Zaawansowane Techniki Programowania
Praca domowa 4
Dominik Freicher
17.11.2016

OPIS ZADANIA

Celem zadania jest wyznaczenie objętości graniastosłupa prostego o podstawie P oraz wysokości h. Podstawę P graniastosłupa stanowi wypukła otoczka rzutu punktów (x,y) na płaszczyznę XY. Wysokość h graniastosłupa wyznaczona jest jako średnia wartość funkcji *z = f(x,y)* określonej nad zadanym zbiorem punktów (x,y). Algorytm obliczania objętości bryły należy zaimplementować w postaci komponentu EJB o nazwie *Block* wraz z niezbędnym interfejsem o nazwie *IBlockRemote* udostępniającym metodę realizującą proces obliczeń i zwracającą wyznaczoną przez komponent wartość objętości bryły z dokładnością do 5 miejsc dziesiętnych

OPIS MECHANIZMU WYSZUKIWANIA

Do zlokalizowania klasy DSManager oraz nawiązania z nią połączenie wykorzystany został interfejs JNDI (ang. Java Naming and Directory Interface). Pozwala on klientom na odkrywanie oraz wyszukiwania danych i obiektów na podstawie nazw.

Aby nawiązać połączenie należy utworzyć obiekt InitialContext:

```
InitialContext ctx = new InitialContext();
```

Następnie należy skorzystać z metody lookup, której implementacja umożliwia odnalezienie zasobu po nazwie:

```
IDataMonitor obj = (IDataMonitor)ctx.lookup("java:global/ejb-
project/DSManager!pl.jrj.dsm.IDSManagerRemote");
```

Ścieżka przestrzeni nazw ma postać:

```
java:global/ejb-project/DSManager!pl.jrj.dsm.IDSManagerRemote
```

Aby nawiązać połączenie ze źródłem danych utworzony został również obiekt InitialContext oraz wykorzystana została metoda lookup:

```
InitialContext context = new InitialContext();

DataSource ds = (DataSource) context.lookup(path);

Connection connection = ds.getConnection();
```

Zmienna `path` zawiera nazwę źródła danych pod którą źródło zarejestrowane zostało w usłudze JNDI.

Zaawansowane Techniki Programowania
Praca domowa 4
Dominik Freicher
17.11.2016

ZASTOSOWANY ALGORYTM

W celu obliczenia objętości graniastosłupa należy wyznaczyć jego pole podstawy oraz wysokość. Wzór na objętość prezentuje się następująco:

$$V = Pp * h$$

gdzie Pp – pole podstawy, h – wysokość

Wysokość h graniastosłupa wyznaczona jest jako średnia wartość funkcji z = f(x,y) określonej nad zadanym zbiorem punktów (x,y).

Aby wyznaczyć pole podstawy graniastosłupa, które jak wiemy stanowi wypukła otoczka rzutu punktów (x,y) na płaszczyznę XY można skorzystać z algorytmu Grahama. Aby wyznaczyć pole powierzchni podstawy należy wykonać następujące kroki:

1. Wybrać skrajnie dolny lewy punkt S, który na pewno należy do otoczki.

2. Dla każdego pozostałego punktu *Pi* wyznaczyć odległość od punktu *S* oraz kąt pomiędzy wektorem *Opi* a dodatnią osią układu współrzędnych.

3. Punkty posortować według kąta oraz odległości.

4. Powtarzać następujące kroki dla punktów, zaczynając od S:

    1. Pobrać trzy kolejne punkty A, B, C.
    2. Jeżeli punkt B leży na zewnątrz trójkąta AOC, to należy on do otoczki wypukłej. Przejść do następnego punktu.
    3. Jeżeli punkt B leży wewnątrz trójkąta AOC, to znaczy, że nie należy do otoczki.

5. Po wyznaczeniu otoczki należy wyznaczyć pole wielokąta jako sumę pól trapezów wyznaczonych przez kolejne boki wielokąta rzutowane na oś Y. Jeżeli wektor wielokąta jest skierowany zgodnie z osią Y to pole powierzchni należy dodać, jeżeli nie to pole powierzchni należy odjąć.

Po wyznaczeniu wysokości oraz pola podstawy należy skorzystać ze wzoru na objętość graniastosłupa, który przedstawiony został na wstępie opisu algorytmu.

STRUKTURY DANYCH

W skład projektu wchodzą klasy `Solver.java` `Block.java` oraz interfejsy `IBlockRemote.java` `IDSManagerRemote.java`. Poniżej przedstawione zostały najważniejsze struktury:

Klasa **Solver.java**

```
void doGet(HttpServletRequest req, HttpServletResponse res)
String getTableName(HttpServletRequest request)
IDSManagerRemote getDSManager()
Connection initDatabaseConnection(final String path)
void fetchingData(Connection connection, String tableName)
void printResult(PrintWriter writer)
```

Klasa Solver realizuje funkcję Servletu, w metodzie doGet zaimplementowana została główna logika. Metoda getTableName zwraca nazwę tabeli otrzymaną z parametru żądania url. Metoda getDSManager odpowiada za pobranie Beana. Następnie następuje wywołanie metody getDS, która zwraca ścieżkę źródła danych. Metoda initDatabaseConnection odpowiada za nawiązanie połączenia ze źródłem danych, natomiast metoda fetchingData odpowiedzialna jest za zwrócenie danych ze źródła. Metoda printResult odpowiada za zwrócenie rezultatu.

Klasa **Block.java**

```
double calculateVolumeOfPrism(List<double[]> points);
void calculateHeightOfPrism()
void calculateSurfaceAreaOfPrism()
void prepareAxisPoints()
void sortAxisPoints(AxisPoint startPoint, AxisPoint neighborPoint)
```

Klasa Block odpowiedzialna jest za realizacje głównego algorytmu oraz zwrócenie wyniku obliczeń za pośrednictwem metody calculateVolumeOfPrism. Metody prywatne kolejno odpowiedzialne są za wyznaczenie wysokości oraz pola podstawy.

Interfejs **IBlockRemote.java**

```
double calculateVolumeOfPrism(List<double[]> points);
```

Interfejs IBlockRemote udostępnia metodę zwracającą wyniki obliczeń.

Interfejs **IDSManagerRemote.java**

```
String getDS();
```

Interfejs IDSManagerRemote udostępnia metodę getDS zwracającą ścieżkę źródła danych.

URUCHOMIENIE

Proces kompilacji jest możliwy z użyciem komendy:

```
javac -cp <app-server-modules> -Xlint Solver.java IBlockRemote.java
Block.java IDSManagerRemote.java
```

Uruchomienie programu możliwe jest w środowisku serwera aplikacyjnego GlassFish 4.