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

import java.io.*;
import java.net.InetSocketAddress;
import java.net.Socket;
import java.net.SocketException;
import pl.nalazek.komunikator.Program;
import pl.nalazek.komunikator.logika.logowanie.UzytkownikZdalny;

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

/** Klasa uruchumiająca nowy wątek do obsługi przychodzących pakietów */
public class WatekObslugujacyPakiet extends Watek{

    private ObjectInputStream strumienPrzych;
    private Socket gniazdoSluchajace = null;
    private String adresIp;

    /** Konstruktor
     * @param iS Obietkowy strumień przychodzący
     * @param gniazdoSluchajace Gniazdo do którego podpięty jest strumień
     */
    public WatekObslugujacyPakiet(ObjectInputStream iS, Socket gniazdoSluchajace)
    {
        strumienPrzych = iS;
        this.gniazdoSluchajace = gniazdoSluchajace;
    }

    /**  Główna metoda wątku */
    public void run()
    {
        try
        {
            InetSocketAddress adresSockt = (InetSocketAddress)gniazdoSluchajace.getRemoteSocketAddress();
            String nazwaUzytkownika;
            if(adresSockt.getAddress().isLoopbackAddress()) adresIp = "127.0.0.1";
            else adresIp = adresSockt.getAddress().getHostAddress();
            while(!interrupted()) {
            Pakiet pakiet = (Pakiet)strumienPrzych.readObject();
            nazwaUzytkownika = pakiet.zwrocIdRozmowyZrodlowej().zwrocNazweUzykownika();
            System.out.println("Pakiet obsłużony");
            UzytkownikZdalny uZdalny = modelRef.zwrocUzytkownikaZdalnego(adresIp, nazwaUzytkownika);


        NrIdPakietu nr = pakiet.getNrId();

        /** Pakiet z wiadomością */
        if(nr == NrIdPakietu.WIADOMOSC)
        {
        Wiadomosc wiadomosc = (Wiadomosc)pakiet;
        Program.komponentSterowanie().odbierzWiadomosc(adresIp, wiadomosc.zwrocIdRozmowyDocelowej(),
  wiadomosc.zwrocTekst());
        }

        /** Pakiet z żądaniem odebrania danych */
        else if(nr == NrIdPakietu.DANE_PYTANIE)
        {
            DanePytanie pytanie = (DanePytanie)pakiet;
            OdpowiedzSterowania kod = Program.komponentSterowanie().odbierzPlik(adresIp,
  pytanie.zwrocIdRozmowyDocelowej(), pytanie.plikInfo, null);
            switch(kod.kod)
            {
            case 0: //odmowa
                synchronized(modelRef.rozmowy().get(pytanie.zwrocIdRozmowyDocelowej().zwrocIdRozmowy()).semaforKoleje
  k)
                { modelRef.rozmowy().get(pytanie.zwrocIdRozmowyDocelowej().zwrocIdRozmowy())
                    .obslugaKolejki(uZdalny, new DaneOdpowiedz(pytanie,false)); }
                break;
            case 1: //zgoda
                modelRef.pytania().put(pytanie.zwrocIdRozmowyDocelowej().zwrocIdRozmowy(), null);
                synchronized(modelRef.rozmowy().get(pytanie.zwrocIdRozmowyDocelowej().zwrocIdRozmowy()).semaforKoleje
  k)
                {
                modelRef.rozmowy().get(pytanie.zwrocIdRozmowyDocelowej().zwrocIdRozmowy())
                    .obslugaKolejki(uZdalny, new DaneOdpowiedz(pytanie,true)); }
```

```
75
76                  try {
77
78
79                          for(int i = 0; i<800; i++)
80                          {
81                              Thread.sleep(20);
82                              /* sprawdzenie czy odpowiedz która przyszła jest opdowiedzią oczekiwaną */
83                              Dane dane = (Dane)
    modelRef.pytania().get(pytanie.zwrocIdRozmowyDocelowej().zwrocIdRozmowy());
84                              if(dane != null)
85                                  {
86                                  /* Odpowiedź pozytywna */
87                                  if(dane.zwrocKlucz() == pytanie.zwrocKlucz())
88                                          {
89                                          dane.zwrocPlik().zapiszPlik(kod.sciezka);
90                                          modelRef.pytania().remove(pytanie.zwrocKlucz());
91                                          }
92                                  /* Odpowiedź negatywna */
93                                  else
94                                          {
95                                          modelRef.pytania().remove(pytanie.zwrocKlucz());
96                                          }
97                                  }
98                          }
99                          modelRef.pytania().remove(pytanie.zwrocKlucz());
100
101                     }
102                     catch (InterruptedException e)
103                     {
104                         throw new Wyjatek("W oczekiwaniu na odpowiedź wystąpił wyjątek.\n" + e.getMessage() + ";;" +
    e.getCause());
105                     }
106                     catch (IOException e)
107                     {
108                     throw new Wyjatek("Przy zapisywaniu pliku wystąpił wyjątek.\n" + e.getMessage() + ";;" +
    e.getCause());
109                     }
110                 break;
111             case -1: //brak odpowiedzi
112                 break;
113             }
114         }
115
116         /** Pakiet z żądaniem rozmowy */
117         else if(nr == NrIdPakietu.ROZMOWA_PYTANIE)
118         {
119             RozmowaPytanie pytanie = (RozmowaPytanie)pakiet;
120             /* sprawdź użytkownika zdalnego w kontaktach*/
121             if(uZdalny == null)
122                 uZdalny = modelRef.dodajUzytkownikaZdalnego(nazwaUzytkownika, adresSockt);
123             /* stworzenie rozmowy */
124             Rozmowa rozmowa = new Rozmowa(modelRef.zwrocAktywnyUzytkownikLokalny(),uZdalny);
125             OdpowiedzSterowania kod = Program.komponentSterowanie().odbierzRozmowe(adresIp,
    rozmowa.zwrocIdRozmowy());
126             switch(kod.kod)
127             {
128             case 0: //odmowa
129                 //( new WatekWysylajacy(new RozmowaOdpowiedz(pytanie, new RozmowaID(0,"UżytkownikNegatywny")),
    adresIp, Model.zwrocAktywnyPortSluchajacyPrzeciwny()) ).start();
130                 synchronized(rozmowa.semaforKolejek)
131                 { rozmowa.obslugaKolejki(uZdalny, new RozmowaOdpowiedz(pytanie, new
    RozmowaID(0,"UżytkownikNegatywny"))); }
132                 break;
133             case 1: //zgoda
134                 rozmowa.dodajIdRozmowyDocelowej(uZdalny, pytanie.zwrocIdRozmowyZrodlowej());
135
136                 /* dodanie do aktywnych rozmów */
137                 modelRef.rozmowy().put(rozmowa.zwrocIdRozmowy().zwrocIdRozmowy(), rozmowa);
138
139                 /* wysłanie pakietu potwierdzającego */
140                 synchronized(rozmowa.semaforKolejek)
141                 { rozmowa.obslugaKolejki(uZdalny, new RozmowaOdpowiedz(pytanie,rozmowa.zwrocIdRozmowy())); }
142                 break;
143             case -1: //brak odpowiedzi
144                 break;
145             }
146
```

```
147         }
148
149         /** Pakiet z opdowiedzią na żądanie rozmowy */
150         else if(nr == NrIdPakietu.ROZMOWA_ODPOWIEDZ)
151         {
152         RozmowaOdpowiedz odpowiedz = (RozmowaOdpowiedz)pakiet;
153         if( modelRef.pytania().containsKey(odpowiedz.zwrocIdRozmowyDocelowej().zwrocIdRozmowy()) )
154             modelRef.pytania().put(odpowiedz.zwrocIdRozmowyDocelowej().zwrocIdRozmowy(), odpowiedz);
155         else throw new Wyjatek("Odebrano przeterminowany  pakiet z odpowiedzią na żądanie rozmowy!");
156         }
157
158
159         /** Pakiet z opdowiedzią na żądanie danych */
160         else if(nr == NrIdPakietu.DANE_ODPOWIEDZ)
161         {
162         DaneOdpowiedz odpowiedz = (DaneOdpowiedz)pakiet;
163         if( modelRef.pytania().containsKey(odpowiedz.zwrocIdRozmowyDocelowej().zwrocIdRozmowy()) )
164             modelRef.pytania().put(odpowiedz.zwrocIdRozmowyDocelowej().zwrocIdRozmowy(), odpowiedz);
165         else throw new Wyjatek("Odebrano przeterminowany pakiet z odpowiedzią na żądanie danych!");
166         }
167
168         /** Pakiet z danymi */
169         else if(nr == NrIdPakietu.DANE)
170         {
171         Dane dane = (Dane)pakiet;
172         if( modelRef.pytania().containsKey(dane.zwrocIdRozmowyDocelowej().zwrocIdRozmowy()) )
173             {modelRef.pytania().remove(dane.zwrocIdRozmowyDocelowej().zwrocIdRozmowy());
174         String sciezka = Program.komponentSterowanie().odebranoPlik(dane.plikInfo.danePliku.getName());
175         dane.zwrocPlik().zapiszPlik(sciezka);
176             }
177         else throw new Wyjatek("Odebrano pakiet z danymi! Pakiet potwierdzający nie był wysyłany...");
178         }
179
180         /** Rozmowa koniec */
181         else if(nr == NrIdPakietu.ROZMOWA_KONIEC)
182         {
183         RozmowaKoniec pakietKoniec = (RozmowaKoniec)pakiet;
184         if( modelRef.rozmowy().containsKey(pakietKoniec.zwrocIdRozmowyDocelowej().zwrocIdRozmowy()))
185             {
186             modelRef.rozmowy().get(pakietKoniec.zwrocIdRozmowyDocelowej().zwrocIdRozmowy()).usunUzytkownikaZdalnego(m
    odelRef.zwrocUzytkownikaZdalnego(adresIp, nazwaUzytkownika), false);
187             Program.komponentSterowanie().odbierzZakonczenieRozmowy(adresIp, pakietKoniec.zwrocIdRozmowyDocelowej());
188             }
189
190         }
191
192         /** Inny pakiet */
193         else if(nr == NrIdPakietu.PAKIET)
194         {
195
196         }
197
198         /** Nieznany pakiet */
199         else
200         {
201             throw new Wyjatek("Odebrano nieznany pakiet!");
202         }
203         }
204         }
205         catch(ClassNotFoundException w)
206         {
207             Program.komponentSterowanie().odbierzWyjatek("W wątku: " + this.getName() + " wystąpił wyjątek.\n" +
    w.getMessage() + ";;" + w.getCause());
208             System.out.println(w.getMessage() + " " + w.getCause());
209         }
210         catch(EOFException w)
211         {
212             Program.komponentSterowanie().odbierzWyjatek("Zakończono połączenie przychodzące z " + adresIp + "!");
213         }
214         catch(SocketException w)
215         {
216             if(w.getMessage().contains("Connection reset")) {
217                 Program.komponentSterowanie().odbierzWyjatek("Zakończono połączenie przychodzące z " + adresIp +
    "!");
218                 Program.komponentSterowanie().odbierzZakonczenieRozmowyWymuszone();
219             }
220         }
221
```

```
222         catch(IOException w)
223         {
224             Program.komponentSterowanie().odbierzWyjatek("W wątku: " + this.getName() + " wystąpił wyjątek.\n" +
    w.getMessage() + ";;" + w.getCause());
225             System.out.println(w.getMessage() + " " + w.getCause());
226         }
227         catch(Exception w)
228         {
229             Program.komponentSterowanie().odbierzWyjatek("W wątku: " + this.getName() + " wystąpił wyjątek.\n" +
    w.getMessage() + ";;" + w.getCause());
230             System.out.println(w.getMessage() + " " + w.getCause());
231         }
232 
233 
234     }
235 
236 
237 }
238 
```