Write a Java Program to implement multithreading (three threads using single run method).

```
class MultiThreadDemo implements Runnable {
    private String threadName;

    MultiThreadDemo(String name) {
        threadName = name;
    }

    public void run() {
        for (int i = 0; i < 5; i++) {
            System.out.println(threadName + " - Count: " + i);
            try {
                // Sleep for a while to simulate some work and allow context switching
                Thread.sleep(500);
            } catch (InterruptedException e) {
                System.out.println(threadName + " interrupted.");
            }
        }
        System.out.println(threadName + " exiting.");
    }

    public static void main(String[] args) {
        Thread thread1 = new Thread(new MultiThreadDemo("Thread 1"));
        Thread thread2 = new Thread(new MultiThreadDemo("Thread 2"));
        Thread thread3 = new Thread(new MultiThreadDemo("Thread 3"));

        thread1.start();
        thread2.start();
        thread3.start();
    }
}
```

Write a Java Program to implement the calculator.

```
import java.util.Scanner;

public class Calculator {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter first number:");
        double num1 = scanner.nextDouble();

        System.out.println("Enter second number:");
        double num2 = scanner.nextDouble();

        System.out.println("Enter an operator (+, -, *, /):");
        char operator = scanner.next().charAt(0);

        double result;

        switch (operator) {
            case '+':
                result = num1 + num2;
                System.out.println("The result is: " + result);
                break;
            case '-':
                result = num1 - num2;
                System.out.println("The result is: " + result);
                break;
            case '*':
                result = num1 * num2;
                System.out.println("The result is: " + result);
                break;
```

```
            case '/':
                if (num2 != 0) {
                    result = num1 / num2;
                    System.out.println("The result is: " + result);
                } else {
                    System.out.println("Error! Division by zero.");
                }
                break;
            default:
                System.out.println("Invalid operator.");
                break;
        }

        scanner.close();
    }
}
```

Write a Java Program to implement the URL.

To demonstrate how to use URLs in Java, we'll create a simple program that performs the following actions:

Parse a given URL.

Extract and print various components of the URL such as protocol, host, port, path, query, and reference.

Handle potential exceptions that may arise during URL processing.

Here's a Java program to achieve this:

```
import java.net.MalformedURLException;
import java.net.URL;

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

```
        // Example URL
        URL url = new
URL("https://www.example.com:8080/path/to/resource?query=example#reference");

        // Display the components of the URL
        System.out.println("URL: " + url.toString());
        System.out.println("Protocol: " + url.getProtocol());
        System.out.println("Host: " + url.getHost());
        System.out.println("Port: " + (url.getPort() != -1 ? url.getPort() : "default port"));
        System.out.println("Path: " + url.getPath());
        System.out.println("Query: " + url.getQuery());
        System.out.println("Reference: " + url.getRef());
      } catch (MalformedURLException e) {
        System.out.println("The URL is malformed: " + e.getMessage());
      }
   }
}
```

Write a Java Program to implement the InetAddress.

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

public class InetAddressExample {
   public static void main(String[] args) {
      try {
        // Get the local host address
        InetAddress localHost = InetAddress.getLocalHost();
        System.out.println("Local Host:");
        System.out.println("Hostname: " + localHost.getHostName());
        System.out.println("IP Address: " + localHost.getHostAddress());
        System.out.println();
```

```
            // Get the IP address from a hostname
            String hostname = "www.example.com";
            InetAddress hostAddress = InetAddress.getByName(hostname);
            System.out.println("Host Information for: " + hostname);
            System.out.println("Hostname: " + hostAddress.getHostName());
            System.out.println("IP Address: " + hostAddress.getHostAddress());
            System.out.println();

            // Get the hostname from an IP address
            String ipAddress = "93.184.216.34"; // IP address for example.com
            InetAddress addressByIP = InetAddress.getByName(ipAddress);
            System.out.println("Host Information for IP: " + ipAddress);
            System.out.println("Hostname: " + addressByIP.getHostName());
            System.out.println("IP Address: " + addressByIP.getHostAddress());
        } catch (UnknownHostException e) {
            System.out.println("Unknown host: " + e.getMessage());
        }
    }
}
```

Write a Java Program for Sending E-mail in Java.

To send an email in Java, you can use the JavaMail API, which is a part of the Java EE platform. To use the JavaMail API, you'll need to include the necessary libraries in your project. If you are using Maven, you can add the following dependency to your pom.xml:

```
<dependency>
    <groupId>com.sun.mail</groupId>
    <artifactId>javax.mail</artifactId>
    <version>1.6.2</version>
</dependency>
```

If you are not using Maven, you can download the JavaMail API from the official site and add the javax.mail.jar to your project's classpath.

Here's a simple Java program to send an email using the JavaMail API:

```
import java.util.Properties;
import javax.mail.*;
import javax.mail.internet.*;

public class EmailSender {
    public static void main(String[] args) {
        // Recipient's email ID needs to be mentioned.
        String to = "recipient@example.com";

        // Sender's email ID needs to be mentioned
        String from = "sender@example.com";
        final String username = "your-email@example.com"; // change accordingly
        final String password = "your-email-password"; // change accordingly

        // Assuming you are sending email through relay.jangosmtp.net
        String host = "smtp.example.com"; // for example, smtp.gmail.com for Gmail

        // Get system properties
        Properties properties = new Properties();
        properties.put("mail.smtp.auth", "true");
        properties.put("mail.smtp.starttls.enable", "true");
        properties.put("mail.smtp.host", host);
        properties.put("mail.smtp.port", "587"); // or 465 for SSL

        // Get the Session object.
        Session session = Session.getInstance(properties,
          new javax.mail.Authenticator() {
            protected PasswordAuthentication getPasswordAuthentication() {
                return new PasswordAuthentication(username, password);
```

```
        }
       });

      try {
        // Create a default MimeMessage object.
        Message message = new MimeMessage(session);

        // Set From: header field of the header.
        message.setFrom(new InternetAddress(from));

        // Set To: header field of the header.
        message.setRecipients(Message.RecipientType.TO,
          InternetAddress.parse(to));

        // Set Subject: header field
        message.setSubject("This is the Subject Line!");

        // Now set the actual message
        message.setText("This is the actual message");

        // Send message
        Transport.send(message);

        System.out.println("Sent message successfully....");

      } catch (MessagingException e) {
        throw new RuntimeException(e);
      }
   }
}
```

Write a Java Program to implement Single Client-Server Communication

Server:

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

public class Server {
   public static void main(String[] args) {
      try (ServerSocket serverSocket = new ServerSocket(1234)) {
         System.out.println("Server is listening on port 1234");

         try (Socket socket = serverSocket.accept()) {
            System.out.println("New client connected");

            InputStream input = socket.getInputStream();
            BufferedReader reader = new BufferedReader(new InputStreamReader(input));

            OutputStream output = socket.getOutputStream();
            PrintWriter writer = new PrintWriter(output, true);

            String text;

            do {
               text = reader.readLine();
               System.out.println("Received: " + text);

               writer.println("Server: " + text);

            } while (!text.equals("bye"));

            System.out.println("Client disconnected");
         }
      } catch (IOException ex) {
```

```
            System.out.println("Server exception: " + ex.getMessage());
            ex.printStackTrace();
        }
    }
}
```

Client:

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

public class Client {
    public static void main(String[] args) {
        String hostname = "localhost";
        int port = 1234;

        try (Socket socket = new Socket(hostname, port)) {

            OutputStream output = socket.getOutputStream();
            PrintWriter writer = new PrintWriter(output, true);

            InputStream input = socket.getInputStream();
            BufferedReader reader = new BufferedReader(new InputStreamReader(input));

            BufferedReader consoleReader = new BufferedReader(new InputStreamReader(System.in));
            String text;

            do {
                System.out.print("Enter message: ");
                text = consoleReader.readLine();

                writer.println(text);
```

```
            String response = reader.readLine();
            System.out.println(response);

        } while (!text.equals("bye"));

    } catch (UnknownHostException ex) {
        System.out.println("Server not found: " + ex.getMessage());
    } catch (IOException ex) {
        System.out.println("I/O error: " + ex.getMessage());
    }
  }
}
```

Running the Programs:

Start the Server:

Run the Server program first. The server will start and listen for incoming connections.

Start the Client:

Run the Client program. The client will connect to the server, send messages, and print responses.

Server Output:

```
Server is listening on port 1234
New client connected
Received: Hello Server
Received: How are you?
Received: bye
Client disconnected
```

Client Output:

Enter message: Hello Server

Server: Hello Server

Enter message: How are you?

Server: How are you?

Enter message: bye

Server: bye

Write a Java Program to implement the Login_Id Form using JDBC

To implement a login form using JDBC (Java Database Connectivity), you need to follow these steps:

Create a Database and a Table: For this example, let's use a MySQL database with a table named users.

Connect to the Database using JDBC: Establish a connection to the database.

Create a Login Form: Use Swing for the GUI to create the login form.

Validate User Credentials: Check the entered username and password against the database records.

Prerequisites:

MySQL database setup:

Database name: testdb

Table name: users

Columns: username (VARCHAR), password (VARCHAR)

sql

Copy code

CREATE DATABASE testdb;

USE testdb;

```
CREATE TABLE users (
    username VARCHAR(50) NOT NULL,
    password VARCHAR(50) NOT NULL
);

INSERT INTO users (username, password) VALUES ('testuser', 'testpass');
```

JDBC Driver for MySQL: Ensure you have the MySQL JDBC driver (mysql-connector-java.jar) in your project's classpath.

Java Code:

1. Database Connection Utility

java

Copy code

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DatabaseConnection {
    public static Connection getConnection() {
        Connection connection = null;
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/testdb", "root", "password");
        } catch (ClassNotFoundException | SQLException e) {
            e.printStackTrace();
        }
        return connection;
    }
}
```

2. Login Form with JDBC

java

Copy code

```
import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class LoginForm extends JFrame {
    private JTextField usernameField;
    private JPasswordField passwordField;
    private JButton loginButton;
    private JPanel loginPanel;

    public LoginForm() {
        setTitle("Login Form");
        setContentPane(loginPanel);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(400, 200);
        setLocationRelativeTo(null);

        loginButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String username = usernameField.getText();
                String password = new String(passwordField.getPassword());
                if (authenticate(username, password)) {
                    JOptionPane.showMessageDialog(LoginForm.this, "Login successful!");
                } else {
                    JOptionPane.showMessageDialog(LoginForm.this, "Invalid username or password.",
"Error", JOptionPane.ERROR_MESSAGE);
```

```
            }
        }
    });
}

private boolean authenticate(String username, String password) {
    boolean isValid = false;
    Connection connection = DatabaseConnection.getConnection();
    if (connection != null) {
        String sql = "SELECT * FROM users WHERE username = ? AND password = ?";
        try (PreparedStatement preparedStatement = connection.prepareStatement(sql)) {
            preparedStatement.setString(1, username);
            preparedStatement.setString(2, password);
            try (ResultSet resultSet = preparedStatement.executeQuery()) {
                if (resultSet.next()) {
                    isValid = true;
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            try {
                connection.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
    return isValid;
}
```

```
    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                new LoginForm().setVisible(true);
            }
        });
    }
}
```

Explanation:

DatabaseConnection Class:

Provides a method to establish a connection to the MySQL database.

Replace "root" and "password" with your MySQL username and password.

LoginForm Class:

Creates a simple Swing-based login form with username and password fields.

On clicking the login button, it calls the authenticate method to validate the credentials.

The authenticate method connects to the database, checks the username and password, and returns true if the credentials are valid, otherwise false.

Main Method:

Launches the login form.

Running the Program:

Ensure your MySQL server is running and the testdb database and users table are set up with the appropriate data.

Ensure the MySQL JDBC driver (mysql-connector-java.jar) is in your project's classpath.

Compile and run the LoginForm class. The login form will appear.

Enter the username and password to test the login functionality.