**Laporan Praktikum Algoritma & Struktur Data**

**Jobsheet 15 – Graph**

****

Nama: Aqil Rahmat Alifiandi

NIM: 2341760099

Prodi: D-IV Sistem Informasi Bisnis

**JURUSAN TEKNOLOGI INFORMASI**

**POLITEKNIK NEGERI MALANG**

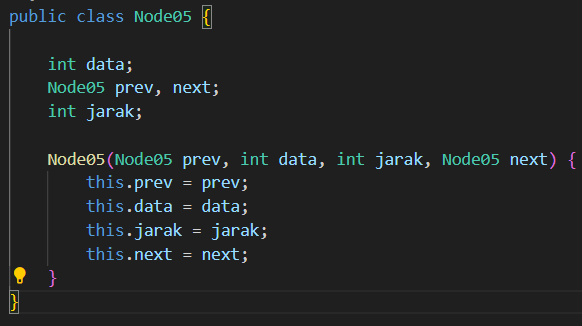
**2023/2024**

**PERCOBAAN 1**

Buka text editor. Buat class Node.java dan class DoubleLinkedList.java sesuai dengan praktikum Double Linked List.

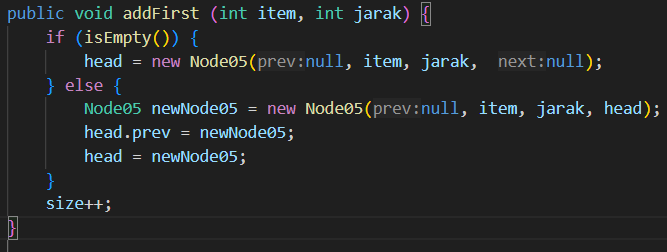
1. **Class Node**

Kode program yang terdapat pada class Node belum dapat mengakomodasi kebutuhan pembuatan graf berbobot, sehingga diperlukan sedikit modifikasi. Setelah Anda menyalin kode program dari class Node pada praktikum Double Linked List, tambahkan atribut jarak bertipe int untuk menyimpan bobot graf

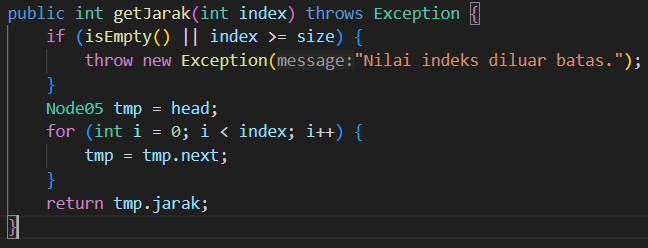
****

1. **Class DoubleLinkedList**

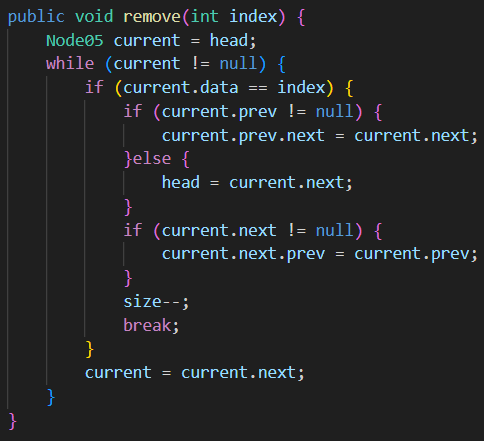
Setelah Anda menyalin kode program dari class DoubleLinkedList pada praktikum Double Linked List, lakukan modifikasi pada method addFirst agar dapat menerima parameter jarak dan digunakan saat instansiasi Node

****

Selanjutnya buat method getJarak (hampir sama seperti method get) yang digunakan untuk mendapatkan nilai jarak edge antara dua node.

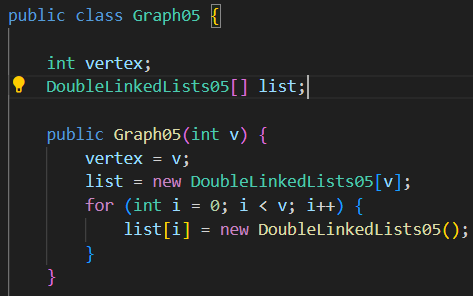


Modifikasi method remove agar dapat melakukan penghapusan edge sesuai dengan node asal dan tujuan pada graf. Pada praktikum Double Linked List, parameter index digunakan untuk menghapus data sesuai posisi pada indeks tertentu, sedangkan pada Graf ini, penghapusan didasarkan pada data node tujuan, sehingga modifikasi kode diperlukan untuk menghindari index out of bound.



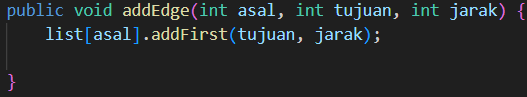
1. **Class Graph**

Lengkapi class Graph dengan atribut yang telah digambarkan di dalam pada class diagram, yang terdiri dari atribut vertex dan DoubleLinkedList dan Tambahkan konstruktor default untuk menginisialisasi variabel vertex dan menambahkan perulangan jumlah vertex sesuai dengan panjang array yang telah ditentukan.



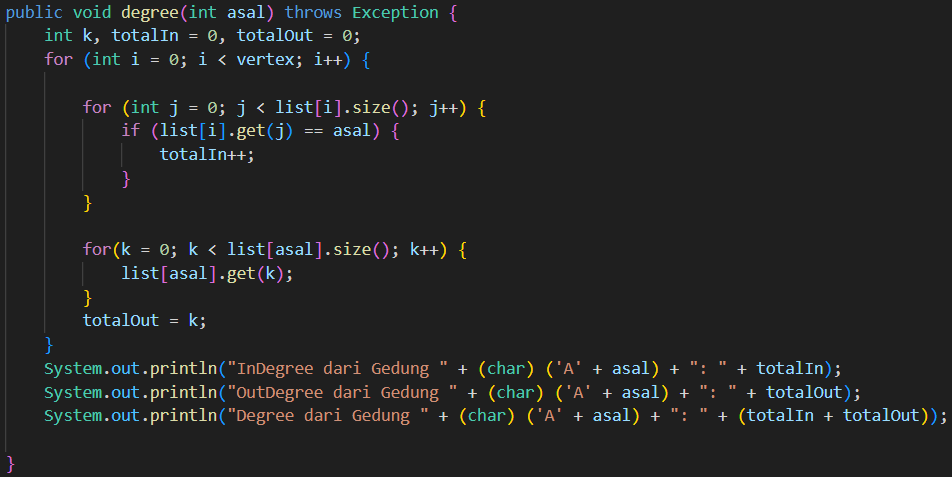
Tambahkan method addEdge() untuk menghubungkan dua node. Baris kode program

berikut digunakan untuk graf berarah (directed).



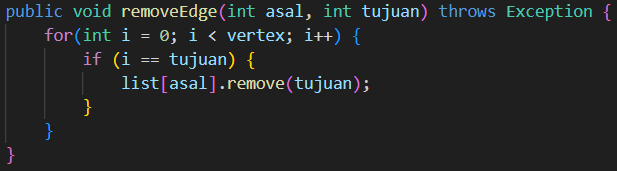
Tambahkan method degree() untuk menampilkan jumlah derajat lintasan pada setiap

vertex. Kode program berikut digunakan untuk menghitung degree pada graf berarah

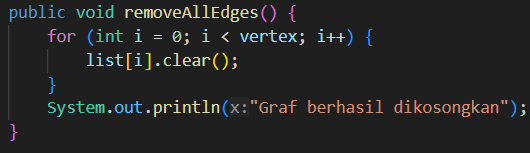


Tambahkan method removeEdge() untuk menghapus lintasan pada suatu graph.

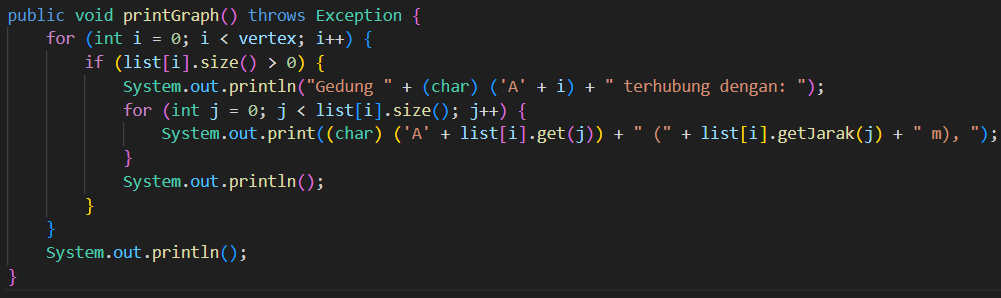
Penghapusan membutuhkan 2 parameter yaitu node asal dan tujuan.

****

Tambahkan method removeAllEdges() untuk menghapus semua vertex yang ada di dalam graf.



Tambahkan method printGraph() untuk mencetak graf.



1. **Class Utama**

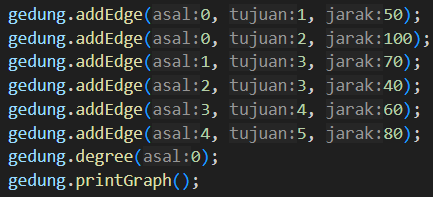
Di dalam fungsi main, lakukan instansiasi object Graph bernama gedung dengan nilai

parameternya adalah 6.

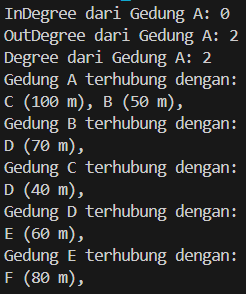


Tambahkan beberapa edge pada graf, tampilkan degree salah satu node, kemudian

tampilkan grafnya.



Hasil Run:

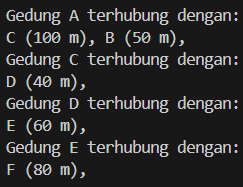


Tambahkan pemanggilan method removeEdge(), kemudian tampilkan kembali graf

tersebut.



Hasil Run:



**PERTANYAAN**

1. Perbaiki kode program Anda apabila terdapat error atau hasil kompilasi kode tidak sesuai!

Jawaban: Terjadi error pada class DoubleLinkedList dan tepatnya pada method remove, perbaikan dengan menambahkan “size—”



1. Pada class Graph, terdapat atribut list[] bertipe DoubleLinkedList. Sebutkan tujuan pembuatan variabel tersebut!

Jawaban: untuk merepresentasikan graf sebagai kumpulan dari beberapa list.

1. Jelaskan alur kerja dari method removeEdge!

Jawaban:

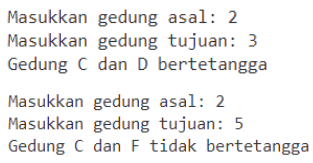
* Metode removeEdge menerima dua parameter, asal dan tujuan, yang mewakili dua simpul (vertex) dalam graf.
* Looping melalui semua simpul di graf menggunakan indeks i dari 0 hingga jumlah vertex (vertex).
* Jika indeks i sama dengan tujuan, maka metode akan mencoba menghapus simpul tujuan dari daftar simpul yang terhubung dengan simpul asal.

1. Apakah alasan pemanggilan method addFirst() untuk menambahkan data, bukan method add jenis lain saat digunakan pada method addEdge pada class Graph?

Jawaban: Pemanggilan addFirst dalam metode addEdge dipilih karena alasan efisiensi, kemudahan implementasi, dan konsistensi dalam pengelolaan adjacency list. Dengan addFirst, penambahan edge dilakukan dalam waktu konstan, menjaga performa graf tetap optimal

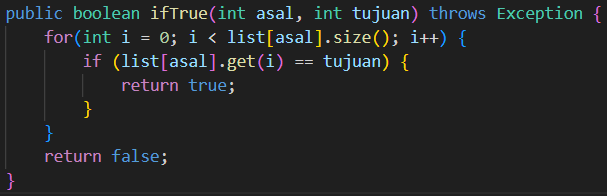
tanpa perlu traversal yang tidak perlu atau penanganan urutan khusus.

1. Modifikasi kode program sehingga dapat dilakukan pengecekan apakah terdapat jalur antara suatu node dengan node lainnya, seperti contoh berikut (Anda dapat memanfaatkan Scanner)



**Jawaban:**

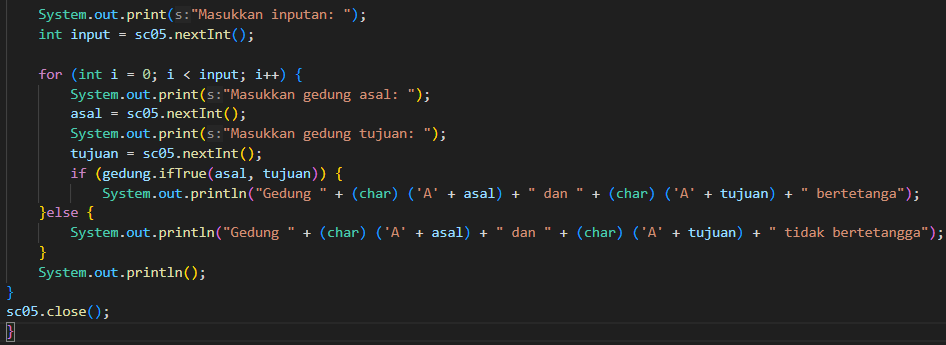
Menambahkan kode pada class Graph01 yaitu method ifTrue bertipe boolean untuk pengecekan apakah terdapat jalur antara uatu node dengan node lainnya.



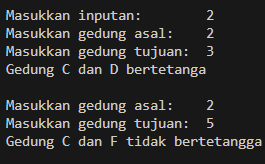
Menambahkan kode pada class GraphMain01 berikut ini yaitu sacnner berupa

inputan variabel asal dan tujuan yang berguna agar pengguna dapat memasukkan

data

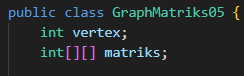


Hasil Run:



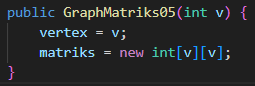
**PERCOBAAN 2**

Buat file baru, beri nama GraphMatriks05.java. Lengkapi class GraphMatriks dengan atribut vertex dan matriks.

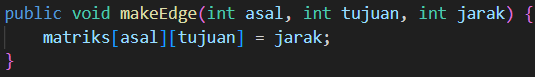


Tambahkan konstruktor default untuk menginisialisasi variabel vertex dan menginstansiasi

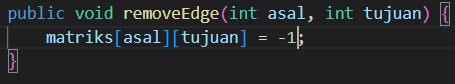
panjang array dua dimensi yang telah ditentukan.



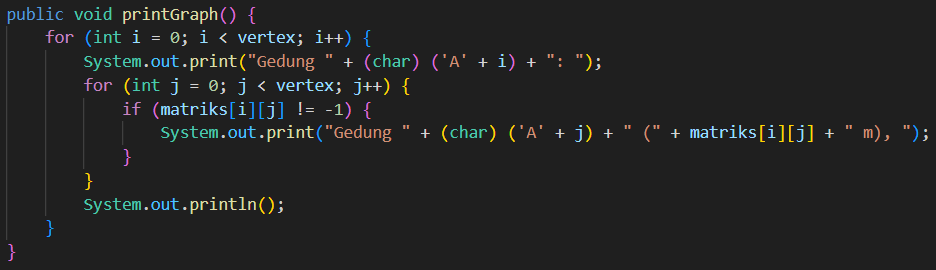
Untuk membuat suatu lintasan yang menghubungkan dua node, maka dibuat method makeEdge() sebagai berikut.



Tambahkan method removeEdge() untuk menghapus lintasan pada suatu graf.

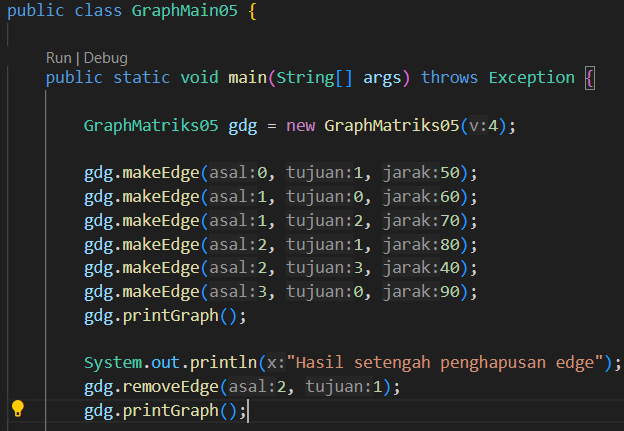


Tambahkan method printGraph() untuk mencetak graf.

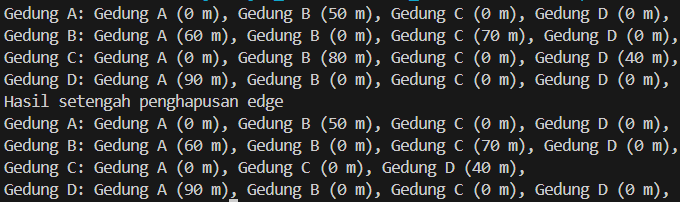


Tambahkan kode berikut pada file GraphMain<NoAbsen>.java yang sudah dibuat pada

Percobaan 1.



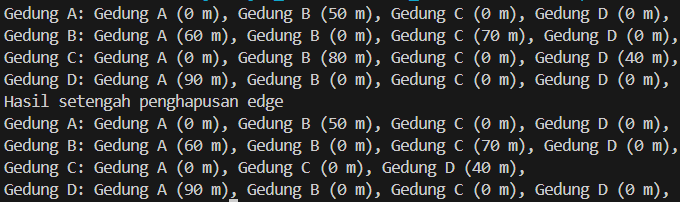
Hasil Run:



**PERTANYAAN**

1. Perbaiki kode program Anda apabila terdapat error atau hasil kompilasi kode tidak sesuai!

Jawaban: berikut adalah yang salah:

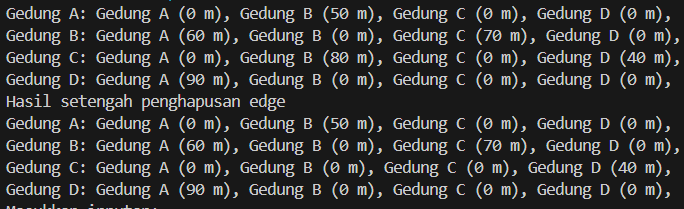


Pembenaran:

Dari -1 diubah menjadi -0



Hasil Run yg benar:



1. Apa jenis graph yang digunakan pada Percobaan 2?

Jawaban: yang digunakan adalah graph berarah dan graph berbobot.

1. Apa maksud dari dua baris kode berikut?



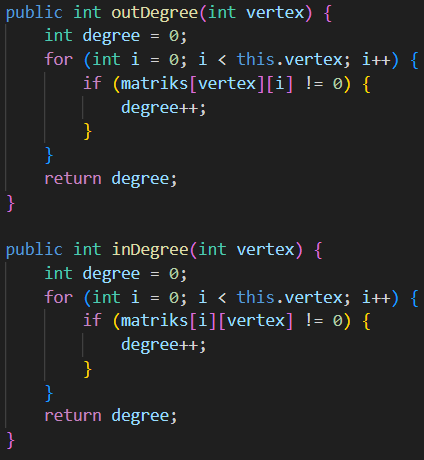
Jawaban: Maksud daripada baris pertama yaitu untuk menambahkan sebuah edge dari vertex 1 ke vertex 2 dengan bobot (jarak) 70. Ini berarti ada sebuah koneksi dari node 1 menuju node 2 dengan jarak atau bobot sebesar 70.

Sedangkan pada baris kedua di maksudkan untuk menambahkan sebuah edge dari vertex 2 ke vertex 1 dengan bobot (jarak) 80. Ini berarti ada sebuah koneksi dari node 2 menuju node 1 dengan jarak atau bobot sebesar 80.

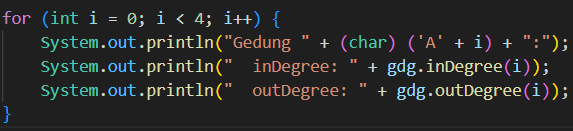
1. Modifikasi kode program sehingga terdapat method untuk menghitung degree, termasuk inDegree dan outDegree!

Jawaban:

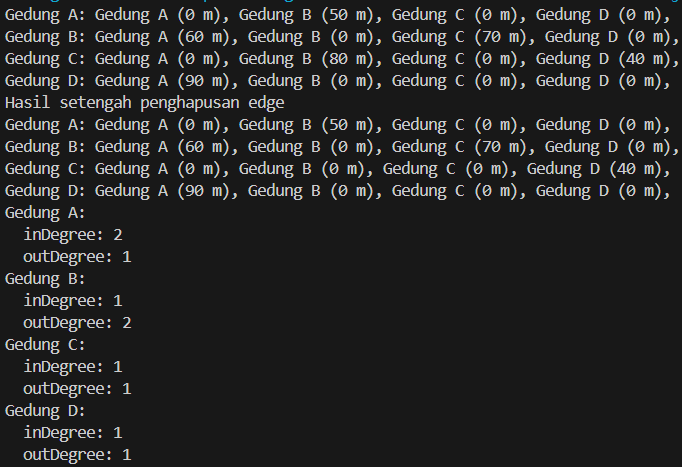
Class GraphMatriks



Class Main



Hasil Run:



**Latihan Praktikum**

1. Modifikasi kode program pada class GraphMain sehingga terdapat menu program yang bersifat dinamis, setidaknya terdiri dari:

a) Add Edge

b) Remove Edge

c) Degree

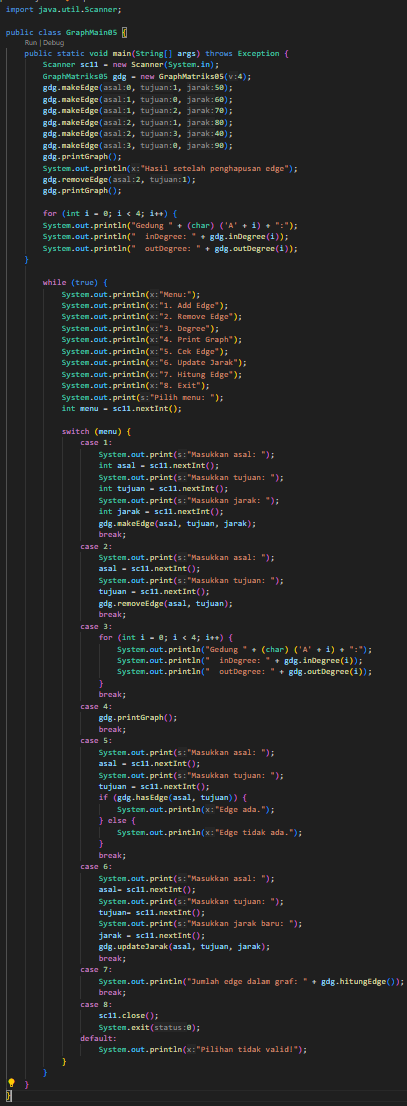
d) Print Graph

e) Cek Edge

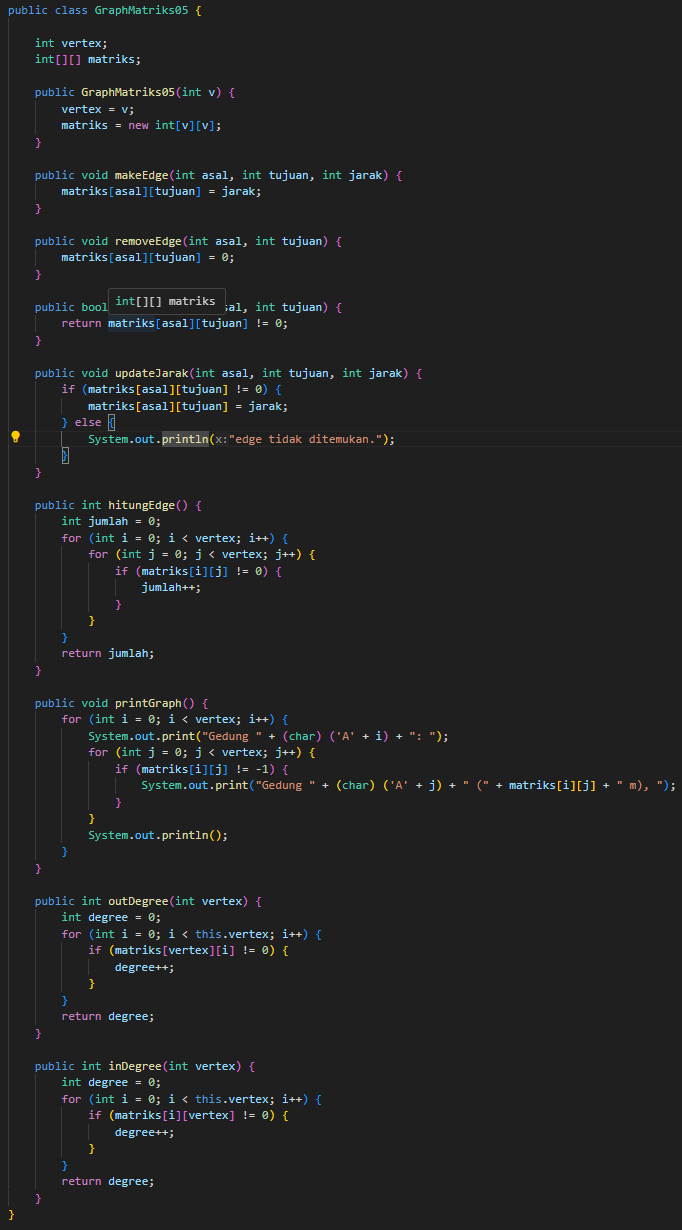
Pengguna dapat memilih menu program melalui input Scanner

1. Tambahkan method updateJarak pada Percobaan 1 yang digunakan untuk mengubah jarak antara dua node asal dan tujuan!
2. Tambahkan method hitungEdge untuk menghitung banyaknya edge yang terdapat di dalam graf!

**Jawaban:**

ClassGraphMain:

ClassGraphMatriks:



Hasil Run:

