```java
package pl.nalazek.komunikator.logika;

import java.awt.FileDialog;
import java.io.IOException;
import java.lang.reflect.InvocationTargetException;
import java.net.InetSocketAddress;
import java.util.Calendar;
import java.util.Iterator;
import java.util.LinkedHashSet;
import javax.swing.SwingUtilities;
import pl.nalazek.komunikator.Program;
import pl.nalazek.komunikator.gui.Gui;
import pl.nalazek.komunikator.logika.logowanie.Logowanie;

/**
 * @author Daniel Nalazek
 * Copyright (C) 2014 Daniel Nalazek
 */

/** Klasa odpowiadająca za sterownie w programie */
public class Sterowanie implements ISterowanie {

    private Gui gui;
    private Sterowanie sterowanie;
    private Model model;
    private Logowanie logowanie;
    private boolean online = false;
    private LinkedHashSet<String[]> listaKluczy;
    private RozmowaID biezacaRozmowa;
    private String zmiennaText;

    /** Konstruktor. Tworzy instancję komponentu Logowanie */
    public Sterowanie()
    {
        logowanie = new Logowanie(this);
    }

    /** Uruchamia sterowanie w programie. */
    public void sterowanieStart()
    {
        sterowanie = this;
        gui = Program.komponentGui(this);
        model = Program.komponentModel(this);

        gui.dezaktywujPanelRozmowa();
        gui.dezaktywujPrzyciskiPaneluUzytkownicy();
        gui.dezaktywujPrzyciskiPaneluUzytkownicyLaczenie();
        logowanie.zaloguj("domyslny", "domyslny");
        odswiezListeKontaktow();
        SwingUtilities.invokeLater(new Runnable(){public void run() { gui.panel2Dodaj.setEnabled(true); }} );
    }

    /** Odświeża listę kontaktów widzialną w głównym oknie programu */
    private void odswiezListeKontaktow()
    {
        String [] t = wygenerujListeUzytkownikow();
        if(t[0] != "")
        SwingUtilities.invokeLater(new Runnable(){public void run() {
  gui.listaUzytkownikow.setListData(wygenerujListeUzytkownikow());
                                                                    gui.listaUzytkownikow.setEnabled(true);
                                                                    gui.panel2Usun.setEnabled(true);}} );
        else SwingUtilities.invokeLater(new Runnable(){public void run() {
  gui.listaUzytkownikow.setListData(wygenerujListeUzytkownikow());
                                                                    gui.listaUzytkownikow.clearSelection();
                                                                    gui.listaUzytkownikow.setEnabled(false);
                                                                    gui.panel2Usun.setEnabled(false);}} );

    }

    /** Tworzy listę kontaktów, która będzie widoczna w głównym oknie programu */
    private String[] wygenerujListeUzytkownikow()
    {
        listaKluczy =  model.listaUZ();
        Iterator<String[]> i = listaKluczy.iterator();
        String[] lista;
        if(listaKluczy.size()!=0)
        lista = new String[listaKluczy.size()];
        else lista =  new String[] { "" };
```

```
77          int j = 0;
78          while(i.hasNext())
79          {
80              lista[j] = i.next()[0];
81              j++;
82          }
83          return lista;
84      }
85
86      /** Odbiera od interfersju graficznego żądanie zamknięcia programu i zamyka program
87       * @param guii Referencja do interfejsu graficznego
88       */
89      public void zadanieZamknieciaProgramu(Gui guii)
90      {
91          if(guii == Program.komponentGui(this))
92          {
93              if(biezacaRozmowa != null)
94              {
95                  rozlacz();
96              }
97              if(online) offlineUstaw();
98              logowanie.wyloguj();
99              System.exit(0);
100         }
101
102     }
103
104     /** Odbiera od interfersju graficznego żądanie dodania użytkownika zdalnego
105      * @param param Tablica z parametrami określającymi nowego użytkownika zdalnego. param[0]:nazwa użytkownika,
    param[1]:adres ip */
106     public void dodanoUzytkownikaZdalnego(String[] param)
107     {
108         InetSocketAddress adr = new InetSocketAddress(param[1],model.zwrocAktywnyPortSluchajacy());
109         try{
110             model.dodajUzytkownikaZdalnego(param[0], adr);
111         }
112         catch(Wyjatek w)
113         {
114             odbierzWyjatek(w.getMessage());
115         }
116         odswiezListeKontaktow();
117
118     }
119
120     /** Odbiera od interfersju graficznego żądanie dodania użytkownika zdalnego
121      * @param guii Referencja do interfejsu graficznego
122      */
123     public void usunietoUzytkownikaZdalnego(Gui guii)
124     {
125         if(guii == gui)
126         if(!gui.listaUzytkownikow.isSelectionEmpty())
127         {
128             try
129             {
130                 String[] refer = zwrocUzytkownikaWolajacegoZListy();
131                 model.usunUzytkownikaZdalnego(refer[0], refer[1]);
132             } catch (Wyjatek e)
133             {
134                 odbierzWyjatek(e.getMessage());
135             }
136             odswiezListeKontaktow();
137             }
138     }
139
140     /** Zwraca użytkownika który obecnie jest zaznaczony na liście
141      * @return Tablicę z parametrami określającymi użytkownika zdalnego obecnie zaznaczonego na liście.
    param[0]:nazwa użytkownika, param[1]:adres ip
142      */
143     private String[] zwrocUzytkownikaWolajacegoZListy()
144     {
145         Iterator<String[]> i = listaKluczy.iterator();
146         int a = gui.listaOstatniIndeks;
147         for(int j = 0; j < a; j++)
148         {
149             i.next();
150         }
151         return i.next();
152     }
```

```
153
154     /** Odbiera od interfejsu graficznego żądanie przejścia w tryb online
155      * @param guii Referencja do interfejsu graficznego
156      * @param stan Tryb przejścia w online. 0:Internet, 1:Localhost1, 2:Localhost2
157      */
158     public void zmienionoNaOnline(Gui guii, int stan)
159     {
160         if(guii == Program.komponentGui(this))
161             {
162                 try {
163                     model.ustawOnline(true, LocalHost.fromInt(stan));
164
165
166                 }
167                  catch (ClassNotFoundException e) {
168                         odbierzWyjatek(e.getMessage());
169                     } catch (IOException e) {
170                         odbierzWyjatek(e.getMessage());
171                     } catch (Wyjatek e) {
172                         if(e.getMessage().contains("otworzy\u0107")) {
173                             odbierzWyjatek(e.getMessage());
174                             gui.menuOnline.setSelected(false);
175                             return;
176                         }
177                         odbierzWyjatek(e.getMessage());
178                     }
179
180                 online = true;
181                 sprawdzPrzyciskiLaczenia();
182                 switch(stan)
183                 {
184                 case 0: SwingUtilities.invokeLater(new Runnable(){public void run() { gui.napisPaska.setText("Online:
    " + Program.komponentModel(sterowanie).mojeIP()); }} ); break;
185                 case 1: SwingUtilities.invokeLater(new Runnable(){public void run() { gui.napisPaska.setText("Online:
    " + "127.0.0.1:5623"); }} ); break;
186                 case 2: SwingUtilities.invokeLater(new Runnable(){public void run() { gui.napisPaska.setText("Online:
    " + "127.0.0.1:5624"); }} ); break;
187                 }
188                 gui.dezaktywujWyborPrzelacznikowOnline();
189
190
191             }
192
193     }
194
195     /** Odbiera od interfejsu graficznego żądanie przejścia w tryb offline
196      * @param guii Referencja do interfejsu graficznego
197      */
198     public void zmienionoNaOffline(Gui guii)
199     {
200         if(guii == Program.komponentGui(this))
201             {
202             offlineUstaw();
203             online = false;
204             }
205
206     }
207
208     /** Ustawia offline w sterowaniu */
209     private void offlineUstaw()
210     {
211         if(biezacaRozmowa != null)
212             rozlacz();
213         try {
214             model.ustawOnline(false, null);
215
216         } catch (ClassNotFoundException e) {
217             odbierzWyjatek(e.getMessage());
218         } catch (IOException e) {
219             odbierzWyjatek(e.getMessage());
220         } catch (Wyjatek e) {
221             odbierzWyjatek(e.getMessage());
222         }
223         online = false;
224         sprawdzPrzyciskiLaczenia();
225         SwingUtilities.invokeLater(new Runnable(){public void run() { gui.napisPaska.setText("Offline"); }} );
226         gui.aktywujWyborPrzelacznikowOnline();
227     }
```

```
228
229     /** Odbiera od interfejsu graficznego zmianę zaznacznia na liście użytkowników
230      * @param guii Referencja do interfejsu graficznego
231      */
232     public void zmienionoStanListy(Gui guii)
233     {
234         if(guii == Program.komponentGui(this))
235         {
236             sprawdzPrzyciskiLaczenia();
237                     }
238
239
240     }
241
242     /** Odbiera od interfejsu graficznego żądanie nawiązania połączenia z użytkownikiem zdalnym
243      * @param guii Referencja do interfejsu graficznego
244      */
245     public void zadaniePolaczenia(Gui guii)
246     {
247         if(guii == gui)
248             if(!gui.listaUzytkownikow.isSelectionEmpty())
249             {
250                 pasekPostepuUstaw(true, "Łączenie...");
251                 String[] refer = zwrocUzytkownikaWolajacegoZListy();
252                 try
253                 {
254                     biezacaRozmowa = model.rozpocznijRozmowe(model.zwrocUzytkownikaZdalnego(refer[1], refer[0]));
255                 } catch (Wyjatek e)
256                 {
257                     odbierzWyjatek(e.getMessage());
258                 }
259                 finally
260                 {
261                     pasekPostepuUstaw(false,null);
262                 }
263                 pasekPostepuUstaw(false,null);
264                 if(biezacaRozmowa == null)
265                 SwingUtilities.invokeLater(new Runnable(){public void run() { gui.oknoDialogoweWyjatek("Brak
    odpowiedzi od użytkownika: " + zwrocUzytkownikaWolajacegoZListy()[0] + "(" + zwrocUzytkownikaWolajacegoZListy()[1] +
    ")");}} );
266                 else if(biezacaRozmowa.zwrocIdRozmowy() == 0)
267                 SwingUtilities.invokeLater(new Runnable(){public void run() { gui.oknoDialogoweWyjatek("Użytkownik: "
    + zwrocUzytkownikaWolajacegoZListy()[0] + "(" + zwrocUzytkownikaWolajacegoZListy()[1] + ") nie zgodził się na
    rozpoczęcie rozmowy..");}} );
268                 else
269                 {
270                 SwingUtilities.invokeLater(new Runnable(){public void run() { gui.panel1Etykieta.setText("Połączono
    z: " + zwrocUzytkownikaWolajacegoZListy()[0] + ", IP: " + zwrocUzytkownikaWolajacegoZListy()[1]);
271                                                                             gui.aktywujPanelRozmowa();}} );
272                 sprawdzPrzyciskiLaczenia();
273                 }
274             }
275
276     }
277
278     /** Odbiera od interfejsu graficznego żądanie zakończenia połączenia z użytkownikiem zdalnym
279      * @param guii Referencja do interfejsu graficznego
280      */
281     public void zadanieRozlaczenia(Gui guii)
282     {
283         if(guii == gui)
284         {
285             rozlacz();
286         }
287     }
288
289     /** Odbiera od interfejsu graficznego żądanie wysłania pliku do uczestników rozmowy
290      * @param guii Referencja do interfejsu graficznego
291      */
292     public void zadanieWyslaniaPliku(Gui guii)
293     {
294         if(guii == gui)
295         {
296         FileDialog oknoPlik = new FileDialog(gui, "Wybierz plik", FileDialog.LOAD);
297         oknoPlik.setVisible(true);
298         String sciezkaPliku = oknoPlik.getDirectory() + oknoPlik.getFile();
299         if(sciezkaPliku.isEmpty()) pasekPostepuUstaw(true, "Wysłano żądanie odebrania pliku. Oczekiwanie na
    odpowiedź...");
```

```
300         try {
301             StanWyslaniaPliku stan = model.wyslijPlik(null, biezacaRozmowa, sciezkaPliku);
302             pasekPostepuUstaw(false,null);
303             if(stan.stan == 1) SwingUtilities.invokeLater(new Runnable(){public void run() {
    gui.oknoDialogoweInfo("Plik został wysłany!", "Stan wysyłania pliku");}} );
304             else if(stan.stan == 0) SwingUtilities.invokeLater(new Runnable(){public void run() {
    gui.oknoDialogoweInfo("Użytkownik nie zgadza się na odbiór pliku!", "Stan wysyłania pliku");}} );
305             else if(stan.stan == -1) SwingUtilities.invokeLater(new Runnable(){public void run() {
    gui.oknoDialogoweInfo("Użytkownik nie odpowiada na żądanie odebrania pliku!", "Stan wysyłania pliku");}} );
306             else ;
307 
308 
309         } catch (Wyjatek e) {
310             odbierzWyjatek(e.toString());
311         }
312         }
313     }
314 
315     /** Zakańcza bieżącą rozmowę */
316     private void rozlacz()
317     {
318         try {
319             model.zakonczRozmowe(biezacaRozmowa);
320         } catch (IOException e) {
321             odbierzWyjatek(e.getMessage() + e.getCause());
322         }
323         //model.zakonczRozmowe(biezacaRozmowa);
324         biezacaRozmowa = null;
325         sprawdzPrzyciskiLaczenia();
326         gui.dezaktywujPanelRozmowa();
327         SwingUtilities.invokeLater(new Runnable(){public void run() { gui.panel1Etykieta.setText("Rozłączono");}} );
328     }
329 
330     /** Odbiera od interfejsu graficznego żądanie wysłania wiadomości do uczestników rozmowy
331      * @param guii Referencja do interfejsu graficznego
332      */
333     public void zadanieWyslaniaWiadomosci(Gui guii)
334     {
335         if(guii == gui)
336         {
337             zmiennaText = gui.panel1PoleTekstowesc.getText();
338 
339             try {
340                 model.wyslijWiadomosc(biezacaRozmowa, new String(gui.panel1PoleTekstoweWpissc.getText()));
341             } catch (IOException e) {
342             odbierzWyjatek(e.getMessage()+e.getCause());
343             }
344             SwingUtilities.invokeLater(new Runnable(){public void run() {
345             zmiennaText = zmiennaText.concat("\n" + gui.panel1PoleTekstoweWpissc.getText());
346             gui.panel1PoleTekstowesc.setText(zmiennaText);
347             gui.panel1PoleTekstoweWpissc.setText("");
348             }} );
349         }
350     }
351 
352     /** Funkcja zarządza przyciskami związanymi z połączeniem */
353     private void sprawdzPrzyciskiLaczenia()
354     {
355         if(online && !gui.listaUzytkownikow.isSelectionEmpty())
356         {
357             SwingUtilities.invokeLater(new Runnable(){public void run() {
358                                                     if(biezacaRozmowa!=null)
359                                                     {
360 
361 
    if(biezacaRozmowa.zwrocNazweUzykownika().equals(zwrocUzytkownikaWolajacegoZListy()[0]))
362                                                                 { gui.panel2Polacz.setEnabled(false);
363                                                                   gui.panel2Rozlacz.setEnabled(true); }
364 
365                                                         else
366                                                         {
367                                                         gui.panel2Polacz.setEnabled(false);
368                                                         gui.panel2Rozlacz.setEnabled(false);
369                                                         }
370                                                     }
371                                                     else
372                                                         {
373                                                             gui.panel2Polacz.setEnabled(true);
```

```
374                                                             gui.panel2Rozlacz.setEnabled(false);
375                                                         }
376                                                     }} );
377         }
378         else
379             gui.dezaktywujPrzyciskiPaneluUzytkownicyLaczenie();
380 
381     }
382 
383     //-----------------------------------------------------------------------------------------Interfejs prawy
384     @Override
385     public void odbierzWiadomosc(String ip, RozmowaID id, String wiadomosc) {
386         String text = gui.panel1PoleTekstowesc.getText();
387         Calendar czas = Calendar.getInstance();
388         gui.panel1PoleTekstowesc.setText(text.concat("\n\t" + czas.get(Calendar.HOUR_OF_DAY)
389                 + ":" + czas.get(Calendar.MINUTE) + ":" + czas.get(Calendar.SECOND) +" " + wiadomosc));
390     }
391 
392     @Override
393     public OdpowiedzSterowania odbierzRozmowe(String ip, RozmowaID rozmowa) {
394         int odp = gui.oknoDialogowePytanie("Czy zgadzasz si\u0119 na rozpocz\u0119cie rozmowy z
    nast\u0119puj\u0105cym u\u017cytkownikiem:\n"
395                 + "u\u017cytkownik " + rozmowa.zwrocNazweUzykownika() + ", IP: " + ip + ".");
396         switch(odp)
397         {
398         case -1: return new OdpowiedzSterowania(-1);
399         case 0: return new OdpowiedzSterowania(0);
400         case 1: {
401 
402             SwingUtilities.invokeLater(new Runnable(){public void run() { gui.panel1Etykieta.setText("Połączono z: "
    + zwrocUzytkownikaWolajacegoZListy()[0] + ", IP: " + zwrocUzytkownikaWolajacegoZListy()[1]);
403                                                                     gui.aktywujPanelRozmowa(); }} );
404             odswiezListeKontaktow();
405             biezacaRozmowa = rozmowa;
406             sprawdzPrzyciskiLaczenia();
407             return new OdpowiedzSterowania(1);
408         }
409         }
410         return null;
411 
412     }
413 
414     /** Odbiera zakończenie rozmowy spowodowane niespodziewanym rozłączeniem */
415     public void odbierzZakonczenieRozmowyWymuszone() {
416             model.rozmowy().get(biezacaRozmowa.zwrocIdRozmowy()).zakonczWymuszenie();
417         SwingUtilities.invokeLater(new Runnable(){public void run() {
418             gui.dezaktywujPanelRozmowa();
419             gui.panel1Etykieta.setText("Rozłączono"); }} );
420             biezacaRozmowa = null;
421             sprawdzPrzyciskiLaczenia();
422 
423     }
424 
425     @Override
426     public void odbierzZakonczenieRozmowy(String ip, RozmowaID id) {
427         SwingUtilities.invokeLater(new Runnable(){public void run() {
428             gui.dezaktywujPanelRozmowa();
429             gui.panel1Etykieta.setText("Rozłączono"); }} );
430             biezacaRozmowa = null;
431             sprawdzPrzyciskiLaczenia();
432 
433     }
434 
435     @Override
436     public void odbierzWyjatek(String opis)
437     {
438         gui.oknoDialogoweWyjatek(opis);
439 
440     }
441 
442     @Override
443     public OdpowiedzSterowania odbierzPlik(String ip, RozmowaID id, PlikNaglowek plik, StanWyslaniaPliku postep)
444     {
445         long dlug = plik.danePliku.length();
446 
447         int odp = gui.oknoDialogowePytanie("Czy zgadzasz się na odebranie następującego pliku:\n"
448                 + "u\u017cytkownik: " + id.zwrocNazweUzykownika() + ", IP: " + ip + "\n"
449                 + "nazwa pliku: " + plik.danePliku.getName() + ", rozmiar: " + dlug + " bajtów");
```

```
450         switch(odp)
451         {
452         case -1: return new OdpowiedzSterowania(-1);
453         case 0: return new OdpowiedzSterowania(0);
454         case 1: pasekPostepuUstaw(true,"Pobieranie pliku...");
455             return new OdpowiedzSterowania(1);
456         }
457         return new OdpowiedzSterowania(-1);
458     }
459 
460     /** Informuje o odebraniu wcześniej uzgodnionego pakietu z danymi (pliku)
461      * @param nazwaKoncowa Nazwa pliku
462      * @return Ścieżka pliku, gdzie plik będzie zapisany
463      */
464     public String odebranoPlik(String nazwaKoncowa)
465     {
466         pasekPostepuUstaw(false,null);
467         FileDialog oknoPlik = new FileDialog(gui, "Wybierz katalog docelowy", FileDialog.SAVE);
468         oknoPlik.setFile(nazwaKoncowa);
469         oknoPlik.setVisible(true);
470         return oknoPlik.getDirectory() + oknoPlik.getFile();
471 
472     }
473 
474     /** Zarządza paskiem postępu
475      * @param aktywny Wartość "true" sprawia że pasek jest widoczny, wartość "false" przeciwnie.
476      * @param tekst Tekst wyświetlany na pasku
477      */
478     public void pasekPostepuUstaw(final boolean aktywny, final String tekst)
479     {
480         try {
481             SwingUtilities.invokeAndWait(new Runnable(){public void run() {
482                 if(aktywny)
483                 gui.pasekPostepu.setVisible(true);
484                 else gui.pasekPostepu.setVisible(false);
485                 if(tekst != null)
486                     {
487                     gui.pasekPostepu.setString(tekst);
488                     gui.pasekPostepu.setStringPainted(true);
489                     }
490                 else gui.pasekPostepu.setStringPainted(false);
491             }});
492         } catch (InvocationTargetException e) {
493             // TODO Auto-generated catch block
494             e.printStackTrace();
495         } catch (InterruptedException e) {
496             // TODO Auto-generated catch block
497             e.printStackTrace();
498         }
499 
500 
501     }
502 
503 }
504 
505 
506 
507 
```