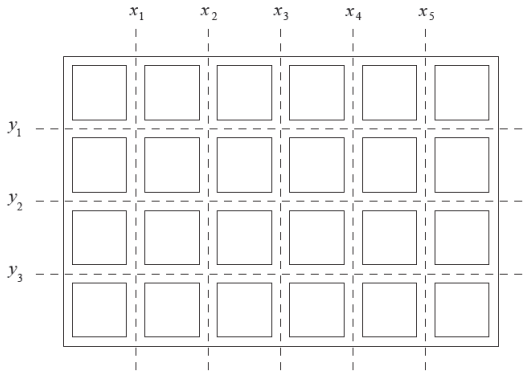
Praca domowa 10 – plate

Termin zwrotu : 03 czerwca godz. 23.00

Zadanie uznaje się za zaliczone, gdy praca oceniona zostanie na co najmniej 6 pkt.

Dana jest tafla niejednorodnego materiału. Taflę należy pociąć na $n * m$ pojedynczych elementów ułożonych w n wierszy i m kolumn. Cięcia możemy dokonywać wzdłuż pionowych i poziomych linii (zaznaczonych na rysunku liniami przerywanymi). Jedno przecięcie tafli wzdłuż wybranej pionowej lub poziomej linii dzieli ten kawałek na dwa mniejsze. Każde cięcie tafli jest obarczone pewnym kosztem wyrażającym się dodatnią liczbą rzeczywistą. Koszt ten nie zależy od długości cięcia, a jedynie od linii wzdłuż której tniemy. Oznaczmy koszty cięcia kolejnych pionowych linii przez $x_1, x_2, \ldots, x_{m-1}$, a wzdłuż poziomych linii przez $y_1, y_2, \ldots, y_{n-1}$. Koszt pocięcia całej tafli na pojedyncze elementy to suma kosztów kolejnych cięć. Należy obliczyć minimalny koszt pocięcia całej tafli na pojedyncze elementy.



Przykładowo, jeżeli potniemy taflę przedstawioną na rysunku, najpierw wzdłuż linii poziomych, a następnie każdy z otrzymanych kawałków wzdłuż linii pionowych, to koszt takiego pocięcia wyniesie y1 +y2 +y3 + 4 * (x1 +x2 +x3 +x4 +x5).

Proces obliczeniowy realizowany jest przez komponent o nazwie `SqPlate` implementujący usługę RESTful'ową inicjowaną z wykorzystaniem wywołania postaci

```
/plate/<datasource>/<table>
```

gdzie `<datasource>` określa zdefiniowane dla potrzeb serwera Glassfish źródło danych (javax.sql.Datasource) niezbędne do komunikacji z SQL-ową bazą danych zawierającą dane ($x_1, x_2, \ldots, x_{m-1}, y_1, y_2, \ldots, y_{n-1}$), a `<table>` nazwę tabeli w której zapisano dane. Struktura tabeli utworzona została z wykorzystaniem instrukcji

```
CREATE TABLE <name> (
   id int NOT NULL,
   x float NOT NULL,
   y float NOT NULL,
)
CONSTRAINT [PK_Table] PRIMARY KEY ( id )
```

Ilość wierszy tabeli nie jest znana. Wiadomo jednak, że tabela zawiera dokładnie *m-1* wierszy z wartością kolumny *x* większą od zera. Analogicznie dokładnie *n-1* wierszy zawiera w kolumnie *y* wartość większą od zera. Porządek poszczególnych ciągów ($x_1, x_2, \ldots, x_{m-1}$) oraz ($y_1, y_2, \ldots, y_{n-1}$) wyznaczony jest jednoznacznie poprzez klucz pierwotny tabeli. Rozwiązanie musi być oparte (dla potrzeb operowania danymi) o wykorzystanie *entity beans*.

Program winien być zapisany w plikach `SqPlate.java` zawierającym implementację usługi sieciowej oraz `MyPlate.java` zawierającym implementację algorytmu podziału oraz ewentualnych innych niezbędnych dla realizacji zadania. Poszczególne elementy rozwiązania nie mogą korzystać z bibliotek zewnętrznych innych niż niezbędne moduły serwera (jak np. gf-client.jar, javaee.jar itp.). Działanie rozwiązania nie może być zależne od jakiegokolwiek dialektu SQL.

Proces kompilacji programu musi być możliwy z użyciem komendy

```
javac -Xlint SqPlate.java MyPlate.java *.java
```

Przykładowy wynik końcowy (w strumieniu wyjściowym nie powinny pojawiać się jakiekolwiek inne elementy – np. wydruki kontrolne) :

```
Koszt cięcia : 15.75000
```

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*.

Zawartość pliku web.xml, który używany będzie w trakcie uruchamiania i testowania usługi 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>javax.ws.rs.core.Application</servlet-name>
  </servlet>
  <servlet-mapping>
    <servlet-name>javax.ws.rs.core.Application</servlet-name>
    <url-pattern>/*</url-pattern>
  </servlet-mapping>
</web-app>
```

Wymagania :

- Klasa implementująca elementy programu głównego winna zostać zdefiniowana w pliku `SqPlate.java`
- Klasa implementująca mechanizm podziału winna być zdefiniowana w pliku `MyPlate.java`
- W pliku README.pdf winien być zawarty opis organizacji struktur danych, szczegółowy opis algorytmu minimalizacji kosztu oraz analiza/dyskusja złożoności obliczeniowej zaproponowanego rozwiązania.

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 zarówno struktury danych, strategię poszukiwania rozwiązania (algorytm) oraz szacowaną złożoność obliczeniową przyjętego rozwiązania.
- 1 pkt – **Styl kodowania** : Styl oceniany jest w oparciu o wskaźniki NCSS oraz CCN (patrz dokument *'Zasady oznaczania i przesyłania prac'*) oraz dodatkowo z uwzględnieniem następujących elementów : Czy funkcje 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 rozwiązanie problemu zostało zaprojektowane i zaimplementowane poprawnie, przy czym za merytorycznie poprawny algorytm można uzyskać dwa punkty, a dwa kolejne za te wyróżniające elementy/cechy rozwiązania (trafność doboru struktury danych, efektywność algorytmu itp.).