Praca domowa 05 – 15ejb-serv

Termin zwrotu : 21 kwietnia 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 (stateless session bean) o nazwie *GameMonitor* wraz z interfejsem *IGameMonitor*, który zdefiniowany jest następująco :

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

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

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.`**

Należy napisać (zaimplementować) grę o nazwie *MasterMind.* Krótki opis gry (wystarczający dla potrzeb niniejszego zadania) można znaleźć np. pod adresem http://www.zagrajsam.pl/dzialy_gier.php?gra=3.

Program winien implementować poprawne zachowanie się gracza przy założeniu, że rozgrywka odbywa się z użyciem 8-miu kolorów w 5-ciu kolumnach. Dla potrzeb zadania kolory oznaczane są dużymi literami A, B, C … H. Kolory mogą się powtarzać, co znaczy, że w dwóch lub większej ilości kolumn możliwe jest ustawienie pionów tego samego koloru.

Metoda *initGame* umożliwia zainicjowanie gry. Wywołanie metody *initGame(”CEBGH”)* ustala poszukiwane przez gracza w kolejnych ruchach początkowe ustawienie pionów (inicjuje grę).

Po podjęciu decyzji o ustawienie pionów dla potrzeb kolejnego ruchu sprawdzamy stan gry wołając metodę *verify*. Wywołanie metody *verify* z argumentem określającym ustawienie kompletu pionów w bieżącym ruchu gracza (np. *verify(”ADEGH”)* zwraca dwucyfrową informację (w postaci znakowej) : pierwszy znak (pierwsza cyfra) określa ilość pionów o właściwym kolorze ustawionych we właściwej kolumnie, drugi - ilość pionów o właściwym kolorze lecz ustawionych w niewłaściwej kolumnie. Np. odpowiedź *”03”* wskazuje, że bieżące ustawienie zawiera wyłącznie trzy piony o właściwym kolorze, żaden z nich nie został jednak umieszczony we właściwej kolumnie. Program kończy działanie gdy w kolejnym ruchu uzyska odpowiedź *”50”*.

Algorytm rozgrywki należy zaimplementować w postaci komponentu EJB o nazwie *MasterMind* wraz z niezbędnym interfejsem *IMasterMind* udostępniającym pojedynczą bezparametrową metodę realizującą proces rozgrywki i zwracająca informację o ilości wykonanych kroków.

Sterowanie procesu rozgrywki winno być zaimplementowane w postaci servletu *MGame*. Servlet otrzymuje przekazywaną w żądaniu (url) jako parametr *m* informację o nieznanym graczowi początkowym ustawieniu pionów (np. m=CEBGH). Dokonuje zainicjowania rozgrywki przekazując ustawienie początkowe do komponentu menadżera gry (metoda *initGame* udostępniona interfejsem *IGameMonitor* komponentu *GameMonitor*). Korzystając z wykonanej i udostępnionej (interfejs *IMasterMind*) metody komponentu *MasterMind* realizującej proces rozgrywki servlet ustala i wyprowadza prawidłowo obliczoną ilość wykonanych kroków (ruchów) algorytmu gry.

Program ma być zapisany w czterech plikach : `IGameMonitor.java` zawierającym definicję interfejsu komponentu `GameMonitor`, `IMasterMind.java` zawierającym definicję interfejsu komponentu zdefiniowanego w pliku `MasterMind.java` , oraz kod servletu `MGame.java`. Poszczególne elementy rozwiązania nie mogą korzystać z bibliotek zewnętrznych innych niż niezbędne moduły serwera (jak np. gf-client.jar, javaee.jar itp.).

Proces kompilacji musi być możliwy z użyciem komendy

```
javac –cp <app-server-modules> –Xlint MGame.java IMasterMind.java IGameMonitor.java MasterMind.java
```

Rozwiązanie testowane będzie w środowisku serwera aplikacyjnego GlassFish 4. Zawartość pliku web.xml, który używany będzie trakcie uruchamiania i testowania komponentu podano niżej :

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns="http://java.sun.com/xml/ns/javaee"
       xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
       xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
       id="WebApp_ID" version="3.0">

  <servlet>
    <servlet-name>ServletNNNNN</servlet-name>
    <servlet-class>Cube</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>ServletNNNNN</servlet-name>
    <url-pattern>/NNNNN/*</url-pattern>
  </servlet-mapping>
</web-app>
```

gdzie NNNNN oznacza numer albumu studenta, którym sygnowana jest praca.

Wymagania :

- Klasa implementująca komponent prowadzący rozgrywkę winna zostać zdefiniowane w pliku `MasterMind.java`.
- Interfejs udostępniający metodę realizująca algorytm gry implementowanej w komponencie `MasterMind.java` winien zostać zdefiniowany w pliku `IMasterMind.java`.
- Servlet nadzorujący proces rozgrywki zapisać należy w pliku `MGame.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
- 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. Autorzy poprawnie działających programów (zwracające właściwe rozwiązanie), które dla zadanego ciągiem testowym zestawu danych znalazły rozwiązanie w najmniejszej ilości kroków otrzymają 4 pkt. Ocena punktowa będzie maleć liniowo wraz ze wzrostem ilości kroków wykonanych przez algorytm ponad niezbędne minimum (przykładowo : rozwiązanie dla pewnego ciągu wejściowego możliwe jest w 6 krokach; autorzy prac, którzy znaleźli rozwiązanie w 6 krokach otrzymają 4 pkt; jeżeli zaimplementowany przez nich algorytm znajdzie rozwiązanie w 7 krokach otrzymają 3 pkt itd.).