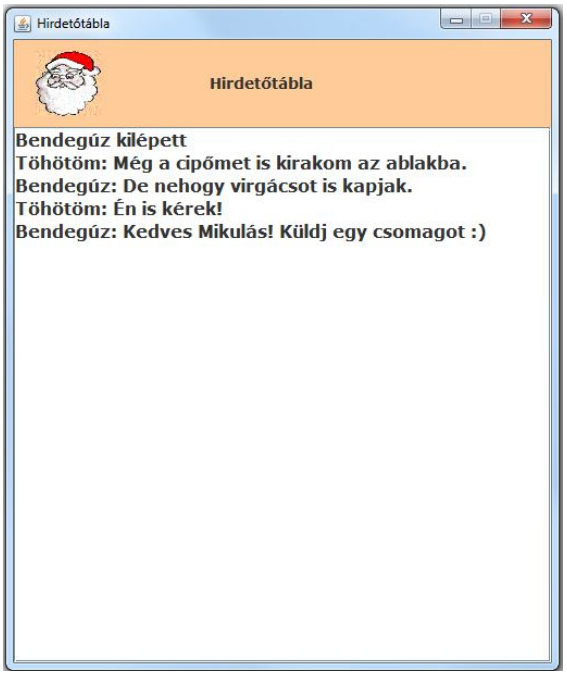
Üzenetküldő kliens - megoldásrészletek

Feladat

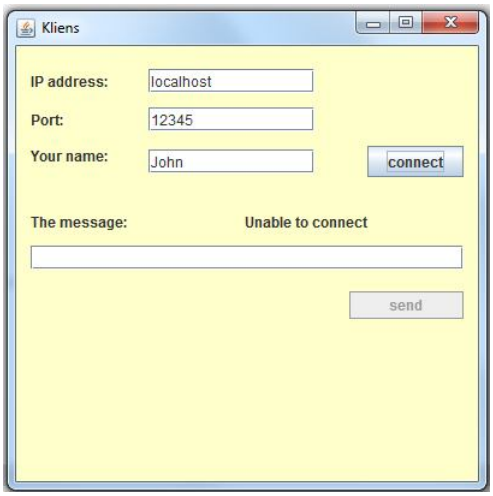


A Mikulás a szerver oldalon várja a gyerekek jelentkezését. Írja meg azt a klienst, amelyről be tudunk jelentkezni hozzá, és üzenetet is tudunk hagyni az üzenőfalán.

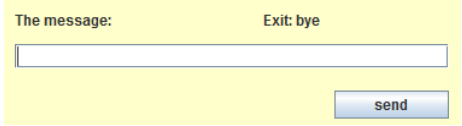
A gombok értelemszerűen legyenek aktívak/inaktívak, az üzenetet pedig az enter lenyomásával is el lehessen küldeni.
Tudnivalók: Előbb meg kell adni a szerver IP-címét, a port most fixen 12345.
Ugyancsak adja meg a nevét.
Ezek után a kapcsolódás gomb segítségével kapcsolódjon a szerverhez.

A kapcsolódás gomb hatására jöjjön létre a szerverrel való kapcsolat, ezután a gomb legyen inaktív. A kapcsolat létrejöttekor a szerver fogadja a megadott nevet, majd visszaküldi azt a szót, amelyet megüzenve a szervernek, lezárul a kliens és a szerver kapcsolata. Ez a kapcsolat akkor is záródjon le, ha az egész alkalmazást zárjuk be. A szervertől kapott szó jelenjen is meg a kliens felületén.

Az üzen gomb hatására, illetve az Enter begépelésekor lehet üzenetet küldeni.
A kliensnek az a dolga, hogy egyetlen String-et átküldjön, a szerver majd tudja kezelni azt.



b) Oldja meg a lokalizációt. (Természetesen a többi üzenetet is.)



Megoldásrészletek:

A megoldásban szét kell bontani a kapcsolódást az üzenetküldéstől, illetve a leállítástól.

Néhány részlet:

A kapcsolódás gomb hatása:

```
private void btnKapcsolodasActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        String ipCim = txtIp.getText();
        int port = Integer.valueOf(txtPort.getText());

        // A socket, out globális, mert másik metódus is használja.
        socket = new Socket(ipCim, port);
        out = new PrintWriter(socket.getOutputStream(), true);

        // Az in csak itt használatos, lehet lokális.
        try (BufferedReader in =
                new BufferedReader(new InputStreamReader(socket.getInputStream()))) {

            String nev = txtNev.getText();
            if (nev.isEmpty()) {
                JOptionPane.showMessageDialog(this, bundle.getString("nevhiany"));
            } else {
                out.println(nev);

                kilepes = in.readLine();
                lblKilepes.setText(bundle.getString("felirat") + " " + kilepes);
                btnUzen.setEnabled(true);
                btnKapcsolodas.setEnabled(false);
            }
        }
    } catch (NumberFormatException | IOException | HeadlessException ex) {
        // lblKilepes.setText("Nem sikerült a kapcsolódás");
        lblKilepes.setText(bundle.getString("hiba"));
    }
}
```

Az üzen gomb hatása:

```
private void btnUzenActionPerformed(java.awt.event.ActionEvent evt) {
    uzen();
}

private void uzen() {
    out.println(txtUzenet.getText());
    if(txtUzenet.getText().equals(kilepes)){
        btnKapcsolodas.setEnabled(true);
        btnUzen.setEnabled(false);
    }
    txtUzenet.setText("");
}
```

Célszerű megoldani azt is, hogy az enter megnyomásának hatására is elmenjen az üzenet:

```
private void txtUzenetKeyPressed(java.awt.event.KeyEvent evt) {
    if(evt.getKeyCode() == KeyEvent.VK_ENTER){
        uzen();
    }
}
```

Végül azt is meg kell oldani, hogy a szerver akkor is tudomást szerezzen a kilépésről, ha a kliens felületéről az ablaklezáró gomb megnyomásával akarunk távozni.
Ezért saját `windowClosing()` metódust kell írnunk. Ezt a frame-n tehetjük meg:

```
public static void main(String args[]) {
    java.awt.EventQueue.invokeLater(new Runnable() {

        public void run() {
            new KliensFrame().start();
        }
    });
}


private void start() {
    setVisible(true);
    this.addWindowListener(new WindowAdapter() {

        @Override
        public void windowClosing(WindowEvent e) {
            kliensPanel1.lezar();
        }
    });
}
```

A panel lezáró metódusa:

```
void lezar() {
    if(socket != null){
        try {
            // értesíti a szervert a kilépésről
            out.println(kilepes);
            out.close();
            socket.close();
            System.out.println("lezártam");
        } catch (IOException ex) {
            Logger.getLogger(KliensPanel.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}
```

Néhány gondolat a szerver oldal megvalósításához:

1. A szervert külön szálként kell megírni, mert különben nem tud működni a grafikus interfész, hiszen a szerver minden kapacitást lefoglalna.

Ennek `run()` metódusában kell megoldani azt, hogy a szerver várja a kliensek jelentkezését, és ha jelentkezik egy, akkor annak kiszolgálására létrehozzon egy `SzerverSzal` (vagy lehet, hogy jobb elnevezés az, hogy `KliensSzal`, csak a kiadott vázlatban a `SzerverSzal` elnevezés szerepel) típusú példányt – erről kicsit később.

Ebben a `Szerver` osztályban kellene megírni a szerverindítást és a szerver lezárását. A szerver indítására vonatkozó metódusban egy adott porton létrehozza a `ServerSocket` példányt és elindítja ezt a szálat. A lezáró metódusban pedig lezárja a még aktív szerverszálakat, majd magát a szerver-socketet is.

Mindkét metódust a frame-ből hívhatjuk, az indítást annak `main()` metódusából, a lezáráshoz pedig hasonlóan felül kell definiálni a `windowClosing()` metódust, mint a kliens esetén.

2. A `SzerverSzal` osztály felel majd egy-egy klienssel való kommunikációért. Paraméterként át kell adni neki a kapcsolódáskor létrejött socket-et, illetve az üzenőfelületet biztosító panel-példányt. (Esetleg a panel helyett lehet vezérlő is, aki kellően bátor ahhoz, hogy ebben az új környezetben is próbálkozzon az MVC szemlélettel.)

Ennek `run()` metódusa valósítja meg a protokollt, azaz fogadja a klienstől jövő üzeneteket a feladatban leírt módon. (Először a nevet olvassa be, visszaküldi a leállító szót, utána pedig ciklusban fogadja a küldött üzeneteket mindaddig, amíg a leállító szót nem küldi a kliens.)

3. Végül kell az a közös felület, amelyen üzengetnek egymásnak. Ezt egy panel felületére rakott szövegmezővel lehet megoldani. Az egyes szerverszálak az üzenet megkapásakor ennek a panelnek a kiíró metódusát hívják meg, és megkérik, hogy írja ki az adott üzenetet.

4. És természetesen kell a frame. ☺