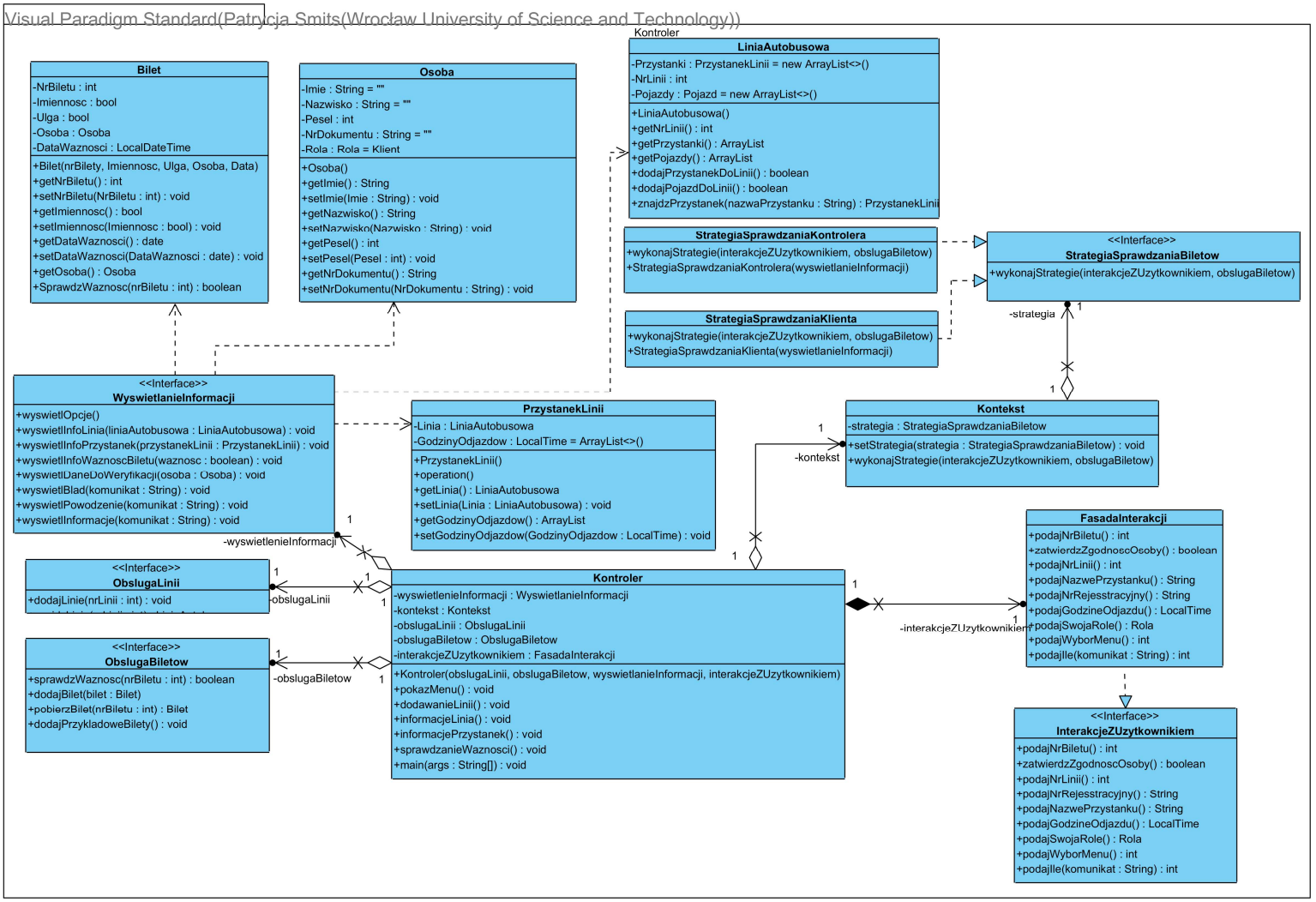
Julia Krok, 272981

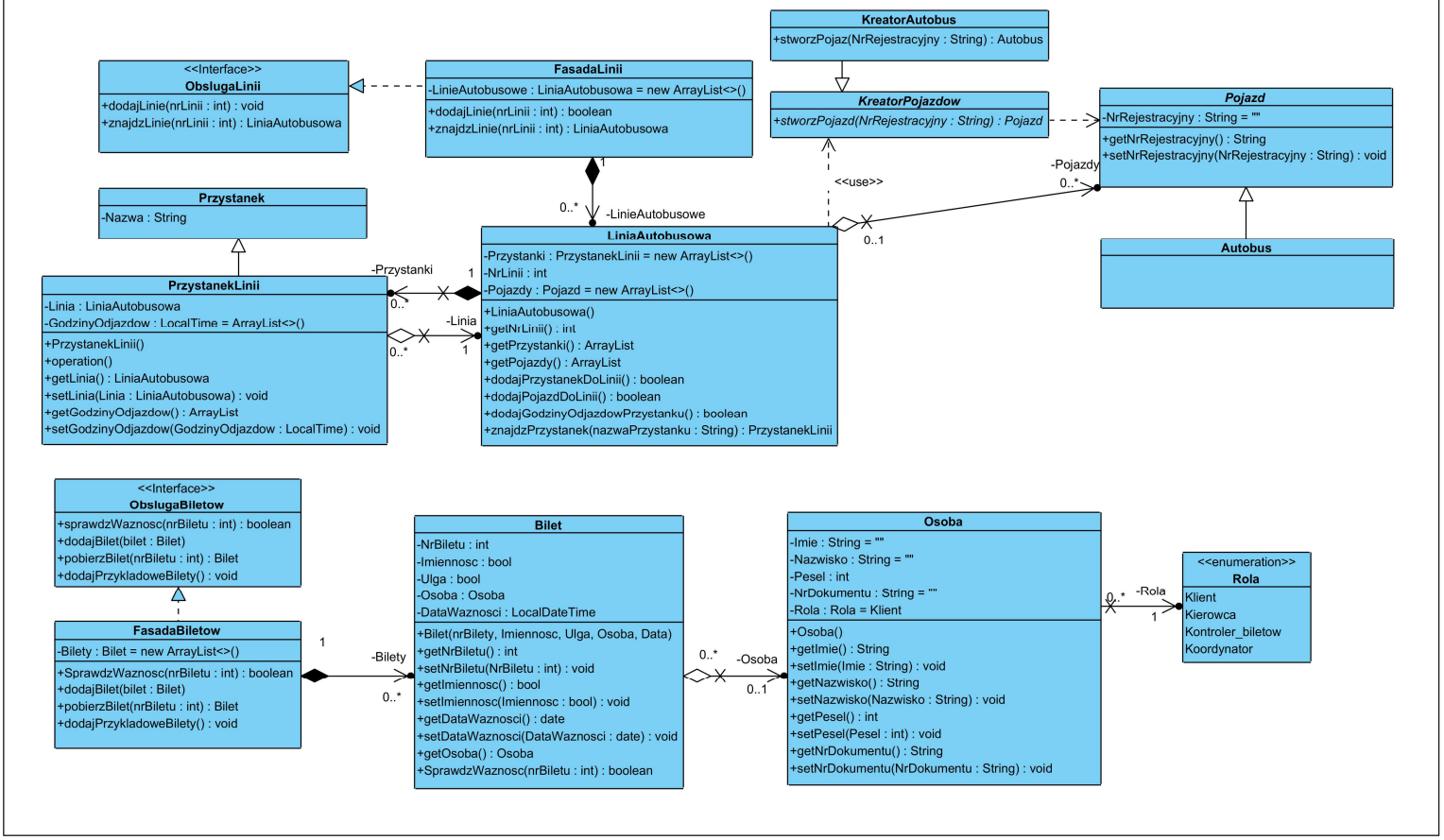
Patrycja Smits, 272940

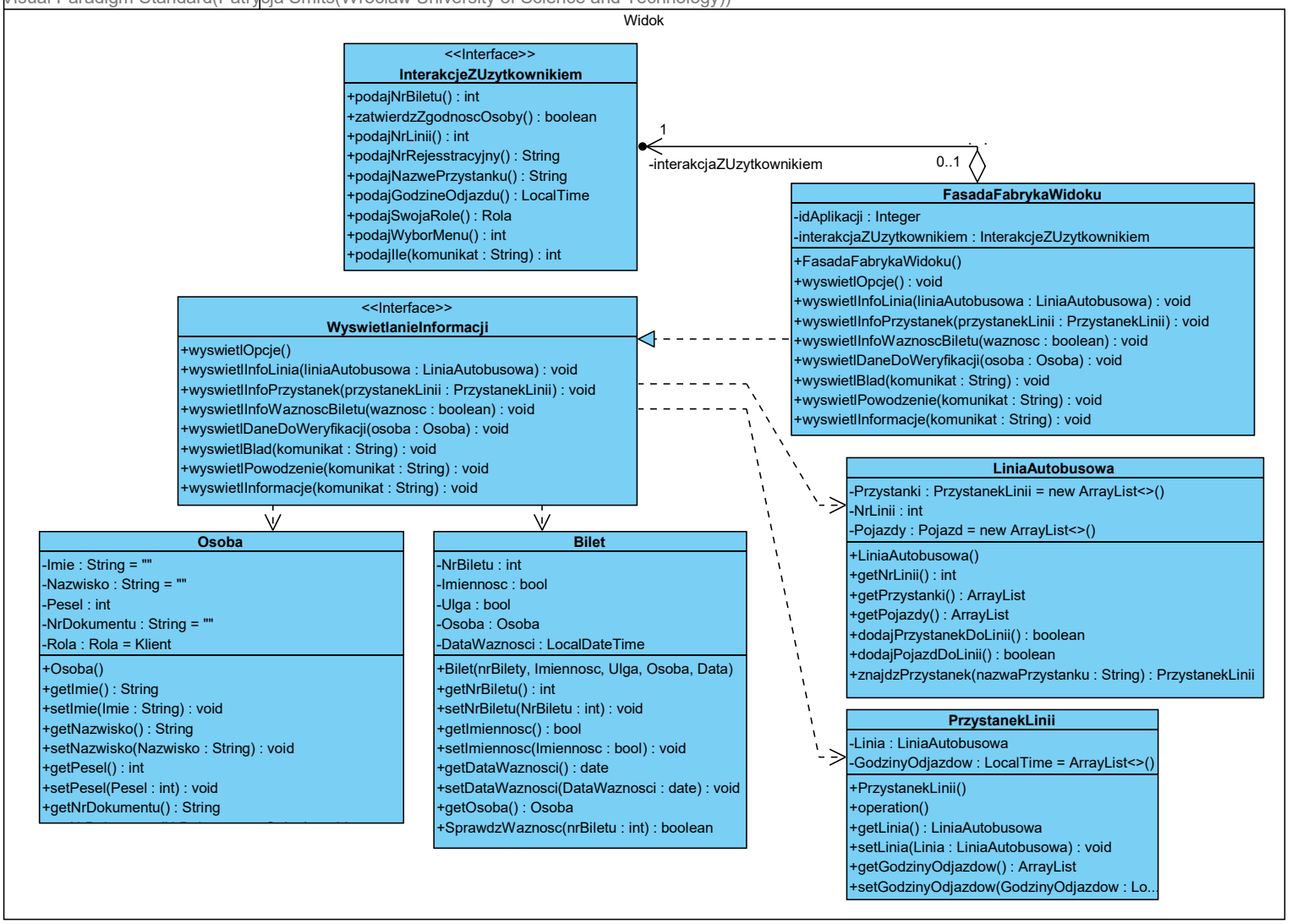
System informacyjny linii autobusowych

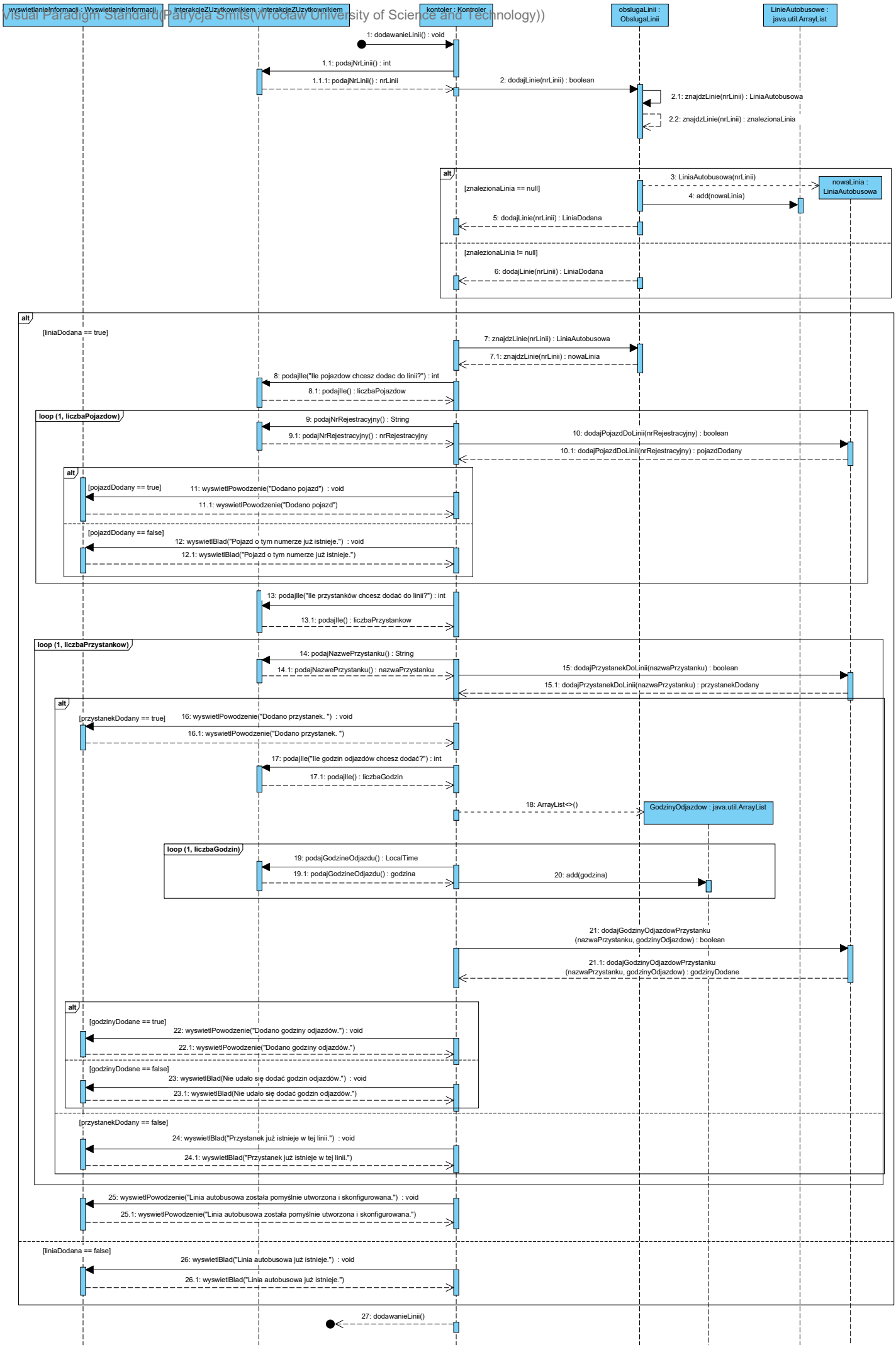
Etap 4: Wykonanie obiektowego modelu zachowania systemu w postaci diagramów sekwencji.

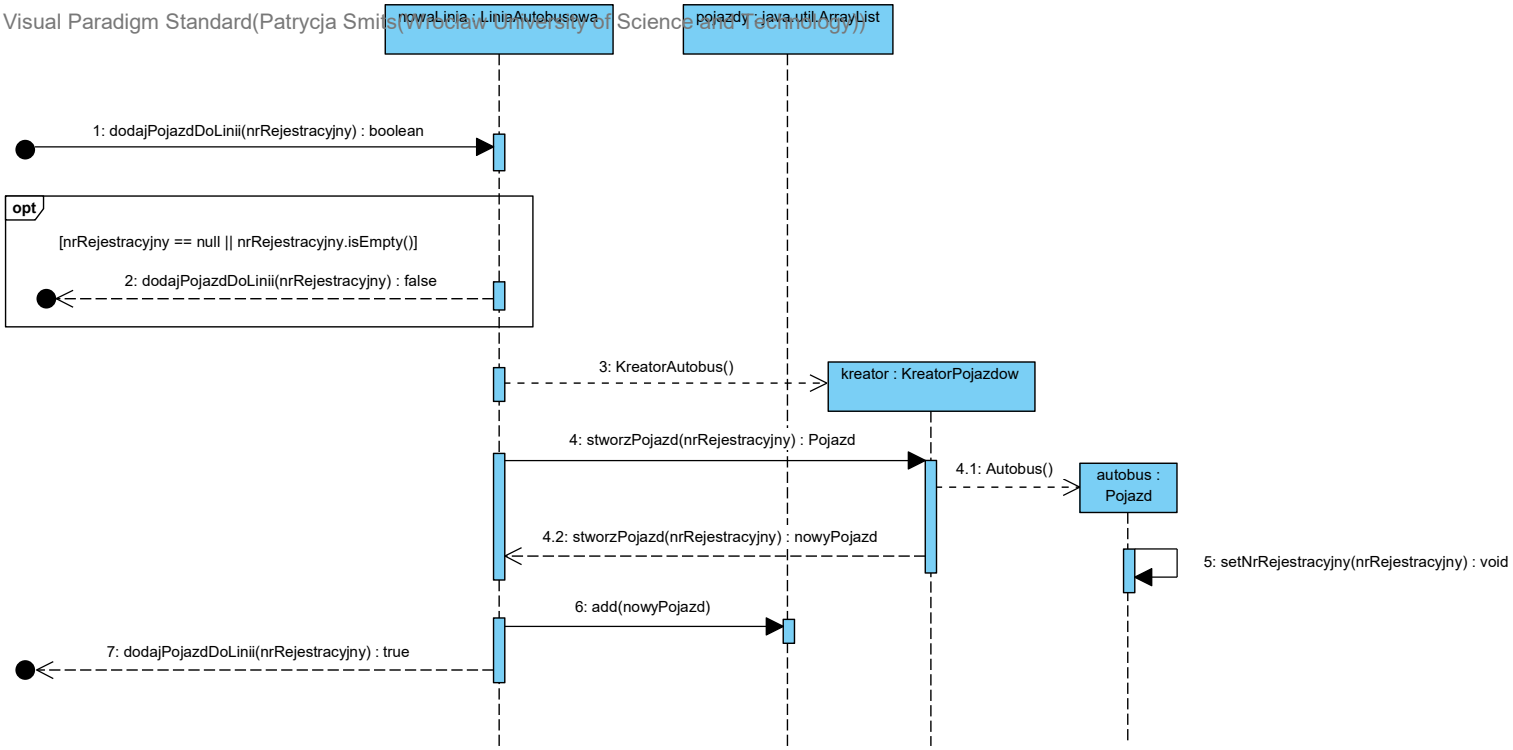


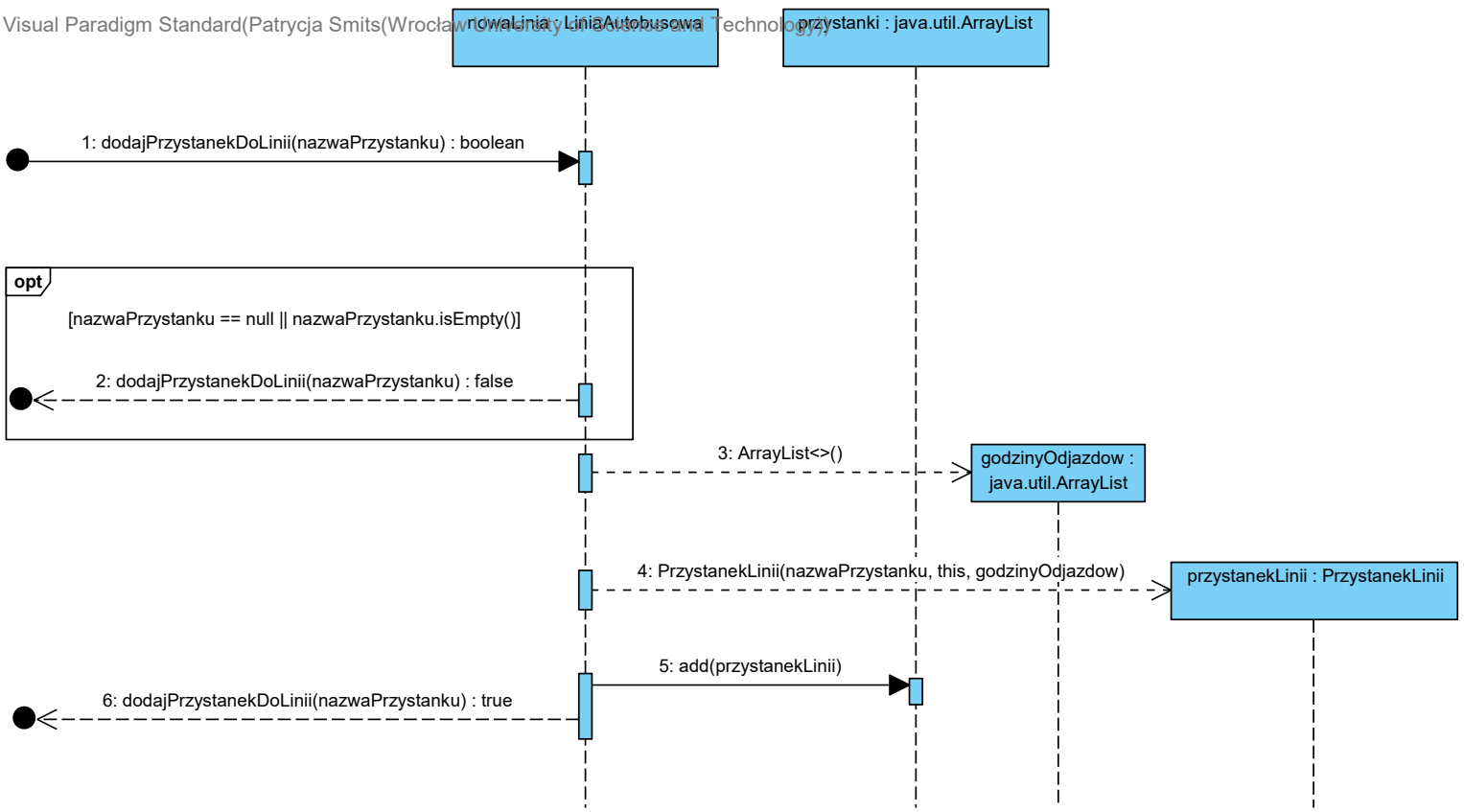
Model

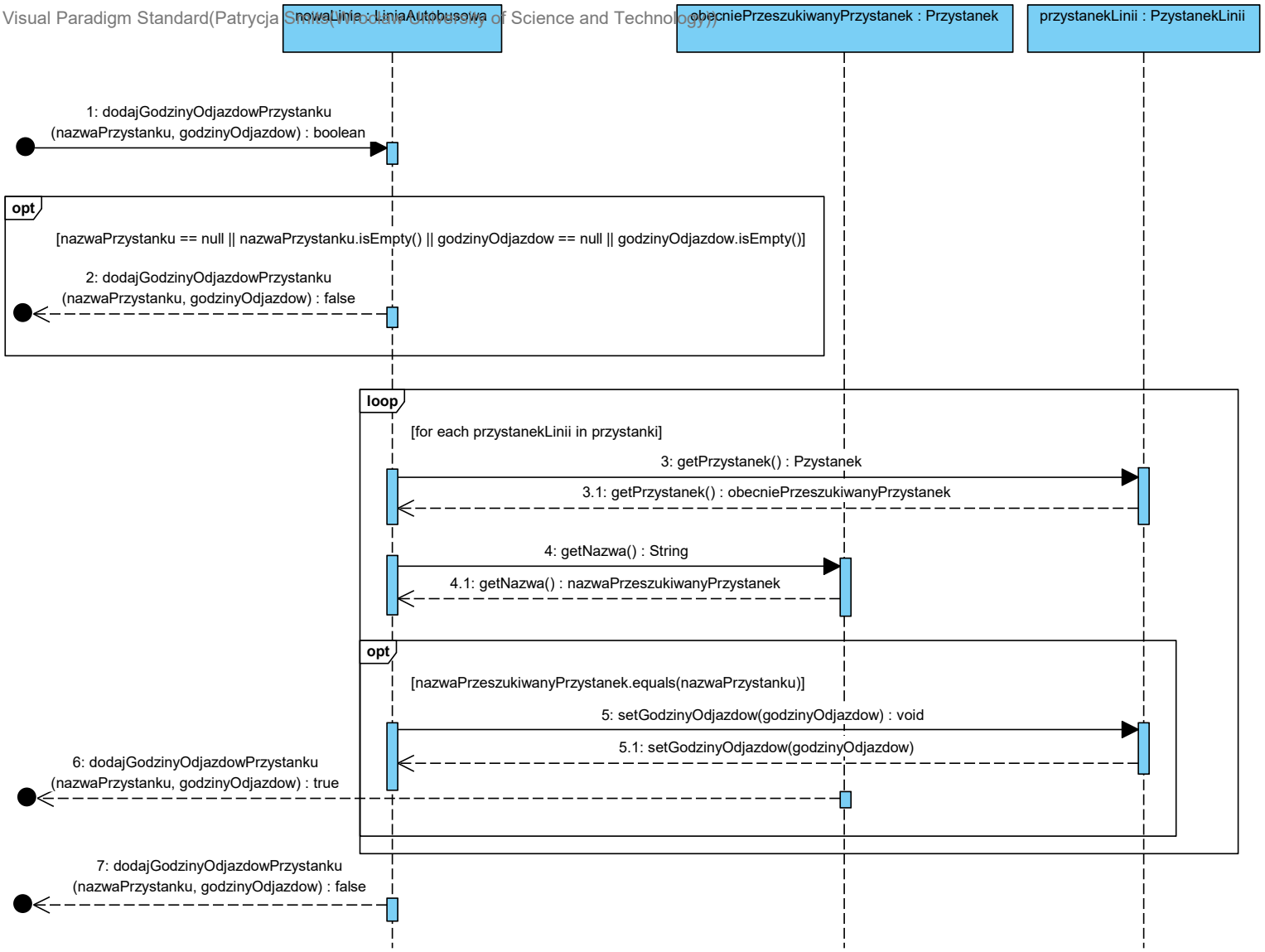


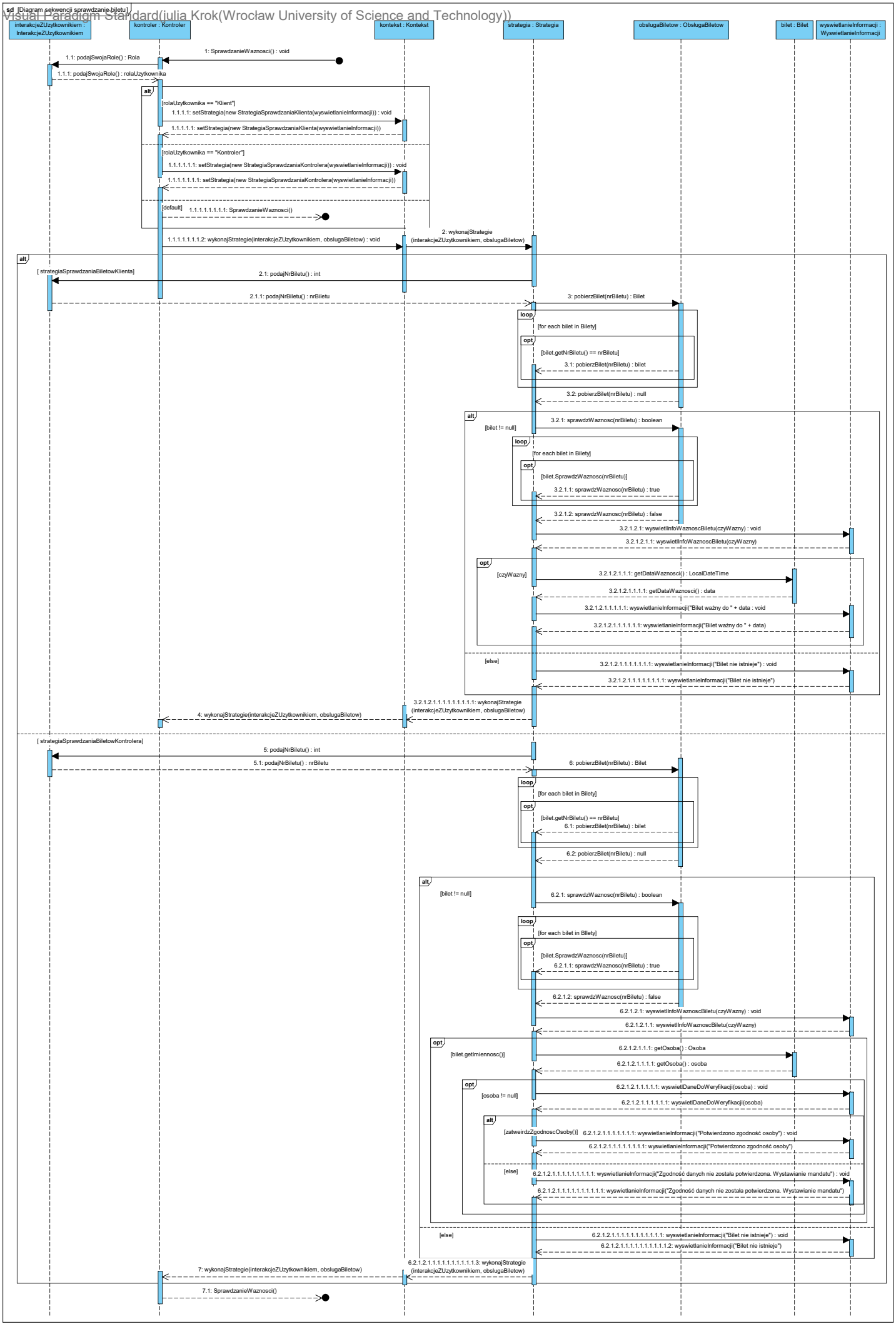







Visual Paradigm Standard(Patrycja Smits(Wrocław University of Science and Technology))
nowaLinia : LiniaAutobusowa
przystanki : java.util.ArrayList
1: dodajPrzystanekDoLinii(nazwaPrzystanku) : boolean
opt
[nazwaPrzystanku == null || nazwaPrzystanku.isEmpty()]
2: dodajPrzystanekDoLinii(nazwaPrzystanku) : false
3: ArrayList<>()
godzinyOdjazdow : java.util.ArrayList
4: PrzystanekLinii(nazwaPrzystanku, this, godzinyOdjazdow)
przystanekLinii : PrzystanekLinii
5: add(przystanekLinii)
6: dodajPrzystanekDoLinii(nazwaPrzystanku) : true





Kontroler.java

```
package Kontroler.Kontroler;

import Model.Model.*;
import Widok.Widok.*;

import java.time.LocalTime;
import java.util.ArrayList;
import java.util.Collection;

public class Kontroler {

   private WyswietlanieInformacji wyswietlanieInformacji;
   private Kontekst kontekst;
   private ObslugaLinii obslugaLinii;
   private ObslugaBiletow obslugaBiletow;
   private InterakcjeZUzytkownikiem interakcjeZUzytkownikiem;

   /**
    * Konstruktor klasy Kontroler.
    *
    * @param obslugaLinii           interfejs obsługi linii
    * @param obslugaBiletow         interfejs obsługi biletów
    * @param wyswietlanieInformacji interfejs do wyświetlania informacji
    * @param interakcjeZUzytkownikiem interfejs do interakcji z
użytkownikiem
    */
   public Kontroler(ObslugaLinii obslugaLinii, ObslugaBiletow
obslugaBiletow, WyswietlanieInformacji wyswietlanieInformacji,
InterakcjeZUzytkownikiem interakcjeZUzytkownikiem) {
      this.obslugaLinii = obslugaLinii;
      this.obslugaBiletow = obslugaBiletow;
      this.wyswietlanieInformacji = wyswietlanieInformacji;
      this.interakcjeZUzytkownikiem = interakcjeZUzytkownikiem;
      this.kontekst = new Kontekst();
   }

   /**
    * Wyświetla menu główne aplikacji i obsługuje wybór użytkownika.
    */
   public void pokazMenu() {
      boolean running = true;
      while (running) {
         wyswietlanieInformacji.wyswietlOpcje();
         int opcja = interakcjeZUzytkownikiem.podajWyborMenu(); //
Wykorzystanie metody do pobrania liczby

         switch (opcja) {
            case 1:    //dodaj linie
               dodawanieLinii();
               break;
            case 2:    //info linia
               informacjeLinia();
               break;
            case 3:    //info przystanek
               informacjePrzystanek();
               break;
            case 4:    //sprawdz waznosc
               sprawdzanieWaznosci();
               break;
```

```
            case 5:    //koniec
               wyswietlanieInformacji.wyswietlInformacje("Zamykanie
aplikacji...");
               running = false;
               break;
            default:
               wyswietlanieInformacji.wyswietlBlad("Nieprawidłowa opcja.
");
         }
      }
   }

   /**
    * Dodaje nową linię autobusową, pojazdy oraz przystanki wraz z
godzinami odjazdów.
    */
   public void dodawanieLinii() {

      int nrLinii = interakcjeZUzytkownikiem.podajNrLinii();

      boolean liniaDodana = obslugaLinii.dodajLinie(nrLinii);

      if (liniaDodana) {
         LiniaAutobusowa nowaLinia = obslugaLinii.znajdzLinie(nrLinii);

         int liczbaPojazdow = interakcjeZUzytkownikiem.podajIle("Ile
pojazdów chcesz dodać do linii? ");
         for (int i = 1; i <= liczbaPojazdow; i++) {
            String nrRejestracyjny =
interakcjeZUzytkownikiem.podajNrRejestracyjny();
            boolean pojazdDodany =
nowaLinia.dodajPojazdDoLinii(nrRejestracyjny);
            if (pojazdDodany) {
               wyswietlanieInformacji.wyswietlPowodzenie("Dodano pojazd");
            } else {
               wyswietlanieInformacji.wyswietlBlad("Pojazd o tym numerze
już istnieje.");
            }
         }

         int liczbaPrzystankow = interakcjeZUzytkownikiem.podajIle("Ile
przystanków chcesz dodać do linii? ");
         for (int i = 1; i <= liczbaPrzystankow; i++) {
            String nazwaPrzystanku =
interakcjeZUzytkownikiem.podajNazwePrzystanku();
            boolean przystanekDodany =
nowaLinia.dodajPrzystanekDoLinii(nazwaPrzystanku);

            if (przystanekDodany) {
               wyswietlanieInformacji.wyswietlPowodzenie("Dodano
przystanek. ");

               // Ustawianie godzin odjazdów dla przystanku
               int liczbaGodzin = interakcjeZUzytkownikiem.podajIle("Ile
godzin odjazdów chcesz dodać dla przystanku \"" + nazwaPrzystanku + "\"?
");
               Collection<LocalTime> godzinyOdjazdow = new ArrayList<>();
               for (int j = 1; j <= liczbaGodzin; j++) {
                  LocalTime godzina =
interakcjeZUzytkownikiem.podajGodzineOdjazdu();
                  godzinyOdjazdow.add(godzina);
```

```
                }
                boolean godzinyDodane =
nowaLinia.dodajGodzinyOdjazdowPrzystanku(nazwaPrzystanku, godzinyOdjazdow);
                if (godzinyDodane) {
                   wyswietlanieInformacji.wyswietlPowodzenie("Dodano godziny
odjazdów.");
                } else {
                   wyswietlanieInformacji.wyswietlBlad("Nie udało się dodać
godzin odjazdów. ");
                }
             } else {
                wyswietlanieInformacji.wyswietlBlad("Przystanek już istnieje
w tej linii.");
             }
          }
          wyswietlanieInformacji.wyswietlPowodzenie("Linia autobusowa
została pomyślnie utworzona i skonfigurowana.");
       } else {
          wyswietlanieInformacji.wyswietlBlad("Linia autobusowa już
istnieje.");
       }
    }

    /**
     * Wyświetla informacje o linii autobusowej.
     */
    public void informacjeLinia() {
       int nrLinii = interakcjeZUzytkownikiem.podajNrLinii();
       LiniaAutobusowa linia = obslugaLinii.znajdzLinie(nrLinii);

       if (linia != null) {
          wyswietlanieInformacji.wyswietlInfoLinia(linia);
       } else {
          wyswietlanieInformacji.wyswietlBlad("Linia o numerze " + nrLinii +
" nie istnieje.");
       }
    }

    /**
     * Wyświetla informacje o przystanku w linii autobusowej.
     */
    public void informacjePrzystanek() {

       int nrLinii = interakcjeZUzytkownikiem.podajNrLinii();
       LiniaAutobusowa linia = obslugaLinii.znajdzLinie(nrLinii);

       if (linia != null) {
          String nazwaPrzystanku =
interakcjeZUzytkownikiem.podajNazwePrzystanku();
          PrzystanekLinii przystanek =
linia.znajdzPrzystanek(nazwaPrzystanku);

          if (przystanek != null) {
             wyswietlanieInformacji.wyswietlInfoPrzystanek(przystanek);
          } else {
             wyswietlanieInformacji.wyswietlBlad("Przystanek \"" +
nazwaPrzystanku + "\" nie istnieje w linii nr " + nrLinii + ".");
          }
       } else {
          wyswietlanieInformacji.wyswietlBlad("Linia o numerze " + nrLinii +
" nie istnieje.");
```

```
        }
    }

    /**
     * Sprawdza ważność biletu na podstawie wybranej strategii.
     */
    public void sprawdzanieWaznosci() {
        Rola rolaUzytkownika = interakcjeZUzytkownikiem.podajSwojaRole();

        switch (rolaUzytkownika) {
            case Klient:
                kontekst.setStrategia(new
StrategiaSprawdzaniaKlienta(wyswietlanieInformacji));
                break;
            case Kontroler_biletow:
                kontekst.setStrategia(new
StrategiaSprawdzaniaKontrolera(wyswietlanieInformacji));
                break;
            default:
                return;
        }

        kontekst.wykonajStrategie(interakcjeZUzytkownikiem, obslugaBiletow);
    }


    /**
     * Główna metoda uruchamiająca program.
     *
     * @param args
     */
    public static void main(String[] args) {
        InterakcjeZUzytkownikiem interakcje = new FasadaInterakcji();
        WyswietlanieInformacji widok = new FasadaFabrykaWidoku();
        ObslugaLinii obslugaLinii = new FasadaLinii();
        ObslugaBiletow obslugaBiletow = new FasadaBiletow();

        obslugaBiletow.dodajPrzykladoweBilety();

        Kontroler kontroler = new Kontroler(obslugaLinii, obslugaBiletow,
widok, interakcje);

        kontroler.pokazMenu();
    }

}
```

InterakcjeZUzytkownkiem,java

```java
package Kontroler.Kontroler;

import Model.Model.Rola;

import java.time.LocalTime;

   public interface InterakcjeZUzytkownikiem {

   int podajNrBiletu();

   boolean zatwierdzZgodnoscOsoby();

   int podajNrLinii();

   String podajNrRejestracyjny();

   String podajNazwePrzystanku();

   LocalTime podajGodzineOdjazdu();

   Rola podajSwojaRole();

   int podajWyborMenu();

   int podajIle(String komunikat);
}
```

FasadaInterakcji.java

```java
package Kontroler.Kontroler;

import Model.Model.Rola;

import java.time.LocalTime;
import java.util.Scanner;

/**
 * Klasa implementująca interfejs {@link InterakcjeZUzytkownikiem}.
 * Odpowiada za interakcje z użytkownikiem i pobieranie od niego danych.
 */
public class FasadaInterakcji implements InterakcjeZUzytkownikiem {

   private final Scanner scanner = new Scanner(System.in);

   /**
    * Pobiera numer biletu od użytkownika.
    *
    * @return numer biletu
    */
   @Override
   public int podajNrBiletu() {
      int nrBiletu = 0;
      boolean poprawneDane = false;

      while (!poprawneDane) {
         System.out.print("Podaj numer biletu: ");
         if (scanner.hasNextInt()) {
            nrBiletu = scanner.nextInt();
            poprawneDane = true;
```

```
            } else {
                System.out.println("Nieprawidłowy numer. Spróbuj ponownie.");
                scanner.next(); // Wyczyść nieprawidłowe dane
            }
        }
        return nrBiletu;
    }

    /**
     * Pobiera od użytkownika informację, czy dane osoby są zgodne z danymi
znajdującymi się na bilecie.
     *
     * @return true, jeśli dane są zgodne; false w przeciwnym razie
     */
    @Override
    public boolean zatwierdzZgodnoscOsoby() {
        System.out.print("Czy dane osoby są zgodne? (tak/nie): ");
        String odpowiedz;

        do {
            odpowiedz = scanner.next().trim().toLowerCase();
            if (odpowiedz.equals("tak")) {
                return true;
            } else if (odpowiedz.equals("nie")) {
                return false;
            } else {
                System.out.println("Nieprawidłowa odpowiedź. Podaj 'tak' lub
'nie'.");
            }
        } while (true);
    }

    /**
     * Pobiera numer linii od użytkownika.
     *
     * @return numer linii
     */
    @Override
    public int podajNrLinii() {
        int nrLinii = 0;
        boolean poprawneDane = false;

        while (!poprawneDane) {
            System.out.print("Podaj numer linii: ");
            if (scanner.hasNextInt()) {
                nrLinii = scanner.nextInt();
                poprawneDane = true;
            } else {
                System.out.println("Nieprawidłowy numer. Spróbuj ponownie.");
                scanner.next();
            }
        }
        return nrLinii;
    }

    /**
     * Pobiera nazwę przystanku od użytkownika.
     *
     * @return nazwa przystanku
     */
    @Override
```

```
    public String podajNazwePrzystanku() {
        System.out.print("Podaj nazwę przystanku: ");
        return scanner.next();
    }

    /**
     * Pobiera numer rejestracyjny pojazdu.
     *
     * @return numer rejestracyjny pojazdu
     */
    @Override
    public String podajNrRejestracyjny() {
        System.out.print("Podaj numer rejestracyjny pojazdu: ");
        return scanner.next();
    }

    /**
     * Pobiera godzinę odjazdu w formacie HH:mm.
     *
     * @return godzina odjazdu jako LocalTime
     */
    @Override
    public LocalTime podajGodzineOdjazdu() {
        LocalTime godzinaOdjazdu = null;
        boolean poprawneDane = false;

        while (!poprawneDane) {
            System.out.print("Podaj godzinę odjazdu (HH:mm): ");
            String input = scanner.next();
            try {
                godzinaOdjazdu = LocalTime.parse(input);
                poprawneDane = true;
            } catch (Exception e) {
                System.out.println("Nieprawidłowy format godziny. Spróbuj
ponownie (HH:mm).");
            }
        }
        return godzinaOdjazdu;
    }

    /**
     * Pobiera rolę użytkownika.
     *
     * @return rola użytkownika jako obiekt typu Rola
     */
    public Rola podajSwojaRole() {
        System.out.println("Wybierz swoją rolę:");
        System.out.println("1 - Klient");
        System.out.println("2 - Kierowca");
        System.out.println("3 - Kontroler biletów");
        System.out.println("4 - Koordynator");

        while (true) {
            String wybor = scanner.next();
            switch (wybor) {
                case "1":
                    return Rola.Klient;
                case "2":
                    return Rola.Kierowca;
                case "3":
                    return Rola.Kontroler_biletow;
```

```
            case "4":
               return Rola.Koordynator;
            default:
               System.out.println("Nieprawidłowy wybór. Wybierz ponownie
(1-4): ");
            }
         }
      }

      /**
       * Pobiera wybór użytkownika z menu.
       *
       * @return numer wybranej opcji
       */
      @Override
      public int podajWyborMenu() {
         int wybor = -1;
         boolean poprawneDane = false;

         while (!poprawneDane) {
            System.out.print("Wybierz opcję z menu: ");
            if (scanner.hasNextInt()) {
               wybor = scanner.nextInt();
               if (wybor >= 1 && wybor <= 5) { // Zakładamy, że mamy 5 opcji w
menu
                  poprawneDane = true;
               } else {
                  System.out.println("Opcja spoza zakresu. Wprowadź liczbę od
1 do 5.");
               }
            } else {
               System.out.println("Nieprawidłowy wybór. Wprowadź liczbę od 1
do 5.");
               scanner.next(); // Wyczyść nieprawidłowe dane
            }
         }
         return wybor;
      }

      /**
       * Pobiera liczbę elementów na podstawie podanego komunikatu.
       *
       * @param komunikat komunikat wyświetlany użytkownikowi
       * @return dodatnia liczba elementów
       */
      @Override
      public int podajIle(String komunikat) {
         int liczba = 0;
         boolean poprawneDane = false;

         while (!poprawneDane) {
            System.out.print(komunikat);
            if (scanner.hasNextInt()) {
               liczba = scanner.nextInt();
               if (liczba > 0) { // Sprawdzamy, czy liczba jest dodatnia
                  poprawneDane = true;
               } else {
                  System.out.println("Liczba musi być większa niż 0. Spróbuj
ponownie.");
               }
            } else {
```

```
            System.out.println("Nieprawidłowy format liczby. Spróbuj
ponownie.");
            scanner.next();
        }
    }
    return liczba;
  }
}
```

WyswietlanieInformacji.java

```
package Widok.Widok;

import Model.Model.*;

public interface WyswietlanieInformacji {

   void wyswietlOpcje();

   void wyswietlInfoLinia(LiniaAutobusowa liniaAutobusowa);

   void wyswietlInfoPrzystanek(PrzystanekLinii przystanekLinii);

   void wyswietlInfoWaznoscBiletu(boolean waznosc);

   void wyswietlDaneDoWeryfikacji(Osoba osoba);

   void wyswietlBlad(String komunikat);

   void wyswietlPowodzenie(String komunikat);

   void wyswietlInformacje(String komunikat);
}
```

FasadaFabrykaWidoku.java

```
package Widok.Widok;

import java.util.*;
import Model.Model.*;

/**
 * Klasa implementująca interfejs WyswietlanieInformacji.
 * Odpowiada za wyświetlanie informacji w aplikacji, takich jak szczegóły
linii,
 * przystanków, biletów oraz komunikatów o błędach lub powodzeniach.
 */
public class FasadaFabrykaWidoku implements WyswietlanieInformacji {

   /**
    * Konstruktor domyślny klasy FasadaFabrykaWidoku.
    */
   public FasadaFabrykaWidoku() {
   }

   /**
    * Wyświetla dostępne opcje menu dla użytkownika.
    */
   @Override
   public void wyswietlOpcje() {
```

```
        System.out.println("1. Dodaj linie autobusowa.");
        System.out.println("2. Wyświetl informacje o linii autobusowej.");
        System.out.println("3. Wyświetl informacje o przystanku.");
        System.out.println("4. Sprawdź ważność biletu.");
        System.out.println("5. Zakończ.");
    }

    /**
     * Wyświetla informacje o linii autobusowej, w tym przystanki i pojazdy.
     *
     * @param liniaAutobusowa Obiekt linii autobusowej, dla której mają
zostać wyświetlone informacje.
     */
    @Override
    public void wyswietlInfoLinia(LiniaAutobusowa liniaAutobusowa) {
        if (liniaAutobusowa != null) {
            System.out.println("Numer linii: " +
liniaAutobusowa.getNrLinii());

            System.out.println("Przystanki na linii:");
            if (liniaAutobusowa.getPrzystanki().isEmpty()) {
                wyswietlBlad("Brak przystanków przypisanych do tej linii.");
            } else {
                liniaAutobusowa.getPrzystanki().forEach(przystanek ->
                        System.out.println("- " +
przystanek.getPrzystanek().getNazwa()));
            }

            System.out.println("Pojazdy przypisane do linii:");
            if (liniaAutobusowa.getPojazdy().isEmpty()) {
                System.out.println("Brak pojazdów przypisanych do tej linii.");
            } else {
                liniaAutobusowa.getPojazdy().forEach(pojazd ->
                        System.out.println("- Numer rejestracyjny: " +
pojazd.getNrRejestracyjny()));
            }
        } else {
            wyswietlBlad("Brak danych o linii autobusowej.");
        }
    }

    /**
     * Wyświetla informacje o przystanku, w tym jego nazwę, numer linii i
godziny odjazdów.
     *
     * @param przystanekLinii Obiekt przystanku z przypisanymi godzinami
odjazdów.
     */
    @Override
    public void wyswietlInfoPrzystanek(PrzystanekLinii przystanekLinii) {
        if (przystanekLinii != null) {
            System.out.println("Przystanek: " +
przystanekLinii.getPrzystanek().getNazwa());
            System.out.println("Numer linii: " +
przystanekLinii.getLinia().getNrLinii());
            System.out.println("Godziny odjazdów:");
            przystanekLinii.getGodzinyOdjazdow().forEach(godzina ->
                    System.out.println("- " + godzina));
        } else {
            wyswietlBlad("Brak danych o przystanku.");
        }
```

```
    }

    /**
     * Wyświetla komunikat o ważności biletu.
     *
     * @param waznosc Wartość boolean określająca, czy bilet jest ważny.
     */
    @Override
    public void wyswietlInfoWaznoscBiletu(boolean waznosc) {
        if (waznosc) {
            System.out.println("Bilet jest ważny. :) ");
        } else {
            System.out.println("Bilet jest nieważny. :( ");
        }
    }

    /**
     * Wyświetla dane osoby do weryfikacji, takie jak imię, nazwisko, PESEL,
numer dokumentu i rolę.
     *
     * @param osoba Obiekt osoby, dla której mają zostać wyświetlone dane.
     */
    @Override
    public void wyswietlDaneDoWeryfikacji(Osoba osoba) {
        if (osoba != null) {
            System.out.println("Imię: " + osoba.getImie());
            System.out.println("Nazwisko: " + osoba.getNazwisko());
            System.out.println("PESEL: " + osoba.getPesel());
            System.out.println("Nr Dokumentu: " + osoba.getNrDokumentu());
            System.out.println("Rola: " + osoba.getRola());
        } else {
            System.out.println("Brak danych o osobie.");
        }
    }

    /**
     * Wyświetla komunikat o błędzie.
     *
     * @param komunikat Treść komunikatu o błędzie.
     */
    @Override
    public void wyswietlBlad(String komunikat) {
        System.out.println("Błąd! " + komunikat);
    }

    /**
     * Wyświetla komunikat o powodzeniu operacji.
     *
     * @param komunikat Treść komunikatu o powodzeniu.
     */
    @Override
    public void wyswietlPowodzenie(String komunikat) {
        System.out.println("Powodzenie! " + komunikat);
    }

    /**
     * Wyświetla ogólny komunikat informacyjny.
     *
     * @param komunikat Treść komunikatu informacyjnego.
     */
    @Override
```

```
    public void wyswietlInformacje(String komunikat) {
        System.out.println(komunikat);
    }
}
```

ObslugaLinii.java

```
package Model.Model;

import java.util.Collection;

public interface ObslugaLinii {

    /**
     * @param nrLinii
     * @return
     */
    boolean dodajLinie(int nrLinii);

    LiniaAutobusowa znajdzLinie(int nrLinii);

}
```

FasadaLinii.java

```
package Model.Model;

import java.util.ArrayList;
import java.util.Collection;

/**
 * Klasa implementująca interfejs {@link ObslugaLinii}.
 * Zarządza kolekcją linii autobusowych, umożliwia ich dodawanie i
wyszukiwanie.
 */
public class FasadaLinii implements ObslugaLinii {

    private Collection<LiniaAutobusowa> LinieAutobusowe = new ArrayList<>();

    /**
     * Dodaje nową linię autobusową, jeśli linia o podanym numerze nie
istnieje.
     *
     * @param nrLinii Numer linii do dodania.
     * @return True, jeśli linia została dodana; false, jeśli linia o takim
numerze już istnieje.
     */
    @Override
    public boolean dodajLinie(int nrLinii) {
        LiniaAutobusowa znalezionaLinia = znajdzLinie(nrLinii);
        if (znalezionaLinia == null) { // Sprawdza, czy linia o podanym
numerze nie istnieje
            LiniaAutobusowa nowaLinia = new LiniaAutobusowa(nrLinii);
            LinieAutobusowe.add(nowaLinia); // Dodaje nową linię
            return true;
        }
        else return false; // Linia już istnieje
    }
```

```
    /**
     * Wyszukuje linię autobusową na podstawie numeru linii.
     *
     * @param nrLinii Numer linii do wyszukania.
     * @return Obiekt {@link LiniaAutobusowa}, jeśli linia istnieje; null,
jeśli nie znaleziono linii.
     */
    @Override
    public LiniaAutobusowa znajdzLinie(int nrLinii) {
        for (LiniaAutobusowa linia : LinieAutobusowe) {
            if (linia.getNrLinii() == nrLinii) {
                return linia;
            }
        }
        return null;
    }

}
```

LiniaAutobusowa.java

```
package Model.Model;

import java.time.LocalTime;
import java.util.*;

/**
 * Klasa reprezentująca linię autobusową, zawierającą przystanki oraz
przypisane pojazdy.
 */
public class LiniaAutobusowa {

    private final int nrLinii;
    private Collection<PrzystanekLinii> przystanki = new ArrayList<>();
    private Collection<Pojazd> pojazdy = new ArrayList<>();

    /**
     * Konstruktor klasy LiniaAutobusowa.
     *
     * @param nrLinii Numer linii autobusowej.
     */
    public LiniaAutobusowa(int nrLinii) {
        this.nrLinii = nrLinii;
    }

    /**
     * Zwraca numer linii autobusowej.
     *
     * @return Numer linii.
     */
    public int getNrLinii() {
        return nrLinii;
    }

    /**
     * Zwraca kolekcję przystanków przypisanych do linii.
     *
     * @return Kolekcja obiektów {@link PrzystanekLinii}.
     */
    public Collection<PrzystanekLinii> getPrzystanki() {
        return przystanki;
```

```
    }

    /**
     * Zwraca kolekcję pojazdów przypisanych do linii.
     *
     * @return Kolekcja obiektów {@link Pojazd}.
     */
    public Collection<Pojazd> getPojazdy() {
        return pojazdy;
    }

    /**
     * Dodaje przystanek do linii autobusowej.
     *
     * @param nazwaPrzystanku Nazwa przystanku do dodania.
     * @return {@code true} jeśli przystanek został dodany pomyślnie, w
przeciwnym razie {@code false}.
     */
    public boolean dodajPrzystanekDoLinii(String nazwaPrzystanku) {
        if (nazwaPrzystanku == null || nazwaPrzystanku.isEmpty()) {
            return false;
        }
        Collection<LocalTime> godzinyOdjazdow = new ArrayList<>();
        PrzystanekLinii przystanekLinii = new
PrzystanekLinii(nazwaPrzystanku, this, godzinyOdjazdow);
        przystanki.add(przystanekLinii);
        return true;
    }

    /**
     * Wyszukuje przystanek o podanej nazwie w linii autobusowej.
     *
     * @param nazwaPrzystanku Nazwa przystanku do wyszukania.
     * @return Obiekt {@link PrzystanekLinii}, jeśli przystanek został
znaleziony, w przeciwnym razie {@code null}.
     */
    public PrzystanekLinii znajdzPrzystanek(String nazwaPrzystanku) {
        if (nazwaPrzystanku == null || nazwaPrzystanku.isEmpty()) {
            return null;
        }
        for (PrzystanekLinii przystanekLinii : przystanki) {
            if
(przystanekLinii.getPrzystanek().getNazwa().equalsIgnoreCase(nazwaPrzystank
u)) {
                return przystanekLinii;
            }
        }
        return null;
    }

    /**
     * Dodaje pojazd do linii autobusowej.
     *
     * @param nrRejestracyjny Numer rejestracyjny pojazdu.
     * @return true jeśli pojazd został dodany pomyślnie, w przeciwnym razie
false.
     */
    public boolean dodajPojazdDoLinii(String nrRejestracyjny) {
        if (nrRejestracyjny == null || nrRejestracyjny.isEmpty()) {
            return false;
        }
```

```
        KreatorPojazdow kreator = new KreatorAutobus();
        Pojazd nowyPojazd = kreator.stworzPojazd(nrRejestracyjny);
        pojazdy.add(nowyPojazd);
        return true;
    }

    /**
     * Dodaje godziny odjazdów dla konkretnego przystanku w linii
autobusowej.
     *
     * @param nazwaPrzystanku Nazwa przystanku, dla którego dodawane są
godziny.
     * @param godzinyOdjazdow Kolekcja godzin odjazdów LocalTime.
     * @return true jeśli godziny zostały dodane pomyślnie, w przeciwnym
razie false.
     */
    public boolean dodajGodzinyOdjazdowPrzystanku(String nazwaPrzystanku,
Collection<LocalTime> godzinyOdjazdow) {
        if (nazwaPrzystanku == null || nazwaPrzystanku.isEmpty() ||
godzinyOdjazdow == null || godzinyOdjazdow.isEmpty()) {
            return false;
        }

        for (PrzystanekLinii przystanekLinii : przystanki) {
            Przystanek obecniePrzeszukiwanyPrzystanek =
przystanekLinii.getPrzystanek();
            String nazwaPrzeszukiwanyPrzystanek =
obecniePrzeszukiwanyPrzystanek.getNazwa();
            if (nazwaPrzeszukiwanyPrzystanek.equals(nazwaPrzystanku)) {
                przystanekLinii.setGgodzinyOdjazdow(godzinyOdjazdow);
                return true;
            }
        }
        return false;
    }
}
```

Przystanek.java

```
package Model.Model;

/**
 * Klasa reprezentująca przystanek.
 */
public class Przystanek {

    /** Nazwa przystanku autobusowego. */
    private String nazwa;

    /**
     * Konstruktor klasy Przystanek.
     *
     * @param nazwa Nazwa przystanku autobusowego.
     */
    public Przystanek(String nazwa) {
        this.nazwa = nazwa;
    }

    /**
     * Zwraca nazwę przystanku.
     *
```

```
     * @return Nazwa przystanku.
     */
    public String getNazwa() {
        return nazwa;
    }

    /**
     * Ustawia nową nazwę przystanku.
     *
     * @param nazwa Nowa nazwa przystanku.
     */
    public void setNazwa(String nazwa) {
        this.nazwa = nazwa;
    }
}
```

PrzystanekLinii.java

```
package Model.Model;

import java.time.LocalTime;
import java.util.ArrayList;
import java.util.Collection;

/**
 * Klasa reprezentująca przystanek przypisany do konkretnej linii
autobusowej.
 * Zawiera informacje o godzinach odjazdów oraz o linii, do której
przystanek należy.
 */
public class PrzystanekLinii extends Przystanek {

    /** Linia autobusowa, do której przystanek jest przypisany. */
    private LiniaAutobusowa linia;

    /** Kolekcja godzin odjazdów dla przystanku. */
    private Collection<LocalTime> godzinyOdjazdow = new ArrayList<>();

    /**
     * Konstruktor klasy PrzystanekLinii.
     *
     * @param nazwaPrzystanku Nazwa przystanku.
     * @param linia Linia autobusowa, do której przystanek jest przypisany.
     * @param godzinyOdjazdow Kolekcja godzin odjazdów dla przystanku.
     */
    public PrzystanekLinii(String nazwaPrzystanku, LiniaAutobusowa linia,
Collection<LocalTime> godzinyOdjazdow) {
        super(nazwaPrzystanku); // Wywołanie konstruktora klasy nadrzędnej
(Przystanek)
        this.linia = linia;
        if (godzinyOdjazdow != null) {
            this.godzinyOdjazdow = godzinyOdjazdow;
        }
    }

    /**
     * Zwraca linię autobusową, do której przystanek jest przypisany.
     *
     * @return Linia autobusowa.
     */
    public LiniaAutobusowa getLinia() {
```

```
        return linia;
    }

    /**
     * Ustawia linię autobusową dla przystanku.
     *
     * @param linia Nowa linia autobusowa.
     */
    public void setLinia(LiniaAutobusowa linia) {
        this.linia = linia;
    }

    /**
     * Zwraca kolekcję godzin odjazdów przypisanych do przystanku.
     *
     * @return Kolekcja godzin odjazdów.
     */
    public Collection<LocalTime> getGodzinyOdjazdow() {
        return godzinyOdjazdow;
    }

    /**
     * Ustawia kolekcję godzin odjazdów dla przystanku.
     *
     * @param godzinyOdjazdow Nowa kolekcja godzin odjazdów.
     */
    public void setGodzinyOdjazdow(Collection<LocalTime> godzinyOdjazdow) {
        if (godzinyOdjazdow != null) {
            this.godzinyOdjazdow = godzinyOdjazdow;
        }
    }

    /**
     * Zwraca obiekt przystanku (this), który dziedziczy po klasie
Przystanek.
     *
     * @return Obiekt przystanku.
     */
    public Przystanek getPrzystanek() {
        return this;
    }
}
```

KreatorPojazdow.java

```
package Model.Model;

/**
 * Klasa KreatorPojazdow odpowiedzialna za tworzenie obiektów {@link
Pojazd}.
 */
public abstract class KreatorPojazdow {
      public abstract Pojazd stworzPojazd(String nrRejestracyjny);
}
```

KreatorAutobus.java

```java
package Model.Model;

public class KreatorAutobus extends KreatorPojazdow {
    /**
     * Tworzy nowy pojazd z podanym numerem rejestracyjnym.
     *
     * @param nrRejestracyjny Numer rejestracyjny pojazdu, który ma zostać
utworzony.
     * @return Obiekt Pojazd z ustawionym numerem rejestracyjnym.
     */
    public Pojazd stworzPojazd(String nrRejestracyjny) {
        Pojazd autobus = new Autobus();
        autobus.setNrRejestracyjny(nrRejestracyjny);
        return autobus;
    }
}
```

Pojazd.java

```java
package Model.Model;

import java.util.Objects;

/**
 * Klasa reprezentująca pojazd przypisany do linii autobusowej.
 * Każdy pojazd jest identyfikowany przez unikalny numer rejestracyjny.
 */
public abstract class Pojazd {

   /** Numer rejestracyjny pojazdu. */
   private String nrRejestracyjny = "";

   /**
    * Zwraca numer rejestracyjny pojazdu.
    *
    * @return Numer rejestracyjny pojazdu.
    */
   public String getNrRejestracyjny() {
      return nrRejestracyjny;
   }

   /**
    * Ustawia numer rejestracyjny pojazdu.
    *
    * @param nrRejestracyjny Nowy numer rejestracyjny pojazdu.
    */
   public void setNrRejestracyjny(String nrRejestracyjny) {
      this.nrRejestracyjny = nrRejestracyjny;
   }
}
```

Autobus.java

```java
package Model.Model;

public class Autobus extends Pojazd {
}
```

Kontekst.java

```
package Kontroler.Kontroler;

import Model.Model.ObslugaBiletow;

public class Kontekst {

   private StrategiaSprawdzaniaBiletow strategia;

   /**
    * Ustawia strategię sprawdzania biletów.
    *
    * @param strategia implementacja interfejsu StrategiaSprawdzaniaBiletow
    */
   public void setStrategia(StrategiaSprawdzaniaBiletow strategia) {
      this.strategia = strategia;
   }

   /**
    * Wykonuje aktualnie ustawioną strategię sprawdzania biletów.
    *
    * @param interakcjeZUzytkownikiem interfejs do interakcji z
użytkownikiem
    * @param obslugaBiletow interfejs do sprawdzania biletów
    */
   public void wykonajStrategie(InterakcjeZUzytkownikiem
interakcjeZUzytkownikiem, ObslugaBiletow obslugaBiletow) {
         strategia.wykonajStrategie(interakcjeZUzytkownikiem,
obslugaBiletow);
   }
}
```

StrategiaSprawdzaniaBieltów.java

```
package Kontroler.Kontroler;

import Model.Model.ObslugaBiletow;

public interface StrategiaSprawdzaniaBiletow {

   public void wykonajStrategie(InterakcjeZUzytkownikiem
interakcjeZUzytkownikiem, ObslugaBiletow obslugaBiletow);
}
```

StrategiaSprawdzaniaKlienta.java

```
package Kontroler.Kontroler;

import Model.Model.*;
import Widok.Widok.*;

import java.time.LocalDateTime;

/**
 * Klasa implementująca interfejs {@link StrategiaSprawdzaniaBiletow}.
 * Klasa implementująca strategię sprawdzania ważności biletu dla klienta.
 */
public class StrategiaSprawdzaniaKlienta implements
StrategiaSprawdzaniaBiletow {
```

```
   private final WyswietlanieInformacji wyswietlanieInformacji;

   /**
    * Konstruktor klasy StrategiaSprawdzaniaKlienta.
    *
    * @param wyswietlanieInformacji interfejs do wyświetlania informacji
    */
   public StrategiaSprawdzaniaKlienta(WyswietlanieInformacji
wyswietlanieInformacji) {
      this.wyswietlanieInformacji = wyswietlanieInformacji;
   }

   /**
    * Wykonuje strategię sprawdzania ważności biletu.
    *
    * @param interakcjeZUzytkownikiem interfejs do interakcji z
użytkownikiem
    * @param obslugaBiletow           interfejs do obsługi biletów
    */
   @Override
   public void wykonajStrategie(InterakcjeZUzytkownikiem
interakcjeZUzytkownikiem, ObslugaBiletow obslugaBiletow) {
      int nrBiletu = interakcjeZUzytkownikiem.podajNrBiletu();
      Bilet bilet = obslugaBiletow.pobierzBilet(nrBiletu);

      if (bilet != null) {
         boolean czyWazny = obslugaBiletow.sprawdzWaznosc(nrBiletu);
         wyswietlanieInformacji.wyswietlInfoWaznoscBiletu(czyWazny);

         if (czyWazny) {
            LocalDateTime data = bilet.getDataWaznosci();
            wyswietlanieInformacji.wyswietlInformacje("Bilet ważny do: " +
data);
         }
      } else {
         wyswietlanieInformacji.wyswietlBlad("Bilet nie istnieje.");
      }
   }
}
```

StrategiaSprawdzaniaKontrolera.java

```
package Kontroler.Kontroler;

import Widok.Widok.*;
import Model.Model.*;

/**
 * Klasa implementująca interfejs {@link StrategiaSprawdzaniaBiletow}.
 * Strategia sprawdzania biletów przeznaczona dla kontrolera.
 * Uwzględnia dokładne sprawdzenie ważności biletu oraz weryfikację danych
w przypadku biletów imiennych.
 */
public class StrategiaSprawdzaniaKontrolera implements
StrategiaSprawdzaniaBiletow {

   private final WyswietlanieInformacji wyswietlanieInformacji;

   /**
    * Konstruktor klasy StrategiaSprawdzaniaKontrolera.
```

```
     *
     * @param wyswietlanieInformacji interfejs do wyświetlania informacji
     */
    public StrategiaSprawdzaniaKontrolera(WyswietlanieInformacji
wyswietlanieInformacji) {
        this.wyswietlanieInformacji = wyswietlanieInformacji;
    }

    /**
     * Wykonuje strategię sprawdzania biletu dla kontrolera.
     *
     * @param interakcjeZUzytkownikiem interfejs do interakcji z
użytkownikiem
     * @param obslugaBiletow           interfejs do obsługi biletów
     */
    @Override
    public void wykonajStrategie(InterakcjeZUzytkownikiem
interakcjeZUzytkownikiem, ObslugaBiletow obslugaBiletow) {
        int nrBiletu = interakcjeZUzytkownikiem.podajNrBiletu();
        Bilet bilet = obslugaBiletow.pobierzBilet(nrBiletu);

        if (bilet != null) {
            boolean czyWazny = obslugaBiletow.sprawdzWaznosc(nrBiletu);
            wyswietlanieInformacji.wyswietlInfoWaznoscBiletu(czyWazny);

            if (bilet.getImiennosc()) {

                Osoba osoba = bilet.getOsoba();
                if (osoba != null) {
                    wyswietlanieInformacji.wyswietlDaneDoWeryfikacji(osoba);
                    if (interakcjeZUzytkownikiem.zatwierdzZgodnoscOsoby()) {
                        wyswietlanieInformacji.wyswietlInformacje("Zgodność
danych została potwierdzona.");
                    } else {
                        wyswietlanieInformacji.wyswietlInformacje("Zgodność
danych nie została potwierdzona. Bilet jest nieważny. Wystawianie
mandatu.");
                    }
                }
            }
        } else {
            wyswietlanieInformacji.wyswietlBlad("Bilet nie istnieje. ");
        }
    }
}
```

ObslugaBiletow.java

```
package Model.Model;

import Model.Model.Bilet;

public interface ObslugaBiletow {
    boolean sprawdzWaznosc(int nrBiletu);
    Bilet pobierzBilet(int nrBiletu);
    void dodajBilet(Bilet bilet);
    void dodajPrzykladoweBilety();
}
```

FasadaBiletow.java

```java
package Model.Model;

import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.Collection;

/**
 * Klasa implementująca interfejs {@link ObslugaBiletow}.
 * Zarządza kolekcją biletów oraz umożliwia sprawdzanie ich ważności i pobieranie.
 */
public class FasadaBiletow implements ObslugaBiletow {

   private Collection<Bilet> Bilety = new ArrayList<>(); // Kolekcja przechowująca bilety

   /**
    * Sprawdza ważność biletu na podstawie jego numeru.
    *
    * @param nrBiletu Numer biletu do sprawdzenia.
    * @return True, jeśli bilet jest ważny; false w przeciwnym przypadku.
    */
   public boolean sprawdzWaznosc(int nrBiletu) {
      for (Bilet bilet : Bilety) {
         if (bilet.SprawdzWaznosc(nrBiletu)) {
            return true;
         }
      }
      return false;
   }

   /**
    * Pobiera bilet na podstawie jego numeru.
    *
    * @param nrBiletu Numer biletu do pobrania.
    * @return Obiekt {@link Bilet}, jeśli istnieje; null, jeśli bilet o podanym numerze nie istnieje.
    */
   @Override
   public Bilet pobierzBilet(int nrBiletu) {
      for (Bilet bilet : Bilety) {
         if (bilet.getNrBiletu() == nrBiletu) {
            return bilet;
         }
      }
      return null;
   }

   /**
    * Dodaje nowy bilet do kolekcji.
    *
    * @param bilet Obiekt {@link Bilet}, który ma zostać dodany.
    */
   public void dodajBilet(Bilet bilet) {
      if (bilet != null) {
         Bilety.add(bilet);
      }
   }

   /**
```

```
    * Dodaje przykładowe bilety do kolekcji na potrzeby testowania.
    */
   public void dodajPrzykladoweBilety() {

      // Przykładowe bilety
      Bilet bilet1 = new Bilet(1, true, false, new Osoba("Jan", "Kowalski",
123456789, "AB123456", Rola.Klient), LocalDateTime.now().plusDays(5));
      Bilet bilet2 = new Bilet(2, true, true, new Osoba("Anna", "Nowak",
987654321, "CD987654", Rola.Klient), LocalDateTime.now().plusDays(3));
      Bilet bilet3 = new Bilet(3, false, false, null,
LocalDateTime.now().minusDays(1)); // Nieważny bilet
      Bilet bilet4 = new Bilet(4, false, false, null,
LocalDateTime.now().plusDays(10));

      // Dodanie biletów do kolekcji
      dodajBilet(bilet1);
      dodajBilet(bilet2);
      dodajBilet(bilet3);
      dodajBilet(bilet4);

   }
}
```

Bilet.java

```
package Model.Model;

import java.time.LocalDateTime;

/**
 * Klasa reprezentująca bilet autobusowy.
 * Zawiera informacje o numerze biletu, jego imienności, uldze, przypisanej
osobie oraz dacie ważności.
 */
public class Bilet {

   private int NrBiletu;                // Numer biletu
   private boolean Imiennosc;           // Czy bilet jest imienny
   private boolean Ulga;                // Czy bilet ma ulgę
   private Osoba Osoba;                 // Osoba przypisana do biletu
   private LocalDateTime DataWaznosci;  // Data ważności biletu

   /**
    * Konstruktor klasy Bilet.
    *
    * @param nrBiletu      Numer biletu.
    * @param imiennosc     Czy bilet jest imienny.
    * @param ulga          Czy bilet ma ulgę.
    * @param osoba         Osoba przypisana do biletu.
    * @param dataWaznosci  Data i czas ważności biletu.
    */
   public Bilet(int nrBiletu, boolean imiennosc, boolean ulga, Osoba osoba,
LocalDateTime dataWaznosci) {
      this.NrBiletu = nrBiletu;
      this.Imiennosc = imiennosc;
      this.Ulga = ulga;
      this.Osoba = osoba;
      this.DataWaznosci = dataWaznosci;
   }

   /**
    * Zwraca numer biletu.
```

```java
     *
     * @return Numer biletu.
     */
    public int getNrBiletu() {
        return NrBiletu;
    }

    /**
     * Ustawia numer biletu.
     *
     * @param nrBiletu Nowy numer biletu.
     */
    public void setNrBiletu(int nrBiletu) {
        this.NrBiletu = nrBiletu;
    }

    /**
     * Sprawdza, czy bilet jest imienny.
     *
     * @return True, jeśli bilet jest imienny, false w przeciwnym przypadku.
     */
    public boolean getImiennosc() {
        return Imiennosc;
    }

    /**
     * Ustawia status imienności biletu.
     *
     * @param imiennosc True, jeśli bilet jest imienny.
     */
    public void setImiennosc(boolean imiennosc) {
        this.Imiennosc = imiennosc;
    }

    /**
     * Sprawdza, czy bilet posiada ulgę.
     *
     * @return True, jeśli bilet posiada ulgę, false w przeciwnym przypadku.
     */
    public boolean getUlga() {
        return Ulga;
    }

    /**
     * Ustawia status ulgi dla biletu.
     *
     * @param ulga True, jeśli bilet ma ulgę.
     */
    public void setUlga(boolean ulga) {
        this.Ulga = ulga;
    }

    /**
     * Zwraca datę ważności biletu.
     *
     * @return Data i czas ważności biletu.
     */
    public LocalDateTime getDataWaznosci() {
        return DataWaznosci;
    }
```

```
    /**
     * Ustawia datę ważności biletu.
     *
     * @param dataWaznosci Nowa data i czas ważności biletu.
     */
    public void setDataWaznosci(LocalDateTime dataWaznosci) {
        this.DataWaznosci = dataWaznosci;
    }

    /**
     * Zwraca osobę przypisaną do biletu.
     *
     * @return Osoba przypisana do biletu.
     */
    public Osoba getOsoba() {
        return Osoba;
    }

    /**
     * Przypisuje osobę do biletu.
     *
     * @param osoba Nowa osoba przypisana do biletu.
     */
    public void setOsoba(Osoba osoba) {
        this.Osoba = osoba;
    }

    /**
     * Sprawdza ważność biletu.
     *
     * @param nrBiletu Numer biletu do sprawdzenia.
     * @return True, jeśli numer biletu jest zgodny i data ważności nie
minęła; false w przeciwnym przypadku.
     */
    public boolean SprawdzWaznosc(int nrBiletu) {
        return this.NrBiletu == nrBiletu && DataWaznosci != null &&
LocalDateTime.now().isBefore(DataWaznosci);
    }
}
```

Osoba.java

```
package Model.Model;

/**
 * Klasa reprezentująca osobę w systemie.
 * Zawiera informacje takie jak imię, nazwisko, numer PESEL, numer
dokumentu i rolę użytkownika.
 */
public class Osoba {

    private String Imie = "";
    private String Nazwisko = "";
    private int Pesel;
    private String NrDokumentu = "";
    private Rola Rola = Model.Model.Rola.Klient;

    /**
     * Konstruktor klasy Osoba.
     *
     * @param imie         Imię osoby.
```

```
     * @param nazwisko    Nazwisko osoby.
     * @param pesel       Numer PESEL osoby.
     * @param nrDokumentu Numer dokumentu tożsamości osoby.
     * @param rola        Rola przypisana osobie {@link Rola}.
     */
    public Osoba(String imie, String nazwisko, int pesel, String
nrDokumentu, Rola rola) {
        this.Imie = imie;
        this.Nazwisko = nazwisko;
        this.Pesel = pesel;
        this.NrDokumentu = nrDokumentu;
        this.Rola = rola;
    }

    /**
     * Zwraca imię osoby.
     *
     * @return Imię osoby.
     */
    public String getImie() {
        return Imie;
    }

    /**
     * Ustawia imię osoby.
     *
     * @param imie Nowe imię osoby.
     */
    public void setImie(String imie) {
        this.Imie = imie;
    }

    /**
     * Zwraca nazwisko osoby.
     *
     * @return Nazwisko osoby.
     */
    public String getNazwisko() {
        return Nazwisko;
    }

    /**
     * Ustawia nazwisko osoby.
     *
     * @param nazwisko Nowe nazwisko osoby.
     */
    public void setNazwisko(String nazwisko) {
        this.Nazwisko = nazwisko;
    }

    /**
     * Zwraca numer PESEL osoby.
     *
     * @return Numer PESEL osoby.
     */
    public int getPesel() {
        return Pesel;
    }

    /**
     * Ustawia numer PESEL osoby.
```

```
     *
     * @param pesel Nowy numer PESEL osoby.
     */
    public void setPesel(int pesel) {
        this.Pesel = pesel;
    }

    /**
     * Zwraca numer dokumentu tożsamości osoby.
     *
     * @return Numer dokumentu tożsamości.
     */
    public String getNrDokumentu() {
        return NrDokumentu;
    }

    /**
     * Ustawia numer dokumentu tożsamości osoby.
     *
     * @param nrDokumentu Nowy numer dokumentu tożsamości.
     */
    public void setNrDokumentu(String nrDokumentu) {
        this.NrDokumentu = nrDokumentu;
    }

    /**
     * Zwraca rolę przypisaną osobie.
     *
     * @return Rola osoby {@link Rola}.
     */
    public Rola getRola() {
        return Rola;
    }

    /**
     * Ustawia rolę osoby.
     *
     * @param rola Nowa rola osoby {@link Rola}.
     */
    public void setRola(Rola rola) {
        this.Rola = rola;
    }
}
```

Rola.java

```
package Model.Model;

public enum Rola {
    Klient,
    Kierowca,
    Kontroler_biletow,
    Koordynator
}
```