1. Write a JAVA Program to Display Image using JFrame

```
import java.awt.FlowLayout;
import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JLabel;
public class Image extends JFrame {
private ImageIcon image1;
private JLabel label1;
private ImageIcon image2;
private JLabel label2;
Image(){
setLayout(new FlowLayout());
image1 = new ImageIcon(getClass().getResource("ice.png"));
label1 = new JLabel(image1);
add(label1);
image2 = new ImageIcon(getClass().getResource("pust.png"));
label2 = new JLabel(image2);
add(label2);
}
public static void main(String args[]) {
Image gui = new Image();
gui.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
gui.setVisible(true);
gui.pack();
gui.setTitle("Image Program");
}
}
```

2. Write a JAVA Program for generating Restaurant Bill

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

public class BillGeneration extends JFrame implements ActionListener {
    JLabel l;
    JSpinner[] spinners;
    JButton b;
    float[] prices = {500, 60, 10};
    String[] itemNames = {"Pizza", "Burger", "Tea"};

    BillGeneration() {
        l = new JLabel("Food Ordering System");
        l.setBounds(100, 50, 300, 20);
        add(l);

        spinners = new JSpinner[itemNames.length];
        for (int i = 0; i < itemNames.length; i++) {
            JLabel label = new JLabel(itemNames[i] + " @ " + prices[i]);
            label.setBounds(100, 100 + 50 * i, 150, 20);
            add(label);

            spinners[i] = new JSpinner();
            spinners[i].setBounds(250, 100 + 50 * i, 50, 20);
            add(spinners[i]);
        }
        b = new JButton("Order");
        b.setBounds(100, 100 + 50 * itemNames.length, 80, 30);
        b.addActionListener(this);
        add(b);

        setSize(400, 400);
        setLayout(null);
        setVisible(true);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }

    public void actionPerformed(ActionEvent e) {
        float amount = 0;
        String msg = "";
        for (int i = 0; i < spinners.length; i++) {
            int quantity = (int) spinners[i].getValue();
            if (quantity > 0) {
                amount += prices[i] * quantity;
                msg += itemNames[i] + ": " + quantity + "\n";
            }
        }
        msg += "-----------------\n";
        JOptionPane.showMessageDialog(this, msg + "Total: " + amount);
    }

    public static void main(String[] args) {
        new BillGeneration();
    }
}
```

3. Write a JAVA Program to Create a Student form in GUI

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;
public class form implements ActionListener {
private static JLabel success;
private static JFrame frame;
private static JLabel label1,label2,label3;
private static JPanel panel;
private static JButton button;
private static JTextField userText1, userText2, userText3;
public static void main(String[] args) {
frame = new JFrame();
panel = new JPanel();
frame.setSize(400, 300);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.add(panel);
panel.setLayout(null);
//Setting all Three Lebels
label1= new JLabel("Name");
label1.setBounds(10,10,80,25);
panel.add(label1);label2 = new JLabel("Roll");
label2.setBounds(10,60,80,25);
panel.add(label2);
label3 = new JLabel("Department");
label3.setBounds(10,110,80,25);
panel.add(label3);
//Creating all Textfields
userText1 = new JTextField("Enter Your Name");
userText1.setBounds(100,10,200,25);
panel.add(userText1);
JTextField userText2 = new JTextField("Enter Your Name");
userText2.setBounds(100,60,200,25);
panel.add(userText2);
JTextField userText3 = new JTextField("Enter Your Name");
userText3.setBounds(100,110,200,25);
panel.add(userText3);
button = new JButton("Save");
button.setBounds(150, 160, 80, 25);
button.addActionListener(new form());
panel.add(button);
success = new JLabel("");
success.setBounds(130,210,300,25);
panel.add(success);
frame.setVisible(true);
}
@Override
public void actionPerformed(ActionEvent e) {
// TODO Auto-generated method stub
success.setText("Saved Successfully");
}
}
```

4. Write a JAVA Program to develop a simple calculator in GUI

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

public class Calculator extends JFrame implements ActionListener {
    JButton b10, b11, b12, b13, b14, b15, b16, b17, b18, b19; // Added b17, b18, b19 for %, (, )
    JButton b[] = new JButton[10];
    double n1, n2, r;
    JTextField res;
    char op;
    boolean decimalClicked = false; // Track if decimal point button is clicked

    public Calculator() {
        super("Calculator");
        setLayout(new BorderLayout());
        JPanel p = new JPanel();
        p.setLayout(new GridLayout(5, 4)); // Increased grid layout to accommodate the new buttons
        for (int i = 0; i <= 9; i++) {
            b[i] = new JButton(i + "");
            p.add(b[i]);
            b[i].addActionListener(this);
        }
        b10 = new JButton("+");
        p.add(b10);
        b10.addActionListener(this);
        b11 = new JButton("-");
        p.add(b11);
        b11.addActionListener(this);
        b12 = new JButton("*");
        p.add(b12);
        b12.addActionListener(this);
        b13 = new JButton("/");
        p.add(b13);
        b13.addActionListener(this);
        b14 = new JButton("=");
        p.add(b14);
        b14.addActionListener(this);
        b15 = new JButton("C");
        p.add(b15);
        b15.addActionListener(this);
        b16 = new JButton(".");
        p.add(b16);
        b16.addActionListener(this);
        b17 = new JButton("%"); // Button for percentage
        p.add(b17);
        b17.addActionListener(this);
        b18 = new JButton("("); // Button for opening bracket
        p.add(b18);
        b18.addActionListener(this);
        b19 = new JButton(")"); // Button for closing bracket
        p.add(b19);
        b19.addActionListener(this);
        res = new JTextField(20); // Increased number of columns to 20
        add(p, BorderLayout.CENTER);
        add(res, BorderLayout.NORTH);
        setVisible(true);
        setSize(400, 500); // Increased frame height to accommodate the new row of buttons
    }

    public void actionPerformed(ActionEvent ae) {
        JButton pb = (JButton) ae.getSource();
        if (pb == b15) {
            r = n1 = n2 = 0;
            res.setText("");
            decimalClicked = false; // Reset decimalClicked flag
        } else if (pb == b14) {
            n2 = Double.parseDouble(res.getText());
            eval();
            res.setText("" + r);
            decimalClicked = false; // Reset decimalClicked flag
        } else if (pb == b16) { // Process decimal point button
            if (!decimalClicked) {
                res.setText(res.getText() + ".");
                decimalClicked = true;
            }
        } else {
            boolean opf = false;
            if (pb == b10) {
                op = '+';
                opf = true;
            }
            if (pb == b11) {
                op = '-';
                opf = true;
            }
            if (pb == b12) {
                op = '*';
                opf = true;
            }
            if (pb == b13) {
                op = '/';
                opf = true;
            }
            if (!opf) {
                for (int i = 0; i < 10; i++) {
                    if (pb == b[i]) {
                        String t = res.getText();
                        t += i;
                        res.setText(t);
                    }
                }
            } else {
                n1 = Double.parseDouble(res.getText());
                res.setText("");
                decimalClicked = false; // Reset decimalClicked flag
            }
        }
    }

    void eval() {
        switch (op) {
            case '+':
                r = n1 + n2;
                break;
            case '-':
                r = n1 - n2;
                break;
            case '*':
                r = n1 * n2;
                break;
            case '/':
                r = n1 / n2;
                break;
        }
    }

    public static void main(String arg[]) {
        new Calculator();
    }
}
```

5. Write a Java program to illustrate suspend, resume and stop operation of thread.

```
class MyThread implements Runnable {
    private volatile boolean running = true;

    @Override
    public void run() {
        for (int i = 0; running; i++) {
            System.out.println("Thread " + Thread.currentThread().getName() + " - " + i);
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
            }
        }
    }

    public void stopThread() {
        running = false;
    }
}

public class ThreadControl {
    public static void main(String[] args) throws InterruptedException {
        MyThread thread = new MyThread();
        Thread thread1 = new Thread(thread, "Thread-1");
        thread1.start();

        // Suspend the thread after 5 seconds
        Thread.sleep(5000);
        System.out.println("Suspending Thread-1");
        // No need to suspend, we just set the flag to false

        // Resume the thread after 3 seconds
        Thread.sleep(3000);
        System.out.println("Resuming Thread-1");
        // No need to resume, we just start the thread with the flag set to true

        // Stop the thread gracefully
        Thread.sleep(2000);
        thread.stopThread();
        System.out.println("Stopping Thread-1");
    }
}
```

6. Write a Java program to create thread class.

```
class MyThread extends Thread {
    @Override
    public void run() {
        // Code that will run in the new thread
        for (int i = 0; i < 5; i++) {
            System.out.println("Thread: " + Thread.currentThread().getName() + ", Count: " + i);
            try {
                Thread.sleep(1000); // Sleep for 1 second
            } catch (InterruptedException e) {
                System.out.println("Thread interrupted.");
            }
        }
    }
}

public class ThreadCreationExample {
    public static void main(String[] args) {
        MyThread thread1 = new MyThread(); // Create a new thread
        MyThread thread2 = new MyThread(); // Create another new thread

        thread1.start(); // Start the first thread
        thread2.start(); // Start the second thread
    }
}
```

7. Write a Java program to illustrate yield(), stop() and sleep() method using thread.

```
class MyThread extends Thread {
    private volatile boolean running = true;

    public void run() {
        for (int i = 0; running && i < 5; i++) {
            System.out.println("Thread: " + Thread.currentThread().getName() + ", Count: " + i);

            // Demonstrate yield()
            if (i == 2) {
                System.out.println("Thread: " + Thread.currentThread().getName() + " is yielding...");
                Thread.yield();
            }

            // Demonstrate sleep()
            try {
                Thread.sleep(1000); // Sleep for 1 second
            } catch (InterruptedException e) {
                System.out.println("Thread interrupted.");
            }
        }
    }

    public void stopThread() {
        running = false;
    }
}

public class ThreadDemo {
    public static void main(String[] args) {
        MyThread thread1 = new MyThread(); // Create a new thread
        MyThread thread2 = new MyThread(); // Create another new thread

        thread1.start(); // Start the first thread
        thread2.start(); // Start the second thread
        // Let threads run for 3 seconds
        try {
            Thread.sleep(3000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        // Stop the threads gracefully
        thread1.stopThread();
        System.out.println("Stopping thread1...");

        // Let thread2 continue running
        try {
            Thread.sleep(3000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        // Stop the second thread gracefully
        thread2.stopThread();
        System.out.println("Stopping thread2...");
    }
}
```

8. Write a Java program to use priority of thread.

```
class MyThread extends Thread {
    public MyThread(String name) {
        super(name);
    }

    @Override
    public void run() {
        for (int i = 0; i < 5; i++) {
            System.out.println("Thread: " + Thread.currentThread().getName() + ", Count: " + i);
        }
    }
}

public class ThreadPriorityDemo {
    public static void main(String[] args) {
        MyThread thread1 = new MyThread("Thread-1");
        MyThread thread2 = new MyThread("Thread-2");

        // Set priorities
        thread1.setPriority(Thread.MIN_PRIORITY); // Minimum priority
        thread2.setPriority(Thread.MAX_PRIORITY); // Maximum priority

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

9. Write a client and server program in Java to establish a connection between them.

Server side code:

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;

public class ServerModel {

    public static void main(String[] args) {
        try {
            // Print a message indicating that the server is waiting for clients
            System.out.println("Waiting for the clients...");

            // Create a server socket on port 4999
            ServerSocket ss = new ServerSocket(4999);

            // Accept a client connection
            Socket soc = ss.accept();

            // Print a message indicating that a client is connected
            System.out.println("Client Connected...");

            // Read the message sent by the client
            BufferedReader in = new BufferedReader(new InputStreamReader(soc.getInputStream()));
            String str = in.readLine();
            System.out.println("Client Sent: " + str);

            // Send a response to the client
            PrintWriter out = new PrintWriter(soc.getOutputStream(), true);
            out.println("I got your msg");
            out.flush();

            // Close the server socket
            ss.close();

        } catch (IOException e) {
            // Print the stack trace if an IO exception occurs
            e.printStackTrace();
        }
    }
}
```

Client side code:

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;

public class ClientModel {

    public static void main(String[] args) {
        try {
            // Connect to the server running on localhost and port 4999
            Socket soc = new Socket("localhost", 4999);

            // Create a message to send to the server
            String str = "hello guys";

            // Send the message to the server
            PrintWriter output = new PrintWriter(soc.getOutputStream());
            output.println(str);
            output.flush();

            // Read the response from the server
            BufferedReader in = new BufferedReader(new InputStreamReader(soc.getInputStream()));
            String strInput = in.readLine();

            // Display the response from the server
            System.out.println("Server Sent: " + strInput);

            // Close the socket connection
            soc.close();

        } catch (IOException e) {
            // Print the stack trace if an IO exception occurs
            e.printStackTrace();
        }
    }
}
```