Praca domowa 03 – 15ejbclient

Termin zwrotu : 31 marca 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 (statefull session bean) o nazwie *DataMonitor* implementujący interfejs *IDataMonitor*, który zdefiniowany jest następująco :

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

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

Metoda `register` dokonuje rejestracji użytkownika w systemie zwracając **`true`** jeżeli proces rejestracji zakończył się poprawnie. W przypadku zakończonej sukcesem rejestracji metody `hasNext` oraz `next` (logika działania jak w typowej implementacji iteratora) umożliwiają pobranie ciągu danych wejściowych niezbędnych do dalszych obliczeń. Jeżeli rejestracja zakończyła się niepowodzeniem, metoda `register` zwraca wartość **`false`** oraz `hasNext` zwraca **`false`** (wartość zwracana przez `next` jest nieokreślona gdy `hasNext` przyjmuje wartość **`false`**).

Należy napisać (zaimplementować) program o nazwie *MyClient*. Program winien wykonać udostępnioną przez komponent metodę *register* podając jako parametr właściwy numer zadania (jak w nagłówku dokumentu) oraz numer albumu (własny i poprawny !). Po zakończonej powodzeniem rejestracji program pobiera bliżej nieokreśloną liczbę danych korzystając z metod `hasNext i next`.

Trzy pierwsze liczby zapisane w strumieniu wejściowym (w ciągu danych wejściowych) reprezentują współczynniki pewnej prostej $ax + by + c$ w przestrzeni 3D. Kolejne liczby w strumieniu wejściowym grupowane są w czwórki, z których każda opisuje położenie masy punktowej w przestrzeni 3D. Każda czwórka zawiera współrzędne punktu w przestrzeni (trzy liczby) oraz wartość umieszczonej w tym punkcie masy (czwarta liczba). Należy obliczyć (wyznaczyć) moment bezwładności opisanego danymi wejściowymi układu punktów materialnych względem zdefiniowanej wcześniej osi obrotu $ax + by + c$.

Program ma być zapisany wyłącznie w dwóch plikach : `IDataMonitor.java` zawierającym definicję interfejsu, oraz `MyClient.java` – zawierającym program główny. Program nie może 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 MyClient.java IDataMonitor.java
```

Uruchomienie programu winno być możliwe z użyciem komendy

```
java -cp <app-server-modules> MyClient
```

Wynik końcowy (w strumieniu wyjściowym nie powinny pojawiać się jakiekolwiek inne elementy – np. wydruki kontrolne) działania programu winien być wyznaczony z dokładnością do 5-ciu miejsc dziesiętnych.

Wymagania :

- Klasa implementująca program winna zostać zdefiniowane w pliku `MyClient.java` a niezbędny interfejs w pliku `IDataMonitor.java`.
- W pliku README.pdf winien być zawarty opis mechanizmu wyszukiwania (lookup) i zestawiania połączenia.
- 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
- 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 ?
- 2 pkt – **README** : plik README.pdf dokumentuje w sposób kompletny i właściwy sposób zestawiania połączenia
- 4 pkt – **Poprawność algorytmu** : czy algorytm został zaimplementowany poprawnie a wynik odpowiada prawidłowej (określonej zbiorem danych testowej) wartości.

Przykład prostego rozwiązania (aplikacja kiencka + EJB)**:** http://javahowto.blogspot.com/2007/06/simple-ejb-3-application-hand-made.html

Uwaga : szczególną uwagę należy zwrócić na poprawne zastosowanie przenośnych (portable) nazw JNDI dla beanów EJB. Warto zatem :

1. pomocniczo zaimplementować (samodzielnie wykonany) komponent (*ejb-project/DataMonitor*) oraz manualnie podjąć próbę umieszczenia (deployment) na serwerze aplikacyjnym, a następnie szczegółowo przeglądnąć zapisy logu systemowego serwera aplikacyjnego w celu zrozumienia mechanizmu tworzenia (składni) nazw JNDI.
2. W oparciu o analizę treści realizowanego wcześniej zadania 2 (cube) ustalić właściwą nazwę projektu, w ramach którego kompilowane jest każdorazowo umieszczane na platformie testowej zadanie. Inaczej mówiąc : samodzielnie odpowiedzieć sobie na kluczowe pytanie : w ramach projektu o jakiej nazwie kompilowany będzie klient *MyClient.java* przez platformę na etapie sprawdzania poprawności przesłanego rozwiązania ?