```java
import java.awt.*;
import java.awt.event.*;
import java.time.LocalDateTime;
import java.util.*;
import java.io.*;
import java.net.*;

public class WetterClientLsg extends Frame {

    // feste Regionnamen
    static private String[] region = { "NordWest", "Nord", "NordOst", "West", "Mitte", "Ost", "SuedWest",
"Sued",
            "SuedOst" };

    // TODO Anlegen einer HashMap fuer die Zuordnung der Region-Namen
    // zu den Label-Objekten, die die jeweiligen Werte anzeigen sollen
    private HashMap<String, Label> labelMap = new HashMap<String, Label>();

    // Statuszeile oben
    private Label status;

    private Socket toServer = null;
    private BufferedReader in = null;
    private PrintWriter out = null;
    private PrintWriter logger = null;

    // TODO
    // Hier innere UpdateThread-Klasse einfuegen
    private WetterUpdate updateReceiver = null;

    class WetterUpdate extends Thread {
        @Override
        public void run() {
            while (!isInterrupted()) {
                try {
                    String updateZeile;
                    // Empfang UPDATE
                    if ((updateZeile = readAndLog()) != null) {
                        if (!updateZeile.equals("<END_UPDATE>")) {
                            String[] tokens = updateZeile.split(":");
                            Label label = labelMap.get(tokens[0]);
                            label.setText(tokens[1] + "°");
                        }
                    }
                } catch (IOException e) {
                    // e.printStackTrace();
                }
            }
            System.out.println("WetterUpdate-Thread ends");
        }
    }

    // Komstruktor Aufbau der GUI
    public WetterClientLsg(String host, int portnummer) {
        super("WetterClient");
        // Statuszeile oben
        status = new Label("Status");
        add(status, BorderLayout.NORTH);

        // Panel mit 9 Labels erzeugen, um die Temperaturen der einzelnen
        // Regionen darzustellen
        Panel wetterkarte = new Panel(new GridLayout(3, 3));
        for (String r : region) {
            Label label = new Label("0,0");
            wetterkarte.add(label);

            // TODO Zuordnung von Label und Region in HashMap speichern
            labelMap.put(r, label);
        }
        add(wetterkarte);

        // Bedienbuttons unten anlegen
```

```
        Panel action = new Panel();
        Button btnInit = new Button("Initialisieren");
        Button btnUpdStart = new Button("Update starten");
        Button btnUpdEnd = new Button("Update beenden");
        action.add(btnInit);
        action.add(btnUpdStart);
        action.add(btnUpdEnd);
        add(action, BorderLayout.SOUTH);
        // Buttons mit Verarbeitungsmethoden verbinden
        btnInit.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                initialize();
            }
        });
        btnUpdStart.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                updateStart();
            }
        });
        btnUpdEnd.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                updateEnd();
            }

        });

        // Bearbeitung Fenster-Close-Button
        addWindowListener(new WindowAdapter() {
            @Override
            public void windowClosing(WindowEvent e) {
                exit();
                dispose();
            }

        });
        // Verbindungsaufbau versuchen
        if (connect(host, portnummer)) {
            status.setText("Verbindung erfolgreich! Weiter mit 'Initialisieren'...");
        } else {
            status.setText("Verbindung fehlgeschlagen");
        }
        // Fenster anzeigen
        pack();
        setVisible(true);
    }

    public boolean connect(String host, int portnummer) {
        // TODO
        // - Verbindungsaufbau zum Server
        // - Initialisierung aller Reader/Writer-Objekte
        // - Rueckgabe true bei erfolgreicher Verbindung
        try {
            toServer = new Socket(host, portnummer);
            in = new BufferedReader(new InputStreamReader(toServer.getInputStream()));
            out = new PrintWriter(toServer.getOutputStream(), true);
            logger = new PrintWriter(new FileWriter("WetterClient.log"), true);
            return true;
        } catch (IOException e) { // 1
            System.out.println(e.getMessage());
        }
        return false;
    }

    public String readAndLog() throws IOException {
        String line = in.readLine();
        LocalDateTime ldt = LocalDateTime.now();
        logger.println(ldt.toString() + ":" + line); // 2
        return line;
    }
```

```
    public void initialize() {
        // TODO
        // - evtl. Update-Modus beenden
        // - Senden <INIT> an Server
        // - enpfang der Init-Werte mit receiveInitValues
        updateEnd();
        status.setText("Neu initialisieren");
        out.println("<INIT>");
        receiveInitValues();
        status.setText("Normal Modus");
    }

    public void receiveInitValues() {
        // TODO
        // - Empfang aller INIT-Nachrichten vom Server
        //   bis "<END_INIT>" gesendet wird.
        // - Zuordnung und Anzeige der empfangenen Werte
        //   in den Region-Labels
        //   mit 'setText(String text)' kann bei Labels ein neuer Text gesetzt werden
        String tickerZeile;
        try {
            while ((tickerZeile = readAndLog()) != null) {
                System.out.println(tickerZeile);
                if (!tickerZeile.equals("<END_INIT>")) {
                    String[] tokens = tickerZeile.split(":");
                    labelMap.get(tokens[0]).setText(tokens[1] + "°");

                } else {
                    break;
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public void updateStart() {
        // TODO
        // Einschalten Update-Modus, sofern nicht schon Update-Modus aktiv
        // Es muss ein eigener Thread zum Empfang
        // von Update-Zeilen erzeugt und gestartet werden.
        // Damit der WetterServer mit dem Versand von Aktualisierungen beginnt,
        // muss das Kommando "<UPDATE>" als eine Textzeile an den Server
        // gesendet werden.
        if (updateReceiver == null) {
            status.setText("Update Modus");
            updateReceiver = new WetterUpdate();
            updateReceiver.start();
            out.println("<UPDATE>");
        }
    }

    public void updateEnd() {
        // TODO
        // Ausschalten Update-Modus (einschalten Normalmodus), wenn Update-Modus
        // aktiv:
        // Senden Kommando "<END UPDATE>" an Server.
        // Der bestehende Update-Thread muss benachrichtigt werden, so dass er
        // sich sauber selbst beendet.
        if (updateReceiver != null) {
            status.setText("Normal Modus");
            out.println("<END_UPDATE>");
            updateReceiver.interrupt();
            System.out.println("End_Update");
            updateReceiver = null;
        }
    }



    public void exit() {
```

```
        // Abmelden vom Server, ggf. Update-Modus vorher beenden
        // lokale Aufraeumarbeiten
        updateEnd();
        try {
            if (toServer != null) {
                out.println("<EXIT>");

                out.close();
                in.close();
                toServer.close();
                logger.close();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }

    }

    public static void main(String[] args) {

        // TODO zum Server passenden Hostname, Portnummer eintragen
        String hostname = "localhost";
        int portnummer = 7077;

        if (args.length ==  2) {
            hostname = args[0];
            portnummer = Integer.parseInt(args[1]);
        }
        new WetterClientLsg(hostname, portnummer);

    }

}
```