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

import java.net.InetSocketAddress;
import java.util.HashMap;
import java.util.Iterator;
import java.util.LinkedHashSet;
import pl.nalazek.komunikator.logika.logowanie.UzytkownikZdalny;

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

/** Klasa obsługująca kontakty dla aktualnego użytownika zalogowanego w programie */
public class AktywneKontakty {

    private LinkedHashSet<UzytkownikZdalny> kontakty;
    private HashMap<String,LinkedHashSet<UzytkownikZdalny>> kontaktyIp;
    private HashMap<String,LinkedHashSet<UzytkownikZdalny>> kontaktyNazwa;
    private LinkedHashSet<String[]> kontaktyNazwy;

    /** Konstruktor klasy
     * @param kontakty Kontakty w postaci LinkedHashSet<UzytkownikZdalny> na których klasa będzie pracować
     */
    public AktywneKontakty(LinkedHashSet<UzytkownikZdalny> kontakty)
    {
        this.kontakty = kontakty;
        przebuduj();
    }

    /** Funkcja sprawdza czy dany kontakt istnieje w spisie
     * @param ip IP kontaktu
     * @param nazwa Nazwa użytkownika
     * @return Wartość "true" w przypadku gdy kontakt istnieje, wartość "false" w przypadku gdy kontakt nie istnieje w spisie
     */
    public boolean sprawdzCzyIstnieje(String ip, String nazwa)
    {
        if(kontaktyIp.containsKey(ip))
        {
            Iterator<UzytkownikZdalny> i = kontaktyIp.get(ip).iterator();
            while(i.hasNext())
            {
                if(i.next().zwrocNazwe() == nazwa) return true;
            }
        }
        return false;
    }

    /** Funkcja szuka uzykownika zdalnego w spisie po adresie ip i jego nazwie
     * @param ip IP szukanego użytkownika
     * @param nazwa Nazwa szukanego użytkownika
     * @return Referencja do UzykownikaZdalnego w przypadku odnalezienia, "null" w przypadku nie odnalezienia.
     */
    public UzytkownikZdalny zwrocUzytkownikaZdalnego(String ip, String nazwa)
    {
        if(kontaktyIp.containsKey(ip))
        {
            LinkedHashSet<UzytkownikZdalny> list = kontaktyIp.get(ip);
            Iterator<UzytkownikZdalny> i = list.iterator();
            while(i.hasNext())
            {
                UzytkownikZdalny uZ = i.next();
                if(uZ.zwrocNazwe().equals(nazwa))
                    return uZ;
            }
        }
        return null;
    }

    /** Dodaje nowy kontakt
     * @param adresSocket Adres gniazdka użytkownika zdalnego
     * @param nazwa Nazwa użytkowika zdalnego
     * @return Referencja do obiektu UzytkownikZdalny, gdy kontakt został prawidłowo dodany, wartość "null" gdy kontakt już istnieje
     */
    public UzytkownikZdalny dodajKontakt(InetSocketAddress adresSocket, String nazwa)
    {
```

```
77          if(!sprawdzCzyIstnieje(adresSocket.getAddress().getHostAddress(),nazwa))
78          {
79              UzytkownikZdalny uZ = new UzytkownikZdalny(adresSocket,nazwa);
80              kontakty.add(uZ);
81              przebuduj();
82              return uZ;
83          }
84          return null;
85      }
86
87      /** Zwraca listę kontaktów w postaci tablicy String
88       * @return Duplikat listy kontaktów: String[3] w postaci [0]:nazwa, [1]:ip, [2]:nr użytkownika w systemie
89       */
90      public LinkedHashSet<String[]> zwrocKontaktyNazwy()
91      {
92          return new LinkedHashSet<String[]>(kontaktyNazwy);
93      }
94
95      /** Usuwa kontakt ze spisu *
96       * @param ip Ip użytkownika zdalnego
97       * @param nazwa Nazwa użytkownika zdalnego
98       * @return Wartość "true" gdy kontakt został usunięty, wartość "false" gdy kontakt nie istnieje
99       */
100     public boolean usunKontakt(String ip, String nazwa)
101     {
102         UzytkownikZdalny uZ = zwrocUzytkownikaZdalnego(ip,nazwa);
103         if(uZ != null)
104         {
105             kontakty.remove(uZ);
106             przebuduj();
107             return true;
108         }
109         return false;
110     }
111
112     /** Tworzy listy kontaktów na potrzeby klasy AktywneKontakty */
113     private void przebuduj()
114     {
115         utworzKontaktyIp();
116         utworzKontaktyNazwa();
117         utworzKontaktyNazwy();
118     }
119
120     /** Tworzy odwzorowanie HashMap adresów ip z użytkownikami zdalnymi */
121     private void utworzKontaktyIp()
122     {
123         Iterator<UzytkownikZdalny> i = kontakty.iterator();
124         kontaktyIp = new HashMap<String,LinkedHashSet<UzytkownikZdalny>>();
125
126         while(i.hasNext())
127         {
128             String ip = i.next().zwrocIp();
129             kontaktyIp.put(ip, new LinkedHashSet<UzytkownikZdalny>());
130             Iterator<UzytkownikZdalny> j = kontakty.iterator();
131             while(j.hasNext())
132             {
133                 UzytkownikZdalny uZ = j.next();
134                 if(uZ.zwrocIp() == ip)
135                     kontaktyIp.get(ip).add(uZ);
136             }
137         }
138     }
139
140     /** Tworzy odwzorowanie HashMap nazw użytkowników z użytkownikami zdalnymi */
141     private void utworzKontaktyNazwa()
142     {
143         Iterator<UzytkownikZdalny> i = kontakty.iterator();
144         kontaktyNazwa = new HashMap<String,LinkedHashSet<UzytkownikZdalny>>();
145
146         while(i.hasNext())
147         {
148             String nazwa = i.next().zwrocNazwe();
149             kontaktyNazwa.put(nazwa, new LinkedHashSet<UzytkownikZdalny>());
150             Iterator<UzytkownikZdalny> j = kontakty.iterator();
151             while(j.hasNext())
152             {
153                 UzytkownikZdalny uZ = j.next();
154                 if(uZ.zwrocNazwe() == nazwa)
```

```
155                     kontaktyNazwa.get(nazwa).add(uZ);
156                 }
157             }
158     }
159 
160     /** Tworzy zbiór LinkedHashSet zawierający tablice obiektów String z nazwą użytkownika, adresem ip i nr użytkownika */
161     private void utworzKontaktyNazwy()
162     {
163         Iterator<UzytkownikZdalny> i = kontakty.iterator();
164         kontaktyNazwy = new LinkedHashSet<String[]>();
165 
166         while(i.hasNext())
167         {
168             UzytkownikZdalny uZ = i.next();
169             String[] obj = new String[3];
170             obj[0] = uZ.zwrocNazwe();
171             obj[1] = uZ.zwrocIp();
172             obj[2] = uZ.zwrocNrUzytkownika().toString();
173             kontaktyNazwy.add(obj);
174 
175         }
176     }
177 
178 }
179 
```