RELATÓRIO TÉCNICO

Wesley de Oliveira Mendes - 828.507

Tarefa 03 - Servidor UDP - Data e Hora

Prof. Rodrigo de Oliveira Plotze

Ribeirão Preoto - SP
2021

```java
package udp;

import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;

/**
 *
 * @author Wesley
 */
public class ServerDateTime extends Thread {

    // Parametros de comunicacao
    private final String URL = "127.0.0.1";
    private final int PORT_RECEIVE = 12345;
    private final int PORT_SEND = 54321;

    enum DateTimeFormat { DATE, TIME, DATETIME, DEFAULT }

    public ServerDateTime(){
        System.out.println("------------------------------");
        System.out.println(" Server running on port " + PORT_RECEIVE);
        System.out.println("------------------------------");
    }

    public int datetimeParser(String message) {
        switch (message) {
            case "date" -> {
                return 0;
            }
            case "time" -> {
                return 1;
            }
            case "realtime" -> {
                return 2;
            }
        }
        return 3;
    }

    public void clock() {
        Thread clock = new Thread() {
            public void run() {
                try {
                    for (;;) {
                        DateTimeFormatter dtf = DateTimeFormatter.ofPattern("dd/MM/yyyy HH:mm:ss");
                        LocalDateTime CurrDatetime = LocalDateTime.now();
                        dtf.format(CurrDatetime);
                        sleep(1000);
                    }
                } catch (InterruptedException e) {
                }
            }
        };

        clock.start();
    }

    public String datetimeFormat(String dtFormat) {
        DateTimeFormatter dtf = DateTimeFormatter.ofPattern(dtFormat);
        LocalDateTime CurrDatetime = LocalDateTime.now();
        return dtf.format(CurrDatetime);
    }

    public String getResponse(String message) {
        int datetimeType = datetimeParser(message);

        DateTimeFormat whichView = DateTimeFormat.values()[datetimeType];
        switch (whichView) {
            case DATE -> {
                return datetimeFormat("dd/MM/yyyy");
            }
            case TIME -> {
                return datetimeFormat("HH:mm:ss");
            }
            case DATETIME -> {
                return datetimeFormat("dd/MM/yyyy HH:mm:ss");
            }
            case DEFAULT -> {
                return "Ops, opção inválida!";
            }
        }

        return "Ops, opção inválida!";
    }
```

```java

    public void response(String message){
        String messageRes = getResponse(message);

        try {
            // Converter a mensagem em bytes
            byte[] buffer = messageRes.getBytes();

            // Criar o pacote que será transmitido
            DatagramPacket packet = new DatagramPacket(
                    buffer,                     // Conteudo
                    buffer.length,              // Tamanho
                    InetAddress.getByName(URL), // Endereco
                    PORT_SEND                   // Porta
            );

            // Enviar o pacote
            new DatagramSocket().send(packet);
        } catch (IOException e) {
            System.err.println("ERRO: " + e.getMessage());
        }
    }

    @Override
    public void run() {
        try {
            // Criar um socket do tipo UDP
            DatagramSocket srv = new DatagramSocket(PORT_RECEIVE);

            while(true) {
                // Definir o buffer para recebimento da mensagem
                byte[] buffer = new byte[256];

                // Definir o pacote que será recebido
                DatagramPacket packet = new DatagramPacket(
                    buffer, buffer.length
                );

                // Receber o pacote
                srv.receive(packet);

                // Transformar o pacote no tipo de dados esperado
                String messageReq = new String(packet.getData()).trim();

                if (messageReq.length() > 0) {
                    // Resquets log
                    System.out.println(
                        "From: " + packet.getAddress().getHostAddress() +
                        ":" + packet.getPort() + "\n" +
                        "Message: " + messageReq + "\n"
                    );

                    // Response
                    response(messageReq);
                }
            }
        } catch (IOException e) {
            System.err.println("ERRO: " + e.getMessage());
        }
    }

    public static void main(String[] args) {
        new ServerDateTime().start();
    }
}
```

```java
package udp;

import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import javax.swing.JTextField;

/**
 *
 * @author Wesley
 */
public class Client extends Thread {

    // Parametros de comunicacao
    private final String URL = "127.0.0.1";
    private final int PORT_RECEIVE = 54321;
    private final int PORT_SEND = 12345;

    private JTextField txt;

    public Client(JTextField txt) {
        this.txt = txt;

        System.out.println("------------------------------");
        System.out.println(" Client running on port " + PORT_RECEIVE);
        System.out.println("------------------------------");
    }

    public void send(String message){
        try {
            // Converter a mensagem em bytes
            byte[] buffer = message.getBytes();

            // Criar o pacote que sera transmitido
            DatagramPacket packet = new DatagramPacket(
                    buffer,                     // Conteudo
                    buffer.length,              // Tamanho
                    InetAddress.getByName(URL), // Endereco
                    PORT_SEND                   // Porta
            );

            // Enviar o pacote
            new DatagramSocket().send(packet);
        } catch (IOException e) {
            System.err.println("ERRO: " + e.getMessage());
        }
    }

    @Override
    public void run() {
        try {
            // Criar um socket do tipo UDP
            DatagramSocket srv = new DatagramSocket(PORT_RECEIVE);

            while(true) {
                // Definir o buffer para recebimento da mensagem
                byte[] buffer = new byte[256];

                // Definir o pacote que sera recebido
                DatagramPacket packet = new DatagramPacket(
                    buffer, buffer.length
                );

                // Receber o pacote
                srv.receive(packet);

                // Transformar o pacote no tipo de dados esperado
                String message = new String(packet.getData()).trim();

                // Exibir a mensagem
                System.out.println(
                    "Server: " + packet.getAddress().getHostAddress() +
                    ":" + packet.getPort() + "\n" +
                    "Message: " + message + "\n"
                );

                // Exibir a mensagem no Text Field
                txt.setText(message);
            }
        } catch (IOException e) {
            System.err.println("ERRO: " + e.getMessage());
        }
    }
}
```

```java
package udp;

import static java.lang.Thread.sleep;

/**
 *
 * @author Wesley
 */
public class ClientForm extends javax.swing.JFrame {

    private Client cli;
    private String choice = "-1";
    private boolean stop;

    public ClientForm() {
        initComponents();

        this.setTitle("Cliente UDP");
        this.setResizable(false);

        cli = new Client(txtDateTime);
        cli.start();
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents
    private void initComponents() {

        txtMessage = new javax.swing.JTextField();
        lblMessage = new javax.swing.JLabel();
        btnSend = new javax.swing.JButton();
        btnDate = new javax.swing.JButton();
        btnTime = new javax.swing.JButton();
        btnDateTime = new javax.swing.JButton();
        btnClean = new javax.swing.JButton();
        btnStopRealTime = new javax.swing.JButton();
        txtDateTime = new javax.swing.JTextField();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        lblMessage.setFont(new java.awt.Font("Tahoma", 1, 18)); // NOI18N
        lblMessage.setText("O QUE DESEJA SABER?");

        btnSend.setFont(new java.awt.Font("Tahoma", 1, 18)); // NOI18N
        btnSend.setText("ENVIAR");
        btnSend.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnSendActionPerformed(evt);
            }
        });

        btnDate.setFont(new java.awt.Font("Tahoma", 1, 12)); // NOI18N
        btnDate.setText("DATA");
        btnDate.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnDateActionPerformed(evt);
            }
        });

        btnTime.setFont(new java.awt.Font("Tahoma", 1, 12)); // NOI18N
        btnTime.setText("HORA");
        btnTime.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnTimeActionPerformed(evt);
            }
        });

        btnDateTime.setFont(new java.awt.Font("Tahoma", 1, 12)); // NOI18N
        btnDateTime.setText("DATA/HORA TEMPO REAL");
        btnDateTime.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnDateTimeActionPerformed(evt);
            }
        });
```

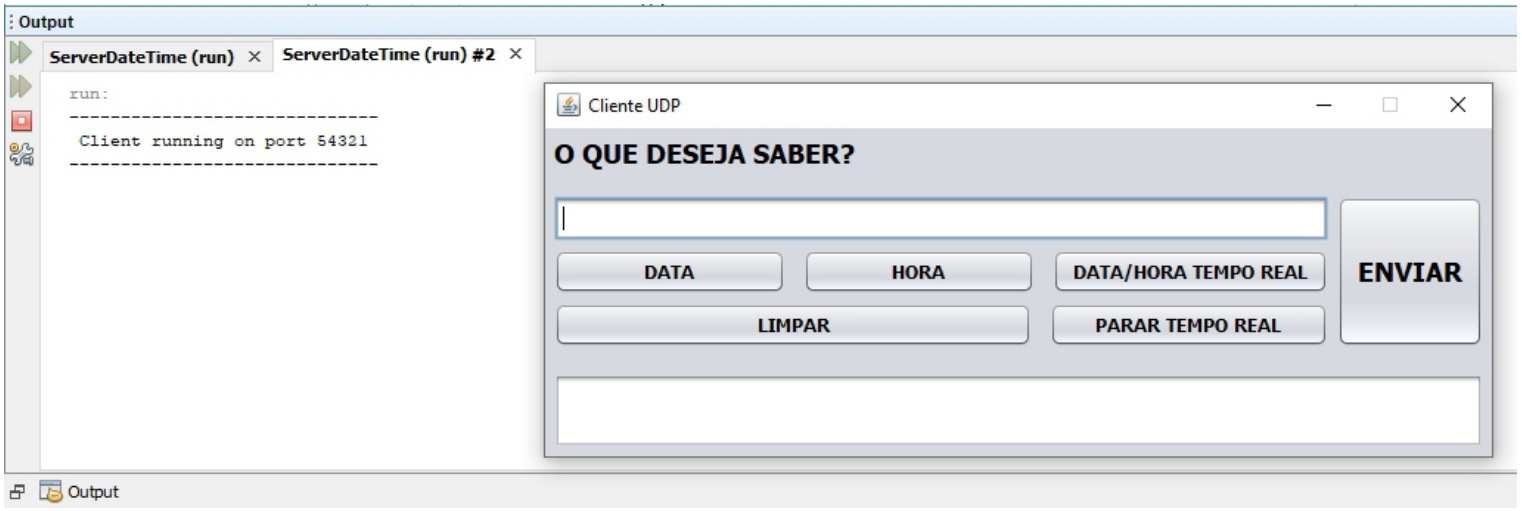
```
        btnClean.setFont(new java.awt.Font("Tahoma", 1, 12)); // NOI18N
        btnClean.setText("LIMPAR");
        btnClean.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnCleanActionPerformed(evt);
            }
        });

        btnStopRealTime.setFont(new java.awt.Font("Tahoma", 1, 12)); // NOI18N
        btnStopRealTime.setText("PARAR TEMPO REAL");
        btnStopRealTime.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnStopRealTimeActionPerformed(evt);
            }
        });

        txtDateTime.setFont(new java.awt.Font("Tahoma", 1, 24)); // NOI18N
        txtDateTime.setHorizontalAlignment(javax.swing.JTextField.CENTER);

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(lblMessage)
                            .addGroup(layout.createSequentialGroup()
                                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)
                                    .addGroup(javax.swing.GroupLayout.Alignment.LEADING, layout.createSequentialGroup()
                                        .addComponent(btnClean, javax.swing.GroupLayout.PREFERRED_SIZE, 324, javax.swing.GroupLayout.PREFERRED_SIZE)
                                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                                        .addComponent(btnStopRealTime, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                                    .addGroup(javax.swing.GroupLayout.Alignment.LEADING, layout.createSequentialGroup()
                                        .addComponent(btnDate, javax.swing.GroupLayout.PREFERRED_SIZE, 157, javax.swing.GroupLayout.PREFERRED_SIZE)
                                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                                        .addComponent(btnTime, javax.swing.GroupLayout.PREFERRED_SIZE, 157, javax.swing.GroupLayout.PREFERRED_SIZE)
                                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                                        .addComponent(btnDateTime))
                                    .addComponent(txtMessage, javax.swing.GroupLayout.Alignment.LEADING))
                                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                .addComponent(btnSend)))
                        .addGap(0, 0, Short.MAX_VALUE))
                    .addComponent(txtDateTime))
                .addContainerGap())
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(lblMessage)
                .addGap(18, 18, 18)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(txtMessage, javax.swing.GroupLayout.PREFERRED_SIZE, 30, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE, false)
                            .addComponent(btnDate, javax.swing.GroupLayout.DEFAULT_SIZE, 30, Short.MAX_VALUE)
                            .addComponent(btnDateTime, javax.swing.GroupLayout.DEFAULT_SIZE, 30, Short.MAX_VALUE)
                            .addComponent(btnTime, javax.swing.GroupLayout.DEFAULT_SIZE, 30, Short.MAX_VALUE))
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                            .addComponent(btnClean, javax.swing.GroupLayout.PREFERRED_SIZE, 30, javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(btnStopRealTime, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))
                    .addComponent(btnSend, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                .addGap(18, 18, 18)
                .addComponent(txtDateTime, javax.swing.GroupLayout.PREFERRED_SIZE, 50, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );

        pack();
    }// </editor-fold>//GEN-END:initComponents
```

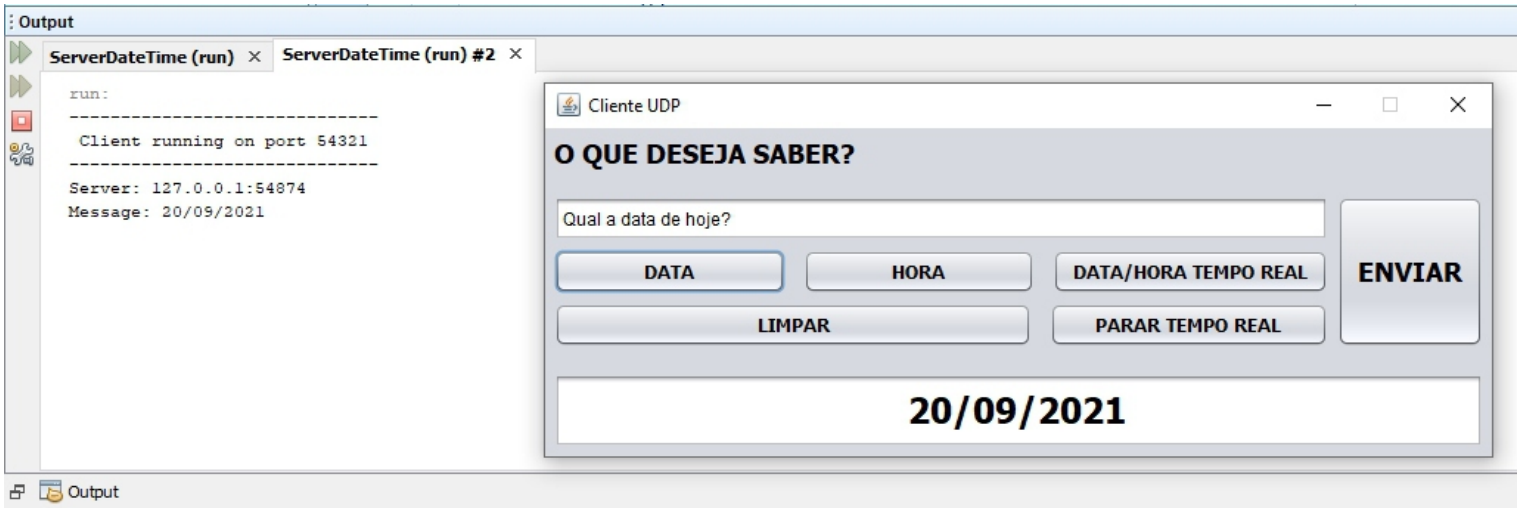
```java
    private void btnSendActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnSendActionPerformed
        String message = txtMessage.getText();

        switch (message) {
            case "Qual a data de hoje?" -> {
                choice = "date";
                cli.send(choice);
            }
            case "Qual a hora?" -> {
                choice = "time";
                cli.send(choice);
            }
            case "Data e hora em tempo real" -> {
                Thread clock = new Thread() {
                    public void run() {
                        try {
                            for (;;) {
                                cli.send("realtime");
                                if (stop) break;
                                sleep(1000);
                            }
                        } catch (InterruptedException e) {
                            System.out.println("Request interrupted");
                        }
                    }
                };

                clock.start();
            }
            default -> {
                cli.send(choice);
            }
        }

        txtMessage.setText("");
        txtMessage.requestFocus();
        choice = "-1";
    }//GEN-LAST:event_btnSendActionPerformed

    private void btnDateActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnDateActionPerformed
        txtMessage.setText("Qual a data de hoje?");
    }//GEN-LAST:event_btnDateActionPerformed

    private void btnTimeActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnTimeActionPerformed
        txtMessage.setText("Qual a hora?");
    }//GEN-LAST:event_btnTimeActionPerformed

    private void btnDateTimeActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnDateTimeActionPerformed
        txtMessage.setText("Data e hora em tempo real");
        stop = false;
    }//GEN-LAST:event_btnDateTimeActionPerformed

    private void btnCleanActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnCleanActionPerformed
        txtDateTime.setText("");
    }//GEN-LAST:event_btnCleanActionPerformed

    private void btnStopRealTimeActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnStopRealTimeActionPerformed
        stop = true;
        txtMessage.setText("");
        txtMessage.requestFocus();
        choice = "-1";
    }//GEN-LAST:event_btnStopRealTimeActionPerformed

    /**
     * @param args the command line arguments
     */
```

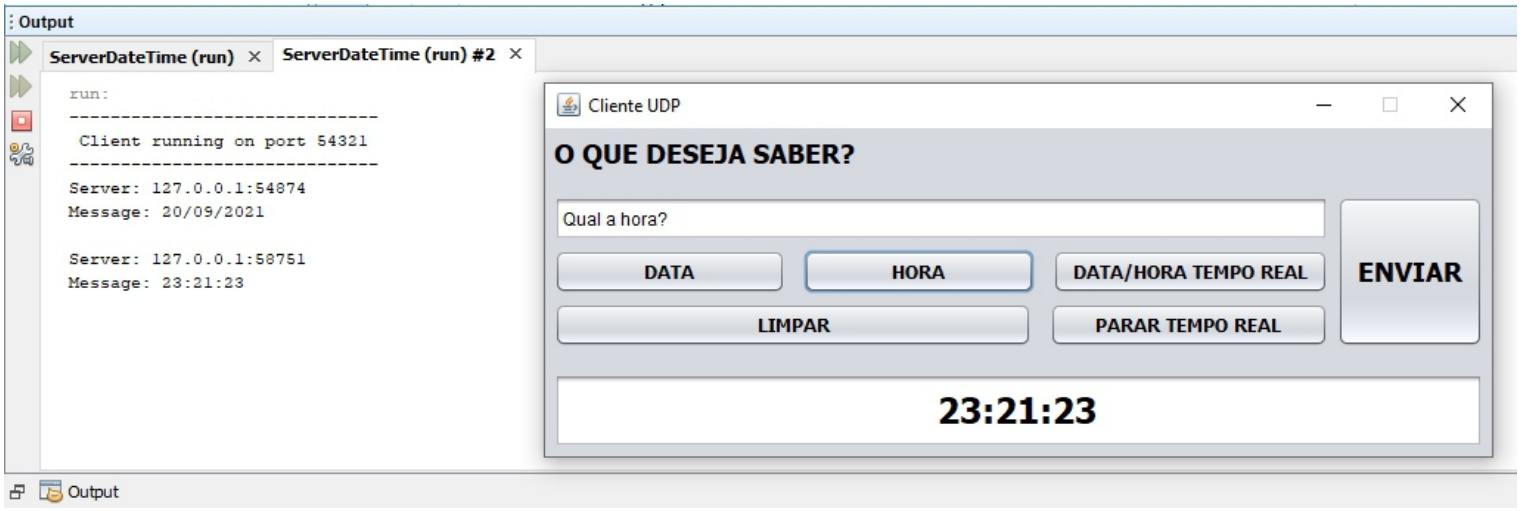
```
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(ClientForm.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(ClientForm.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(ClientForm.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(ClientForm.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new ClientForm().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton btnClean;
    private javax.swing.JButton btnDate;
    private javax.swing.JButton btnDateTime;
    private javax.swing.JButton btnSend;
    private javax.swing.JButton btnStopRealTime;
    private javax.swing.JButton btnTime;
    private javax.swing.JLabel lblMessage;
    private javax.swing.JTextField txtDateTime;
    private javax.swing.JTextField txtMessage;
    // End of variables declaration//GEN-END:variables
}
```

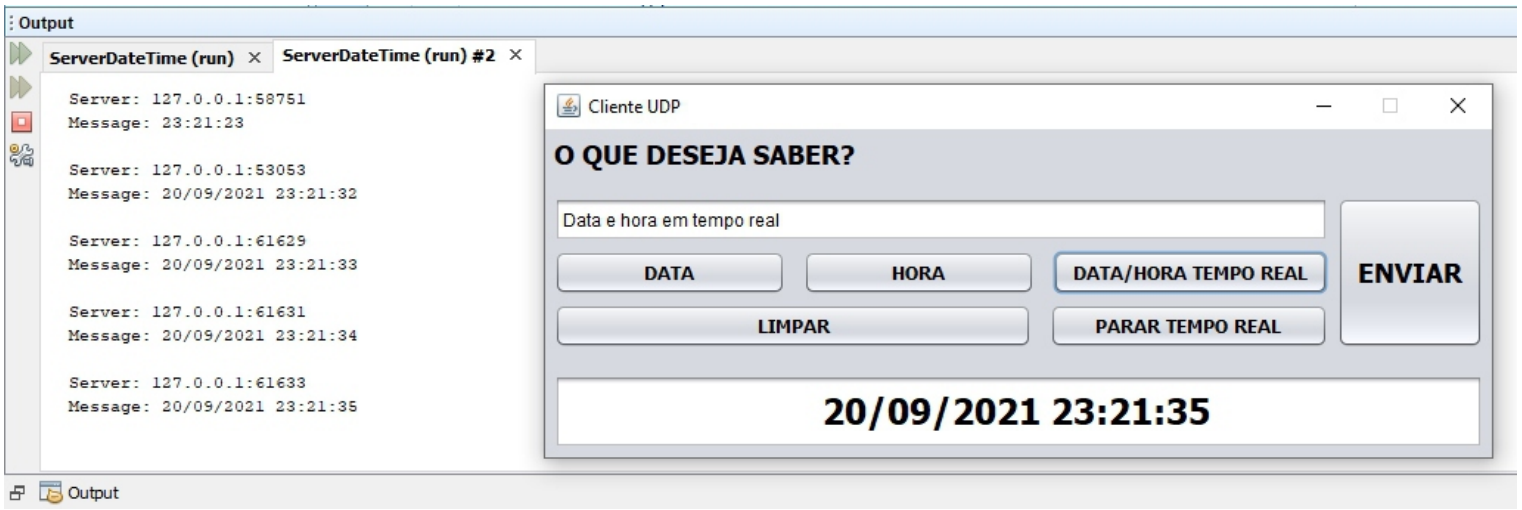
PROJECT:

https://github.com/WesGtoX/Distributed-and-Concurrent-Systems/tree/main/ServerDateTime


Output
ServerDateTime (run)
ServerDateTime (run) #2
run:
Client running on port 54321
Cliente UDP
O QUE DESEJA SABER?
DATA
HORA
DATA/HORA TEMPO REAL
ENVIAR
LIMPAR
PARAR TEMPO REAL
Output


Output
ServerDateTime (run)
ServerDateTime (run) #2
run:
Client running on port 54321
Server: 127.0.0.1:54874
Message: 20/09/2021
Cliente UDP
O QUE DESEJA SABER?
Qual a data de hoje?
DATA
HORA
DATA/HORA TEMPO REAL
ENVIAR
LIMPAR
PARAR TEMPO REAL
20/09/2021
Output


Output
ServerDateTime (run)
ServerDateTime (run) #2
run:
Client running on port 54321
Server: 127.0.0.1:54874
Message: 20/09/2021
Server: 127.0.0.1:58751
Message: 23:21:23
Cliente UDP
O QUE DESEJA SABER?
Qual a hora?
DATA
HORA
DATA/HORA TEMPO REAL
ENVIAR
LIMPAR
PARAR TEMPO REAL
23:21:23
Output


Output
ServerDateTime (run)
ServerDateTime (run) #2
Server: 127.0.0.1:58751
Message: 23:21:23
Server: 127.0.0.1:53053
Message: 20/09/2021 23:21:32
Server: 127.0.0.1:61629
Message: 20/09/2021 23:21:33
Server: 127.0.0.1:61631
Message: 20/09/2021 23:21:34
Server: 127.0.0.1:61633
Message: 20/09/2021 23:21:35
Cliente UDP
O QUE DESEJA SABER?
Data e hora em tempo real
DATA
HORA
DATA/HORA TEMPO REAL
ENVIAR
LIMPAR
PARAR TEMPO REAL
20/09/2021 23:21:35
Output