Politechnika Wrocławska
Katedra Informatyki Technicznej

Inżynieria Oprogramowania

Zapoznanie się z wybranym narzędziem UML - wprowadzenie do UML

Magdalena Biernat
Mateusz Bortkiewicz

Opiekun
prof. dr hab. inż. Jan Magott

22 października 2017

1 Wprowadzenie

Niniejsze sprawozdanie jest dokumentem z pierwszego laboratorium *Inżynierii Oprogramowania*.

1.1 Cel laboratorium

Wprowadzenie do UML – wykonanie prostego projektu programu za pomocą wybranych diagramów UML i implementacja projektu programu w języku Java.

1.2 Plan pracy

Postawiono wykonać zadania wg instrukcji prowadzącego:

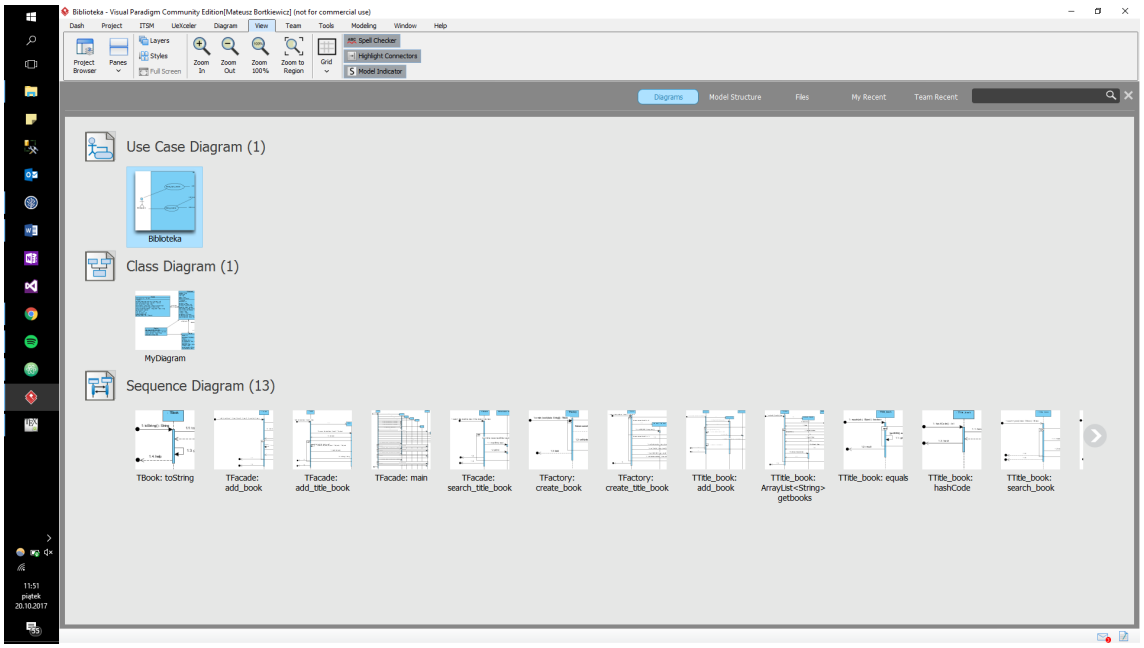
- Zapoznanie się z punktami 1 i 2 instrukcji laboratorium.
- Wykonanie projektu UML w środowisku Visual Paradigm.
- Odwzorowanie programu w języku Java w środowisku Eclipse.

2 Laboratorium

2.1 Przepisanie

2.1.1 Projekt UML

Na potrzeby zapoznania się z językiem UML, stworzono projekt na podstawie instrukcji. Widok na wszystkie diagramy stworzone w Visual Paradigm wygląda następująco:



Rysunek 1: Stworzone diagramy

2.1.2 Odwzorowanie w Javie

Kolejnym krokiem było odwzorowanie środowiska w języku Java. W celu efektywniejszego stworzenia projektu, porzucono rozwiązanie w środowisku NetBeans i wykonano projekt w środowisku Eclipse. W wyniku odwzorowania i skompilowania projektu, uzyskano odpowiedź w konsoli:

```
[Title: Title1 Author: Author1 ISBN: ISBN1 Publisher: Publisher1,
Title: Title2 Author: Author2 ISBN: ISBN2 Publisher: Publisher2,
Title: Title3 Author: Author3 ISBN: ISBN3 Publisher: Publisher3]
[Title: Title1 Author: Author1 ISBN: ISBN1 Publisher: Publisher1 Number: 1]
[Title: Title2 Author: Author2 ISBN: ISBN2 Publisher: Publisher2 Number: 1]
[Title: Title2 Author: Author2 ISBN: ISBN2 Publisher: Publisher2 Number: 1]
[Title: Title2 Author: Author2 ISBN: ISBN2 Publisher: Publisher2 Number: 2]
```

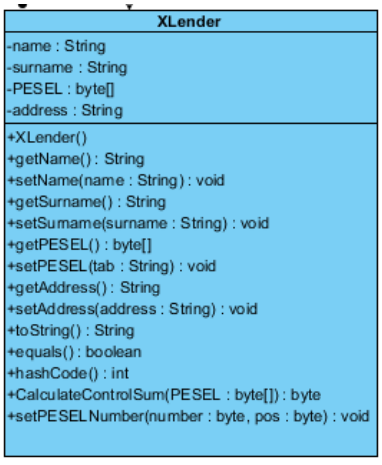
Uzyskana odpowiedź jest zgodna z tą, która jest zawarta w instrukcji, a co za tym idzie: ćwiczenie zostało wykonane poprawnie.

2.2 Część kreatywna

Na potrzeby ćwiczenia stworzono klasę XLender, która odzwierciedla wypożyczającego. Klasa posiada pola tj. imię, nazwisko, PESEL, adres. W celu obsługi nowopowstałej klasy, dodano lub zmodyfikowano pola lub metody w klasach istniejących. I tak klasa TFacade otrzymała metody *search_lender(), add_lender(), getLenders()* oraz pole *lenders* klasy XLender. Metoda *main()* została odpowiednio zmodyfikowana. Klasa TFactory otrzymała natomiast metodę *create_lender*.

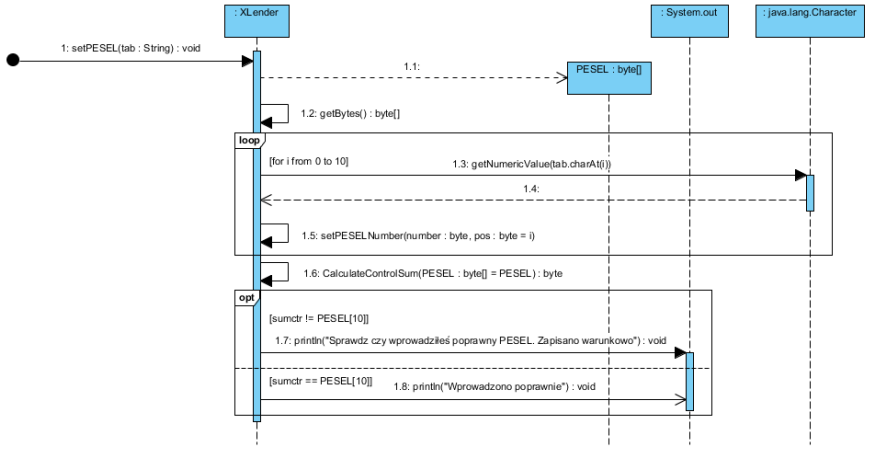
2.2.1 Project UML

W diagramie klas dodano klasę XLender oraz zmodyfikowano odpowiednie klasy dodając im odpowiednie metody.

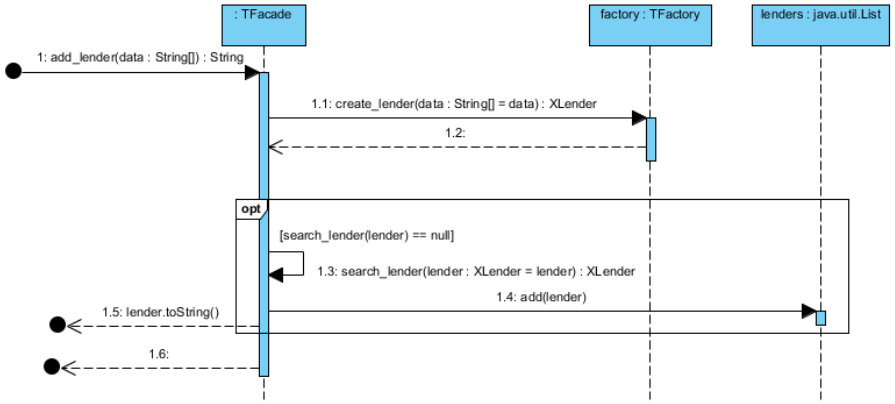


Rysunek 2: Stworzona klasa

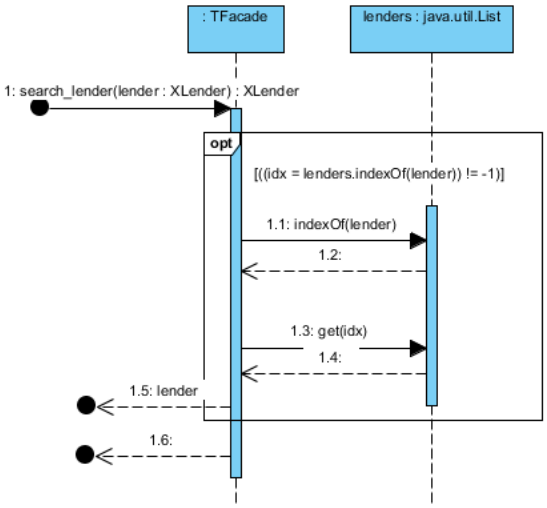
W diagramach sekwencji dodano diagramy dla setera PESEL-u klasy XLender, add_lender i search_lender klasy TFacade, create_lender klasy TFactory. Zmodyfikowano m.in metodę main w klasie TFacade.



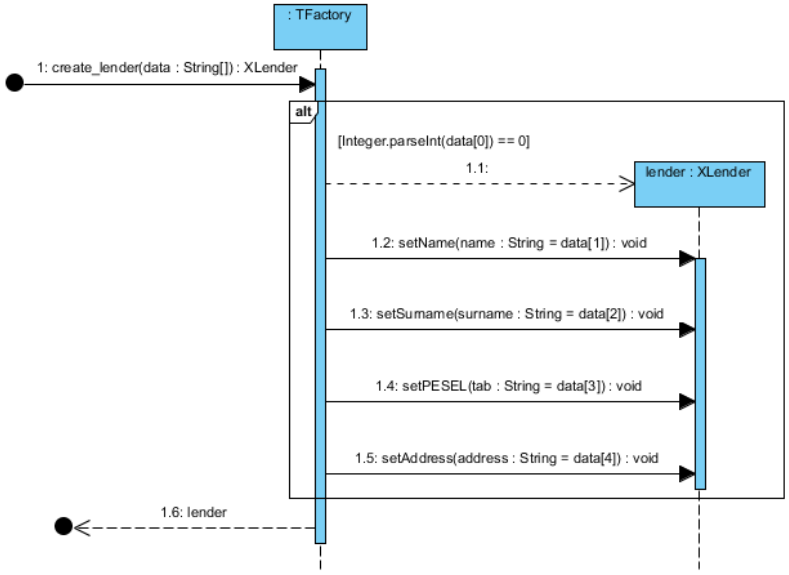
Rysunek 3: XLender: setPESEL



Rysunek 4: TFacade: add_lender



Rysunek 5: TFacade: search_lender



Rysunek 6: TFactory: create_lender

2.2.2 Odzwierciedlenie w Javie

Klasa XLender posiada pola:

```
private String name;
private String surname;
private String address;
private byte[] PESEL;
```

oraz odpowiednie settery i gettery. Tutaj m.in. setter PESELU:

```
public void setPESEL(String tab) {
this.PESEL = new byte[11];

for (byte i = 0; i < 11; i++)
setPESELNumber((byte) Character.getNumericValue(tab.charAt(i)), i);

byte sumctr = CalculateControlSum(this.PESEL);
if (sumctr != this.PESEL[10])
System.out.println("Sprawdz czy wprowadziłeś poprawny PESEL. Zapisano warunkowo");
else
System.out.println("Wprowadzono poprawnie");
}
```

W klasie TFacade mamy nowe metody:

```
public XLender search_lender(XLender lender){

int idx;
if((idx = lenders.indexOf(lender)) != -1){
lender = lenders.get(idx);
return lender;
}
return null;
}

public String add_lender(String data[]){
TFactory factory = new TFactory();
XLender lender = factory.create_lender(data);
if(search_lender(lender) == null){
lenders.add(lender);
return lender.toString();
}
return null;
}
```

Podobnie w klasie TFactory

```
public XLender create_lender(String data[]){
XLender lender = null;
switch (Integer.parseInt(data[0])){
case 0:
lender = new XLender();
```

```
lender.setName(data[1]);
lender.setSurname(data[2]);
lender.setPESEL(data[3]);
lender.setAddress(data[4]);
break;
}
return lender;
}
```

Pełen kod oraz diagramy znajdują się w załączonych plikach.