Wojskowa Akademia Techniczna im. Jarosława Dąbrowskiego w Warszawie



SPRAWOZDANIE

z przedmiotu

Systemy baz danych

Temat: *Projekt obiektowej bazy danych Uczelnia*

Studenci: Konrad Sposób, Marek Wąsowski

Grupa: I7B4S4

Kierunek: Informatyka, Systemy informatyczne

Prowadzący:

mgr inż. Maciej Szymczyk

Warszawa, 06.05.2018r.

Spis treści

[Treść zadania projektowego 3](#_Toc513400366)

[1. Analiza problemu 4](#_Toc513400367)

[2. Projekt 4](#_Toc513400368)

[2.1 Opis klas: 5](#_Toc513400369)

[2.2 Opis metod: 5](#_Toc513400370)

[2.3 Opis związków 6](#_Toc513400371)

[2.4 Polimorfizm 6](#_Toc513400372)

[2.5 Nowo zdefiniowane typy danych 8](#_Toc513400373)

[3. Versant 9](#_Toc513400374)

[3.1 Wypełnienie bazy danych 9](#_Toc513400375)

[3.1 Wykonanie metod 12](#_Toc513400376)

[4. Repozytorium 13](#_Toc513400377)

[5. Podsumowanie 13](#_Toc513400378)

[5.1 Napotkane problemy i ich rozwiązania 13](#_Toc513400379)

[5.2 Pomysły/idee 13](#_Toc513400380)

[5.3 Ocena środowiska 13](#_Toc513400381)

[5.4 Tabela do wypełnienia 14](#_Toc513400382)

# Treść zadania projektowego

Dla wybranego zagadnienia należy (w grupach 2 osobowych):

1. Identyfikacja klas, metod, związków między klasami dla wybranego obszaru dziedzinowego
2. Implementacja obiektowej bazy danych dla wybranego obszaru dziedzinowego wykorzystując wybraną OBD z listy <https://db-engines.com/en/ranking/object+oriented+dbms> (uwaga, każda grupa używa innego oprogramowania) :

* model bazy danych powinien składać się z co najmniej 6 klas;
* model bazy danych powinien zawierać:
  + metody;
  + dziedziczenie;
  + związki;
  + klasy abstrakcyjne i polimorfizm;
  + nowo zdefiniowane typy danych;

1. Wypełnienie bazy danych testowymi danymi
2. Pokaz wykonania metod obiektów w terminalu
3. Umieszczenie projektu w repozytorium git na <https://github.com/>. Wymagana jest aktywność wszystkich uczestników grupy.
4. Udokumentowanie pracy w formie sprawozdania laboratoryjnego.
5. Podsumowanie wykonanej pracy w szczególności uwzględniając:
   * napotkane problemy i ich rozwiązania,
   * pomysły/idee,
   * ocenę środowiska.
   * wypełnioną tabelkę zamieszczoną na końcu tego dokumentu
6. Wysłanie sprawozdania (format .docx lub pdf) razem z linkiem do repozytorium na adres mailowy prowadzącego maksymalnie 20 dni po zajęciach laboratoryjnych do godziny 23:59
7. Temat maila powinien rozpoczynać się od #OBD

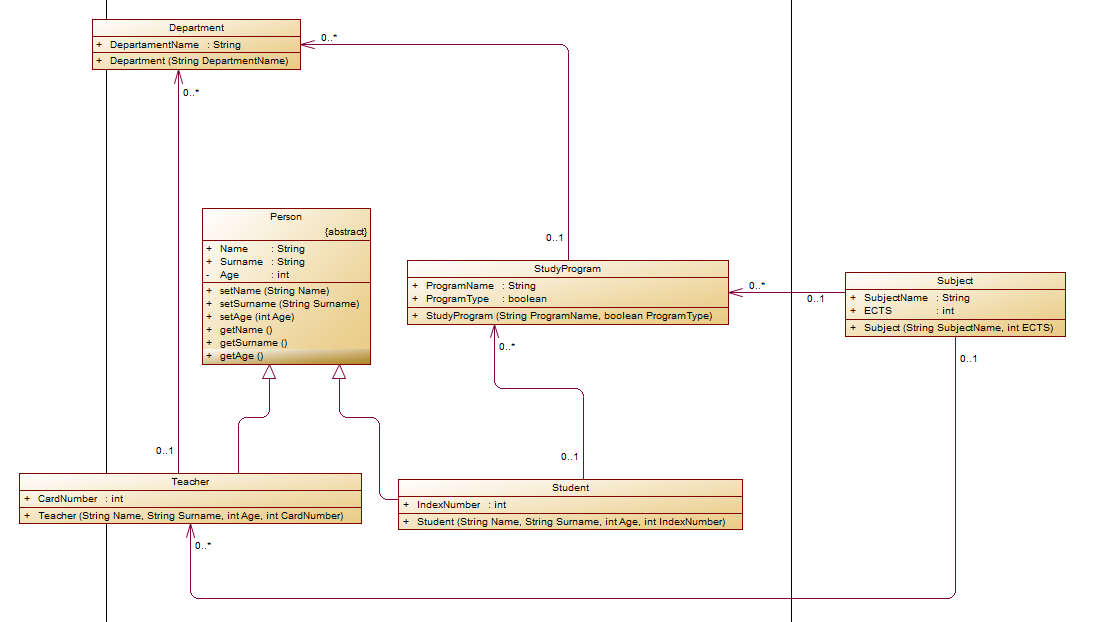
# Analiza problemu

Projekt opiera się na modelu bazy danych dla pewnej uczelni, która przechowuje informacje na temat:

* pracowników zatrudnionych na poszczególnych wydziałach uczelni,
* studentów uczęszczających na zajęcia w ramach danego kierunku,
* wydziałów, z których składa się uczelnia,
* programów studiów obejmujących zbiór przedmiotów jakie uczelnia oferuje na poszczególnych wydziałach,
* przedmiotów jakie są wykładane na danym kierunku.

# Projekt

Projektowanie bazy danych uczelni rozpoczęto od zaprojektowania diagramu klas obiektowej bazy danych w środowisku PowerDesigner i na jego podstawie wykonano bazę danych w środowisku Versant Object Database. Taki proces wytwórczy oprogramowania pozwala na kompletną analizę architektury systemu informatycznego przy projektowaniu bazy danych w nowym systemie bazodanowym. Poniżej zaprezentowano wykonany diagram klas projektu bazy danych uczelni:



## Opis klas:

* Person: klasa abstrakcyjna, przechowuje informacje na temat wszystkich osób zarejestrowanych na uczelni
* Student: dziedziczy po klasie Person. Dodatkowo zawiera atrybut Numer indeksu.
* Teacher: dziedziczy po klasie Person. Dodatkowo zawiera atrybut Numer karty dostępu
* Department: przechowuje informacje, jakie wydziały należą do uczelni.
* StudyProgram: przechowuje informacje na temat kierunków studiów oraz ich typu: stacjonarny/niestacjonarny
* Subject: przechowuje informacje o wykładanych na uczelni przedmiotach.

## Opis metod:

* **Person:**
  + setName(String Name):ustawia imię
  + setSurname(string Surname): ustawia nazwisko
  + setAge(int Age): ustawia wiek
  + getName():pobiera imię
  + getSurname(): pobiera nazwisko
  + getAge(): pobiera wiek
* **Student:**
* Student(String Name, String Surname, int Age, int IndexNumber): konstruktor klasy
* **Teacher:**
  + Teacher(String Name, String Surname, int Age, int CardNumber): konstruktor klasy
* **Department:**
  + Department(String DepartmentName): konstruktor klasy
* **StudyProgram:**
  + StudyProgram(String ProgramName, boolean ProgramType): konstruktor klasy
* **Subject:**
  + Subject(String SubjectName, int ECTS): konstruktor klasy

## Opis związków

* **Department**

@OneToMany(cascade=CascadeType.***PERSIST***)

**private** Collection<Teacher> teachers;

**private** Collection<StudyProgram> studyProgrames;

Do jednego wydziału należy wielu nauczycieli oraz podlega wiele programów studiów

* **StudyProgram:**

@OneToMany(cascade=CascadeType.***PERSIST***)

**private** Collection<Student> students;

**private** Collection<Subject> subjects;

Na jednym programie studiów uczy się wielu studentów oraz prowadzonych jest wiele przedmiotów

* **Teacher:**

@OneToMany(cascade=CascadeType.***PERSIST***)

**private** Collection<Subject> subject;

Jeden nauczyciel może prowadzić wiele przedmiotów

## Polimorfizm

Na potrzeby wykorzystania paradygmatu polimorfizmu w projekcie wykonano abstrakcyjną klasę Person, z której dziedziczą klasy Student i Teacher, co znajduje odzwierciedlenie w Tworzeniu obiektów klas studenta i nauczyciela.

Poniżej kod klasy Person:

**package** com.versant.jpa.helloworld.model;

**public** **abstract** **class** Person {

**int** age;

String name, surname;

**abstract** String **setName**(String name);

**abstract** String **setSurname**(String surname);

**abstract** **int** **setAge**(**int** age);

**abstract** String **getName**();

**abstract** String **getSurname**();

**abstract** **int** **getAge**();

**Person**(){}

Poniżej kod klasy Student:

**package** com.versant.jpa.helloworld.model;

**import** javax.persistence.\*;

/\*\*

\* a very simple JPA Entity for the jpa\_hello\_world example project

\*/

@Entity

**public** **class** Student **extends** Person {

@Id

**private** **long** id;

**int** IndexNumber;

**public** **Student**(String name, String surname, **int** age, **int** IndexNumber) {

**super**();

**this**.name = name;

**this**.surname = surname;

**this**.age=age;

**this**.IndexNumber=IndexNumber;

}

@SuppressWarnings("unused")

**private** **Student**(){

// a no-args constructor is required, it may be private or protected

}

@Override

**public** String **toString**() {

**return** "Student["+name+","+surname+","+age+","+IndexNumber+"]";

}

@Override

String **setName**(String name) {

**this**.name=name;

**return** **this**.name;

}

@Override

String **setSurname**(String surname) {

**this**.surname=surname;

**return** **this**.surname;

}

@Override

**int** **setAge**(**int** age) {

**this**.age=age;

**return** **this**.age;

}

@Override

String **getName**() {

**return** **this**.name;

}

@Override

String **getSurname**() {

**return** **this**.surname;

}

@Override

**int** **getAge**() {

**return** **this**.age;

Poniżej utworzenie obiektu klasy Student:

Person **ps1** = **new** Student("Jan", "Jankowski", 19,0);

Sytuacja odbywa się analogicznie w przypadku nauczyciela.

## Nowo zdefiniowane typy danych

Środowisko Versant umożliwia definiowanie nowych typów danych. W tym celu utworzono tabele wyświetlające dane, które są zwracane przez zapytania:

TypedQuery<Department> **query2** = em.createQuery(jpq2, Department.**class**);

List<Department> **resultList2** = query2.getResultList();

**int** **nDep**=1;

System.***out***.println("@VERSANT\_SERVER:: Runing query: "+jpq2);

**for** (Department **dep** : resultList2) {

System.***out***.println(nDep+". "+ dep);

nDep++;

TypedQuery<Subject> **query2** = em.createQuery(jpq2, Subject.**class**);

List<Subject> **resultList2** = query2.getResultList();

**int** **n**=1;

System.***out***.println("@VERSANT\_SERVER:: Runing query: "+jpq2);

**for** (Subject **sub** : resultList2) {

System.***out***.println(n+". "+ sub);

n++;

}

TypedQuery<StudyProgram> **query2** = em.createQuery(jpq2, StudyProgram.**class**);

List<StudyProgram> **resultList2** = query2.getResultList();

**int** **n**=1;

System.***out***.println("@VERSANT\_SERVER:: Runing query: "+jpq2);

**for** (StudyProgram **sp** : resultList2) {

System.***out***.println(n+". "+ sp);

n++;

}

# Versant

Versant Object Database (VOD) to oprogramowanie bazy danych obiektów opracowane przez firmę Versant Corporation. Baza danych obiektu Versant umożliwia programistom używanie języków zorientowanych obiektowo do przechowywania danych w celach transakcyjnych, umożliwiając odpowiedniemu językowi działanie jako język definicji danych (DDL) dla bazy danych. Innymi słowy, model pamięci jest modelem schematu bazy danych . Ogólnie rzecz biorąc, utrzymywanie w VOD zaimplementowane poprzez deklarowanie listy klas, a następnie udostępnienie interfejsu programowania aplikacji do rozgraniczenia transakcji w przypadkach użycia. Istnieją dodatkowe interfejsy API, wykraczające poza proste rozgraniczenie transakcji, zapewniające bardziej zaawansowane możliwości niezbędne do rozwiązania praktycznych problemów związanych z optymalizacją wydajności i skalowalnością w przypadku systemów z dużą ilością danych, wieloma równoczesnymi użytkownikami, opóźnieniami sieci , wąskimi gardłami itp.

Projekt obiektowej bazy danych uczelni został wykonany przy użyciu API w języku JAVA w środowisku Eclipse.

## Wypełnienie bazy danych

W celu wypełnienia bazy danych należy najpierw połączyć się z serwerem i utworzyć na nim bazę danych. Następnie korzystając z zaimplementowanych klas, należy dodawać do bazy obiekty:

Department **d1** = **new** Department("Cybernetyka");

em.persist(d1);

System.***out***.println("@VERSANT\_SERVER:: Object created - "+d1);

Department **d2** = **new** Department("Elektronika");

em.persist(d2);

System.***out***.println("@VERSANT\_SERVER:: Object created - "+d2);

Department **d3** = **new** Department("Logistyka");

em.persist(d3);

System.***out***.println("@VERSANT\_SERVER:: Object created - "+d3);

Department **d4** = **new** Department("Mechanika");

em.persist(d4);

System.***out***.println("@VERSANT\_SERVER:: Object created - "+d4);

Department **d5** = **new** Department("Optoelektronika");

em.persist(d5);

System.***out***.println("@VERSANT\_SERVER:: Object created - "+d5);

StudyProgram **sp1** = **new** StudyProgram("Informatyka",**false**);

em.persist(sp1);

System.***out***.println("@VERSANT\_SERVER:: Object created - "+sp1);

StudyProgram **sp2** = **new** StudyProgram("Kryptologia",**false**);

em.persist(sp2);

System.***out***.println("@VERSANT\_SERVER:: Object created - "+sp2);

StudyProgram **sp3** = **new** StudyProgram("Informatyka w medycynie",**true**);

em.persist(sp3);

System.***out***.println("@VERSANT\_SERVER:: Object created - "+sp3);

StudyProgram **sp4** = **new** StudyProgram("Logistyka",**false**);

em.persist(sp4);

System.***out***.println("@VERSANT\_SERVER:: Object created - "+sp4);

StudyProgram **sp5** = **new** StudyProgram("Budownictwo",**false**);

em.persist(sp5);

System.***out***.println("@VERSANT\_SERVER:: Object created - "+sp5);

StudyProgram **sp6** = **new** StudyProgram("Ekonomia",**false**);

em.persist(sp6);

System.***out***.println("@VERSANT\_SERVER:: Object created - "+sp6);

StudyProgram **sp7** = **new** StudyProgram("Ekonometria",**false**);

em.persist(sp7);

System.***out***.println("@VERSANT\_SERVER:: Object created - "+sp7);

StudyProgram **sp8** = **new** StudyProgram("Inżynieria środowiska",**false**);

em.persist(sp8);

System.***out***.println("@VERSANT\_SERVER:: Object created - "+sp8);

StudyProgram **sp9** = **new** StudyProgram("Geodezja",**false**);

em.persist(sp9);

System.***out***.println("@VERSANT\_SERVER:: Object created - "+sp9);

StudyProgram **sp10** = **new** StudyProgram("Bezpieczeństwo publiczne",**false**);

em.persist(sp9);

System.***out***.println("@VERSANT\_SERVER:: Object created - "+sp9);

Subject **s1** = **new** Subject("Analiza matematyczna",5);

em.persist(s1);

System.***out***.println("@VERSANT\_SERVER:: Object created - "+s1);

Subject **s2** = **new** Subject("Algebra liniowa",4);

em.persist(s2);

System.***out***.println("@VERSANT\_SERVER:: Object created - "+s2);

Subject **s3** = **new** Subject("Statystyka",3);

em.persist(s3);

System.***out***.println("@VERSANT\_SERVER:: Object created - "+s3);

Subject **s4** = **new** Subject("Programowanie obiektowe",2);

em.persist(s4);

System.***out***.println("@VERSANT\_SERVER:: Object created - "+s4);

Subject **s5** = **new** Subject("Rachunek prawdopodobieństwa",3);

em.persist(s5);

System.***out***.println("@VERSANT\_SERVER:: Object created - "+s5);

Person **pt1** = **new** Teacher("Krzysztof2", "Kowalski2", 65, 100000);

em.persist(pt1);

System.***out***.println("@VERSANT\_SERVER:: Object created - "+pt1);

Person **ps1** = **new** Student("Jan", "Jankowski", 19,0);

em.persist(ps1);

System.***out***.println("@VERSANT\_SERVER:: Object created - "+ps1);

Person **ps2** = **new** Student("Konrad", "Kowalewski", 20,1);

em.persist(ps2);

System.***out***.println("@VERSANT\_SERVER:: Object created - "+ps2);

Person **ps3** = **new** Student("Adam", "Król", 19,2);

em.persist(ps3);

System.***out***.println("@VERSANT\_SERVER:: Object created - "+ps3);

Person **ps4** = **new** Student("Tomasz", "Wiśniewski", 22,3);

em.persist(ps4);

System.***out***.println("@VERSANT\_SERVER:: Object created - "+ps4);

Person **ps5** = **new** Student("Alojzy", "Normalny", 21,4);

em.persist(ps5);

System.***out***.println("@VERSANT\_SERVER:: Object created - "+ps5);

Person **ps6** = **new** Student("Albert", "Jach", 22,5);

em.persist(ps6);

System.***out***.println("@VERSANT\_SERVER:: Object created - "+ps6);

Person **ps7** = **new** Student("Bartosz", "Martyniuk", 23,6);

em.persist(ps7);

System.***out***.println("@VERSANT\_SERVER:: Object created - "+ps7);

Person **ps8** = **new** Student("Błażej", "Jurkowski", 24,7);

em.persist(ps8);

System.***out***.println("@VERSANT\_SERVER:: Object created - "+ps8);

Person **ps9** = **new** Student("Bartłomiej", "Skiba", 25,8);

em.persist(ps9);

System.***out***.println("@VERSANT\_SERVER:: Object created - "+ps9);

Person **ps10** = **new** Student("Czesław", "Kot", 26,9);

em.persist(ps10);

System.***out***.println("@VERSANT\_SERVER:: Object created - "+ps10);

## Wykonanie metod

Działanie środowiska Versant opiera się na wykonywaniu odpowiednich metod w odpowiedniej kolejności. Na samym początku należy ,,postawić serwer”, następnie utworzyć bazę. W tym momencie można wykonywać na niej zapytania. Następnie środowisko umożliwia usunięcie bazy danych i zamknięcie połączenia z serwerem.

setUpDB();

create();

query("select d from Department d", "Department");

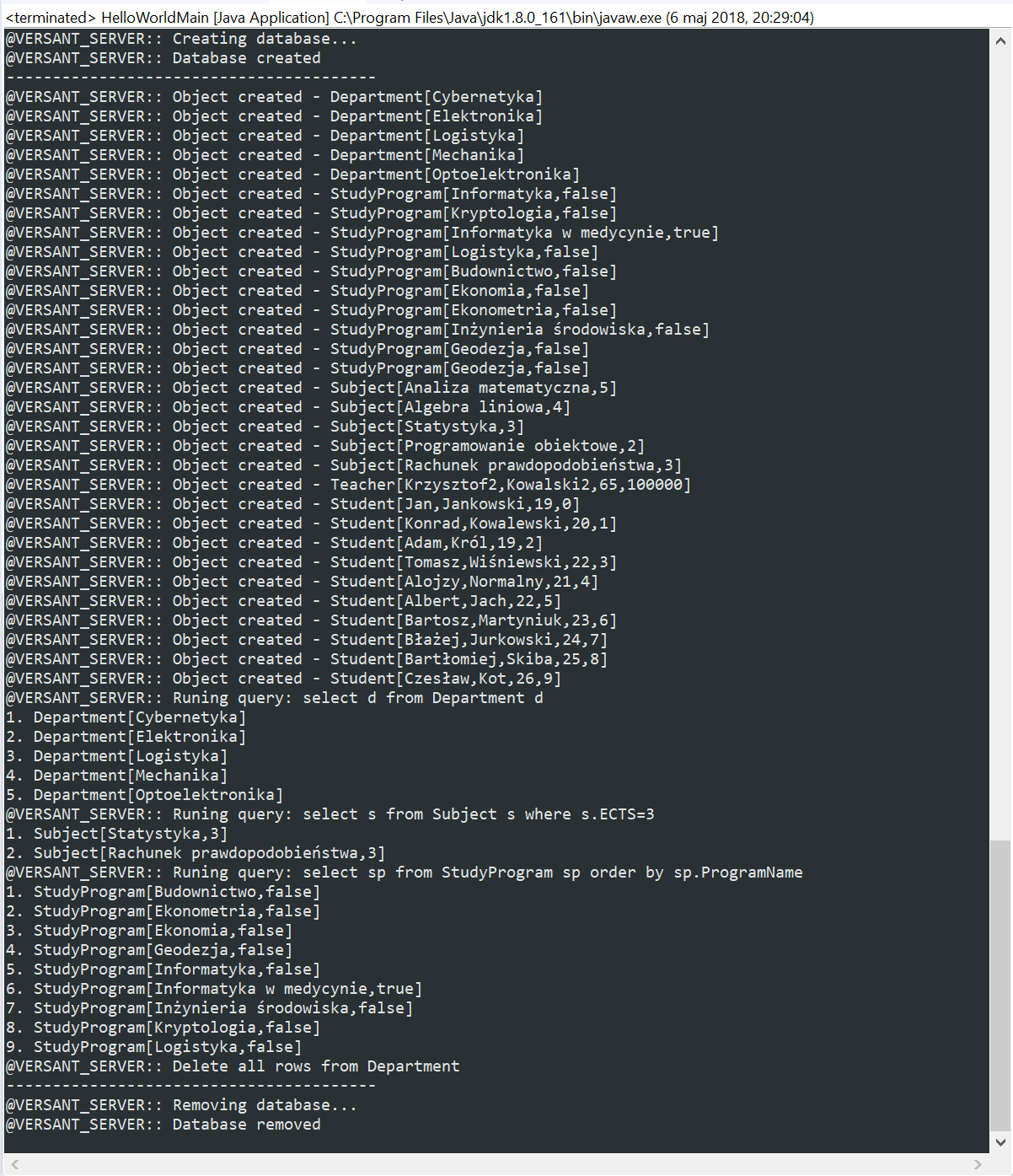
query("select s from Subject s where s.ECTS=3", "Subject");

query("select sp from StudyProgram sp order by sp.ProgramName", "StudyProgram");

delete();

tearDownDB();

Powyższe metody realizują następujące funkcjonalności:



# Repozytorium

Pełny kod został zamieszczony w repozytorium: <https://github.com/w0lko/SBD>

# Podsumowanie

## Napotkane problemy i ich rozwiązania

Pierwszy problem pojawił się już podczas konfigurowania środowiska. Podczas dodawania plugin’u Versant do środowiska Eclipse pojawiał się błąd przez co niemożliwym było uruchomienie bazy poza Eclipsem. Sprawiło to również, że wszystkie pliki XML o nazwie persistence w których znajdują się informacje o klasach musiały być edytowane ręcznie. Z tego powodu całą bazę możemy obejrzeć jedynie w konsoli środowiska Eclipse. Problem sprawiało również mała liczba publikacji/tutorialów pomagających przy pierwszym spotkaniu z Versantem. Ze względu na to implementacja często przebiegała metodą prób i błędów.

## Pomysły/idee

Wykonany przez nas projekt miał na celu zrozumienie działania obiektowych baz danych. Staraliśmy się poprawnie wykonać wszystkie zadane przez prowadzącego wymagania. Klasy zostały wypełnione podstawowymi atrybutami oraz danymi tak, aby była możliwość zademonstrowania poprawności działania zaimplementowanych metod. Projekt jest jedynie wycinkiem realnej bazy, która mogłaby funkcjonować na prawdziwej uczelni.

## Ocena środowiska

Versant pozwala na programowanie obiektowych baz danych w wielu językach: Java, C, C#, C++, Smalltalk, Python. Do implemetacji wybraliśmy język Java. Implementacja w tym jęzku zapewnia spełnienie wszystkich paradygmatów obiektowości: polimorfizmu, enkapsulacji itd.

Sam Versant nie posiada swojego interfejsu przez co administracja bazą jest niewygodna. Problemy z dodaniem pluginu do środowiska Eclipse sprawiły, że Versant nie byłby naszym pierwszym wyborem w przyszłości.

## Tabela do wypełnienia

|  |  |
| --- | --- |
| Jak środowisko realizuje zagadnienia związane z: | Odpowiedź |
| Obiektowością - Metodami | 10/10 |
| Obiektowością – Dziedziczeniem, typami abstrakcyjnymi | 5/10 |
| Obiektowością – Związkami między klasami (asocjacja, kompozycja) | 10/10 |
| Obiektowością – Typy danych – proste, złożone | 10/10 |
| Obiektowością - Polimorfizmem | 10/10 |
| Obiektowością – Tożsamością danych | 10/10 |
| Obiektowością – Enkapsulacją | 7/10 |
| Obiektowością – Trwałością danych | 8/10 |
| Administracja – Zarządzanie środowiskiem | 2/10 |
| Interfejs – Czy narzędzie posiada API? Dla jakich języków? | Tak, Java, C, C#, C++, Smalltalk, Python |
| Środowisko – Czy narzędzie zawiera w sobie środowisko programistyczne? | Nie |
| Skalowalność – Czy narzędzie umożliwia horyzontalne skalowanie środowiska? (rozproszone przetwarzanie, magazynowanie, replikacja) | Tak |
| Multi-model – Czy narzędzie zapewnia inne rodzaje bazy danych? | Nie |