ETAP 3

Modelowanie obiektowej 3-warstwowej architektury

Diagramy komponentów i klas

Zintegrowany System Sprzedaży Biletów dla Sieci Multikin

Autorzy projektu:

Oleksandr Radionenko
nr albumu: 274003

Yaroslav Perepilka
nr albumu: 282279

Politechnika Wrocławska
Wydział Informatyki i Telekomunikacji
Kierunek: Informatyka Techniczna

Przedmiot: Inżynieria Oprogramowania

Wrocław, 2 grudnia 2025

Spis treści

1 Wprowadzenie

1.1 Nazwa tworzonego oprogramowania

Zintegrowany System Sprzedaży Biletów dla Sieci Multikin – kompleksowe rozwiązanie informatyczne służące do zarządzania ofertą filmową, seansami oraz rezerwacją biletów w ramach sieci kin.

1.2 Temat etapu

Modelowanie obiektowej 3-warstwowej architektury – budowa warstwowego modelu architektury oprogramowania w postaci diagramu komponentów oraz obiektowego modelu struktury komponentów w postaci diagramów klas z zastosowaniem wzorców projektowych zgodnie z zasadami SOLID.

2 Zadanie 1 – Diagram komponentów

2.1 Opis architektury

System został zaprojektowany w oparciu o architekturę 3-warstwową typu MVC (Model-View-Controller), która składa się z następujących warstw:

- **Widok2 (View)** – warstwa prezentacji, odpowiedzialna za interfejs użytkownika i interakcję z aktorem systemu (Administrator Sieci).
- **Kontroler2 (Controller)** – warstwa kontroli, zarządzająca logiką biznesową, przepływem sterowania między warstwami oraz realizacją przypadków użycia.
- **Model2 (Model)** – warstwa encji, modelująca dane biznesowe systemu oraz zapewniająca operacje ich przetwarzania i dostępu do magazynu danych.

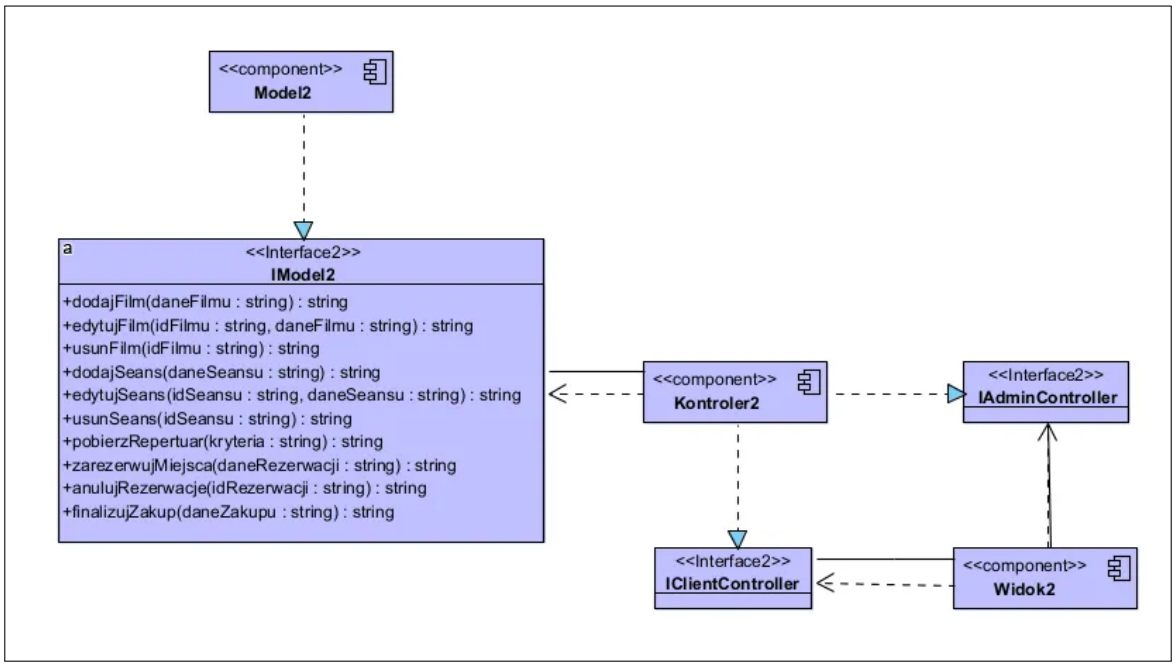
2.2 Połączenia między warstwami

Warstwy komunikują się ze sobą poprzez interfejsy, zgodnie z zasadami architektury MVC:

- Warstwa **Widok2** używa interfejsu `IClientController` udostępnianego przez warstwę **Kontroler2** w celu inicjowania operacji biznesowych.
- Warstwa **Kontroler2** używa interfejsu `IModel2` udostępnianego przez warstwę **Model2** w celu wykonywania operacji na danych.
- Warstwa **Kontroler2** udostępnia interfejs `IAdminController` dla warstwy **Widok2**, umożliwiając administratorowi wykonywanie operacji zarządzania systemem.

Wszystkie połączenia są realizowane pośrednio poprzez interfejsy w postaci tzw. „lizaka i łapki", co zapewnia luźne powiązanie między warstwami i zgodność z zasadami SOLID.

2.3 Diagram komponentów



Rysunek 1: Diagram komponentów – architektura 3-warstwowa MVC

3 Zadanie 2 – Diagram klas warstwy Controller

3.1 Opis struktury warstwy kontroli

Warstwa kontroli (Controller) odpowiada za zarządzanie logiką biznesową systemu oraz realizację przypadków użycia. Została zaprojektowana z zastosowaniem następujących wzorców projektowych:

3.1.1 Zastosowane wzorce projektowe

1. **Fasada** – klasa `ClientController` i `AdminController` działają jako fasady, zapewniając uproszczony interfejs dostępu do funkcjonalności warstwy kontroli. Ukrywają złożoność wewnętrznej struktury przed innymi warstwami.

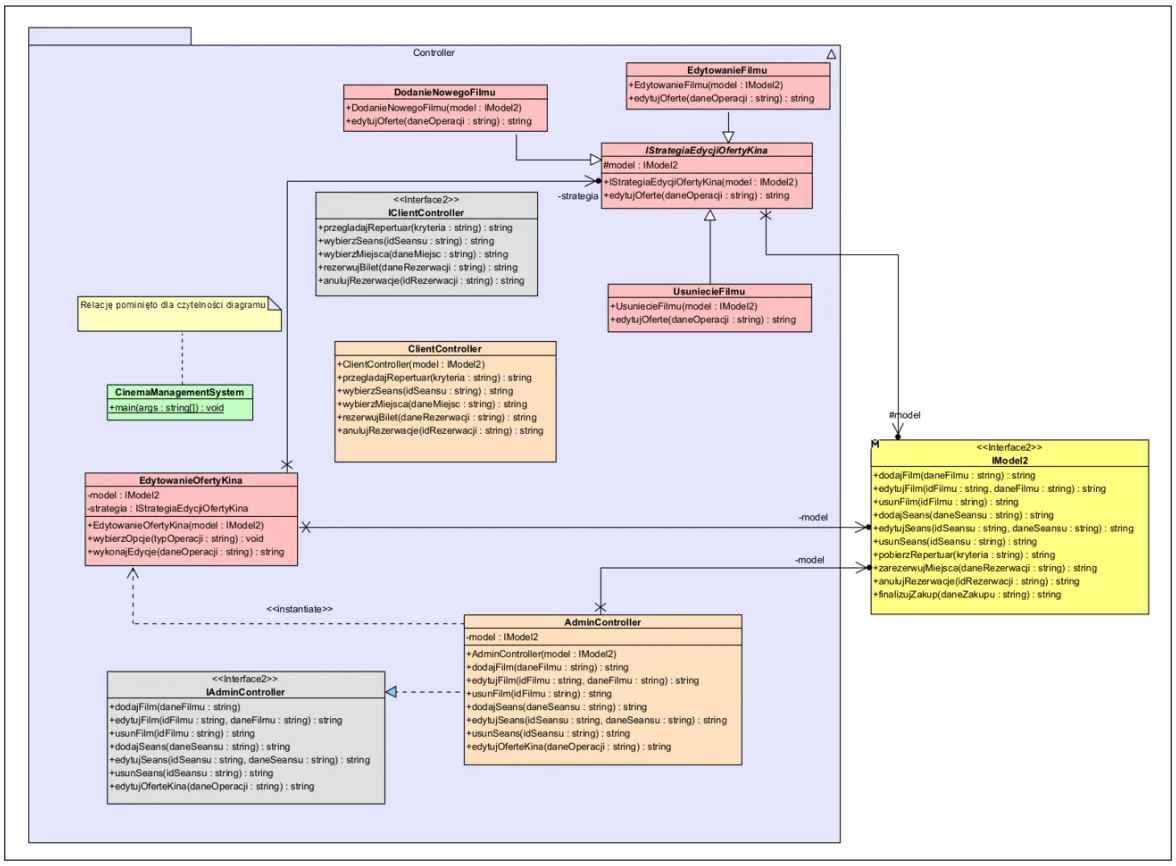
2. **Strategia** – wzorzec zastosowany w klasie `IStrategiaEdycjiOfertyKina`, która definiuje interfejs dla różnych strategii edycji oferty kina. Konkretne strategie (`DodanieNowegoFilmu`, `EdytowanieFilmu`, `UsuniecieFilmu`) implementują różne sposoby modyfikacji oferty filmowej.

3. **Metoda szablonowa** – klasa abstrakcyjna `EdytowanieOfertyKina` definiuje szablon algorytmu edycji oferty. Szczegółowe kroki są implementowane przez klasy pochodne, co pozwala na elastyczne rozszerzanie funkcjonalności.

3.1.2 Główne klasy i ich odpowiedzialności

- **CinemaManagementSystem** – główna klasa systemu zawierająca operację `main()`, odpowiedzialna za inicjalizację i uruchomienie aplikacji.
- **ClientController** – fasada dla operacji klienckich, realizująca interfejs `IClientController`.
- **AdminController** – fasada dla operacji administracyjnych, realizująca interfejs `IAdminController`, zarządzająca edycją oferty kin, dodawaniem, edycją i usuwaniem filmów oraz seansów.
- **EdytowanieOfertyKina** – klasa abstrakcyjna implementująca metodę szablonową dla procesu edycji oferty kina.
- **IStrategiaEdycjiOfertyKina** – interfejs strategii edycji oferty, implementowany przez konkretne strategie.

3.2 Diagram klas warstwy Controller



Rysunek 2: Diagram klas warstwy Controller

4 Zadanie 3 – Diagram klas warstwy Model

4.1 Opis struktury warstwy encji

Warstwa encji (Model) odpowiada za modelowanie danych biznesowych systemu oraz zapewnienie operacji ich przetwarzania. Została zaprojektowana z zastosowaniem następujących wzorców projektowych:

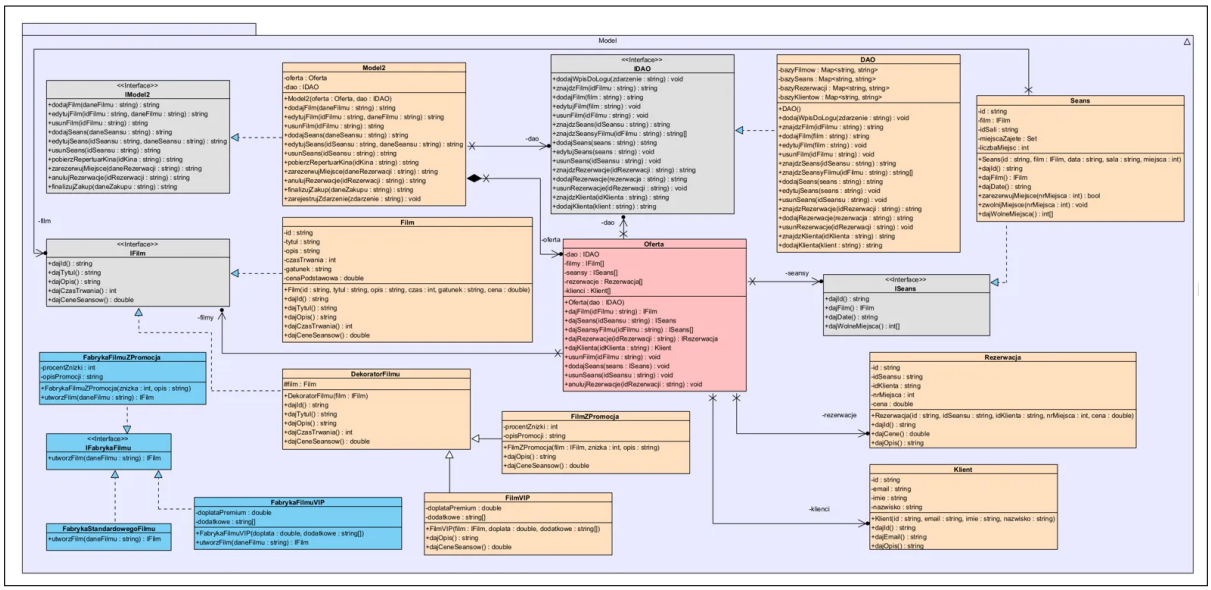
4.1.1 Zastosowane wzorce projektowe

1. **Fasada (Facade)** – klasa `Model2` działa jako fasada dla całej warstwy modelu, udostępniając interfejs `IModel2` i zarządzając dostępem do operacji na danych.
2. **Adapter (DAO – Data Access Object)** – klasa `IDAO` wraz z konkretną implementacją `DAO` stanowi adapter do warstwy zasobów (bazy danych), tłumacząc żądania modelu na operacje CRUD.
3. **Dekorator (Decorator)** – wzorzec zastosowany w hierarchii klas `Film`, `DekoratorFilmu` i `FilmZPromocja`, pozwalający na dynamiczne dodawanie funkcjonalności (np. promocji) do obiektów filmów.
4. **Fabryka abstrakcyjna (Abstract Factory)** – klasy `FabrykaFilmu`, `FabrykaFilmuZPromocja`, `FabrykaStandardowegoFilmu` oraz `FabrykaFilmuVIP` implementują wzorzec fabryki do tworzenia różnych typów filmów.
5. **Kompozyt (Composite)** – struktura `Oferta` zawierająca kolekcje obiektów `Film` i `Seans` realizuje wzorzec kompozytu, umożliwiając zarządzanie hierarchią obiektów.

4.1.2 Główne klasy i ich odpowiedzialności

- **Model2** – centralna klasa fasadowa warstwy, zarządzająca wszystkimi operacjami na danych i koordynująca pracę pozostałych klas.
- **IDAO i DAO** – interfejs i implementacja wzorca DAO, zapewniające dostęp do warstwy zasobów (bazy danych).
- **IFilm i Film** – interfejs i klasa bazowa reprezentująca film w systemie.
- **ISeans i Seans** – interfejs i klasa reprezentująca seans filmowy.
- **Oferta** – klasa agregująca filmy i seanse, reprezentująca ofertę kina.
- **Rezerwacja** – klasa reprezentująca rezerwację biletu przez klienta.
- **Klient** – klasa reprezentująca klienta systemu.
- **FabrykaFilmu** – hierarchia klas fabrycznych do tworzenia różnych typów filmów (standardowych, VIP, z promocją).

4.2 Diagram klas warstwy Model



Rysunek 3: Diagram klas warstwy Model

5 Zadanie 4 – Kod źródłowy klas

W niniejszej sekcji przedstawiono kompletny kod źródłowy wszystkich klas wygenerowanych automatycznie przez narzędzie Visual Paradigm (Instant Generator) na podstawie diagramów klas z Zadań 2 i 3. Kod został zorganizowany alfabetycznie według w pełni kwalifikowanych nazw klas z uwzględnieniem struktury pakietów.

5.1 Pakiet Controller

5.1.1 AdminController.java

```java
package controller;

import model.IModel2;

/**
 * Kontroler administracyjny - fasada dla operacji Administratora Sieci.
 * Realizuje interfejs IAdminController.
 */
public class AdminController implements IAdminController {
    private IModel2 model;

    /**
     * Konstruktor kontrolera administracyjnego.
     * @param model fasada warstwy Model
     */
    public AdminController(IModel2 model) {
        this.model = model;
    }

    /**
     * Dodanie nowego filmu do systemu.
     * @param daneFilmu dane filmu w formacie tekstowym
     * @return identyfikator dodanego filmu
     */
    public String dodajFilm(String daneFilmu) {
        throw new UnsupportedOperationException();
    }

    /**
     * Edycja istniejacego filmu.
     * @param idFilmu identyfikator filmu
     * @param daneFilmu nowe dane filmu
     * @return potwierdzenie edycji
     */
    public String edytujFilm(String idFilmu, String daneFilmu) {
        throw new UnsupportedOperationException();
    }
```

```
38
39      /**
40       * Usuniecie filmu z systemu.
41       * @param idFilmu identyfikator filmu
42       * @return potwierdzenie usuniecia
43       */
44      public String usunFilm(String idFilmu) {
45          throw new UnsupportedOperationException();
46      }
47
48      /**
49       * Dodanie nowego seansu do systemu.
50       * @param daneSeansu dane seansu w formacie tekstowym
51       * @return identyfikator dodanego seansu
52       */
53      public String dodajSeans(String daneSeansu) {
54          throw new UnsupportedOperationException();
55      }
56
57      /**
58       * Edycja istniejacego seansu.
59       * @param idSeansu identyfikator seansu
60       * @param daneSeansu nowe dane seansu
61       * @return potwierdzenie edycji
62       */
63      public String edytujSeans(String idSeansu, String daneSeansu)
            {
64          throw new UnsupportedOperationException();
65      }
66
67      /**
68       * Usuniecie seansu z systemu.
69       * @param idSeansu identyfikator seansu
70       * @return potwierdzenie usuniecia
71       */
72      public String usunSeans(String idSeansu) {
73          throw new UnsupportedOperationException();
74      }
75
76      /**
77       * Edytowanie oferty kina w systemie.
78       * @param daneOperacji dane operacji (typ operacji i
             parametry)
79       * @return wynik operacji
80       */
81      public String edytujOferteKina(String daneOperacji) {
82          throw new UnsupportedOperationException();
83      }
84  }
```

5.1.2 CinemaManagementSystem.java

```java
package controller;

/**
 * Glowna klasa systemu zarzadzania kinem.
 * Zawiera metode main() uruchamiajaca aplikacje.
 */
public class CinemaManagementSystem {

    /**
     * Glowna metoda programu.
     * @param args argumenty wiersza polecen
     */
    public static void main(String[] args) {
        throw new UnsupportedOperationException();
    }
}
```

5.1.3 ClientController.java

```java
package controller;

import model.IModel2;

/**
 * Kontroler kliencki - fasada dla operacji Klienta.
 * Realizuje interfejs IClientController.
 */
public class ClientController implements IClientController {
    private IModel2 model;

    /**
     * Konstruktor kontrolera klienckiego.
     * @param model fasada warstwy Model
     */
    public ClientController(IModel2 model) {
        this.model = model;
    }

    /**
     * Przegladanie repertuaru kin.
     * @param kryteria kryteria wyszukiwania
     * @return lista seansow spelniajacych kryteria
     */
    public String przegladajRepertuar(String kryteria) {
        throw new UnsupportedOperationException();
    }
```

```
28
29      /**
30       * Wybor konkretnego seansu.
31       * @param idSeansu identyfikator seansu
32       * @return szczegoly wybranego seansu
33       */
34      public String wybierzSeans(String idSeansu) {
35          throw new UnsupportedOperationException();
36      }
37
38      /**
39       * Wybor miejsc na seansie.
40       * @param daneMiejsc dane wybranych miejsc
41       * @return potwierdzenie wyboru miejsc
42       */
43      public String wybierzMiejsca(String daneMiejsc) {
44          throw new UnsupportedOperationException();
45      }
46
47      /**
48       * Rezerwacja biletu.
49       * @param daneRezerwacji dane rezerwacji
50       * @return potwierdzenie rezerwacji
51       */
52      public String rezerwujBilet(String daneRezerwacji) {
53          throw new UnsupportedOperationException();
54      }
55
56      /**
57       * Anulowanie rezerwacji.
58       * @param idRezerwacji identyfikator rezerwacji
59       * @return potwierdzenie anulowania
60       */
61      public String anulujRezerwacje(String idRezerwacji) {
62          throw new UnsupportedOperationException();
63      }
64  }
```

5.1.4 DodanieNowegoFilmu.java

```
1   package controller;
2
3   import model.IModel2;
4
5   /**
6    * Strategia dodawania nowego filmu do oferty kina.
7    * Implementuje IStrategiaEdycjiOfertyKina.
8    */
```

```
9  public class DodanieNowegoFilmu extends
       IStrategiaEdycjiOfertyKina {
10
11     /**
12      * Konstruktor strategii dodawania filmu.
13      * @param model fasada warstwy Model
14      */
15     public DodanieNowegoFilmu(IModel2 model) {
16         super(model);
17     }
18
19     /**
20      * Wykonanie edycji oferty - dodanie nowego filmu.
21      * @param daneOperacji dane nowego filmu
22      * @return wynik operacji dodawania
23      */
24     @Override
25     public String edytujOferte(String daneOperacji) {
26         throw new UnsupportedOperationException();
27     }
28 }
```

5.1.5 EdytowanieFilmu.java

```
1  package controller;
2
3  import model.IModel2;
4
5  /**
6   * Strategia edycji istniejacego filmu w ofercie kina.
7   * Implementuje IStrategiaEdycjiOfertyKina.
8   */
9  public class EdytowanieFilmu extends IStrategiaEdycjiOfertyKina {
10
11     /**
12      * Konstruktor strategii edycji filmu.
13      * @param model fasada warstwy Model
14      */
15     public EdytowanieFilmu(IModel2 model) {
16         super(model);
17     }
18
19     /**
20      * Wykonanie edycji oferty - modyfikacja filmu.
21      * @param daneOperacji dane zawierajace ID filmu i nowe dane
22      * @return wynik operacji edycji
23      */
24     @Override
```

```
25      public String edytujOferte(String daneOperacji) {
26          throw new UnsupportedOperationException();
27      }
28  }
```

5.1.6 EdytowanieOfertyKina.java

```
1   package controller;
2
3   import model.IModel2;
4
5   /**
6    * Klasa zarzadzajaca edycja oferty kina.
7    * Implementuje wzorzec Metoda Szablonowa.
8    */
9   public class EdytowanieOfertyKina {
10      private IModel2 model;
11      private IStrategiaEdycjiOfertyKina strategia;
12
13      /**
14       * Konstruktor klasy edytowania oferty.
15       * @param model fasada warstwy Model
16       */
17      public EdytowanieOfertyKina(IModel2 model) {
18          this.model = model;
19      }
20
21      /**
22       * Wybor strategii edycji oferty (dodanie, edycja, usuniecie)
             .
23       * @param typOperacji typ operacji do wykonania
24       */
25      public void wybierzOpcje(String typOperacji) {
26          throw new UnsupportedOperationException();
27      }
28
29      /**
30       * Wykonanie edycji oferty przy uzyciu wybranej strategii.
31       * @param daneOperacji dane niezbedne do wykonania operacji
32       * @return wynik wykonania operacji
33       */
34      public String wykonajEdycje(String daneOperacji) {
35          throw new UnsupportedOperationException();
36      }
37  }
```

5.1.7 IAdminController.java

```java
package controller;

/**
 * Interfejs fasady kontrolera dla Administratora Sieci.
 */
public interface IAdminController {

    /**
     * Dodanie nowego filmu.
     * @param daneFilmu dane filmu
     * @return identyfikator dodanego filmu
     */
    String dodajFilm(String daneFilmu);

    /**
     * Edycja filmu.
     * @param idFilmu identyfikator filmu
     * @param daneFilmu nowe dane filmu
     * @return potwierdzenie edycji
     */
    String edytujFilm(String idFilmu, String daneFilmu);

    /**
     * Usuniecie filmu.
     * @param idFilmu identyfikator filmu
     * @return potwierdzenie usuniecia
     */
    String usunFilm(String idFilmu);

    /**
     * Dodanie nowego seansu.
     * @param daneSeansu dane seansu
     * @return identyfikator dodanego seansu
     */
    String dodajSeans(String daneSeansu);

    /**
     * Edycja seansu.
     * @param idSeansu identyfikator seansu
     * @param daneSeansu nowe dane seansu
     * @return potwierdzenie edycji
     */
    String edytujSeans(String idSeansu, String daneSeansu);

    /**
     * Usuniecie seansu.
     * @param idSeansu identyfikator seansu
     * @return potwierdzenie usuniecia
     */
```

```
50      String usunSeans(String idSeansu);
51
52      /**
53       * Edytowanie oferty kina.
54       * @param daneOperacji dane operacji
55       * @return wynik operacji
56       */
57      String edytujOferteKina(String daneOperacji);
58  }
```

5.1.8 IClientController.java

```
1   package controller;
2
3   /**
4    * Interfejs fasady kontrolera dla Klienta.
5    */
6   public interface IClientController {
7
8       /**
9        * Przegladanie repertuaru.
10       * @param kryteria kryteria wyszukiwania
11       * @return lista seansow
12       */
13      String przegladajRepertuar(String kryteria);
14
15      /**
16       * Wybor seansu.
17       * @param idSeansu identyfikator seansu
18       * @return szczegoly seansu
19       */
20      String wybierzSeans(String idSeansu);
21
22      /**
23       * Wybor miejsc.
24       * @param daneMiejsc dane miejsc
25       * @return potwierdzenie wyboru
26       */
27      String wybierzMiejsca(String daneMiejsc);
28
29      /**
30       * Rezerwacja biletu.
31       * @param daneRezerwacji dane rezerwacji
32       * @return potwierdzenie rezerwacji
33       */
34      String rezerwujBilet(String daneRezerwacji);
35
36      /**
```

```
37          * Anulowanie rezerwacji.
38          * @param idRezerwacji identyfikator rezerwacji
39          * @return potwierdzenie anulowania
40          */
41         String anulujRezerwacje(String idRezerwacji);
42     }
```

5.1.9 IStrategiaEdycjiOfertyKina.java

```
1  package controller;
2
3  import model.IModel2;
4
5  /**
6   * Abstrakcyjna klasa bazowa dla strategii edycji oferty kina.
7   * Implementuje wzorzec Strategia.
8   */
9  public abstract class IStrategiaEdycjiOfertyKina {
10     protected IModel2 model;
11
12     /**
13      * Konstruktor strategii.
14      * @param model fasada warstwy Model
15      */
16     public IStrategiaEdycjiOfertyKina(IModel2 model) {
17         this.model = model;
18     }
19
20     /**
21      * Metoda abstrakcyjna wykonania edycji oferty.
22      * Implementowana przez konkretne strategie.
23      * @param daneOperacji dane operacji
24      * @return wynik operacji
25      */
26     public abstract String edytujOferte(String daneOperacji);
27 }
```

5.1.10 UsuniecieFilmu.java

```
1  package controller;
2
3  import model.IModel2;
4
5  /**
6   * Strategia usuwania filmu z oferty kina.
7   * Implementuje IStrategiaEdycjiOfertyKina.
8   */
```

```java
public class UsuniecieFilmu extends IStrategiaEdycjiOfertyKina {

    /**
     * Konstruktor strategii usuwania filmu.
     * @param model fasada warstwy Model
     */
    public UsuniecieFilmu(IModel2 model) {
        super(model);
    }

    /**
     * Wykonanie edycji oferty - usuniecie filmu.
     * @param daneOperacji dane zawierajace ID filmu do usuniecia
     * @return wynik operacji usuwania
     */
    @Override
    public String edytujOferte(String daneOperacji) {
        throw new UnsupportedOperationException();
    }
}
```

5.2 Pakiet Model

5.2.1 DAO.java

```java
package model;

import java.util.Map;
import java.util.HashMap;

/**
 * Data Access Object - adapter do warstwy zasobow (bazy danych).
 * Symuluje operacje CRUD na bazie danych.
 */
public class DAO implements IDAO {
    private Map<String, String> bazyFilmow;
    private Map<String, String> bazySeans;
    private Map<String, String> bazyRezerwacji;
    private Map<String, String> bazyKlientow;

    /**
     * Konstruktor DAO - inicjalizacja struktur danych.
     */
    public DAO() {
        this.bazyFilmow = new HashMap<>();
        this.bazySeans = new HashMap<>();
        this.bazyRezerwacji = new HashMap<>();
        this.bazyKlientow = new HashMap<>();
```

```
24      }
25
26      /**
27       * Dodanie wpisu do logu zdarzen.
28       * @param zdarzenie opis zdarzenia
29       */
30      public void dodajWpisDoLogu(String zdarzenie) {
31          throw new UnsupportedOperationException();
32      }
33
34      /**
35       * Wyszukanie filmu w bazie danych.
36       * @param idFilmu identyfikator filmu
37       * @return dane filmu w formacie tekstowym
38       */
39      public String znajdzFilm(String idFilmu) {
40          throw new UnsupportedOperationException();
41      }
42
43      /**
44       * Dodanie filmu do bazy danych.
45       * @param film dane filmu
46       * @return identyfikator dodanego filmu
47       */
48      public String dodajFilm(String film) {
49          throw new UnsupportedOperationException();
50      }
51
52      /**
53       * Edycja filmu w bazie danych.
54       * @param film zaktualizowane dane filmu
55       */
56      public void edytujFilm(String film) {
57          throw new UnsupportedOperationException();
58      }
59
60      /**
61       * Usuniecie filmu z bazy danych.
62       * @param idFilmu identyfikator filmu
63       */
64      public void usunFilm(String idFilmu) {
65          throw new UnsupportedOperationException();
66      }
67
68      /**
69       * Wyszukanie seansu w bazie danych.
70       * @param idSeansu identyfikator seansu
71       * @return dane seansu
72       */
```

```
73      public String znajdzSeans(String idSeansu) {
74          throw new UnsupportedOperationException();
75      }
76
77      /**
78       * Wyszukanie wszystkich seansow danego filmu.
79       * @param idFilmu identyfikator filmu
80       * @return tablica identyfikatorow seansow
81       */
82      public String[] znajdzSeansyFilmu(String idFilmu) {
83          throw new UnsupportedOperationException();
84      }
85
86      /**
87       * Dodanie seansu do bazy danych.
88       * @param seans dane seansu
89       * @return identyfikator dodanego seansu
90       */
91      public String dodajSeans(String seans) {
92          throw new UnsupportedOperationException();
93      }
94
95      /**
96       * Edycja seansu w bazie danych.
97       * @param seans zaktualizowane dane seansu
98       */
99      public void edytujSeans(String seans) {
100         throw new UnsupportedOperationException();
101     }
102
103     /**
104      * Usuniecie seansu z bazy danych.
105      * @param idSeansu identyfikator seansu
106      */
107     public void usunSeans(String idSeansu) {
108         throw new UnsupportedOperationException();
109     }
110
111     /**
112      * Wyszukanie rezerwacji w bazie danych.
113      * @param idRezerwacji identyfikator rezerwacji
114      * @return dane rezerwacji
115      */
116     public String znajdzRezerwacje(String idRezerwacji) {
117         throw new UnsupportedOperationException();
118     }
119
120     /**
121      * Dodanie rezerwacji do bazy danych.
```

```
122         * @param rezerwacja dane rezerwacji
123         * @return identyfikator dodanej rezerwacji
124         */
125        public String dodajRezerwacje(String rezerwacja) {
126            throw new UnsupportedOperationException();
127        }
128
129        /**
130         * Usuniecie rezerwacji z bazy danych.
131         * @param idRezerwacji identyfikator rezerwacji
132         */
133        public void usunRezerwacje(String idRezerwacji) {
134            throw new UnsupportedOperationException();
135        }
136
137        /**
138         * Wyszukanie klienta w bazie danych.
139         * @param idKlienta identyfikator klienta
140         * @return dane klienta
141         */
142        public String znajdzKlienta(String idKlienta) {
143            throw new UnsupportedOperationException();
144        }
145
146        /**
147         * Dodanie klienta do bazy danych.
148         * @param klient dane klienta
149         * @return identyfikator dodanego klienta
150         */
151        public String dodajKlienta(String klient) {
152            throw new UnsupportedOperationException();
153        }
154    }
```

5.2.2 DekoratorFilmu.java

```
1   package model;
2
3   /**
4    * Abstrakcyjna klasa dekoratora filmu.
5    * Implementuje wzorzec Dekorator.
6    */
7   public abstract class DekoratorFilmu implements IFilm {
8       protected IFilm film;
9
10      /**
11       * Konstruktor dekoratora.
12       * @param film obiekt filmu do udekorowania
```

```
     */
    public DekoratorFilmu(IFilm film) {
        this.film = film;
    }

    public String dajId() {
        return this.film.dajId();
    }

    public String dajTytul() {
        return this.film.dajTytul();
    }

    public String dajOpis() {
        return this.film.dajOpis();
    }

    public int dajCzasTrwania() {
        return this.film.dajCzasTrwania();
    }

    public double dajCeneSeansow() {
        return this.film.dajCeneSeansow();
    }
}
```

5.2.3 FabrykaFilmuVIP.java

```
package model;

/**
 * Fabryka filmow VIP z dodatkowymi uslugami.
 * Implementuje IFabrykaFilmu.
 */
public class FabrykaFilmuVIP implements IFabrykaFilmu {
    private double doplataPremium;
    private String[] dodatkowe;

    /**
     * Konstruktor fabryki filmow VIP.
     * @param doplata kwota doplaty za usluge VIP
     * @param dodatkowe dodatkowe uslugi VIP
     */
    public FabrykaFilmuVIP(double doplata, String[] dodatkowe) {
        this.doplataPremium = doplata;
        this.dodatkowe = dodatkowe;
    }

```

```
    /**
     * Utworzenie filmu VIP.
     * @param daneFilmu dane bazowe filmu
     * @return film z dekoracj  VIP
     */
    public IFilm utworzFilm(String daneFilmu) {
        throw new UnsupportedOperationException();
    }
}
```

5.2.4 FabrykaFilmuZPromocja.java

```
package model;

/**
 * Fabryka filmow z promocja cenowa.
 * Implementuje IFabrykaFilmu.
 */
public class FabrykaFilmuZPromocja implements IFabrykaFilmu {
    private int procentZnizki;
    private String opisPromocji;

    /**
     * Konstruktor fabryki filmow promocyjnych.
     * @param znizka procent znizki
     * @param opis opis promocji
     */
    public FabrykaFilmuZPromocja(int znizka, String opis) {
        this.procentZnizki = znizka;
        this.opisPromocji = opis;
    }

    /**
     * Utworzenie filmu z promocja.
     * @param daneFilmu dane bazowe filmu
     * @return film z dekoracj  promocji
     */
    public IFilm utworzFilm(String daneFilmu) {
        throw new UnsupportedOperationException();
    }
}
```

5.2.5 FabrykaStandardowegoFilmu.java

```
package model;

/**
```

```
 * Fabryka standardowych filmow bez dodatkowych uslug.
 * Implementuje IFabrykaFilmu.
 */
public class FabrykaStandardowegoFilmu implements IFabrykaFilmu {

    /**
     * Utworzenie standardowego filmu.
     * @param daneFilmu dane filmu
     * @return obiekt standardowego filmu
     */
    public IFilm utworzFilm(String daneFilmu) {
        throw new UnsupportedOperationException();
    }
}
```

5.2.6 Film.java

```
package model;

/**
 * Klasa reprezentujaca film w systemie.
 */
public class Film implements IFilm {
    private String id;
    private String tytul;
    private String opis;
    private int czasTrwania;
    private String gatunek;
    private double cenaPodstawowa;

    /**
     * Konstruktor filmu.
     * @param id identyfikator filmu
     * @param tytul tytul filmu
     * @param opis opis fabuly
     * @param czas czas trwania w minutach
     * @param gatunek gatunek filmu
     * @param cena podstawowa cena biletu
     */
    public Film(String id, String tytul, String opis, int czas,
                String gatunek, double cena) {
        this.id = id;
        this.tytul = tytul;
        this.opis = opis;
        this.czasTrwania = czas;
        this.gatunek = gatunek;
        this.cenaPodstawowa = cena;
    }
```

```
32
33      public String dajId() {
34          return this.id;
35      }
36
37      public String dajTytul() {
38          return this.tytul;
39      }
40
41      public String dajOpis() {
42          return this.opis;
43      }
44
45      public int dajCzasTrwania() {
46          return this.czasTrwania;
47      }
48
49      public double dajCeneSeansow() {
50          return this.cenaPodstawowa;
51      }
52  }
```

5.2.7 FilmVIP.java

```
1   package model;
2
3   /**
4    * Dekorator filmu - wersja VIP z dodatkowymi uslugami.
5    */
6   public class FilmVIP extends DekoratorFilmu {
7       private double doplataPremium;
8       private String[] dodatkowe;
9
10      /**
11       * Konstruktor dekoratora VIP.
12       * @param film bazowy film
13       * @param doplata kwota doplaty
14       * @param dodatkowe dodatkowe uslugi
15       */
16      public FilmVIP(IFilm film, double doplata, String[] dodatkowe
            ) {
17          super(film);
18          this.doplataPremium = doplata;
19          this.dodatkowe = dodatkowe;
20      }
21
22      @Override
23      public String dajOpis() {
```

```
        return this.film.dajOpis() + " [VIP: " +
                String.join(", ", this.dodatkowe) + "]";
    }

    @Override
    public double dajCeneSeansow() {
        return this.film.dajCeneSeansow() + this.doplataPremium;
    }
}
```

5.2.8 FilmZPromocja.java

```
package model;

/**
 * Dekorator filmu - wersja promocyjna ze znizka.
 */
public class FilmZPromocja extends DekoratorFilmu {
    private int procentZnizki;
    private String opisPromocji;

    /**
     * Konstruktor dekoratora promocyjnego.
     * @param film bazowy film
     * @param znizka procent znizki
     * @param opis opis promocji
     */
    public FilmZPromocja(IFilm film, int znizka, String opis) {
        super(film);
        this.procentZnizki = znizka;
        this.opisPromocji = opis;
    }

    @Override
    public String dajOpis() {
        return this.film.dajOpis() + " [PROMOCJA: " +
                this.opisPromocji + " -" + this.procentZnizki + "
                   %]";
    }

    @Override
    public double dajCeneSeansow() {
        return this.film.dajCeneSeansow() *
                (1.0 - this.procentZnizki / 100.0);
    }
}
```

5.2.9 IDAO.java

```java
package model;

/**
 * Interfejs Data Access Object.
 * Definiuje operacje dostepu do warstwy zasobow.
 */
public interface IDAO {
    void dodajWpisDoLogu(String zdarzenie);
    String znajdzFilm(String idFilmu);
    String dodajFilm(String film);
    void edytujFilm(String film);
    void usunFilm(String idFilmu);
    String znajdzSeans(String idSeansu);
    String[] znajdzSeansyFilmu(String idFilmu);
    String dodajSeans(String seans);
    void edytujSeans(String seans);
    void usunSeans(String idSeansu);
    String znajdzRezerwacje(String idRezerwacji);
    String dodajRezerwacje(String rezerwacja);
    void usunRezerwacje(String idRezerwacji);
    String znajdzKlienta(String idKlienta);
    String dodajKlienta(String klient);
}
```

5.2.10 IFabrykaFilmu.java

```java
package model;

/**
 * Interfejs fabryki filmow.
 * Implementuje wzorzec Fabryka Abstrakcyjna.
 */
public interface IFabrykaFilmu {
    /**
     * Utworzenie obiektu filmu.
     * @param daneFilmu dane filmu w formacie tekstowym
     * @return utworzony obiekt filmu
     */
    IFilm utworzFilm(String daneFilmu);
}
```

5.2.11 IFilm.java

```java
package model;

```

```java
/**
 * Interfejs filmu.
 * Definiuje podstawowe operacje dla obiektow filmow.
 */
public interface IFilm {
    String dajId();
    String dajTytul();
    String dajOpis();
    int dajCzasTrwania();
    double dajCeneSeansow();
}
```

5.2.12 IModel2.java

```java
public interface IModel2 {

    /**
     *
     * @param daneFilmu
     */
    string dodajFilm(string daneFilmu);

    /**
     *
     * @param idFilmu
     * @param daneFilmu
     */
    string edytujFilm(string idFilmu, string daneFilmu);

    /**
     *
     * @param idFilmu
     */
    string usunFilm(string idFilmu);

    /**
     *
     * @param daneSeansu
     */
    string dodajSeans(string daneSeansu);

    /**
     *
     * @param idSeansu
     * @param daneSeansu
     */
    string edytujSeans(string idSeansu, string daneSeansu);

```

```
35      /**
36       *
37       * @param idSeansu
38       */
39      string usunSeans(string idSeansu);
40
41      /**
42       *
43       * @param idKina
44       */
45      string pobierzRepertuarKina(string idKina);
46
47      /**
48       *
49       * @param daneRezerwacji
50       */
51      string zarezerwujMiejsce(string daneRezerwacji);
52
53      /**
54       *
55       * @param idRezerwacji
56       */
57      string anulujRezerwacje(string idRezerwacji);
58
59      /**
60       *
61       * @param daneZakupu
62       */
63      string finalizujZakup(string daneZakupu);
64
65  }
```

5.2.13 Rezerwacja.java

```
1   public class Rezerwacja {
2
3       private string id;
4       private string idSeansu;
5       private string idKlienta;
6       private int nrMiejsca;
7       private double cena;
8
9       /**
10       *
11       * @param id
12       * @param idSeansu
13       * @param idKlienta
14       * @param nrMiejsca
```

```
15          * @param cena
16          */
17         public Rezerwacja(string id, string idSeansu, string
               idKlienta, int nrMiejsca, double cena) {
18             // TODO - implement Rezerwacja.Rezerwacja
19             throw new UnsupportedOperationException();
20         }
21
22         public string dajId() {
23             // TODO - implement Rezerwacja.dajId
24             throw new UnsupportedOperationException();
25         }
26
27         public double dajCene() {
28             // TODO - implement Rezerwacja.dajCene
29             throw new UnsupportedOperationException();
30         }
31
32         public string dajOpis() {
33             // TODO - implement Rezerwacja.dajOpis
34             throw new UnsupportedOperationException();
35         }
36
37 }
```

6 Podsumowanie

W ramach Etapu 3 laboratorium z Inżynierii Oprogramowania wykonano:

1. Diagram komponentów przedstawiający 3-warstwową architekturę MVC systemu sprzedaży biletów dla sieci multikin.
2. Diagram klas warstwy Controller z zastosowaniem wzorców projektowych: Fasada, Strategia i Metoda Szablonowa.
3. Diagram klas warstwy Model z zastosowaniem wzorców projektowych: Fasada, Adapter (DAO), Dekorator, Fabryka Abstrakcyjna i Kompozyt.
4. Przygotowanie struktury do implementacji kodu źródłowego w języku Java na podstawie zaprojektowanych diagramów.

Zaprojektowana architektura przestrzega zasad SOLID i wykorzystuje sprawdzone wzorce projektowe, co zapewnia:

- Luźne powiązanie między warstwami
- Łatwość rozszerzania funkcjonalności
- Wysoką testowalność kodu
- Przejrzystą strukturę systemu
- Łatwość utrzymania i modyfikacji