**Laporan Teori Algoritma dan Struktur Data**

**Jobsheet 15 - Graph**

**Dosen Pengampu : Triana Fatmawati, S.T., M.T.**



**Nama : Annisa**

**Nim : 2341760032**

**Kelas : SIB 1E**

**Prodi : D-IV Sistem Informasi Bisnis**

**JURUSAN TEKNOLOGI INFORMASI**

**POLITEKNIK NEGERI MALANG**

**2023/2024**

**2. Praktikum**

**2.1 Percobaan 1: Implementasi Graph menggunakan Linked List**

**2.1.1 Langkah-langkah Percobaan**

Berikut merupakan kode program class Node



Class GraphMain



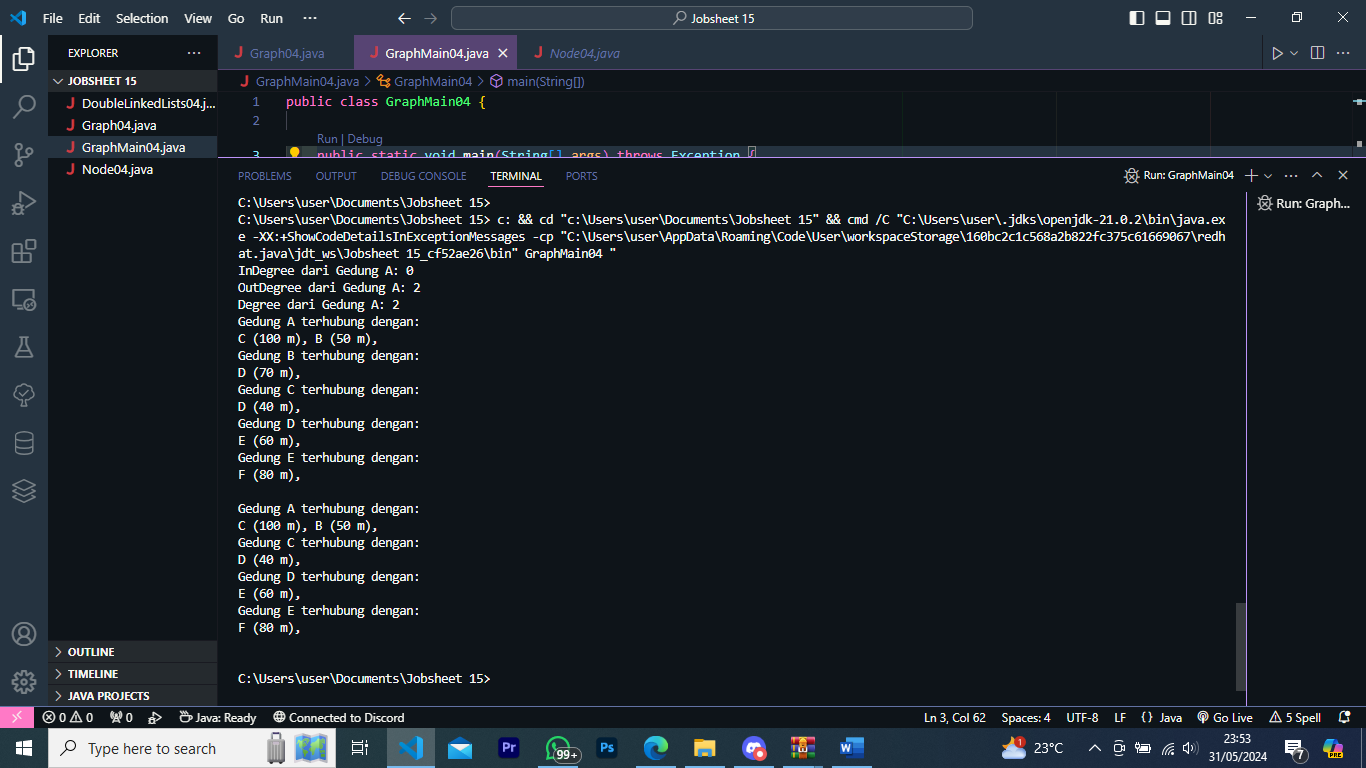
Class DoubleLinked List



Class Graph



**2.1.2 Verifikasi Hasil Percobaan**

****

**2.1.3 Pertanyaan**

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

Perbaikan kode program yang error yaitu dengan menambahkan size—pada method remove



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

Pada class Graph, atribut list[] yang bertipe DoubleLinkedList digunakan untuk merepresentasikan graf sebagai kumpulan dari beberapa list.

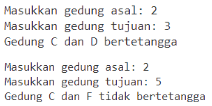
**3. Jelaskan alur kerja dari method removeEdge!**

Method removeEdge dalam kelas Graph berfungsi untuk menghapus sebuah edge antara dua simpul dalam graf. Alur kerja method ini melibatkan penghapusan entri yang sesuai dari daftar adjacency (linked list) untuk simpul asal (asal). Metode removeEdge bertujuan untuk menghapus sebuah edge (sisi) yang menghubungkan dua node dalam graf.

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

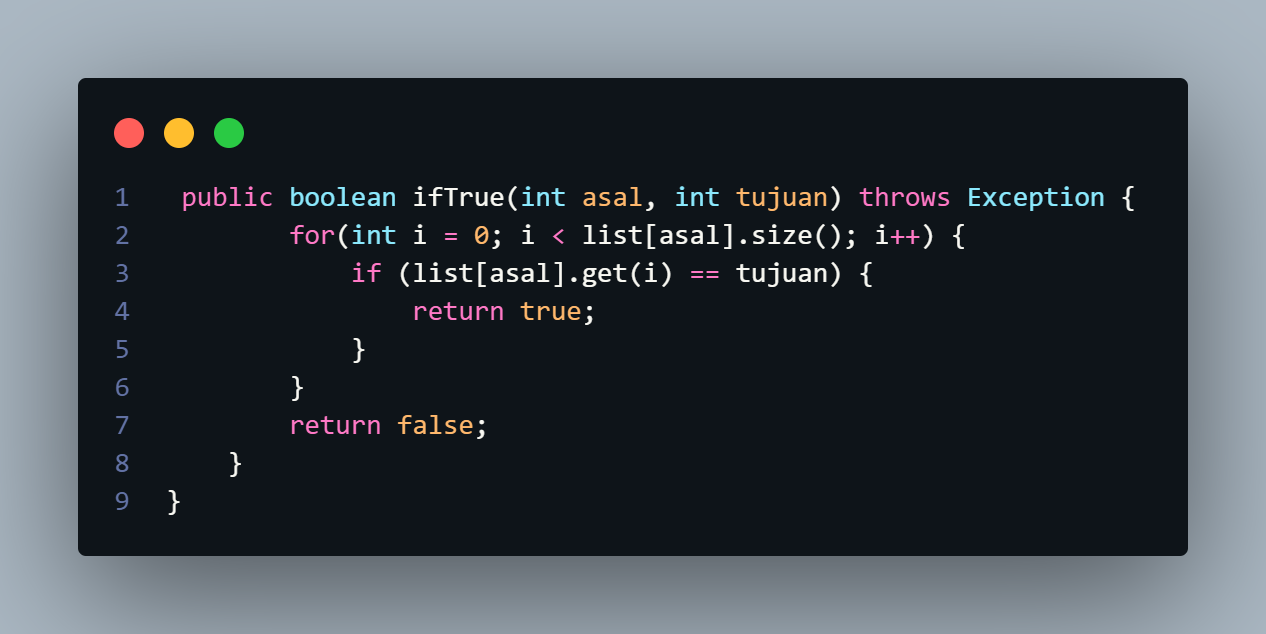
Pemanggilan addFirst dalam metode addEdge dipilih karena alasan efisiensi, kemudahan implementasi, dan konsistensi dalam pengelolaan adjacency list. Dengan addFirst, penambahan edge dilakukan dalam waktu konstan, menjaga performa graf tetap optimal tanpa perlu traversal yang tidak perlu atau penanganan urutan khusus.

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



Berikut merupakan hasil modifikasi daro studi kasus di atas yaitu dengan menambahkan kode program pada class graph dan graphmain

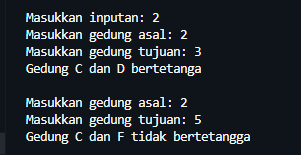
Class Graph



Class GraphMain



Output



**2.2 Percobaan 2: Implementasi Graph menggunakan Matriks**

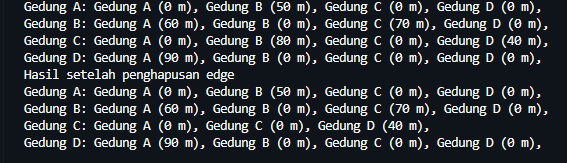
**2.2.1 Langkah-langkah Percobaan**

Berikut merupakan Class GraphMatriks



Class GraphMain

**2.2.2 Verifikasi Hasil Percobaan**



**2.2.3 Pertanyaan**

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

**2. Apa jenis graph yang digunakan pada Percobaan 2?**

Menggunakan graph berarah dan berbobot. Dikarenakan cara pengaturan nilai pada matriks adjacency dalam metode makeEdge di mana nilai jarak (yang merupakan bobot) disimpan pada posisi matriks[asal][tujuan]. Sedangkan dalam metode removeEdge untuk mengindikasikan penghapusan edge juga mendukung bahwa graph ini berarah, karena hanya arah dari asal ke tujuan yang dihapus, bukan sebaliknya

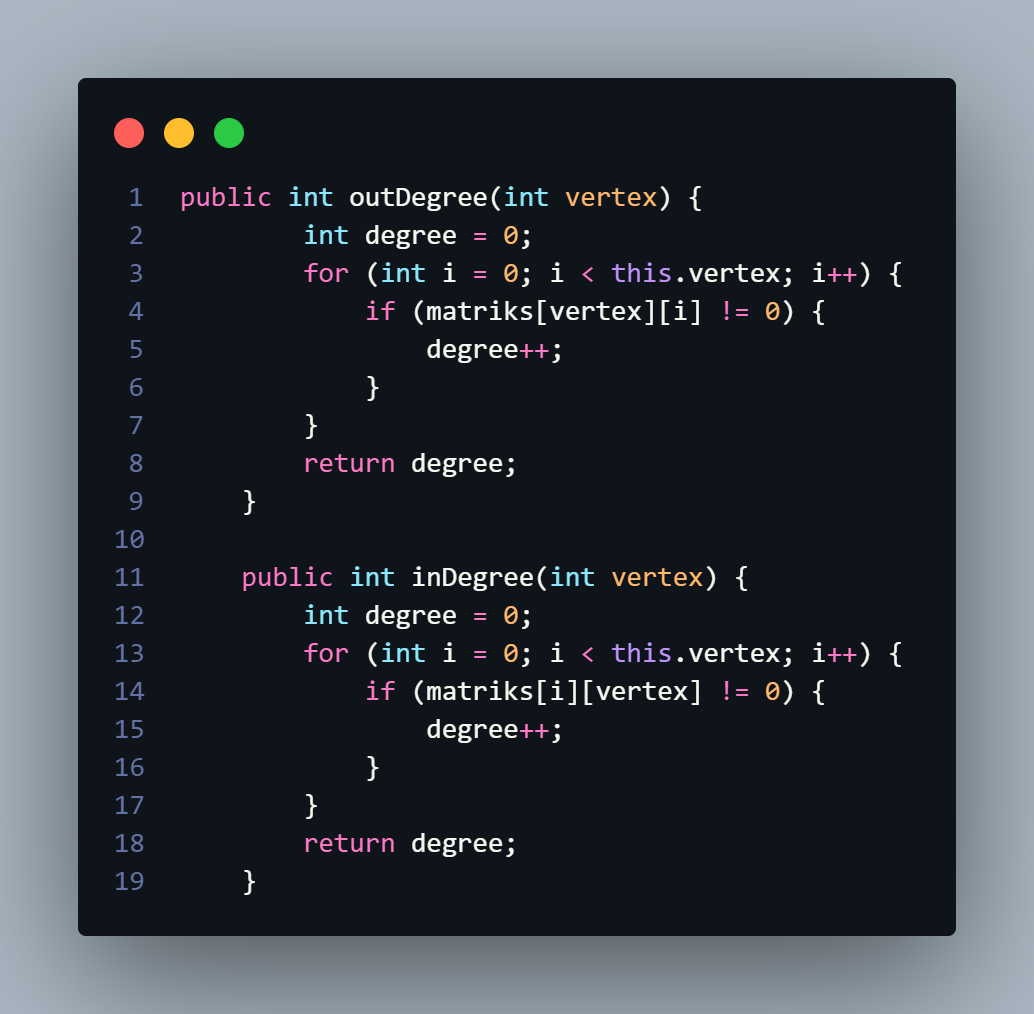
3**. Apa maksud dari dua baris kode berikut?**



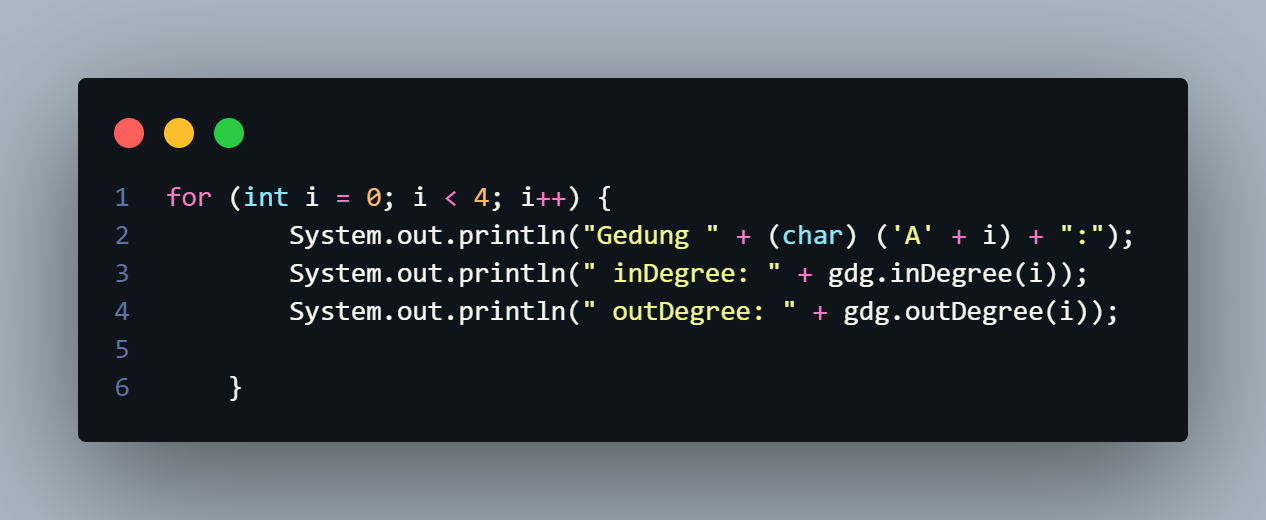
Maksut dari dua bari tersbeut yaitu pada baris pertama untuk menambahkan sebuah edge dari vertex 1 ke vertex 2 dengan bobot (jarak) 70. Ini berarti ada sebuah koneksi dari node 1 menuju node 2 dengan jarak atau bobot sebesar 70. Sedangkan pada baris kedua di maksudkan untuk menambahkan sebuah edge dari 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.

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

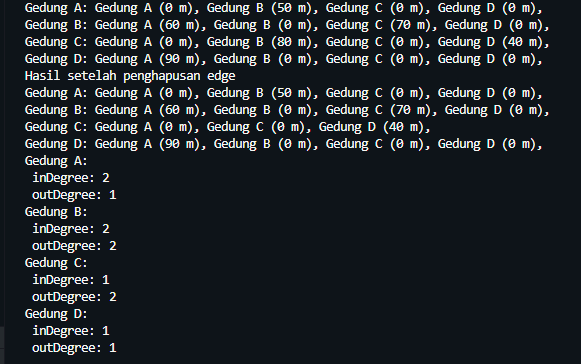
Class GraphMatriks



Class GraphMain



Output



**3. 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**

**2. Tambahkan method updateJarak pada Percobaan 1 yang digunakan untuk mengubah jarak antara dua node asal dan tujuan!**

**3. Tambahkan method hitungEdge untuk menghitung banyaknya edge yang terdapat di dalam graf!**

Berikut merupakan kode program hasil modifikasi di atas





Output

