Graphic Era Hill University, Dehradun
(Answer Sheet for Online Examination ~~Aug~~. 2021)

Please tick (√) your campus: (DEHRADUN/BHIMTAL/HALDWANI)

Name: Deepankar Sharma Univ.Roll No. 2092014 Student ID 20041299
Date: Jan 10, 2022 Course: BCA Branch: Sem.: 03 Section:
Subject Name: Java Programming Subject Code: TBC.301 Page No.

Ques 5) A - /* table "employee"
database : "db1" */
/* id, name, price */
// Java Program

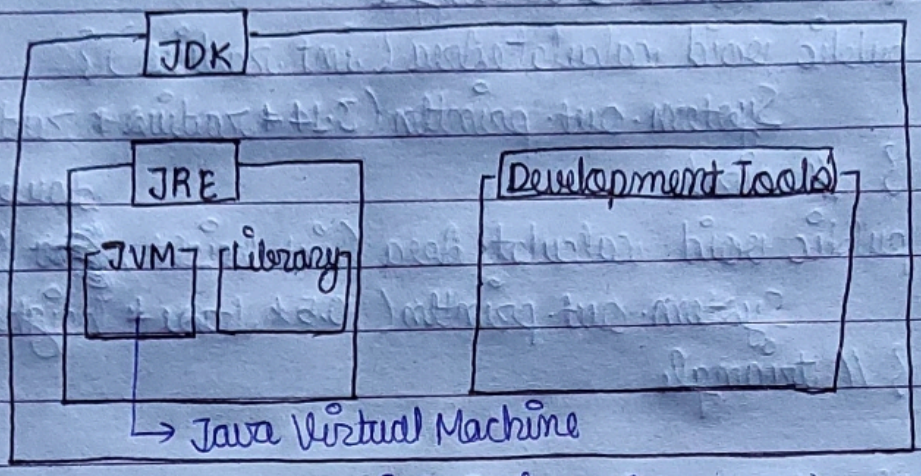
```
import java.sql.*;          // Step 1 → Importing the package
import java.util.Scanner;
public class test {
    public static void main (String[] args) throws Exception {
        int id; String name; double price;
        Scanner sc = new Scanner (System.in);
        id = sc.nextInt();
        name = sc.next();
        price = sc.nextDouble();
String url = "jdbc:mysql://localhost:3306/db1"; // url to db1
        Class.forName ("com.jdbc.Driver"); // load and register
                                           driver Step ②
    for mysql → "com.mysql.cj.jdbc.Driver"
        Connection con = DriverManager.getConnection (url,
                                "username", "password");
    //establishing connection → Step ③
PreparedStatement pst = con.prepareStatement ("insert into employee
                        values (?,?,?); // Step ④ → statement
pst.setInt (1, id);
pst.setString (2, name); pst.setDouble (3, price);
int c = pst.executeUpdate(); // Step ⑤ → execute Query
if (c!=0) System.out.println ("success"); // Step ⑥ → Process Result
pst.close(); con.close(); // Step ⑥ → close connection and
} // end of main                          statement
} // end of class
```

PAGE NO.:
DATE: / /20

Name - Deepankar Sharma
University Roll No - 2092014
Student ID - 20041299
Subject Name - Java Programming
Subject Code - TBC-301

Date - January 10, 2022
Course - BCA
Sem - 03

① (A)



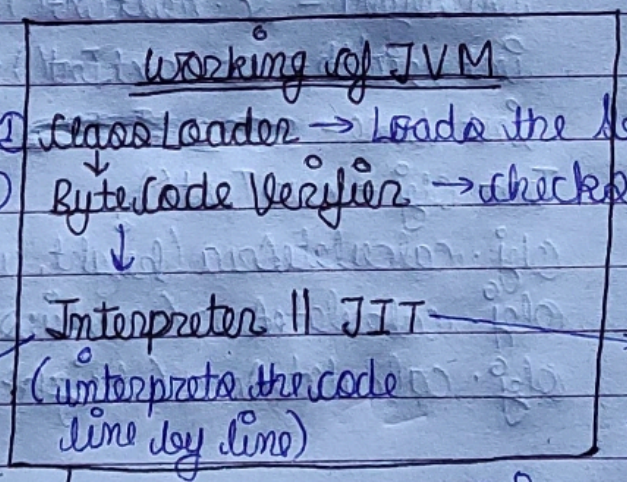
Source code → The original file with (.java) extension (Test.java)

↓

Compilation → Compiled by Java compiler (.class)
> javac Test.java

↓

Byte Code



correlates and

Libraries classes

② ③

software interpreter written in C language

JIT (Just in Time Compiler)
It is used to optimize redundant byte code

↓

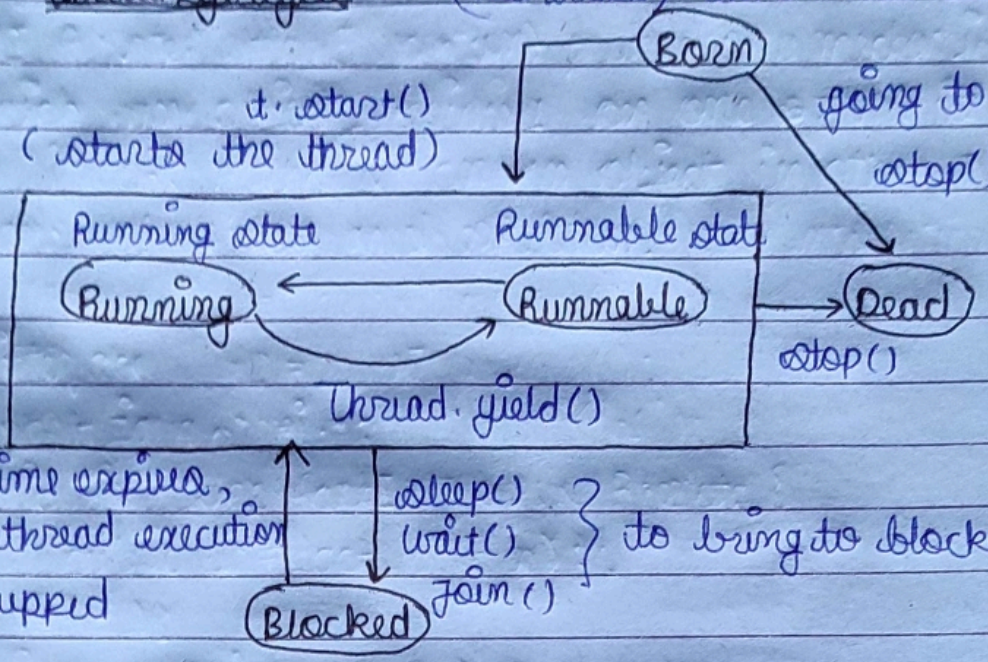
Machine code / OS/CPU

This is how JVM provides Architecture Neutrality to the code.

PAGE NO.:
DATE: / /20

```
/* Java Program */
import java.util.Scanner;
class Test {
    public void calculateArea(int height, int width) {
        System.out.println(height * width);
    } // square & rectangle
    public void calculateArea(int radius) {
        System.out.println(3.14 * radius * radius);
    } // circle
    public void calculateArea(int height, double base) {
        System.out.println(0.5 * base * height);
    } // triangle
}
public class MainClass {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int height = sc.nextInt();
        int width = sc.nextInt(); int radius = sc.nextInt();
        int base = sc.nextDouble();
        Test obj = new Test();
        obj.calculateArea(height, width);
        obj.calculateArea(radius);
        obj.calculateArea(height, base);
    }
}
```

Ques ② (a) Thread Life Cycle (new thread) Test t = new Test()




```
// demonstrate thread sync.
class test{
    int a;
    synchronized void show(){
        System.out.println("value of a is" + a);
    }
}
class testThread extends java.lang.Thread { test obj;
    public void run(){
        obj.a += 10;  obj.show();
    }
}
class testSync {
    public static void main(String[] args){
        Test ob = new Test(); ob.a = 5;
        testThread t1 = new testThread(); t1.obje = ob;
        testThread t2 = new testThread(); t2.obj = ob;
        t1.start();  t2.start();
    }}
```

here both t1 and t2 are sharing same object of 'test' class

PAGE NO.:
DATE: / /20

Ques ③ (B) Remote Method Invocation

RMI (Remote Method Invocation) provides a mechanism through which an object residing on one JVM is allowed to invoke methods on an object in other JVM by means of stubs and skeleton.

```
// interface
import java.rmi.*;
public interface FactI extends Remote {
    public int fact(int n);
}
// implementation of Remote Interface
import java.rmi.server.*;
public class FactTest extends UnicastRemoteObject
    implements FactI {
    public FactTest() throws Exception { super(); }
    public int fact(int n){
        if(n==0 || n==1) { return 1; }
        return n*fact(n-1);
    }
}
// Server
import java.rmi.*;
public class Server {
    public static void main(String[] args){
        FactTest obj = new FactTest();
        Naming.rebind("fact", obj);
    }
}
```

PAGE NO.:
DATE: / /20

```
//client
import java.rmi.*;
public class Client {
    public static void main(String[] args) {
        FactI x = (FactI) Naming.lookup("fact");
        int n = x.fact(4);
        System.out.println("Factorial of 4 is" + n);
    }
}
```

PAGE NO.:
DATE: / /20

// Swing GUI

Ques ④ (B)

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class test extends JFrame implements ActionListener{
    JTextField t1 = new JTextField();
    JTextField t2 = new JTextField();
    JButton b1 = new JButton("b1");
    JButton b2 = new JButton("b2");
    JLabel l1 = new JLabel();
    test(){
        setVisible(true);
        setLayout(new FlowLayout());
        b1.addActionListener(this);
        b2.addActionListener(this);
      add(t1); add(t2); add(b1); add(b2); add(l1);
    }
  public void actionPerformed(ActionEvent ae){
      int num1 = Integer.parseInt(t1.getText());
      int num2 = Integer.parseInt(t2.getText());
      if(ae.getSource()==b1) num1 = num1+num2;
      if(ae.getSource()==b2) num1 = num1-num2;
      t1.setText(String.valueOf(num1));
  }
  public static void main(String[] args){
      new test();
  }
}
```