NAMA : HIZKIA ELSADANTA
NIM : 2341720253
KELAS : TI-1G
MATERI : GRAPH

Percobaan 1

Class node15

```java
package P13;

public class Node15 {
    int data;
    Node15 prev, next;
    int jarak;

    Node15(Node15 prev, int data, int jarak, Node15 next) {
        this.prev = prev;
        this.data = data;
        this.next = next;
        this.jarak = jarak;
    }
}
```

Class doublelinkedlist

```java
package P13;

public class DoubleLinkedList15 {
    Node15 head;
    int size;

    public DoubleLinkedList15() {
        head = null;
        size = 0;
    }

    public boolean isEmpty() {
        return head == null;
    }

    void addFirst(int item, int jarak) {
        if (isEmpty()) {
            head = new Node15(prev:null, item, jarak, next:null);
        } else {
            Node15 newNode = new Node15(prev:null, item, jarak, head);
            head.prev = newNode;
            head = newNode;
        }
        size++;
    }

    public int size() {
        return size;
    }

    public void clear() {
        head = null;
        size = 0;
    }

    int get(int index) throws Exception {
        if (isEmpty() || index >= size) {
            throw new Exception(message:"Nilai indeks di luar batas.");
        }
        Node15 tmp = head;
        for (int i = 0; i < index; i++) {
            tmp = tmp.next;
        } return tmp.data;
    }

    int getJarak(int index) throws Exception {
        if (isEmpty() || index >= size) {
            throw new Exception(message:"Nilai indeks di luar batas");
        }
        Node15 tmp = head;
        for (int i = 0; i < index; i++) {
            tmp = tmp.next;
        }
        return tmp.jarak;
    }

    void remove(int index) {
        Node15 current = head;
        while (current != null) {
            if (current.data == index) {
                if (current.prev != null) {
                    current.prev.next = current.next;
                } else {
                    head = current.next;
                }
                if (current.next != null) {
                    current.next.prev = current.prev;
                }
                break;
            }
            current = current.next;
        }
    }

}
```

NAMA : HIZKIA ELSADANTA
NIM : 2341720253
KELAS : TI-1G
MATERI : GRAPH

Class graph15

```java
package P13;

public class Graph15 {
    int vertex;
    DoubleLinkedList15 list[];

    Graph15(int v) {
        vertex = v;
        list = new DoubleLinkedList15[v];
        for (int i = 0; i < v; i++) {
            list[i] = new DoubleLinkedList15();
        }
    }

    void addEdge(int asal, int tujuan, int jarak) {
        list[asal].addFirst(tujuan, jarak);
    }

    void degree(int asal) throws Exception {
        int k;
        int totalIn = 0;
        int totalOut = 0;
        for (int i = 0; i < vertex; i++) {
            for (int j = 0; j < list[i].size(); j++) {
                if (list[i].get(j) == asal) {
                    ++totalIn;
                }
            }
            for (k = 0; k < list[asal].size(); k++) {
                list[asal].get(k);
            }
            totalOut = k;
        }
        System.out.println("InDegree dari Gedung " + (char) ('A' + asal) + ": " + totalIn);
        System.out.println("OutDegree dari Gedung " + (char) ('A' + asal) + ": " + totalOut);
        System.out.println("Degree dari Gedung " + (char) ('A' + asal) + ": " + (totalIn + totalOut));
    }

    void removeEdge(int asal, int tujuan) throws Exception {
        for (int i = 0; i < vertex; i++) {
            if (i == tujuan) {
                list[asal].remove(tujuan);
            }
        }
    }

    void removeAllEdges() {
        for (int i = 0; i < vertex; i++) {
            list[i].clear();
        }
        System.out.println("Graf berhasil dikosongkan");
    }

    void printGraph() throws Exception {
        for (int i = 0; i < vertex; i++) {
            if (list[i].size() > 0) {
                System.out.println("Gedung " + (char) ('A' + i) + " terhubung dengan ");
                for (int j = 0; j < list[i].size(); j++) {
                    System.out.print((char) ('A' + list[i].get(j)) + " (" + list[i].getJarak(j) + " m), ");
                }
                System.out.println("");
            }
        }
        System.out.println("");
    }
}
```

Main

```java
package P13;

public class GraphMain15 {
    Run | Debug
    public static void main(String[] args) throws Exception {
        Graph15 gedung = new Graph15(6);
        gedung.addEdge(0, 1, 50);
        gedung.addEdge(0, 2, 100);
        gedung.addEdge(1, 3, 70);
        gedung.addEdge(2, 3, 40);
        gedung.addEdge(3, 4, 60);
        gedung.addEdge(4, 5, 80);
        gedung.degree(0);
        gedung.printGraph();
    }
}
```

NAMA : HIZKIA ELSADANTA
NIM : 2341720253
KELAS : TI-1G
MATERI : GRAPH

Hasil running pada langkah 14

```java
package P13;

public class GraphMain15 {
    Run | Debug
    public static void main(String[] args) throws Exception {
        Graph15 gedung = new Graph15(v:6);
        gedung.addEdge(asal:0, tujuan:1, jarak:50);
        gedung.addEdge(asal:0, tujuan:2, jarak:100);
        gedung.addEdge(asal:1, tujuan:3, jarak:70);
        gedung.addEdge(asal:2, tujuan:3, jarak:40);
        gedung.addEdge(asal:3, tujuan:4, jarak:60);
        gedung.addEdge(asal:4, tujuan:5, jarak:80);
        gedung.degree(asal:0);
        gedung.printGraph();
        // gedung.removeEdge(1, 3);
        // gedung.printGraph();
    }
}
```

```
InDegree dari Gedung A: 0
OutDegree dari Gedung A: 2
Degree dari Gedung A: 2
Gedung A terhubung dengan
C (100 m), B (50 m),
Gedung B terhubung dengan
D (70 m),
Gedung C terhubung dengan
D (40 m),
Gedung D terhubung dengan
E (60 m),
Gedung E terhubung dengan
F (80 m),
```

Hasil running pada langkah 17

```java
package P13;

public class GraphMain15 {
    Run | Debug
    public static void main(String[] args) throws Exception {
        Graph15 gedung = new Graph15(v:6);
        gedung.addEdge(asal:0, tujuan:1, jarak:50);
        gedung.addEdge(asal:0, tujuan:2, jarak:100);
        gedung.addEdge(asal:1, tujuan:3, jarak:70);
        gedung.addEdge(asal:2, tujuan:3, jarak:40);
        gedung.addEdge(asal:3, tujuan:4, jarak:60);
        gedung.addEdge(asal:4, tujuan:5, jarak:80);
        // gedung.degree(0);
        // gedung.printGraph();
        gedung.removeEdge(asal:1, tujuan:3);
        gedung.printGraph();
    }
}
```

```
Gedung A terhubung dengan
C (100 m), B (50 m),
Gedung C terhubung dengan
D (40 m),
Gedung D terhubung dengan
E (60 m),
Gedung E terhubung dengan
F (80 m),
```

NAMA : HIZKIA ELSADANTA
NIM : 2341720253
KELAS : TI-1G
MATERI : GRAPH

Pertanyaan :

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

   Jawab : Terjadi error ketika removeEdge

   Sebelum

```
void remove(int index) {
    Node15 current = head;
    while (current != null) {
        if (current.data == index) {
            if (current.prev != null) {
                current.prev.next = current.next;
            } else {
                head = current.next;
            }
            if (current.next != null) {
                current.next.prev = current.prev;
            }
            break;
        }
        current = current.next;
    }
}
```

   Setelah

```
void remove(int index) throws Exception {
    if (isEmpty() || index >= size) {
        throw new Exception(message:"Nilai indeks di luar batas.");
    }
    Node15 current = head;
    for (int i = 0; i < index; i++) {
        current = current.next;
    }
    if (current.prev != null) {
        current.prev.next = current.next;
    } else {
        head = current.next;
    }
    if (current.next != null) {
        current.next.prev = current.prev;
    }
    size--;
}
```

   Sebelum

```
void removeEdge(int asal, int tujuan) throws Exception {
    for (int i = 0; i < vertex; i++) {
        if (i == tujuan) {
            list[asal].remove(tujuan);
        }
    }
}
```

   Setelah

```
void removeEdge(int asal, int tujuan) throws Exception {
    for (int i = 0; i < list[asal].size(); i++) {
        if (list[asal].get(i) == tujuan) {
            list[asal].remove(i);
            break;
        }
    }
}
```

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

   Jawab : Tujuan pembuatan variabel list[] bertipe DoubleLinkedList pada kelas Graph adalah untuk menyimpan daftar adjacency (tetangga) dari setiap vertex dalam graph, yang memungkinkan penyimpanan dan pengelolaan edges beserta jaraknya secara efisien.

3. Jelaskan alur kerja dari method removeEdge!

   Jawab : Method removeEdge bekerja dengan cara mengiterasi daftar adjacency pada vertex asal, mencari edge yang menuju ke tujuan, dan menghapusnya begitu ditemukan.

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

   Jawab : Penggunaan addFirst dalam method addEdge pada class graph dipilih karena efisiensinya dalam menambahkan elemen ke awal daftar, kesederhanaan implementasi, dan karena dalam banyak kasus graf, urutan tetangga tidak menjadi faktor penting, sehingga implementasi method addFirst lebih sederhana dibandingkan dengan method penambahan lainnya seperti addLast atau addAtIndex.

5. Modifikasi kode program sehingga dapat dilakukan pengecekan apakah terdapat jalur antara suatu node

NAMA : HIZKIA ELSADANTA
NIM : 2341720253
KELAS : TI-1G
MATERI : GRAPH

dengan node lainnya, seperti contoh berikut (Anda dapat memanfaatkan Scanner).

```
Masukkan gedung asal: 2
Masukkan gedung tujuan: 3
Gedung C dan D bertetangga

Masukkan gedung asal: 2
Masukkan gedung tujuan: 5
Gedung C dan F tidak bertetangga
```

Jawab : Modifikasi pada class graph dan main

Class graph

```
boolean adjacency(int start, int end) throws Exception {
    return list[start].size() > 0 && list[start].get(index:0) == end;
}
```

Main

```
Scanner kiak = new Scanner(System.in);
boolean lanjut = true;
while (lanjut) {
    System.out.println(x:"Menu:");
    System.out.println(x:"1. Cek gedung");
    System.out.println(x:"0. Keluar");
    System.out.print(s:"Pilih menu (0/1): ");
    int pilihan = kiak.nextInt();

    switch (pilihan) {
        case 1:
            System.out.print(s:"Masukkan asal gedung (0-5): ");
            int asal = kiak.nextInt();
            System.out.print(s:"Masukkan gedung tujuan (0-5): ");
            int tujuan = kiak.nextInt();

            if (asal < 0 || asal > 5 || tujuan < 0 || tujuan > 5) {
                System.out.println(x:"Gedung yang dimasukkan tidak valid. Masukkan gedung dalam rentang 0-5.");
            } else {
                if (gedung.adjacency(asal, tujuan)) {
                    System.out.println("Gedung " + (char) ('A' + asal) + " dan " + (char) ('A' + tujuan) + " bertetangga");
                } else {
                    System.out.println("Gedung " + (char) ('A' + asal) + " dan " + (char) ('A' + tujuan) + " tidak bertetangga");
                }
            }
            break;
        case 0:
            lanjut = false;
            System.out.println(x:"Terima kasih telah menggunakan program.");
            break;
        default:
            System.out.println(x:"Pilihan tidak valid. Silakan pilih lagi.");
            break;
    }
}
kiak.close();
```

Hasil

```
Masukkan asal gedung (0-5): 2
Masukkan gedung tujuan (0-5): 3
Gedung C dan D bertetangga
```

```
Masukkan asal gedung (0-5): 2
Masukkan gedung tujuan (0-5): 5
Gedung C dan F tidak bertetangga
```

NAMA : HIZKIA ELSADANTA
NIM : 2341720253
KELAS : TI-1G
MATERI : GRAPH

Percobaan 2

Class GraphMatriks

```java
package P13;

public class GraphMatriks15 {
    int vertex;
    int[][] matriks;

    GraphMatriks15(int v) {
        vertex = v;
        matriks = new int[v][v];
    }

    void makeEdge(int asal, int tujuan, int jarak) {
        matriks[asal][tujuan] = jarak;
    }

    void removeEdge(int asal, int tujuan) {
        matriks[asal][tujuan] = 0;
    }

    void printGraph() {
        for (int i = 0; i < vertex; i++) {
            System.out.print("Gedung " + (char) ('A' + i) + ": ");
            for (int j = 0; j < vertex; j++) {
                if (matriks[i][j] != -1) {
                    System.out.print("Gedung " + (char) ('A' + j) + " (" + matriks[i][j] + " m), ");
                }
            }
            System.out.println();
        }
    }
}
```

Main

```java
GraphMatriks15 gdg = new GraphMatriks15(v:4);
gdg.makeEdge(asal:0, tujuan:1, jarak:50);
gdg.makeEdge(asal:1, tujuan:0, jarak:60);
gdg.makeEdge(asal:1, tujuan:2, jarak:70);
gdg.makeEdge(asal:2, tujuan:1, jarak:80);
gdg.makeEdge(asal:2, tujuan:3, jarak:40);
gdg.makeEdge(asal:3, tujuan:0, jarak:90);
gdg.printGraph();
System.out.println(x:"Hasil setelah penghapusan edge");
gdg.removeEdge(asal:2, tujuan:1);
gdg.printGraph();
```

Hasil

```
Gedung A: Gedung A (0 m), Gedung B (50 m), Gedung C (0 m), Gedung D (0 m),
Gedung B: Gedung A (60 m), Gedung B (0 m), Gedung C (70 m), Gedung D (0 m),
Gedung C: Gedung A (0 m), Gedung B (80 m), Gedung C (0 m), Gedung D (40 m),
Gedung D: Gedung A (90 m), Gedung B (0 m), Gedung C (0 m), Gedung D (0 m),
Hasil setelah penghapusan edge
Gedung A: Gedung A (0 m), Gedung B (50 m), Gedung C (0 m), Gedung D (0 m),
Gedung B: Gedung A (60 m), Gedung B (0 m), Gedung C (70 m), Gedung D (0 m),
Gedung C: Gedung A (0 m), Gedung B (0 m), Gedung C (0 m), Gedung D (40 m),
Gedung D: Gedung A (90 m), Gedung B (0 m), Gedung C (0 m), Gedung D (0 m),
```

Pertanyaan :

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

   Jawab : Sebelumnya = -1

```java
void removeEdge(int asal, int tujuan) {
    matriks[asal][tujuan] = 0;
}
```

NAMA : HIZKIA ELSADANTA
NIM : 2341720253
KELAS : TI-1G
MATERI : GRAPH

2. Apa jenis graph yang digunakan pada Percobaan 2?

   Jawab : Graph berarah yang artinya tepi memiliki arah yang jelas, dimana dari A ke B dan dari B ke A adalah dua tepi berbeda dengan bobot/jarak yang berbeda. Graph tersebut juga merupakan graph berbobot karena setiap tepi memiliki nilai bobot tertentu, yang disimpan dalam matriks.

3. Apa maksud dari dua baris kode berikut?

```
gdg.makeEdge(1, 2, 70);
gdg.makeEdge(2, 1, 80);
```

   Jawab : Dua baris kode diatas sedang menambahkan dua tepi berarah (directed edges) dengan bobot / jarak (weights) ke dalam graph.

   `gdg.makeEdge(1, 2, 70);` Baris kode ini menambahkan sebuah tepi dari vertex 1 (B) ke vertex 2 (C) dengan bobot/jarak 70 yang artinya, ada hubungan arah dari vertex B ke vertex C yang jaraknya 70 meter.
   `gdg.makeEdge(2, 1, 80);` Baris kode ini menambahkan sebuah tepi dari vertex 2 (C) ke vertex 1 (B) dengan bobot 80 yang artinya, ada hubungan arah dari vertex C ke vertex B yang jaraknya 80 meter.

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

   Jawab :

```
int inDegree(int gedung) {
    int inDegree = 0;
    for (int i = 0; i < vertex; i++) {
        if (matriks[i][gedung] != 0) {
            inDegree++;
        }
    }
    return inDegree;
}

int outDegree(int gedung) {
    int outDegree = 0;
    for (int j = 0; j < vertex; j++) {
        if (matriks[gedung][j] != 0) {
            outDegree++;
        }
    }
    return outDegree;
}

int degree(int gedung) {
    return inDegree(gedung) + outDegree(gedung);
}
```

```
Gedung A: InDegree = 2, OutDegree = 1, Total Degree = 3
Gedung B: InDegree = 1, OutDegree = 2, Total Degree = 3
Gedung C: InDegree = 1, OutDegree = 1, Total Degree = 2
Gedung D: InDegree = 1, OutDegree = 1, Total Degree = 2
```

Link GitHub

https://github.com/Kiaakk/Algoritma_Struktur_Data_1G_15.git