Praca domowa 07 – 15ejb-entity

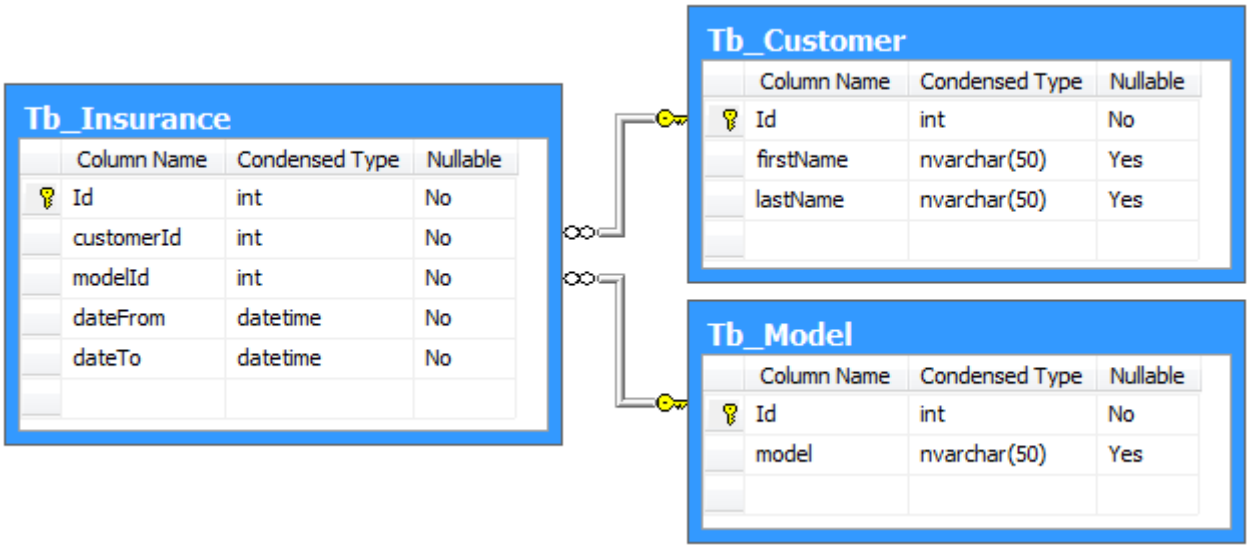
Termin zwrotu : 12 maja godz. 23.00
Zadanie uznaje się za zaliczone, gdy praca oceniona zostanie na co najmniej 6 pkt.

Na serwerze aplikacyjnym Glassfish 4 w kontenerze *ejb* zainstalowany jest pod nazwą *ejb-project* (deployment descriptor) komponent (stateful session bean) o nazwie *GameManager* wraz z interfejsem *IGameRemote*, który zdefiniowany jest następująco :

```
package pl.jrj.game;
import javax.ejb.Remote;

@Remote
public interface IGameRemote {
	public boolean register(int hwork, String album);   // hwork - numer zadania, album – numer albumu studenta
}
```

Metoda *register* dokonuje rejestracji użytkownika w systemie zwracając `true` jeżeli proces rejestracji zakończył się poprawnie. Jeżeli rejestracja zakończyła się niepowodzeniem, metoda *register* zwraca wartość `false`.



W repozytorium danych przechowywane są informacje o okresach ważności polis ubezpieczeniowych samochodów klientów (kolumny *dateFrom, dateTo*). Diagram odpowiedniego fragmentu bazy danych przedstawiono powyżej.

Należy stworzyć aplikację klienta `Client.java` odpowiadającą na pytanie jaki procent zaewidencjonowanych w bazie klientów nie posiada ważnego ubezpieczenia na wskazany model samochodu oraz dzień (np. *Jaki procent klientów nie posiadało wykupionego ubezpieczenia samochodu marki 'Range Rover' w dniu '05/26/2014'*). Poprawnie zaokrąglony wynik należy wyznaczyć w procentach z dokładnością do jednego miejsca dziesiętnego

.

Rozwiązanie musi być oparte (dla potrzeb operowania danymi) o wykorzystanie *entity beans*. Dla potrzeb zadania zaprojektować można zgodną z potrzebami rozwiązania pewną ilość komponentów oraz niezbędnych interfejsów (nie wprowadza się dalszych ograniczeń co do ilości oraz nazewnictwa wykorzystanych komponentów).

Program ma być zapisany w plikach : `Client.java` zawierającym aplikację klienta, `IGameRemote.java` zawierającym definicję interfejsu komponentu `GameManager` oraz pozostałych – wykonanych dla potrzeb zadania – elementów rozwiązania. Poszczególne elementy rozwiązania nie mogą korzystać z bibliotek zewnętrznych innych niż niezbędne moduły serwera (jak np. javaee.jar itp.).

Proces kompilacji musi być możliwy z użyciem komendy

```
javac –cp <app-server-modules> –Xlint Client.java IGameRemote.java *.java
```

Dla potrzeb dostępu do danych wykorzystane zostaną mechanizmy JPA. Niezbędny plik persistence.xml, który będzie używany w procesie testowania zadania (opisujący *persistence context*) podano poniżej :

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
  <persistence xmlns="http://java.sun.com/xml/ns/persistence" version="1.0">
      <persistence-unit name="persistenceNNNNN" transaction-type="JTA">
          <exclude-unlisted-classes>false</exclude-unlisted-classes>
          <jta-data-source>XXXXX</jta-data-source>
      </persistence-unit>
</persistence>
```

gdzie *XXXXX* jest zależne od instalacji (wskazuje na zasób JDBC specyfikujący źródło danych – bazę danych), a *persistenceNNNNN* jest nazwą kontekstu wykorzystywaną dla potrzeb zadania studenta o numerze albumu *NNNNN*.

Uruchomienie programu winno być możliwe z użyciem komendy

```
java Client <file>
```

Parametr `<file>` wskazuje na plik tekstowy zawierający w pierwszej linii nazwę badanej marki samochodu (*model*) a w linii kolejnej datę, o którą pytamy. Program winien akceptować datę zapisaną w formacie zgodnym z ustawieniami systemu operacyjnego (ustawienia

regionalne) komputera, na którym jest uruchamiany. Wynik końcowy (w strumieniu wyjściowym nie powinny pojawiać się jakiekolwiek inne elementy – np. wydruki kontrolne) działania programu musi zawierać pojedynczą liczbę, a więc np.

```
Wynik : 14,2%
```

Wymagania :

- Klasa implementująca aplikację winna zostać zdefiniowane w pliku `Client.java`.
- Interfejs umożliwiający poprawną rejestrację zadania winien zostać zdefiniowany w pliku `IGameMonitor.java`.
- W pliku README.pdf winien być zawarty opis mechanizm operowania danymi oraz algorytm wyznaczania wyniku.
- Proces obliczenia rozwiązania winien się kończyć w czasie nie przekraczającym 1 min (orientacyjnie dla typowego notebooka). Po przekroczeniu limitu czasu zadanie będzie przerywane, i traktowane podobnie jak w sytuacji błędów wykonania (czyli nie podlega dalszej ocenie).

Sposób oceny **:**

- 1 pkt – **Weryfikacja** : czy program jest skompletowany i spakowany zgodnie z ogólnymi zasadami przesyłania zadań.
- 1 pkt – **Kompilacja** : każdy z plików winien być kompilowany bez jakichkolwiek błędów lub ostrzeżeń (w sposób omówiony wyżej)
- 1 pkt – **Wykonanie** : program powinien wykonywać się bez jakichkolwiek błędów i ostrzeżeń (dla pliku danych wejściowych zgodnych z wyżej zamieszczoną specyfikacją) z wykorzystaniem omówionych wyżej parametrów linii komend
- 2 pkt – **README** : plik README.pdf dokumentuje w sposób kompletny i właściwy sposób zestawiania połaczenia
- 1 pkt – **Styl kodowania** : czy funkcji i zmienne posiadają samo-wyjaśniające nazwy ? Czy podział na funkcje ułatwia czytelność i zrozumiałość kodu ? Czy funkcje eliminują (redukują) powtarzające się bloki kodu ? Czy wcięcia, odstępy, wykorzystanie nawiasów itp. (formatowanie kodu) są spójne i sensowne ?
- 4 pkt – **Poprawność algorytmu** : czy algorytm został zaimplementowany poprawnie a wynik odpowiada prawidłowej (określonej zbiorem danych testowej) wartości.