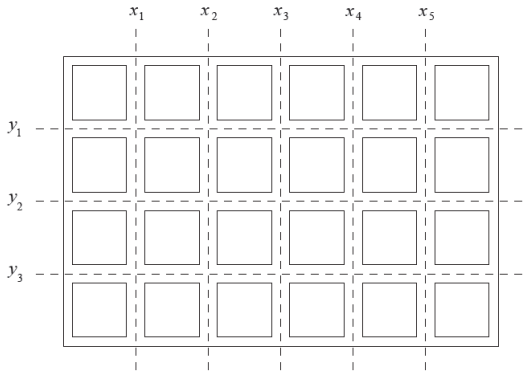
Praca domowa 07 – plate

Termin zwrotu : 11 grudnia 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).

W oparciu o technologię EJB należy zbudować komponent `Plate` organizujący niezbędne struktury danych oraz implementujący metody potrzebne do tworzenia oraz operowania na tych danych. Bezpośrednie odwołania do elementów struktury są niedozwolone. Dostęp winien odbywać się wyłącznie z użyciem odpowiednio zaimplementowanych metod. Zaproponowane metody winny być udostępnione z wykorzystaniem interfejsu `IPlateRemote`.

Dane do obliczeń przechowywane są w zewnętrznym repozytorium przy czym `<datasource>` określa zdefiniowane dla potrzeb serwera Glassfish 4 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 odpowiedniej 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.

Należy zbudować komponent `Cost` implementujący algorytm obliczania kosztów operacji cięcia. Metoda służąca obliczaniu kosztu winna być udostępnione z wykorzystaniem interfejsu `ICostRemote`. Ewentualne wszelkie inne operacje związane z operowaniem strukturami danych realizowane być mogą wyłącznie w oparciu o metody udostępnione interfejsem `IPlateRemote`.

Należy napisać program klienta `AppClient` umożliwiający wczytanie danych z SQL'owego repozytorium (wskazywanego przez parametry `<datasource>` oraz `<table>` programu). Aplikacja – korzystając z operacji udostępnionych interfejsem `IPlateRemote` - winna utworzyć właściwą strukturę danych, a następnie korzystając z udostępnionej interfejsem `ICostRemote` metody wyznaczyć koszt operacji cięcia.

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 –cp <app-server-modules> –Xlint AppClient.java Plate.java IPlateRemote.java  \
                                      Cost.java ICostRemote.java
```

Uruchomienie programu winno być możliwe z użyciem komendy

```
java AppClient <datasource> <table>
```

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

```
Koszt cięcia : 15.75000
```

Rozwiązanie testowane będzie w środowisku serwera aplikacyjnego GlassFish 4.

Wymagania :

- Komponent umożliwiający tworzenie i operowanie na niezbędnych strukturach danych winien być umieszczony w pliku `Plate.java`.
- Interfejs udostępniający zaprojektowane metody realizowane przez komponent `Plate.java` winien zostać zdefiniowany w pliku `IPlateRemote.java`.
- Algorytm umożliwiający wyznaczanie ilości kosztu operacji cięcia winien być umieszczony w pliku `Cost.java`.
- Interfejs udostępniający zaprojektowane metody realizowane przez komponent `Cost.java` winien zostać zdefiniowany w pliku `ICostRemote.java`.
- Aplikacja klienta wprowadzająca dane oraz organizująca proces obliczeń winna być umieszczona w pliku `AppClient.java`
- W pliku README.pdf winien być zawarty szczegółowy opis zastosowanych do przechowywania struktur danych, proponowanych metod dostępu/przeszukiwania oraz algorytmu wyznaczania kosztu operacji.
- 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 – **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 – **Komentarze wewnętrzne** : czy program jest skomentowany w sposób zapewniający zrozumienie jego działania, oraz wyjaśniający warunki, które muszą zachodzić przed i po wykonaniu każdej z funkcji.
- 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 elementy/cechy rozwiązania (trafność doboru struktury danych, efektywność algorytmu itp.).