1. Write a program in Java with class Rectangle with the data fields width,length, area and color .The length, width and area are of double type and color is of string type .The methods are setLength() , setWidth (), setColor(), and findArea(). Create two object of Rectangle and compare their area and color. If area and color both are same for the objects then display “Matching Rectangles” otherwise display “Non matching Rectangle”.

```
class Rectangle {
    private double length;
    private double width;
    private double area;
    private String color;

    public void setLength(double length) {
        this.length = length;
    }

    public void setWidth(double width) {
        this.width = width;
    }

    public void setColor(String color) {
        this.color = color;
    }

    public void findArea() {
        this.area = length * width;
    }

    public double getArea() {
        return area;
    }

    public String getColor() {
        return color;
    }
}
```

```
public class RectangleComparison {
    public static void main(String[] args) {
        // Create two Rectangle objects
        Rectangle rect1 = new Rectangle();
        rect1.setLength(5.0);
        rect1.setWidth(4.0);
        rect1.setColor("Blue");
        rect1.findArea();

        Rectangle rect2 = new Rectangle();
        rect2.setLength(5.0);
        rect2.setWidth(4.0);
        rect2.setColor("Blue");
        rect2.findArea();

        // Compare the area and color of the rectangles
        if (rect1.getArea() == rect2.getArea()
             && rect1.getColor().equals(rect2.getColor())) {
            System.out.println("Matching Rectangles");
        } else {
            System.out.println("Non-matching Rectangles");
        }
    }
}
```

2. Create a class Account with two overloaded constructors. First constructor is used for initializing, name of account holder, account number and initial amount in account. Second constructor is used for initializing name of account holder, account number, addresses, type of account and current balance. Account class is having methods deposit(), withdraw(), and getBalance().Make necessary assumption for data members and return types of the methods. Create objects of Account class and use them.

```
class Account {
    private String accountHolderName;
    private int accountNumber;
    private double balance;
    private String address;
    private String accountType;
    public Account(String accountHolderName, int accountNumber, double initialAmount) {
        this.accountHolderName = accountHolderName;
        this.accountNumber = accountNumber;
        this.balance = initialAmount;
    }
    public Account(String accountHolderName, int accountNumber, String address, String accountType, double currentBalance) {
        this.accountHolderName = accountHolderName;
        this.accountNumber = accountNumber;
        this.address = address;
        this.accountType = accountType;
        this.balance = currentBalance;
    }
    public void deposit(double amount) {
        balance += amount;
    }
    public void withdraw(double amount) {
        if (amount <= balance) {
            balance -= amount;
        } else {
            System.out.println("Insufficient balance");
        }
    }
    public double getBalance() {
        return balance;
    }
}
```

```
public class AccountExample {
    public static void main(String[] args) {
        // Create an account using the first constructor
        Account account1 = new Account("John Smith", 12345, 1000.0);

        // Deposit some amount
        account1.deposit(500.0);

        // Withdraw an amount
        account1.withdraw(200.0);

        // Get the balance
        double balance1 = account1.getBalance();
        System.out.println("Account 1 balance: " + balance1);

        // Create an account using the second constructor
        Account account2 = new Account("Jane Doe", 54321, "123 Main St", "Savings", 2000.0);

        // Withdraw from account 2
        account2.withdraw(1500.0);

        // Get the balance
        double balance2 = account2.getBalance();
        System.out.println("Account 2 balance: " + balance2);
    }
}
```

3. Write a program in Java to create a stack class with push() and pop () methods. Create two objects of stack with 10 data item in both. Compare the top elements of both stack and print the comparison result.

```
class Stack {
    private int maxSize;
    private int top;
    private int[] stackArray;

    public Stack(int maxSize) {
        this.maxSize = maxSize;
        this.top = -1;
        this.stackArray = new int[maxSize];
    }

    public void push(int item) {
        if (top < maxSize - 1) {
            stackArray[++top] = item;
        } else {
            System.out.println("Stack is full");
        }
    }

    public int pop() {
        if (top >= 0) {
            return stackArray[top--];
        } else {
            System.out.println("Stack is empty");
            return -1;
        }
    }

    public int peek() {
        if (top >= 0) {
            return stackArray[top];
        } else {
            System.out.println("Stack is empty.");
            return -1;
        }
    }
}
```

```
public class StackComparison {
    public static void main(String[] args) {
        // Create stack 1
        Stack stack1 = new Stack(10);
        for (int i = 1; i <= 10; i++) {
            stack1.push(i);
        }

        // Create stack 2
        Stack stack2 = new Stack(10);
        for (int i = 10; i >= 1; i--) {
            stack2.push(i);
        }

        // Compare the top elements of both stacks
        int topElement1 = stack1.peek();
        int topElement2 = stack2.peek();
        if (topElement1 == topElement2) {
            System.out.println("Top elements are same");
        } else {
            System.out.println("Top elements are different.");
        }
    }
}
```

4. Write a program in Java to create a Player class. Intermit classes CricketPlayer, FootballPlayer and HockeyPlayer from Player class.

```
class Player {
    private String name;

    public Player(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }
}

class CricketPlayer extends Player {
    private String team;

    public CricketPlayer(String name, String team) {
        super(name);
        this.team = team;
    }

    public String getTeam() {
        return team;
    }
}
```

```
class FootballPlayer extends Player {
    private String club;

    public FootballPlayer(String name, String club) {
        super(name);
        this.club = club;
    }

    public String getClub() {
        return club;
    }
}

class HockeyPlayer extends Player {
    private String country;

    public HockeyPlayer(String name, String country) {
        super(name);
        this.country = country;
    }

    public String getCountry() {
        return country;
    }
}
```

```
public class PlayerExample {
    public static void main(String[] args) {
        CricketPlayer cricketPlayer = new CricketPlayer("Virat Kohli", "India");
        FootballPlayer footballPlayer = new FootballPlayer("Lionel Messi", "Barcelona");
        HockeyPlayer hockeyPlayer = new HockeyPlayer("Sidney Crosby", "Canada");

        System.out.println("Cricket Player: " + cricketPlayer.getName() + ", Team: " + cricketPlayer.getTeam());
        System.out.println("Football Player: " + footballPlayer.getName() + ", Club: " + footballPlayer.getClub());
        System.out.println("Hockey Player: " + hockeyPlayer.getName() + ", Country: " + hockeyPlayer.getCountry());
    }
}
```

5. Write a class Worker and derive classes DailyWorker and SalariedWorker from it. Every worker is has a name and a salary rate. Write method computePay(int hours) to compute the week pay of every worker. A DailyWorker is paid on the basis of number of days he/she work. The SalariedWorker gets paid the wage for 40 hours a week no matter what actual hours is. Test this program to calculate the pay of workers. You are expected to use concept of polymorphism to write this program.

```
class Worker {
    private String name;
    private double salaryRate;
    public Worker(String name, double salaryRate) {
        this.name = name;
        this.salaryRate = salaryRate;
    }
    public double computePay(int hours) {
        return salaryRate * hours;
    }
}

class DailyWorker extends Worker {
    private int daysWorked;
    public DailyWorker(String name, double salaryRate, int daysWorked) {
        super(name, salaryRate);
        this.daysWorked = daysWorked;
    }
    public double computePay(int hours) {
        return super.computePay(daysWorked * hours);
    }
}
class SalariedWorker extends Worker {
    public SalariedWorker(String name, double salaryRate) {
        super(name, salaryRate);
    }
    public double computePay(int hours) {
        return super.computePay(40);
}
```

```
public class WorkerExample {
    public static void main(String[] args) {
        // Create worker objects
        Worker worker1 = new DailyWorker("John", 15.0, 4);
        Worker worker2 = new SalariedWorker("Jane", 20.0);

        // Compute and display their pay
        System.out.println("Worker 1 (Daily Worker): "  + worker1.computePay(8) + " INR");
        System.out.println("Worker 2 (Salaried Worker): " + worker2.computePay(30) + " INR");
    }
}
```

6. Write a program in Java to display name and roll number of students. Initialize respective array variables for 10 students. Handle ArrayIndexOutOfBoundsExeption, so that any such problem doesn't cause illegal termination of program

```
public class StudentDetails {
    public static void main(String[] args) {
        // Initialize arrays for names and roll numbers of students
        String[] names = { "John", "Emma", "Michael", "Olivia", "William", "Sophia", "James", "Isabella", "Benjamin", "Mia" };
        int[] rollNumbers = { 101, 102, 103, 104, 105, 106, 107, 108, 109, 110 };

        try {
            // Display the details of each student
            for (int i = 0; i < 12; i++) {  // Intentionally accessing 12 elements to trigger ArrayIndexOutOfBoundsException
                System.out.println("Name: " + names[i] + ", Roll Number: " + rollNumbers[i]);
            }
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Error: Array index out of bounds!");
        }
    }
}
```

7. On singles track two vehicles are running for as vehicles are going in same direction there is no problem. If the vehicles are running in same direction there is a chance of collision. To avoid collisions, write a java program using exception handling. You are free to make necessary assumptions.

```
class VehicleCollisionException extends Exception {
    public VehicleCollisionException(String message) {
        super(message);
    }
}

class Vehicle {
    private String name;
    private int position;

    public Vehicle(String name) {
        this.name = name;
        this.position = 0;
    }

    public void move(int distance) throws VehicleCollisionException {
        int newPosition = position + distance;

        // Check if there is a chance of collision with another vehicle
        if (newPosition >= 0 && newPosition <= 100) {
            position = newPosition;
            System.out.println(name + " moved to position " + position);
        } else {
            throw new VehicleCollisionException(name + " is at risk of colliding with another vehicle!");
        }
    }
}
```

```
public class VehicleSimulation {
    public static void main(String[] args) {
        Vehicle car = new Vehicle("Car");
        Vehicle truck = new Vehicle("Truck");

        try {
            car.move(50);
            truck.move(80);
            car.move(20);
            truck.move(40);
            car.move(30);
        } catch (VehicleCollisionException e) {
            System.out.println("Vehicle collision detected: " + e.getMessage());
        }
    }
}
```

8. Write a java program using thread synchronization in multithreading

```
class Test {
    public static void main(String args[]) {
        final Customer c = new Customer();
        new Thread() {
            public void run() {
                c.withdraw(15000);
            }
        }.start();
        new Thread() {
            public void run() {
                c.deposit(10000);
            }
        }.start();
    }
}
```

```
class Customer {
    int amount = 10000;
    synchronized void withdraw(int amount) {
        System.out.println("going to withdraw...");
        if (this.amount < amount) {
            System.out.println("Less balance; waiting for deposit...");
            try {
                wait();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        this.amount -= amount;
        System.out.println("withdraw completed...");
    }
    synchronized void deposit(int amount) {
        System.out.println("going to deposit...");
        this.amount += amount;
        System.out.println("deposit completed... ");
        notify();
    }
}
```

```
class TestSynchronization1 {
    public static void main(String args[]) {
        Table obj = new Table(); //only one object
        MyThread1 t1 = new MyThread1(obj);
        MyThread2 t2 = new MyThread2(obj);
        t1.start();
        t2.start();
    }
}
```

9. Write a program in java to create a String object. Initialize this object with your name. Find the length of your name using appropriate String method. Find whether character “a” is in your name or not, if yes find the number of time “a” it appear in your name. Print locations of occurrences of “a”. Try same for different String objects.

```
public class StringOperations {
    public static void main(String[] args) {
        // Create a String object and initialize it with a name
        String name = args[0];
        // Find the length of the name
        int length = name.length();
        System.out.println("Length of the name: " + length);
        // Check if character "a" is present in the name
        boolean containsA = name.contains("a");
        System.out.println("Character 'a' is present in the name: " + containsA);
        if (containsA) {
            // Count the occurrences of character "a" in the name
            int count = 0;
            for (int i = 0; i < length; i++) {
                if (name.charAt(i) == 'a') {
                    count++;
                }
            }
            System.out.println("Number of times 'a' appears in the name: " + count);
            // Print the locations of occurrences of character "a"
            System.out.print("Locations of occurrences of 'a': ");
            for (int i = 0; i < length; i++) {
                if (name.charAt(i) == 'a') {
                    System.out.print(i + " ");
                }
            }
            System.out.println();
        }
    }
}
```

10.Write a program in java to read a statement from console, convert it in to uppercase and again print on console.

```
import java.util.Scanner;

public class UppercaseConversion {
    public static void main(String[] args) {
        // Create a Scanner object to read input from the console
        Scanner scanner = new Scanner(System.in);

        // Prompt the user to enter a statement
        System.out.print("Enter a statement: ");

        // Read the input statement from the console
        String statement = scanner.nextLine();

        // Convert the statement to uppercase
        String uppercaseStatement = statement.toUpperCase();

        // Print the uppercase statement
        System.out.println("Uppercase Statement: " + uppercaseStatement);

        // Close the scanner
        scanner.close();
    }
}
```

11.Write a java program to copy a file into another file.

```
import java.io.*;

public class FileCopy {
    public static void main(String[] args) {
        String sourceFile = "c:/ignou/srcfile.txt";
        String destinationFile = "c:/ignou/destfile.txt";

        try {
            File inputFile = new File(sourceFile);
            File outputFile = new File(destinationFile);

            FileInputStream fis = new FileInputStream(inputFile);
            FileOutputStream fos = new FileOutputStream(outputFile);

            byte[] buffer = new byte[1024];
            int bytesRead;

            while ((bytesRead = fis.read(buffer)) != -1) {
                fos.write(buffer, 0, bytesRead);
            }

            System.out.println("File copied successfully.");

            fis.close();
            fos.close();
        } catch (IOException e) {
            System.out.println("An error occurred while copying the file: " + e.getMessage());
        }
    }
}
```

12. Write a Java Applet program, which provide a text area with horizontal and vertical scrollbars. Type some lines of text in text area and use scrollbars for movements in text area. Read a word in a text field and find whether the word is in the content of text area or not.

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
public class TextSearchApplet extends Applet implements ActionListener {
    private TextArea textArea;
    private TextField searchField;
    private Button searchButton;
    public void init() {
        setLayout(new BorderLayout());
        textArea = new TextArea(10, 30);
        textArea.setEditable(true);
        searchField = new TextField(20);
        searchButton = new Button("Search");
        searchButton.addActionListener(this);
        add(textArea, BorderLayout.CENTER);
        add(searchField, BorderLayout.NORTH);
        add(searchButton, BorderLayout.SOUTH);
    }
    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == searchButton) {
            String content = textArea.getText();
            String word = searchField.getText();
            if (content.contains(word)) {
                System.out.println("Word '" + word + "' found in the text area.");
            } else {
                System.out.println("Word '" + word + "' not found in the text area.");
            }
        }
    }
}
```

13. Write a Java program to find the numeric address of the following websites.
    a. www.ignou.ac.in
    b. www.indiatimes.com
    c. www.rediff.com
    d. www.apple.com

```
import java.net.InetAddress;
import java.net.UnknownHostException;

public class WebsiteAddress {
    public static void main(String[] args) {
        String[] websites = {
            "www.ignou.ac.in",
            "www.indiatimes.com",
            "www.rediff.com",
            "www.apple.com"
        };

        for (String website : websites) {
            try {
                InetAddress address = InetAddress.getByName(website);
                System.out.println("Website: " + website);
                System.out.println("Numeric Address: " + address.getHostAddress());
                System.out.println();
            } catch (UnknownHostException e) {
                System.out.println("Unable to find address for " + website);
                System.out.println();
            }
        }
    }
}
```

14.Write a program to test Socket functionality for appropriate hostname and port number

```
import java.io.IOException;
import java.net.Socket;

public class SocketTest {
    public static void main(String[] args) {
        String hostname = "www.ignou.ac.in";
        int port = 80;

        try {
            Socket socket = new Socket(hostname, port);
            System.out.println("Socket connection established with " + hostname + " on port " + port);
            socket.close();
        } catch (IOException e) {
            System.out.println("Error connecting to " + hostname + " on port " + port + ": " + e.getMessage());
        }
    }
}
```

15.Database example

INSERT

```
import java.sql.*;
class StudentInsertP{
    public static void main(String as[]){
        try{
            String name = as[0];
            int age = Integer.parseInt(as[1]);
            String course = as[2];
            String SQL = "INSERT INTO tabstudent(sname,sage,scourse) VALUES(?,?,?)";
            //load the driver(bridge creation)
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            //Connect the driver to the DSN
            Connection con = DriverManager.getConnection("jdbc:odbc:dsnschool");
            //Allocate space in memory to execute the query
            PreparedStatement pstmt = con.prepareStatement(SQL);
            //set the parameters
            pstmt.setString(1,name);
            pstmt.setInt(2,age);
            pstmt.setString(3,course);
            //Execute the query
            pstmt.executeUpdate();
            //Close the connection
            pstmt.close();
            con.close();
            System.out.println("INSERTED Successfuly");
        }
        catch(Exception err){
            err.printStackTrace();
        }
    }
}
```

VIEW

```
import java.sql.*;
class ViewStudents{
        public static void main(String as[]){
                try{
                        String SQL = "SELECT sid,sname,sage,scourse FROM tabStudent";
                        //load the driver(bridge creation)
                        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
                        //Connect the driver to the DSN
                        Connection con = DriverManager.getConnection("jdbc:odbc:dsnschool");
                        //Allocate space in memory to execute the query
                        Statement stmt = con.createStatement();
                        //Execute the query
                        ResultSet rs = stmt.executeQuery(SQL);
                        System.out.println("ID\tNAME\tAGE\tCOURSE");
                        System.out.println("-------------------------------------------------------");
                        while(rs.next()){
                                int id = rs.getInt("sid");
                                String name = rs.getString("sname");
                                String age = rs.getString("sage");
                                String course = rs.getString("scourse");
                                //display the output
                                System.out.println(id+"\t"+name+"\t"+age+"\t"+course);
                        }
                        //Close the connection
                        stmt.close();
                        con.close();
                }
                catch(Exception err){
                        err.printStackTrace();
                }
        }
}
```