Praca domowa 03 – plane

Termin zwrotu : 28 marca godz. 23.00
Zadanie uznaje się za zaliczone, gdy praca oceniona zostanie na co najmniej 6 pkt.

Wartości pewnej funkcji dwóch zmiennych z = f(x, y), gdzie x, y, z przyjmują wartości zmiennoprzecinkowe, przechowywane są w SQL-owym repozytorium danych (w bazie danych) w tabeli o nazwie Otable. Struktura tabeli utworzona została z wykorzystaniem instrukcji

```
CREATE TABLE Otable (
   id int NOT NULL,
   x float NOT NULL,
   y float NOT NULL,
   z float NOT NULL
)
CONSTRAINT [PK_Otable] PRIMARY KEY
(
   id
)
```

Połączenie do SQL-owej bazy danych (dostęp do bazy) realizowany jest z wykorzystaniem driverów JDBC poprzez wykonanie metody

```
String database = <db>;  // gdzie <db> przekazywany jako parametr URL
Connection conn = DriverManager.getConnection(database);
```

Zbiór punktów (x, y) definiuje obszar T nad którym rozpostarta jest funkcja f.

Należy obliczyć (wyznaczyć) powierzchnię największego obszaru płaskiego O. Przez ‘obszar płaski’ O rozumiemy pewien podobszar obszaru T w którym dla każdej pary x i y leżącej wewnątrz lub na brzegu obszaru O wartość funkcji jest stała (z = const). Z oczywistych względów obszar T zawierać może 0 lub więcej obszarów płaskich. Pole obszaru płaskiego to pole największego wielokąta wypukłego rozpiętego nad zbiorem punktów (x,y) należących do obszaru O.

Algorytm należy zaimplementować z wykorzystaniem technologii servletów jako komponent o nazwie *Plane*. Servlet otrzymuje jako dane wejściowe parametr o nazwie *db*, który przekazywany jest w żądaniu (url). Odpowiedź na żądanie GET oraz POST protokołu http zawiera wyznaczoną przez komponent wartość powierzchni szukanego obszaru z dokładnością do 5 miejsc dziesiętnych, a więc np.

```
Maksimum : 456.93172
```

Proces kompilacji (w środowisku serwera aplikacyjnego Tomcat 7.0) musi być możliwy z użyciem komendy

```
javac –extdirs <path-to-appserver>/lib –Xlint Plane.java
```

Uruchomienie programu w środowisku serwera aplikacyjnego musi być możliwe wyłącznie z wykorzystaniem dwóch plików:

```
WEB-INF/classes/Plane.class
WEB-INF/web.xml
```

Zawartość pliku web.xml, który używany będzie w 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>servletNNNNNN</servlet-name>
    <servlet-class>Plane</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>servletNNNNNN</servlet-name>
    <url-pattern>/*</url-pattern>
  </servlet-mapping>
</web-app>
```

Wymagania :

- Klasa implementująca komponent winna zostać zdefiniowane w pliku `Plane.java`
- Należy zwrócić uwagę, że parametry url'a zawierać mogą napisy złożone z liter cyfr oraz znaków specjalnych, które kodowane będą z wykorzystaniem UTF-8.
- W pliku README.pdf winien być zawarty szczegółowy opis organizacji struktur danych oraz szczegółowy opis zastosowanego algorytmu obliczeniowego.
- 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 struktury danych, oraz opis przyjętej koncepcji algorytmu
- 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.