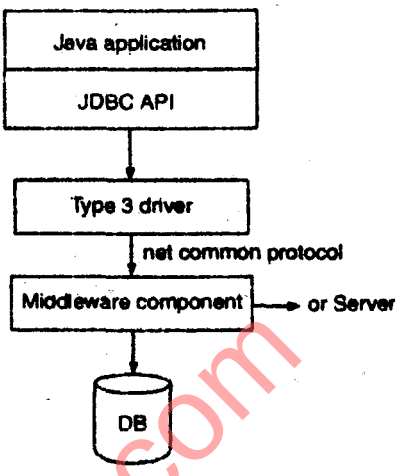
Native-API/partly java driver: The distinctive characteristics of type 2 jdbc drivers are that Type 2 drivers convert JDBC calls into database-specific calls *i.e.*, this driver is specific to a particular database. Some distinctive characteristics of type 2 jdbc drivers are shown below. Example: Oracle will have oracle native API.



Type 2: Native api/partly Java driver

Type 3 JDBC Driver

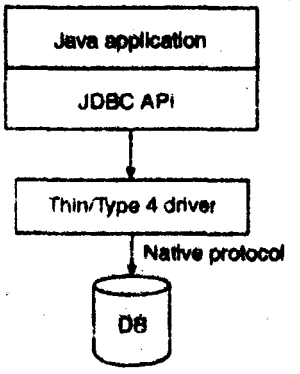
All Java/Net-protocol driver: Type 3 database requests are passed through the network to the middle-tier server. The middle tier then translates the requedsts to the database. If the middle-tier server can in turn use Type 1, Type 2 or Type 4 drivers.



Type 3: All Java/Net-protocol driver

Type 4: JDBC-protocol/all-java driver

The type 4 uses java networking libraries to communicate directly with the database server.



Type 4: Native-protocol/all-Java driver

Q. 5 (*a*) (*ii*) Write an interactive program in Java that can update and delete employee information via an employee database using statement class. Make suitable assumption yourself.

Ans.

```
Import java.io*;
Impon java.sql.*;
Class A
{
Public static void main (String args[])
{
Class.forName("sun.jdbc.orbc.jdbcOdbcDriver");
Connection con=DriverManager.getConnection();
Statementstmt=con.createStatement(con);
```

```
Stmt.executeupdate("deletef from tablename where empid-001");
Stmt. execute update ("updatetablename set empname=" ram" where empid=001");
con.close();
}
}
```

Q. 5 (*b*) Describe the difference between CGI and Servlet. Explain how a Servlet handles HTTP requests via an example. Write a simple Servlet program displaying the current Data for each request.

Ans. 1. Servlets are effectively a Java version of CGI scripts, which are written in perl, C, C ++, UNIX shell scripts etc.

2. Increase CGI creates a process for every execution. So, this is a time taking process whereas servlet executes by using threadings, So this is light weight process. Hence Servlets are in more use

3. CGI is a process based (Heavyweight) and Servlet is a Thread based (Lightweight)

How Servlet handles HTTP request

1. Http Servlet: The AuctionServlet class extends HttpServlet, which is an abstract class.

```
public class AuctionServlet extends HttpServlet{
```

2. A servlet can be either loaded when the web server starts up or loaded when requested by way of an HTTP URL that specifies the servlet. The servlet is usually loaded by a separate classloader in the web server because this allows the servlet to be reloaded by unloading the class loader that loaded the servlet class. However, if the servlet depends on other classes and one of those classes changes, you will need to update the date stamp on the servlet for it to reload.

3. After a servlet loads, the first stage in its lifecycle is the web server calls the servlets init method. Once loaded and initialized the next stage in the servlets lifecycle is to serve requests. The servlets serves requests through its servers, doGet, or doPost method implementations.

The servlet can optionally implement a destroy method to perform cleanup operations before the seb server unloads the servlet.

Program to Display Current Date and time by Servlet

```
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class DataServlet extends HttpServlet{

public void deoGet (HttpServletRequest resuest, "<HTML>" serviceResponse response) throws
ServletException, IOException{

PrintWriter pw = response.getWrite()'

Data today = new Date();

pw.println"<(html>" "+"<body bg color="\"#99996\>

<h1> Date and Time with Servlet (< h1 >);
pw.println("<b>" + today+"<\b> <\body>" + "/html");
```

Q. 5 (*c*) Write short notes on any two:

(*i*) Java AWT

(*ii*) Dynamic Billboard Applet

(*iii*) Java Beans

Ans. (*i*) Please see Q.4(*a*) of 2008-09

(*ii*) Please see Q.5(*d*) of 2007-08

(*iii*) Please see Q.5(*c*) of 2009-10