Slip 1

Q2. Q.2) Write a JDBC program to display all the details of the Person table in proper format on the screen. Create a Person table with fields as PID, name, gender, birth_year. Insert values in Person table.

```
import java.sql.*;


public class PersonDetails {
    public static void main(String[] args) {
        try {
            Class.forName("org.postgresql.Driver");
            Connection con = DriverManager.getConnection(
                "jdbc:postgresql1:DYP", "postgres", "");


            Statement st = con.createStatement();
            ResultSet rs = st.executeQuery("SELECT * FROM person");


            System.out.println("PID\tName\tGender\tBirth Year");
            System.out.println("----------------------------------");


            while (rs.next()) {
                System.out.print(rs.getInt("PID") + "\t");
                System.out.print(rs.getString("name") + "\t");
                System.out.print(rs.getString("gender") + "\t");
                System.out.println(rs.getInt("birth_year"));
            }


            con.close();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

Slip 2

Q2. Write a JDBC program to display all the countries located in the West Region. Create a table Country with fields (Name, continent, Capital,Region). Insert values in the table.

```
import java.sql.*;

public class WestRegionCountries {
    public static void main(String[] args) {
        try {
            Class.forName("org.postgresql.Driver");
            Connection con = DriverManager.getConnection(
                "jdbc:postgresql1:DYP", "postgres", "");

            Statement st = con.createStatement();
            ResultSet rs = st.executeQuery("SELECT * FROM Country WHERE Region = 'West'");

            System.out.println("Name\tContinent\tCapital\tRegion");
            System.out.println("---------------------------------------------");

            while (rs.next()) {
                System.out.print(rs.getString("Name") + "\t");
                System.out.print(rs.getString("Continent") + "\t");
                System.out.print(rs.getString("Capital") + "\t");
                System.out.println(rs.getString("Region"));
            }

            con.close();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

Slip 3.

Q2 Write a JDBC program to insert the records into the table Employee(ID,name,salary) using PreparedStatement interface. Accept details of Employees from user.

```
import java.sql.*;
import java.util.Scanner;

public class InsertEmployee {
    public static void main(String[] args) {
        try {
            Class.forName("org.postgresql.Driver");
            Connection con = DriverManager.getConnection(
                "jdbc:postgresql1:DYP", "postgres", "");

            String query = "INSERT INTO Employee (ID, name, salary) VALUES (?, ?, ?)";
            PreparedStatement ps = con.prepareStatement(query);
            Scanner sc = new Scanner(System.in);


            System.out.print("Enter Employee ID: ");

            int id = sc.nextInt();

            sc.nextLine(); // consume newline


            System.out.print("Enter Employee Name: ");

            String name = sc.nextLine();


            System.out.print("Enter Employee Salary: ");

            double salary = sc.nextDouble();


            // Set values in PreparedStatement

            ps.setInt(1, id);

            ps.setString(2, name);

            ps.setDouble(3, salary);


            int rows = ps.executeUpdate();

            System.out.println(rows + " record inserted successfully.");


            con.close();

            sc.close();

        } catch (Exception e) {

            System.out.println(e);

        }

    }
```

}

Slip 4.

Q2 Write a JDBC program to update number_of_students of “BCA Science” to 1000.Create a table Course (Code,name, department,number_of_students). Insert values in the table.

```
import java.sql.*;


public class UpdateCourse {
   public static void main(String[] args) {
      try {
         Class.forName("org.postgresql.Driver");
         Connection con = DriverManager.getConnection(
            "jdbc:postgresql1:DYP", "postgres", "");


         String query = "UPDATE Course SET number_of_students = 1000 WHERE name = 'BCA
Science'";
         Statement st = con.createStatement();
         int rows = st.executeUpdate(query);


         System.out.println(rows + " record(s) updated successfully.");


         con.close();
      } catch (Exception e) {
         System.out.println(e);
      }
   }
}
```

Slip 5

Q2Write a client-server program which displays the server machine’s date and time on

the client machine.

ServerDateTime.java)

```
import java.io.*;
import java.net.*;
import java.util.Date;

public class ServerDateTime {
    public static void main(String[] args) throws Exception {
        ServerSocket server = new ServerSocket(5000);
        System.out.println("Server is running...");

        Socket s = server.accept();
        PrintWriter out = new PrintWriter(s.getOutputStream(), true);

        Date date = new Date();
        out.println("Server Date and Time: " + date.toString());

        s.close();
        server.close();
    }
}
```

ClientDateTime.java

```
import java.io.*;
import java.net.*;
```

```
public class ClientDateTime {
   public static void main(String[] args) throws Exception {
      Socket s = new Socket("localhost", 5000);
      BufferedReader in = new BufferedReader(new InputStreamReader(s.getInputStream()));

      String msg = in.readLine();
      System.out.println("Received from server: " + msg);

      s.close();
   }
}
```

Slip 6

Q2. Define a thread called “PrintText_Thread” for printing text on command prompt for n number of times. Create three threads and run them. Pass the text and n as parameters to the thread constructor. Example:

i. First thread prints “I am in FY” 10 times

ii. Second thread prints “I am in SY” 20 times

iii. Third thread prints “I am in TY” 30 times

```
public class PrintTextThread extends Thread {
   private String text;
   private int n;

   // Constructor to accept the text and the number of prints
   public PrintTextThread(String text, int n) {
      this.text = text;
```

```
        this.n = n;
    }

    // Override run method to define thread's behavior
    @Override
    public void run() {
        for (int i = 0; i < n; i++) {
            System.out.println(text);
        }
    }

    // Main method to run the threads
    public static void main(String[] args) {
        // Creating threads with different text and repetition count
        PrintTextThread thread1 = new PrintTextThread("I am in FY", 10);
        PrintTextThread thread2 = new PrintTextThread("I am in SY", 20);
        PrintTextThread thread3 = new PrintTextThread("I am in TY", 30);

        // Starting the threads
        thread1.start();
        thread2.start();
        thread3.start();
    }
}
```

Slip 10

Q2Write a java program using multithreading to execute the threads sequentially.

(Use Synchronized Method)

```
public class SequentialThreads extends Thread {
    private String text;
    private int n;

    public SequentialThreads(String text, int n) {
        this.text = text;
        this.n = n;
    }

    // Synchronized method to ensure thread execution is sequential
    public synchronized void printText() {
        for (int i = 0; i < n; i++) {
            System.out.println(text);
            try {
                Thread.sleep(100); // Add a slight delay to make thread switching visible
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }

    @Override
    public void run() {
        printText();
```

```
}

public static void main(String[] args) {
    // Creating threads with different messages and repetition counts
    SequentialThreads thread1 = new SequentialThreads("I am in FY", 10);
    SequentialThreads thread2 = new SequentialThreads("I am in SY", 20);
    SequentialThreads thread3 = new SequentialThreads("I am in TY", 30);

    // Starting threads
    thread1.start();
    try {
        thread1.join(); // Ensures thread1 completes before thread2 starts
    } catch (InterruptedException e) {
        e.printStackTrace();
    }

    thread2.start();
    try {
        thread2.join(); // Ensures thread2 completes before thread3 starts
    } catch (InterruptedException e) {
        e.printStackTrace();
    }

    thread3.start();
    try {
        thread3.join(); // Ensures thread3 completes last
    } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

Slip 11

Q2. Write a java program using Inter Thread Communication.

```
class SharedResource {
    private boolean flag = false; // Shared resource (condition flag)

    // Method that will make the current thread wait
    public synchronized void printMessage1() {
        try {
            while (flag == false) {
                wait(); // Wait until notified by another thread
            }
            System.out.println("I am in FY");
            flag = false; // Reset flag for the next thread to proceed
            notify(); // Notify the next thread
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }

    // Method that will make the current thread wait
    public synchronized void printMessage2() {
        try {
            while (flag == true) {
                wait(); // Wait until notified by another thread
            }
            System.out.println("I am in SY");
            flag = true; // Set flag to notify the other thread
            notify(); // Notify the next thread
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}

class Thread1 extends Thread {
    SharedResource resource;

    public Thread1(SharedResource resource) {
        this.resource = resource;
    }

    @Override
    public void run() {
        while (true) {
            resource.printMessage1(); // Call method to print message
        }
    }
}
```

```
class Thread2 extends Thread {
    SharedResource resource;

    public Thread2(SharedResource resource) {
        this.resource = resource;
    }

    @Override
    public void run() {
        while (true) {
            resource.printMessage2(); // Call method to print message
        }
    }
}

public class InterThreadCommunication {
    public static void main(String[] args) {
        SharedResource resource = new SharedResource();

        // Creating and starting threads
        Thread1 thread1 = new Thread1(resource);
        Thread2 thread2 = new Thread2(resource);

        thread1.start();
        thread2.start();
    }
}
```

Slip 12

Q2Write a multithreading program using Runnable interface to blink Text on the frame.

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

class BlinkTextRunnable implements Runnable {
  private JLabel label;

  public BlinkTextRunnable(JLabel label) {
    this.label = label;
  }

  @Override
  public void run() {
    boolean visible = true;
    while (true) {
      try {
        // Toggle text visibility
        label.setVisible(visible);
        visible = !visible; // Flip the visibility
        Thread.sleep(500); // Blink every 500ms
      } catch (InterruptedException e) {
        e.printStackTrace();
      }
    }
  }
}

public class BlinkTextFrame {
  public static void main(String[] args) {
    // Create a JFrame
    JFrame frame = new JFrame("Blinking Text Example");
    frame.setSize(400, 200);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    // Create a JLabel to display text
    JLabel label = new JLabel("Blinking Text", SwingConstants.CENTER);
    label.setFont(new Font("Arial", Font.BOLD, 30));

    // Add label to the frame
    frame.add(label);
    frame.setVisible(true);

    // Create and start the thread to blink the text
    BlinkTextRunnable blinkRunnable = new BlinkTextRunnable(label);
    Thread blinkThread = new Thread(blinkRunnable);
    blinkThread.start();
  }
}
```

Slip13

Q2

Write a program to display information about the ResultSet like number of columns available in the ResultSet and SQL type of the column. Use Person table.

(Use ResultSetMetaData).

```
import java.sql.*;


public class ResultSetMetaDataExample {
    public static void main(String[] args) {
        try {
            // Use the database connection details from your uploaded code
            Class.forName("org.postgresql.Driver");


            // Connection string according to your format
            Connection con = DriverManager.getConnection(
                "jdbc:postgresql1:DYP", "postgres", "");  // Correct format as per your example


            // SQL query to fetch data from the Person table
            String query = "SELECT * FROM Person"; // Adjust to your table name


            // Execute the query
            Statement stmt = con.createStatement();
            ResultSet rs = stmt.executeQuery(query);


            // Get the metadata of the ResultSet
            ResultSetMetaData metaData = rs.getMetaData();
            // Get the number of columns in the ResultSet
            int columnCount = metaData.getColumnCount();
            System.out.println("Number of columns: " + columnCount);

            // Loop through each column and print its details
            for (int i = 1; i <= columnCount; i++) {
                String columnName = metaData.getColumnName(i);
                int columnType = metaData.getColumnType(i);
                String columnTypeName = metaData.getColumnTypeName(i);

                System.out.println("Column " + i + ":");
                System.out.println("Name: " + columnName);
                System.out.println("SQL Type: " + columnTypeName);
                System.out.println("SQL Type Code: " + columnType);
                System.out.println();
            }

            // Close the connection
            con.close();

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Slip 14.

Q2

Q.2) Write a JDBC program to display all the countries located in the West Region. Create a table Country with fields (Name, continent, Capital,Region). Insert values in the table.

```
import java.sql.*;

public class CountryRegionDisplay {
    public static void main(String[] args) {
        try {
            // Step 1: Register JDBC driver
            Class.forName("org.postgresql.Driver");

            // Step 2: Open a connection (use the connection string you provided earlier)
            Connection con = DriverManager.getConnection(
                "jdbc:postgresql1:DYP", "postgres", "");  // Your PostgreSQL connection string

            // Step 3: Create a Statement object to execute SQL
            String query = "SELECT * FROM Country WHERE Region = 'West'";
            Statement stmt = con.createStatement();
            ResultSet rs = stmt.executeQuery(query);

            // Step 4: Process the ResultSet and display the countries in the West region
            System.out.println("Countries in the West region:");
            while (rs.next()) {
                String name = rs.getString("Name");
                String continent = rs.getString("Continent");
                String capital = rs.getString("Capital");
                String region = rs.getString("Region");

                // Display country details
                System.out.println("Name: " + name);
                System.out.println("Continent: " + continent);
                System.out.println("Capital: " + capital);
                System.out.println("Region: " + region);
                System.out.println("--------------");
            }

            // Step 5: Close the connection
            con.close();

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Slip15

2. Q.2) Write a JDBC program to perform search operation on Person table.

1. Search all the person born in the year 1986.

2. Search all the females born between 2000- 2005.

```
import java.sql.*;

public class PersonSearch {
    public static void main(String[] args) {
```

```
    try {
      // Step 1: Register the JDBC driver
      Class.forName("org.postgresql.Driver");

      // Step 2: Open a connection (using your database connection details)
      Connection con = DriverManager.getConnection(
        "jdbc:postgresql1:DYP", "postgres", "");  // Your PostgreSQL connection string

      // Step 3: Search for persons born in the year 1986
      String query1 = "SELECT * FROM Person WHERE EXTRACT(YEAR FROM DateOfBirth) = 1986";
      Statement stmt1 = con.createStatement();
      ResultSet rs1 = stmt1.executeQuery(query1);

      System.out.println("Persons born in the year 1986:");
      while (rs1.next()) {
        String name = rs1.getString("Name");
        String gender = rs1.getString("Gender");
        Date dob = rs1.getDate("DateOfBirth");

        System.out.println("Name: " + name + ", Gender: " + gender + ", Date of Birth: " + dob);
      }
      System.out.println("--------------");

      // Step 4: Search for females born between 2000 and 2005
      String query2 = "SELECT * FROM Person WHERE Gender = 'Female' AND EXTRACT(YEAR FROM
DateOfBirth) BETWEEN 2000 AND 2005";
      Statement stmt2 = con.createStatement();
      ResultSet rs2 = stmt2.executeQuery(query2);
```

```
            System.out.println("Females born between 2000 and 2005:");
            while (rs2.next()) {
                String name = rs2.getString("Name");
                String gender = rs2.getString("Gender");
                Date dob = rs2.getDate("DateOfBirth");

                System.out.println("Name: " + name + ", Gender: " + gender + ", Date of Birth: " + dob);
            }

            // Step 5: Close the connection
            con.close();

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```