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

import java.util.HashMap;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.Queue;
import pl.nalazek.komunikator.Program;
import pl.nalazek.komunikator.logika.logowanie.UzytkownikLokalny;
import pl.nalazek.komunikator.logika.logowanie.UzytkownikZdalny;

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

/** Klasa reprezentująca rozmowę w programie */
public class Rozmowa {
    private RozmowaID idRozmowy;
    private UzytkownikLokalny uLokalny;
    private volatile HashMap<UzytkownikZdalny,Queue<Pakiet>> kolejkaWych;
    private volatile HashMap<UzytkownikZdalny,WatekWysylajacy> watkiWysylajacy;
    private volatile HashMap<UzytkownikZdalny,RozmowaID> idRozmowyDocelowej;
    Object semaforKolejek;
    private boolean aktywna = false;

    /** Konstruktor
     * @param uLokalny Użytkownik lokalny inicjujący rozmowę
     * @param uZdalny Użytkownik zdalny z którym rozpoczynana jest rozmowa
     * @throws Wyjatek
     */
    public Rozmowa(UzytkownikLokalny uLokalny, UzytkownikZdalny uZdalny) throws Wyjatek
    {
        Integer[] zmienne = {uLokalny.zwrocNrUzytkownika(), uZdalny.zwrocNrUzytkownika()};
        idRozmowy = new RozmowaID(zmienne);
        idRozmowy.ustawNazweUzytkownika(uLokalny.zwrocNazwe());
        this.uLokalny = uLokalny;
        kolejkaWych = new HashMap<UzytkownikZdalny,Queue<Pakiet>>();
        watkiWysylajacy = new HashMap<UzytkownikZdalny,WatekWysylajacy>();
        idRozmowyDocelowej = new HashMap<UzytkownikZdalny,RozmowaID>();
        semaforKolejek = new Object();
        dodajUzytkownikaZdalnego(uZdalny);

    }

    /** Kończy wszystkie nawiązane połączenia w rozmowie. Wysyła pakiet RozmowaKoniec do uczestników rozmowy */
    public void zakoncz()
    {

        Iterator<UzytkownikZdalny> u = watkiWysylajacy.keySet().iterator();
        while(u.hasNext()) usunUzytkownikaZdalnego(u.next(),true);
        aktywna = false;
    }

    /** Kończy wszystkie nawiązane połączenia w rozmowie. Nie wysyła pakietu RozmowaKoniec do uczestników rozmowy */
    public void zakonczWymuszenie()
    {

        Iterator<UzytkownikZdalny> u = watkiWysylajacy.keySet().iterator();
        while(u.hasNext()) usunUzytkownikaZdalnego(u.next(),false);
        aktywna = false;
    }

    /** Zwraca id rozmowy
     * @return id rozmowy
     */
    synchronized public RozmowaID zwrocIdRozmowy()
    {
        return idRozmowy;
    }

    /** Zwraca użytkownika lokalnego - gospodarza rozmowy w programie
     * @return Użytkownik lokalny
     */
    synchronized public UzytkownikLokalny zwrocUzytkownikaLokalnego()
    {
        return uLokalny;
    }

```

```java
    /** Dodaje użytkownika zdalnego do rozmowy, tworząc nowy wątek wysyłający oraz kolejkę wychodzącą
     * @param uZdalny Dodawany użytkownik zdalny
     * @throws Wyjatek
     */
    synchronized public void dodajUzytkownikaZdalnego(UzytkownikZdalny uZdalny) throws Wyjatek
    {
        int port;
        port = Model.zwrocAktywnyPortSluchajacyPrzeciwny();
        WatekWysylajacy nowyWatek = new WatekWysylajacy(this,uZdalny,port);
        watkiWysylajacy.put(uZdalny, nowyWatek);
        watkiWysylajacy.get(uZdalny).start();
        while(nowyWatek.getState() != Thread.State.WAITING)
            if(nowyWatek.getState() == Thread.State.TERMINATED) {
                break;
            };
        if(nowyWatek.getState() == Thread.State.WAITING) {
            kolejkaWych.put(uZdalny, new LinkedList<Pakiet>());
            aktywna = true;
        }
        else {
            watkiWysylajacy.remove(uZdalny);
            zakoncz();
            throw new Wyjatek("Nie można połączyć ze zdalnym użytkownikiem!");
        }


    }

    /** Usuwa użytkownika zdalnego z rozmowy.
     * @param uZdalny Usuwany użytkownik zdalny
     * @param jaRozlaczam Wartość "true" oznacza, że inicjującym rozłączenie jest użytkownik lokalny. Wartość "false" oznacza, że inicjującym rozłączenie jest użytkownik zdalny
     */
    synchronized public void usunUzytkownikaZdalnego(UzytkownikZdalny uZdalny, boolean jaRozlaczam)
    {
        if(jaRozlaczam)obslugaKolejki(uZdalny, new RozmowaKoniec(idRozmowyDocelowej.get(uZdalny),idRozmowy));
        if(watkiWysylajacy.get(uZdalny) != null)
        while(!( watkiWysylajacy.get(uZdalny).getState() == Thread.State.WAITING ||
watkiWysylajacy.get(uZdalny).getState() == Thread.State.TERMINATED )) ;

        kolejkaWych.remove(uZdalny);
        synchronized(watkiWysylajacy.get(uZdalny).semafor)
        {
            watkiWysylajacy.get(uZdalny).interrupt();
        }
        watkiWysylajacy.remove(uZdalny);
        usunIdRozmowyDocelowej(uZdalny);
    }

    /** Dodaje id rozmowy docelowej dla danego użytkownika zdalnego w celu poprawnego adresowania pakietów
     * @param uZdalny Użytkownik zdalny
     * @param id Dodawane id rozmowy docelowej
     */
    synchronized public void dodajIdRozmowyDocelowej(UzytkownikZdalny uZdalny, RozmowaID id)
    {
        idRozmowyDocelowej.put(uZdalny, id);
    }

    /** Usuwa id rozmowy docelowej dla danego użytkownika zdalnego
     * @param uZdalny Użytkownik zdalny
     */
    synchronized private void usunIdRozmowyDocelowej(UzytkownikZdalny uZdalny)
    {
        idRozmowyDocelowej.remove(uZdalny);
    }

    /** Zwraca informację o tym czy rozmowa jest aktywna, tzn. czy jest połączenie co najmniej z jednym użytkownikiem zdalnym
     * @return Wartość "true" w przypadku gdy rozmowa jest aktywna, wartość "false" w przeciwnym wypadku
     */
    boolean czyAktywna()
    {
        return aktywna;
    }

    /** Obsługuje kolejkę wychodzącą dla danego użytkownika. Wyjmuje lub wkłada pakiety do kolejki, a także informuje wątek wysyłający o nowym pakiecie w kolejce.
     * @param uZdalny Użytkownik zdalny do którego kolejki chcemy się odnieść. Wartość "null" powoduje włożenie
```

```
   pakietu do wszytskich kolejek w rozmowie.
153     * @param pakiet Pakiet do przesłania. Wartość "null" powoduję wyjęcie z kolejki pierwszego elementu (funkcja
   używana przez wątek wysyłający).
154     * @return W przypadku podania jako drugi parametr wartości "null" zwraca Pakiet, w przeciwnym razie zwraca
   wartość "null".
155     */
156    Pakiet obslugaKolejki(UzytkownikZdalny uZdalny, Pakiet pakiet)
157    {
158        if(uZdalny != null)
159        {
160            if(pakiet == null)
161                synchronized(semaforKolejek)
162                { return kolejkaWych.get(uZdalny).poll(); }
163            else
164                {
165                if(pakiet.zwrocIdRozmowyDocelowej() == null)
   pakiet.dodajIdRozmowyDocelowej(idRozmowyDocelowej.get(uZdalny)); //oznaczenie pakietu
166                    try{
167                    synchronized(semaforKolejek)
168                    {kolejkaWych.get(uZdalny).offer(pakiet);}
169                    }
170                    catch(IllegalStateException w)
171                    {
172                    Program.komponentSterowanie().odbierzWyjatek("W rozmowie: " + idRozmowy.zwrocIdRozmowy() + "
   wystąpił wyjątek.\n" + w.getMessage() + ";;" + w.getCause());
173                    }
174                    synchronized (watkiWysylajacy.get(uZdalny).semafor)
175                    {
176                        watkiWysylajacy.get(uZdalny).semafor.notify();
177                        }
178                    return null;
179                }
180        }
181        else {
182            Iterator<UzytkownikZdalny> i;
183            synchronized(semaforKolejek) {i = kolejkaWych.keySet().iterator();}
184                while(i.hasNext())
185                {
186                    UzytkownikZdalny uZ = i.next();
187                    if(pakiet.zwrocIdRozmowyDocelowej() == null)
   pakiet.dodajIdRozmowyDocelowej(idRozmowyDocelowej.get(uZ)); //oznaczenie pakietu
188                    try{
189                        synchronized(semaforKolejek){
190                    kolejkaWych.get(uZ).offer(pakiet);}
191                    }
192                    catch(IllegalStateException w)
193                    {
194                    Program.komponentSterowanie().odbierzWyjatek("W rozmowie: " + idRozmowy.zwrocIdRozmowy() + "
   wystąpił wyjątek.\n" + w.getMessage() + ";;" + w.getCause());
195                    }
196                    synchronized (watkiWysylajacy.get(uZ).semafor)
197                    {
198                        watkiWysylajacy.get(uZ).semafor.notify();
199                        }
200                    return null;
201                }
202            }
203        return null;
204    }
205}
206
```