## Projekt 1 Baza studentów – aplikacja trójwarstwowa (grupa 3-osobowa)

Korzystając z JavaFX, JPA oraz własnego protokołu (Socket, RMI) zaimplementuj bazę studentów.

Student to obiekt, posiadający standardowe pola: imię, nazwisko, numer indeksu.

Student posiada również oceny.

Studentowi przydzielana jest dowolna liczba przedmiotów. Z każdego przedmiotu otrzymuje jedną ocenę, albo oceny nie ma.

Przygotuj odpowiednią strukturę obiektową. Zwróć uwagę na to, że oceny brane są z pewnego zbioru. W bazie danych powinna być tabela przedmiotów i tabela ocen.

Zaproponuj i zaimplementuj odpowiednie relacje many-to-one (lub many-to-many) (przykład: <http://www.mkyong.com/hibernate/hibernate-one-to-many-relationship-example/>)

Opracuj listę przypadków użycia:

1. Dodaj/usuń studenta
2. Dodaj/usuń przedmiot
3. Modyfikuj listy przedmiotów i ocen dla wybranego studenta
4. Wyświetl listę przedmiotów z ocenami dla wybranego studenta.
5. Wyświetl listę przedmiotów wybranych chociaż raz przez któregoś ze studentów wraz ze średnią oceną dla wszystkich studentów

Uważaj na zachowanie więzów integralności (JPA przychodzi tu z pomocą)

1. Jeżeli usuwamy przedmiot, to są usuwane wszystkie oceny uzyskane przez studentów dla tego przedmiotu
2. Jeżeli usuwamy studenta - usuwane są wszystkie jego oceny
3. Student na liście ocen nie może mieć dwa lub więcej razy tego samego przedmiotu.

Realizacja ćwiczenia

Celem ćwiczenia jest wykonanie aplikacji składającej się z trzech komponentów (osobnych, działających równocześnie aplikacji):

* klient
* serwer
* baza danych

**Klient** jest interaktywną aplikacją JavaFX. Klient łączy się z serwerem poprzez mechanizm socketów lub RMI i własny protokół, natomiast serwer łączy się z bazą danych poprzez JPA.

Baza danych jest postawiona na MySQL, np. na programie XAMPP. Można też użyć PostreSQL lub SQLite.

W ogólnym założeniu każdy z komponentów może stać na innej maszynie. Ponieważ pracę wykonujemy w zespołach trzyosobowych proszę o zaprezentowanie jej działania w ten sposób, że klient, oraz baza MySQL są na jednej maszynie a serwer na drugiej maszynie. Aby przygotować sprawną prezentację, należy zbadać, czy ruch sieciowy jest przepuszczany i ewentualnie wprowadzić tymczasowe zmiany na zaporze.

Aplikacja serwera – serwer jest aplikacją wielowątkową bazującą na Socketach lub RMI w celu połączenia się z klientem i JPA do połączenia z bazą.

Wymagania dokumentacyjne:   
dobrze zrobiony Javadoc  lub dokumentacja projektowa (UML, PU).

Projekt 2 Baza studentów – aplikacja wykorzystująca SpringBoot   
(grupa 2-osobowa)

Korzystając z frameworka SpringBoot zaimplementuj bazę studentów.

Student to obiekt, posiadający standardowe pola: imię, nazwisko, numer indeksu.

Student posiada również oceny.

Studentowi przydzielana jest dowolną ilość przedmiotów. Z każdego przedmiotu otrzymuje jedną ocenę lub może nie mieć oceny.

Przygotuj odpowiednią strukturę obiektową. Zwróć uwagę na to, że oceny brane są z pewnego zbioru. W bazie danych powinna być tabela przedmiotów i tabela ocen. Przygotowane obiekty zakoduj jako encje JPA.

Zaproponuj i zaimplementuj odpowiednie relacje many-to-one (lub many-to-many) (przykład: <http://www.mkyong.com/hibernate/hibernate-one-to-many-relationship-example/>)

Opracuj listę przypadków użycia:

1. Dodaj / usuń studenta
2. Dodaj / usuń przedmiot
3. Modyfikuj listy przedmiotów i ocen dla wybranego studenta
4. Wyświetl listę przedmiotów z ocenami dla wybranego studenta.
5. Wyświetl listę przedmiotów wybranych chociaż raz przez któregoś ze studentów wraz ze średnią oceną

Uważaj na zachowanie więzów integralności (JPA przychodzi tu z pomocą)

1. Jeżeli usuwamy przedmiot, to są usuwane wszystkie oceny uzyskane przez studentów dla tego przedmiotu
2. Jeżeli usuwamy studenta – usuwane są wszystkie jego oceny
3. Student na liście ocen nie może mieć dwa lub więcej razy tego samego przedmiotu.

Realizacja ćwiczenia

Dla operacji z listy przypadków użycia przygotuj klasę serwisową i kontroler z mapowaniem ścieżek URL na poszczególne metody.

Przygotuj szablony HTML dla operacji. Szablony nie muszą używać wyszukanej grafiki ani interakcji – wystarczy czysty HTML. Jeśli jednak jest to wygodne dla waszego projektu, możecie użyć dowolnego frameworka frontendowego (React, Angular).

Wymagania dokumentacyjne:   
dobrze zrobiony Javadoc  lub dokumentacja projektowa (UML, PU).

**Projekt 3 Lista obecności – aplikacja trójwarstwowa (grupa 3-osobowa)**

Korzystając z JavaFX, JPA oraz własnego protokołu (Socket, RMI) zaimplementuj listę obecności.

Student to obiekt, posiadający standardowe pola: imię, nazwisko, numer indeksu.

Student przydzielony jest do grupy. Grupa składa się ze studentów z utworzonej wcześniej listy.

Opracuj listę przypadków użycia

* Dodaj / usuń studenta
* Dodaj / usuń grupy
* Dodaj / usuń studenta z grupy
* Dodaj termin dla grupy
* Sprawdź obecności (obecny, spóźniony, nieobecny)
* Wyświetl dziennik obecności

Opracuj strukturę obiektową, czyli odpowiednie klasy:

* Student
* Grupa
* Termin
* Obecność

i relacje many-to-one (lub many-to-many) (przykład: <http://www.mkyong.com/hibernate/hibernate-one-to-many-relationship-example/>).

Baza danych powinna zawierać tabelę grup i tabelę studentów.

Realizacja ćwiczenia

Celem ćwiczenia jest wykonanie aplikacji składającej się z trzech komponentów (osobnych, działających równocześnie aplikacji):

* klient
* serwer
* baza danych

**Klient** jest interaktywną aplikacją JavaFX. Klient łączy się z serwerem poprzez mechanizm socketów lub RMI i własny protokół, natomiast serwer łączy się z bazą danych poprzez JPA.

Baza danych jest postawiona na MySQL, np. na programie XAMPP. Można też użyć PostreSQL lub SQLite.

W ogólnym założeniu każdy z komponentów może stać na innej maszynie. Ponieważ pracę wykonujemy w zespołach trzyosobowych proszę o zaprezentowanie jej działania w ten sposób, że klient, oraz baza MySQL są na jednej maszynie a serwer na drugiej maszynie. Aby przygotować sprawną prezentację, należy zbadać, czy ruch sieciowy jest przepuszczany i ewentualnie wprowadzić tymczasowe zmiany na zaporze.

Aplikacja serwera – serwer jest aplikacją wielowątkową bazującą na Socketach lub RMI w celu połączenia się z klientem i JPA do połączenia z bazą.

Wymagania dokumentacyjne:   
dobrze zrobiony Javadoc  lub dokumentacja projektowa (UML, PU).

Projekt 4 Lista obecności – aplikacja wykorzystująca SpringBoot   
(grupa 2-osobowa)

Korzystając z frameworka SpringBoot zaimplementuj listę obecności.

Student to obiekt, posiadający standardowe pola: imię, nazwisko, numer indeksu.

Student przydzielony jest do grupy. Grupa składa się ze studentów z określonej listy.

Przygotuj odpowiednią strukturę obiektową. Zwróć uwagę na to, że lista studentów, których przypisujemy do grup jest wcześniej określona. Baza danych powinna zawierać tabelę grup i tabelę studentów.

Opracuj listę przypadków użycia

* Dodaj / usuń studenta
* Dodaj / usuń grupy
* Dodaj / usuń studenta z grupy
* Dodaj termin dla grupy
* Sprawdź obecności (obecny, spóźniony, nieobecny)
* Wyświetl dziennik obecności

Opracuj strukturę obiektową, czyli odpowiednie klasy:

* Student
* Grupa
* Termin
* Obecność

i relacje many-to-one (lub many-to-many) (przykład: <http://www.mkyong.com/hibernate/hibernate-one-to-many-relationship-example/>).

Przygotowane obiekty zakoduj jako encje JPA.

Realizacja ćwiczenia

Dla operacji z listy przypadków użycia przygotuj klasę serwisową i kontroler z mapowaniem ścieżek URL na poszczególne metody.

Przygotuj szablony HTML dla operacji. Szablony nie muszą używać wyszukanej grafiki ani interakcji – wystarczy czysty HTML. Jeśli jednak jest to wygodne dla waszego projektu, możecie użyć dowolnego frameworka frontendowego (React, Angular).

Wymagania dokumentacyjne:   
dobrze zrobiony Javadoc  lub dokumentacja projektowa (UML, PU).

**Projekt 5 Tablica współdzielona – aplikacja trójwarstwowa  
(grupa 3-osobowa)**

Korzystając z JavaFX, JPA oraz własnego protokołu (Socket, RMI) zaimplementuj tablicę współdzieloną.

Opracuj listę przypadków użycia:

* Rysowanie prostych kształtów
* Pisanie (rysowanie linii)
* Gumka
* Zmiana koloru linii
* Zapis do bazy

Przygotuj odpowiednią strukturę obiektową.

Istnieją użytkownicy (ich lista może być niezmienna). Stan tablicy powinien być na bieżąco zapisywany w bazie. Każdy użytkownik ma swój rysunek, ale tablica może być współdzielona przez wielu użytkowników równocześnie.

Realizacja ćwiczenia

Celem ćwiczenia jest wykonanie aplikacji składającej się z trzech komponentów (osobnych, działających równocześnie aplikacji):

* klient
* serwer
* baza danych

**Klient** jest interaktywną aplikacją JavaFX. Klient łączy się z serwerem poprzez mechanizm socketów lub RMI, natomiast serwer łączy się z bazą danych poprzez JPA.

Baza danych jest postawiona na MySQL, np. na programie XAMPP. Można też użyć PostreSQL lub SQLite.

W ogólnym założeniu każdy z komponentów może stać na innej maszynie. Ponieważ pracę wykonujemy w zespołach trzyosobowych proszę o zaprezentowanie jej działania w ten sposób, że klient, oraz baza MySQL są na jednej maszynie a serwer na drugiej maszynie. Aby przygotować sprawną prezentację, należy zbadać, czy ruch sieciowy jest przepuszczany i ewentualnie wprowadzić tymczasowe zmiany na zaporze.

Aplikacja serwera – serwer jest aplikacją wielowątkową bazującą na Socketach lub RMI w celu połączenia się z klientem i JPA do połączenia z bazą.

Wymagania dokumentacyjne:   
dobrze zrobiony Javadoc  lub dokumentacja projektowa (UML, PU).

## Projekt 6 Tablica współdzielona – aplikacja wykorzystująca SpringBoot (grupa 2-osobowa)

Korzystając z frameworka SpringBoot zaimplementuj tablicę współdzieloną.

Opracuj listę przypadków użycia:

* Rysowanie prostych kształtów
* Pisanie (rysowanie linii)
* Gumka
* Zmiana koloru linii
* Zapis do bazy

Przygotuj odpowiednią strukturę obiektową.

Istnieją użytkownicy (ich lista może być niezmienna). Stan tablicy powinien być na bieżąco zapisywany w bazie. Każdy użytkownik ma swój rysunek, ale tablica może być współdzielona przez wielu użytkowników równocześnie.

Realizacja ćwiczenia

Dla operacji z listy przypadków użycia przygotuj klasę serwisową i kontroler z mapowaniem ścieżek URL na poszczególne metody.

Dla frontendu wykorzystaj dowolne biblioteki JavaScript obsługujące Canvas (interaktywne rysowanie).

Wymagania dokumentacyjne:   
dobrze zrobiony Javadoc  lub dokumnetacja projektowa (UML, PU).

**Projekt 7 Bankomat – aplikacja trójwarstwowa  
(grupa 2-osobowa)**

Korzystając z JavaFX, JPA oraz własnego protokołu (Socket, RMI) zaimplementuj aplikację symulującą działanie bankomatu.

Opracuj listę przypadków użycia:

* Wypłata / wpłata na konto
* Wypłata w EUR
* Sprawdzenie stanu konta
* Wydrukowanie historii operacji
* Zmiana PINu
* Doładowanie telefonu

Przygotuj odpowiednią strukturę obiektową. Baza danych powinna zawierać tabelę kont i tabelę transakcji.

Zaproponuj i zaimplementuj odpowiednie relacje many-to-one (lub many-to-many) (przykład: <http://www.mkyong.com/hibernate/hibernate-one-to-many-relationship-example/>)

Realizacja ćwiczenia

Celem ćwiczenia jest wykonanie aplikacji składającej się z trzech komponentów (osobnych, działających równocześnie aplikacji):

* klient
* serwer
* baza danych

**Klient** jest aplikacją JavaFX, przypominającą wyglądem bankomat. Klient łączy się z serwerem poprzez mechanizm socketów lub RMI, natomiast serwer łączy się z bazą danych poprzez JPA.

Baza danych jest postawiona na MySQL, np. na programie XAMPP. Można też użyć PostreSQL lub SQLite.

W ogólnym założeniu każdy z komponentów może stać na innej maszynie. Ponieważ pracę wykonujemy w zespołach dwuosobowych proszę o zaprezentowanie jej działania w ten sposób, że klient, oraz baza MySQL są na jednej maszynie a serwer na drugiej maszynie. Aby przygotować sprawną prezentację, należy zbadać, czy ruch sieciowy jest przepuszczany i ewentualnie wprowadzić tymczasowe zmiany na zaporze.

Aplikacja serwera – serwer jest aplikacją wielowątkową bazującą na Socketach lub RMI w celu połączenia się z klientem i JPA do połączenia z bazą.

Wymagania dokumentacyjne:   
dobrze zrobiony Javadoc  lub dokumentacja projektowa (UML, PU).