**LAPORAN HASIL PRAKTIKUM**

**ALGORITMA DAN STRUKTUR DATA**

**GRAPH**



**Oleh:**

**FAUZIYYAH ADELIA RAMANDA**

**NIM. 2341760145**

**SIB-1F / 10**

**D-IV SISTEM INFORMASI BISNIS**

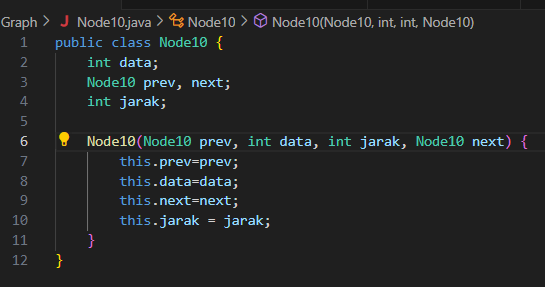
**JURUSAN TEKNOLOGI INFORMASI**

**POLITEKNIK NEGERI MALANG**

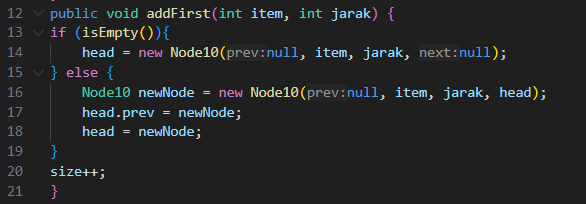
* 1. **2.1 Percobaan 1: Implementasi Graph menggunakan Linked List**

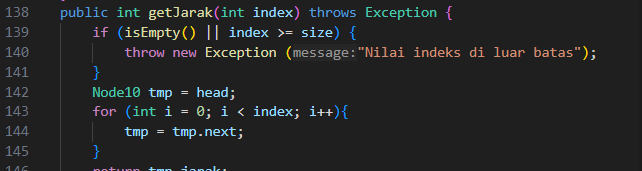
**2.2.1 Langkah Percobaan**

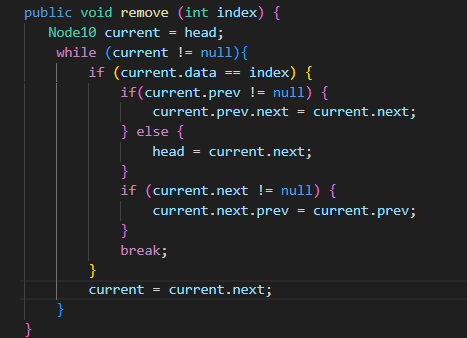
1. Membuat Class Node



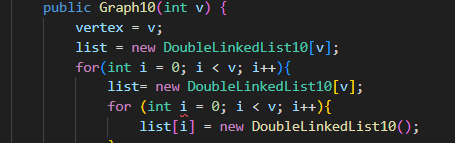
2. Menyalin class DoubleLinkedList

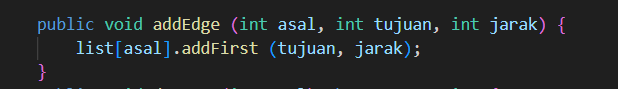
****

****

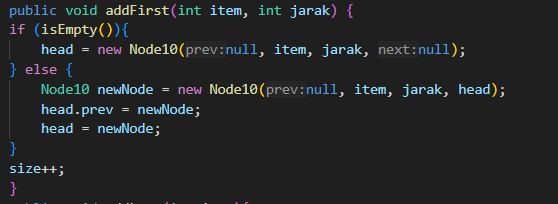
****

3. Membuat class graph dan konstruktornya

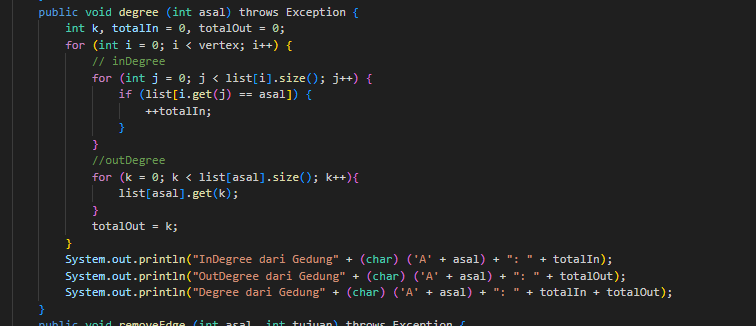


4. Membuat method addEdge

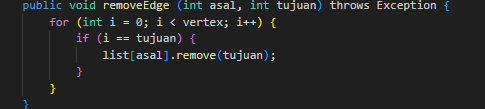
5. Menambahkan method addFirst



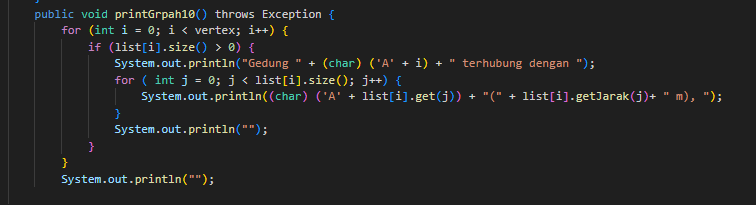
6. Menambahkan method degree



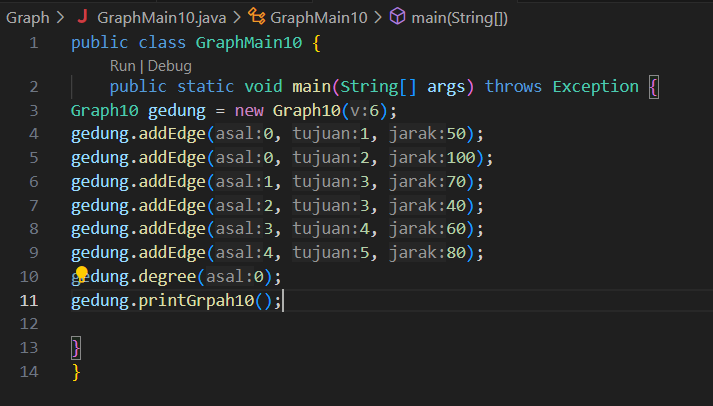
7. Menambahkan method remove edge



8. Membuat method print graph



9. Membuat main class



**2.1.3 Langkah Percobaan**

1. Perbaiki kode program Anda apabila terdapat error atau hasil kompilasi kode tidak sesuai!
2. Pada class Graph, terdapat atribut **list[]** bertipe DoubleLinkedList. Sebutkan tujuan pembuatan variabel tersebut!

* Untuk menyimpan daftar tetangga (adjency list) dari stiap vertex dalam graph

1. Jelaskan alur kerja dari method **removeEdge**!
2. Validasi : Pastikan node yang ingin dihapus ada dan terhubung
3. Penghapusan Koneksi : Hapus node dari daftar tetangga masing-masing, lalu perbarui degree node, yang trakhir hapus edge (jika graf terarah)
4. Pembaruan Struktur Graf : Perbarui struktur data internal graf, perbarui matriks adjasensi (jika ada), hapus entri daftar adjasensi (jika ada)
5. Penanganan Kasus Khusus : Perbarui bobot graf(jika graf tertimbang), hapus node terisolasi (jika ada)
6. Pengembangan Nilai : Kembalikan status operasi (berhasil/gagal)
7. Apakah alasan pemanggilan method **addFirst()** untuk menambahkan data, bukan method add jenis lain saat digunakan pada method **addEdge** pada class Graph?

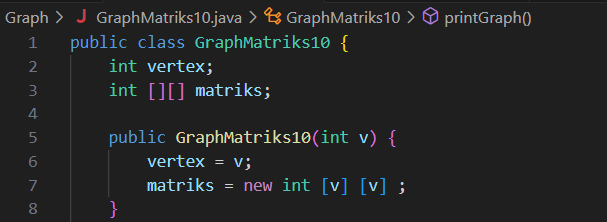
Karena efisiensi addFirst() lebih cepat karna tidak menggeser elemen lain, dan representasi koneksi addFirst() dapat mewakili urutan edge diproses, lalu kemudahan implementasi addFirst() menyederhanakan implementasi addEdge

1. A close up of black text  Description automatically generatedA close up of black text  Description automatically generatedModifikasi kode program sehingga dapat dilakukan pengecekan apakah terdapat jalur antara suatu node dengan node lainnya, seperti contoh berikut (Anda dapat memanfaatkan Scanner).

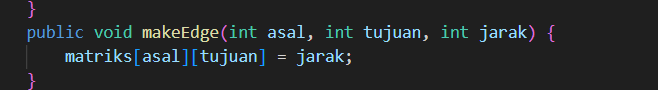
# Percobaan 2: Implementasi Graph menggunakan Matriks

2.2.1 Langkah Percobaan

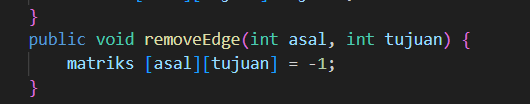
1. Membuat file GraphMatriks.java dan menambahlan konstruktor



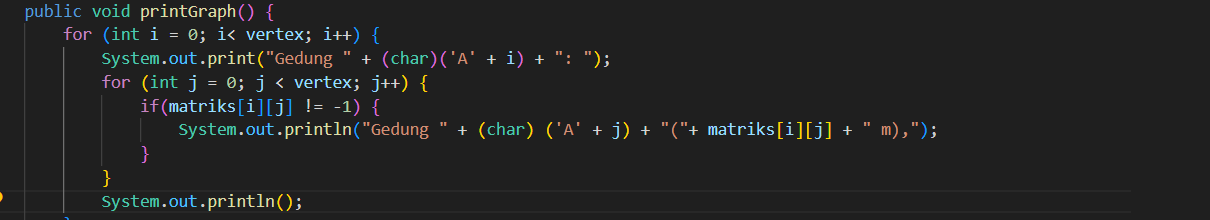
2. Membuat method makeEdge()



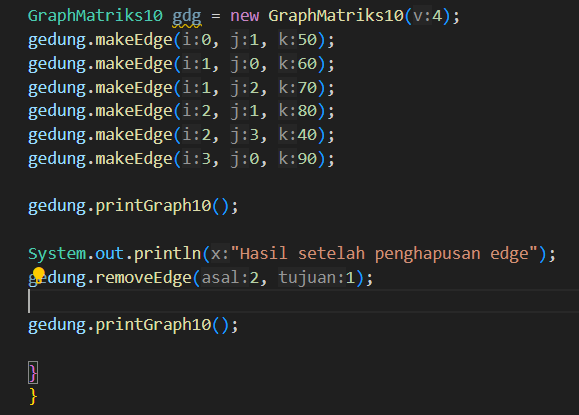
3. Membuat removeEdge()



4. Tambahkan method printGraph()



5.Menambahkan kode pada file GraphMain.java



# Pertanyaan

* + - 1. Perbaiki kode program Anda apabila terdapat error atau hasil kompilasi kode tidak sesuai!
      2. Apa jenis graph yang digunakan pada Percobaan 2?

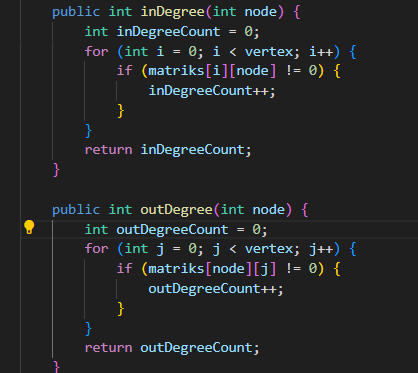
Graf matriks, ialah cara unruk merepresentasikan struktur graf dengan menggunakan matriks, yaitu susunan bilangan yang tersusun dalam baris dan kolom

* + - 1. Apa maksud dari dua baris kode berikut?



Untuk menambahkan edge diantara 2 node dalam graf baris matriks

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

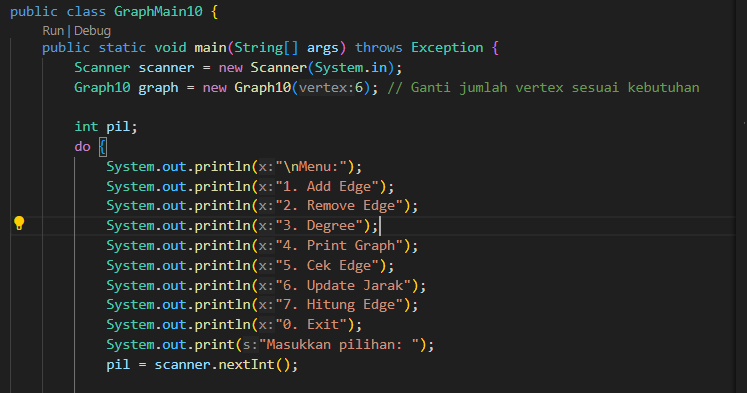


# Latihan Praktikum

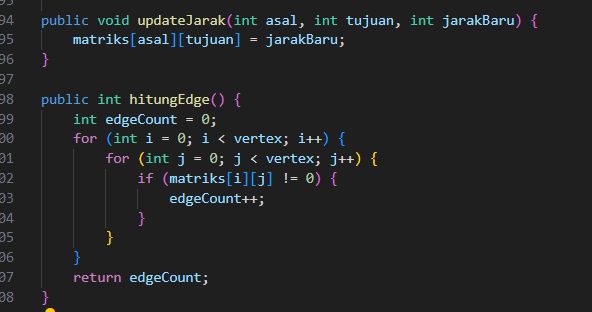
**Waktu percobaan: 90 menit**

1. Modifikasi kode program pada class **GraphMain** sehingga terdapat menu program yang bersifat dinamis, setidaknya terdiri dari:
   1. Add Edge
   2. Remove Edge
   3. Degree
   4. Print Graph
   5. 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!



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

