Praca domowa 2 – cone

Termin zwrotu : 27 października godz. 23.00
Zadanie uznaje się za zaliczone, gdy praca oceniona zostanie na co najmniej 6 pkt.

Należy stworzyć (zaimplementować) z wykorzystaniem technologii servletów komponent (servlet) o nazwie *Cone*. Servlet otrzymuje jako dane wejściowe trzy parametry o nazwach *ra, rb* oraz *h*, które przekazywane są w żądaniu (url). Parametry (przyjmujące wartość rzeczywistą) określają odpowiednio promień podstawy *ra*, wysokość stożka ściętego *h* oraz promień przekroju *rb* na wysokości *h*. Środek podstawy stożka umieszczony jest w punkcie (0,0) płaszczyzny (x,y) a oś symetrii stożka pokrywa się z dodatnią półosią współrzędnej z. Wymiary bryły podane są w metrach [m]. Bryła zbudowana jest z materiału o gęstości właściwej *c* wyrażonej w [kg]/[m^3]. W materiale występują defekty, które mają charakter przestrzeni o kształcie kulistym i gęstości materiału *g*.

Wyniki badania struktury materiału zawierające opis defektów – z których każdy zapisywany jest w postaci równania kuli o środku w punkcie *(x, y, z)* oraz promieniu *r* przekazywane są do servletu w postaci kolejnych żądań typu GET. Kolejność parametrów *x, y, z* i *r* w żądaniu GET jest dowolna. Otrzymanie przez servlet żądania typu POST z parametrami : *ra, rb* oraz *h* określającymi rozmiary stożka, *c* charakteryzującym gęstość materiału, z którego kostka została wykonana oraz *g* charakteryzującym gęstość materiału w obszarach defektów oznacza, że przekazano komplet danych.

Otrzymanie przez servlet żądania typu POST oznacza przejście do fazy analizy danych i wyznaczania wyniku. Żądanie POST protokołu http winno zwrócić wyznaczoną przez komponent masę rzeczywistą analizowanego bloku materiału. Dla rozwiązania zadania wykorzystać należy metodę Monte Carlo, przy czym otrzymany wynik wyznaczony winien być z dokładnością nie mniejszą niż 10^{-2}.

Proces kompilacji w środowisku serwera aplikacyjnego musi być możliwy z użyciem komendy

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

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

```
NNNNN/WEB-INF/classes/Cone*.class
NNNNN/WEB-INF/web.xml
```

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

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>servletNNNNN</servlet-name>
    <servlet-class>Cone</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>servletNNNNN</servlet-name>
    <url-pattern>/*</url-pattern>
  </servlet-mapping>
</web-app>
```

Program ma być zapisany wyłącznie w pojedynczym pliku `Cone.java` zawierającym kod servletu implementującego mechanizm poszukiwania rozwiązania. Program nie może korzystać z jakichkolwiek bibliotek zewnętrznych.

Wynik działania programu winien być poprawnie zaokrągloną i sformatowaną liczbą o wartości równej poszukiwanej masie bryły – nie ma potrzeby formatowania i zwracania kompletnej strony html.

Wymagania :

- Klasa implementująca problem winna zostać zdefiniowana w pliku `Cone.java`
- W pliku README.pdf winien być zawarty szczegółowy opis organizacji struktur danych oraz szczegółowy opis zastosowanego mechanizmu (metody) poszukiwania rozwiązania z opisem mechanizmu zapewnienia osiągnięcia zakładanej dokładności wyznaczania wyniku.
- Proces poszukiwania rozwiązania dla zestawu danych o ilości defektów rzędu 10^3 winien się kończyć w czasie nie przekraczającym 3 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.