LAPORAN PRAKTIKUM

MATA KULIAH PRAKTIKUM ALGORITMA STRUKTUR DATA

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

PERTEMUAN 15 : GRAPH

Nama : Yonanda Mayla Rusdiaty

NIM : 2341760184

Prodi : D-IV Sistem Informasi Bisnis

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

2024

PRAKTIKUM 1 : Impleentasi Graph Menggunakan Linked List

Kode program class Node29 :

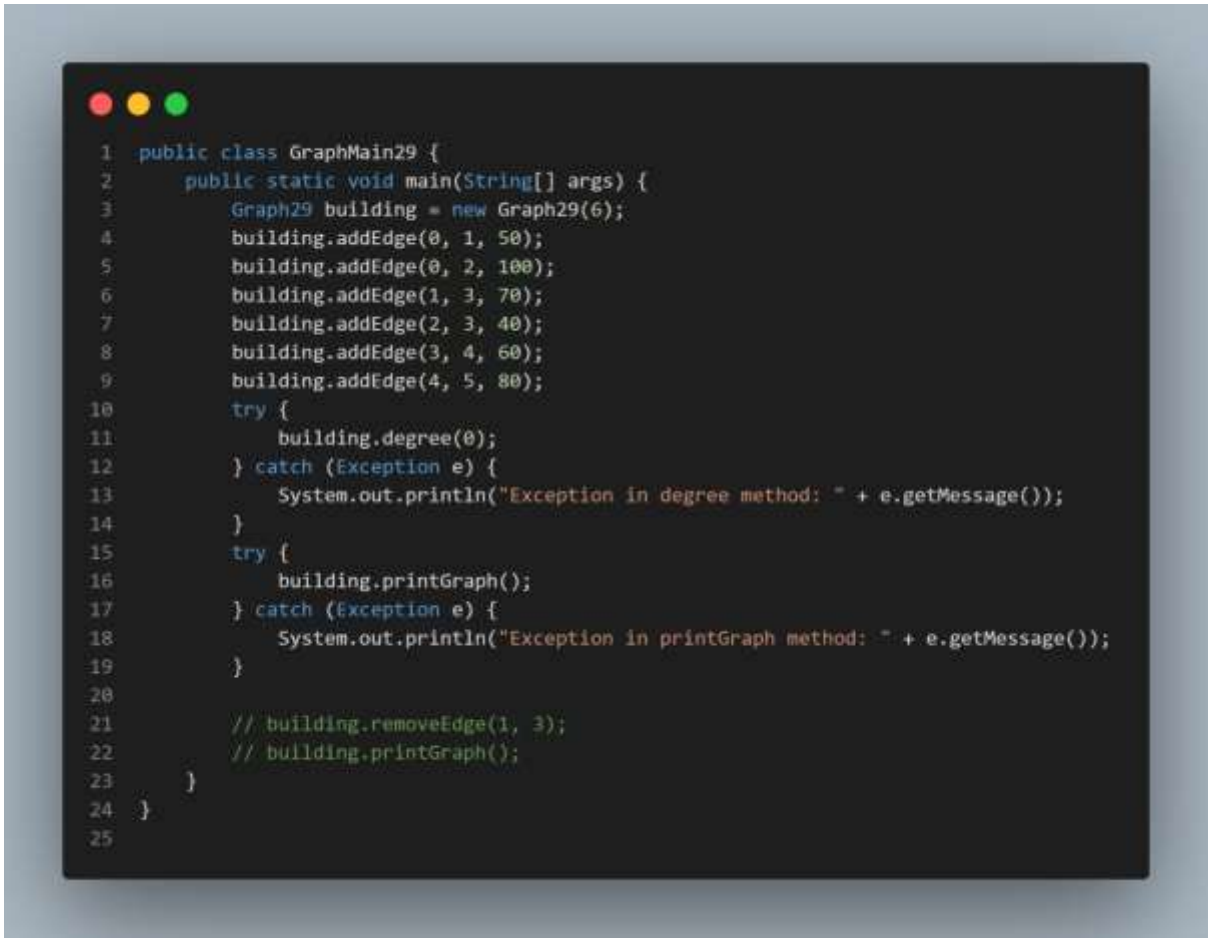
```
public class Node29 {
    int data;
    Node29 prev;
    Node29 next;
    int jarak;

    public Node29(Node29 prev, int data, int jarak, Node29 next) {
        this.prev = prev;
        this.data = data;
        this.next = next;
        this.jarak = jarak;
    }
}
```

Kode program class DoubleLinkedList29 :

Kode program class Graph :

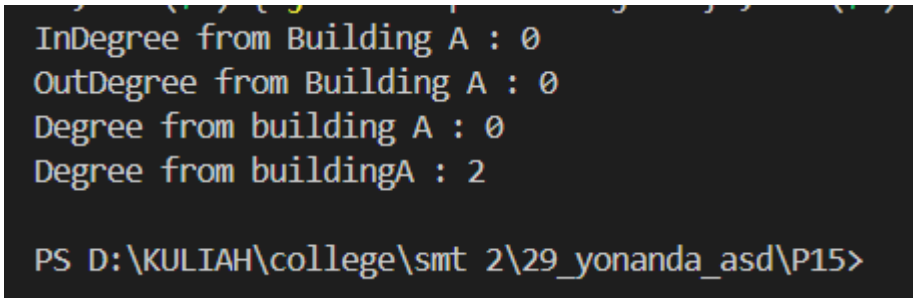
```java
public class Graph29 {
    int vertex;
    DoubleLinkedLists29 list[];

    public Graph29(int v) {
        list = new DoubleLinkedLists29[v];
        for (int i = 0; i < v; i++) {
            list[i] = new DoubleLinkedLists29();
        }
    }

    public void addEdge(int source, int destination, int jarak) {// for directed graph
        list[source].addFirst(destination, jarak);
        list[destination].addFirst(source, jarak);// for undirected graph
    }

    public void degree(int source) throws Exception {
        int k, totalIn = 0, totalOut = 0;
        for (int i = 0; i < vertex; i++) {
            // in degree
            for (int j = 0; j < list[i].size(); j++) {
                if (list[i].get(j) == source) {
                    ++totalIn;
                }
            }
            // out degree
            for (k = 0; k < list[source].size(); k++) {
                list[source].get(k);
            }
            totalOut = k;
        }
        System.out.println("InDegree from Building " + (char) ('A' + source) + " : " + totalIn);
        System.out.println("OutDegree from Building " + (char) ('A' + source) + " : " + totalOut);
        System.out.println("Degree from building " + (char) ('A' + source) + " : " + (totalIn + totalOut));
        System.out.println("Degree from building" + (char) ('A' + source) + " : " + list[source].size());
    }

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

    public void removeAllEdges() {
        for (int i = 0; i < vertex; i++) {
            list[i].clear();
        }
        System.out.println("Graph successfully emptied");
    }

    public void printGraph() throws Exception {
        for (int i = 0; i < vertex; i++) {
            if (list[i].size() > 0) {
                System.out.print("Building " + (char) ('A' + i) + " connected with ");
                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("");
    }
}
```

Kode program class GraphMain29 :



```java
public class GraphMain29 {
    public static void main(String[] args) {
        Graph29 building = new Graph29(6);
        building.addEdge(0, 1, 50);
        building.addEdge(0, 2, 100);
        building.addEdge(1, 3, 70);
        building.addEdge(2, 3, 40);
        building.addEdge(3, 4, 60);
        building.addEdge(4, 5, 80);
        try {
            building.degree(0);
        } catch (Exception e) {
            System.out.println("Exception in degree method: " + e.getMessage());
        }
        try {
            building.printGraph();
        } catch (Exception e) {
            System.out.println("Exception in printGraph method: " + e.getMessage());
        }

        // building.removeEdge(1, 3);
        // building.printGraph();
    }
}
```

Output :



```
InDegree from Building A : 0
OutDegree from Building A : 0
Degree from building A : 0
Degree from buildingA : 2

PS D:\KULIAH\college\smt 2\29_yonanda_asd\P15>
```

Kedua hasil output masih tidak sesuai, baik untuk Langkah 14 ataupun 17

Pertanyaan :

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

   Jawab :

   Berikut adalah hasil modifikasinya :

   - Kode program class Graph29 :

```java
import java.util.*;

public class Graph29 {
    private class Edge {
        int destination, distance;

        public Edge(int destination, int distance) {
            this.destination = destination;
            this.distance = distance;
        }
    }

    private int vertex;
    private LinkedList<Edge>[] list;

    public Graph29(int vertex) {
        this.vertex = vertex;
        list = new LinkedList[vertex];
        for (int i = 0; i < vertex; i++) {
            list[i] = new LinkedList<>();
        }
    }

    public void addEdge(int source, int destination, int distance) {
        list[source].addFirst(new Edge(destination, distance));
    }

    public void removeEdge(int source, int destination) {
        list[source].removeIf(edge -> edge.destination == destination);
    }

    public void degree(int source) {
        int inDegree = 0, outDegree = list[source].size();
        for (LinkedList<Edge> edges : list) {
            for (Edge edge : edges) {
                if (edge.destination == source) {
                    inDegree++;
                }
            }
        }
        System.out.println("InDegree dari Gedung " + (char) ('A' + source) + ": " + inDegree);
        System.out.println("OutDegree dari Gedung " + (char) ('A' + source) + ": " + outDegree);
        System.out.println("Degree dari Gedung " + (char) ('A' + source) + ": " + (inDegree + outDegree));
    }

    public void printGraph() {
        for (int i = 0; i < vertex; i++) {
            if (!list[i].isEmpty()) {
                System.out.println("Gedung " + (char) ('A' + i) + " terhubung dengan");
                for (Edge edge : list[i]) {
                    System.out.println((char) ('A' + edge.destination) + " (" + edge.distance + " m),");
                }
            }
        }
    }
}
```

- Kode program main :

```java
public static void main(String[] args) {
        Graph29 building = new Graph29(6);
        building.addEdge(0, 1, 50);
        building.addEdge(0, 2, 100);
        building.addEdge(1, 3, 70);
        building.addEdge(2, 3, 40);
        building.addEdge(3, 4, 60);
        building.addEdge(4, 5, 80);
        try {
            building.degree(0);
        } catch (Exception e) {
            System.out.println("Exception in degree method: " + e.getMessage());
        }
        try {
            building.printGraph();
        } catch (Exception e) {
            System.out.println("Exception in printGraph method: " + e.getMessage());
        }

        try {
            building.removeEdge(1, 3);
        } catch (Exception e) {
            System.out.println("Exception in removeEdge method: " + e.getMessage());
        }
        try {
            building.printGraph();
        } catch (Exception e) {
            System.out.println("Exception in printGraph method: " + e.getMessage());
        }
    }
```

- Output Langkah 14 :

```
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),
PS D:\KULIAH\college\smt 2\29_yonanda_asd\P15>
```

- Output Langkah 17 :

```
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),
PS D:\KULIAH\college\smt 2\29_yonanda_asd\P15>
```

2. Pada class Graph, terdapat atribut list[] bertipe DoubleLinkedList. Sebutkan tujuan pembuatan variabel tersebut!
   Jawab : Hal tersebut digunakan untuk merepresentasikan struktur data graf. Dalam konteks ini, list[] digunakan untuk menyimpan daftar simpul (vertex) yang terhubung dengan setiap simpul dalam graf. Setiap indeks dalam array list[] merepresentasikan satu simpul dalam graf, dan setiap elemen dalam DoubleLinkedList di indeks tersebut merepresentasikan simpul lain yang terhubung dengan simpul tersebut.

   Dengan kata lain, list[i] berisi daftar simpul yang terhubung dengan simpul i. Jika ada edge antara simpul i dan simpul j, maka j akan ada dalam list list[i] dan sebaliknya. Hal ini menggunakan pendekatan adjacency list.

3. Jelaskan alur kerja dari method removeEdge!
   Jawab : Metode removeEdge digunakan untuk menghapus edge (sisi) antara dua simpul (vertex) dalam graf. Dalam kode tersebut, method ini menerima dua argumen: yaitu source dan destination, yang mewakili simpul asal dan simpul tujuan dari edge yang akan dihapus. Berikut adalah alur kerja dari metode removeEdge:

   1) removeIf adalah method dari LinkedList yang menghapus semua elemen yang memenuhi kondisi yang diberikan. Dalam hal ini, kondisinya adalah edge.destination == destination, yang berarti "jika tujuan dari edge adalah simpul tujuan".
   2) Jadi, ekspresi ini akan menghapus semua edge dari list[source] yang tujuannya adalah simpul destination. Dengan kata lain, maka akan menghapus semua edge yang mengarah dari simpul source ke simpul destination.
   3) Setelah baris kode ini dijalankan, edge antara simpul source dan destination telah dihapus dari graf.

   4) Hal ini hanya akan menghapus edge dari source ke destination, bukan sebaliknya. Jika graf nya adalah graf tidak berarah (undirected graph),kita mungkin juga perlu menghapus edge dari destination ke source.

4. Apakah alasan pemanggilan method addFirst() untuk menambahkan data, bukan method add jenis lain saat digunakan pada method addEdge pada class Graph?
   Jawab : Pemanggilan metode addFirst() dalam konteks ini digunakan untuk menambahkan elemen ke awal LinkedList. Alasan utama penggunaan addFirst() dibandingkan dengan metode add lainnya (seperti addLast() atau add(int index, E element)) adalah efisiensi waktu.

   Dalam LinkedList, operasi addFirst() adalah operasi O(1), yang berarti ia memiliki waktu eksekusi konstan, tidak peduli seberapa besar ukuran list. Ini karena LinkedList menyimpan referensi langsung ke elemen pertama dalam list, sehingga dapat langsung menambahkan elemen baru di awal tanpa harus melewati elemen lainnya.

   Sebaliknya, metode seperti addLast() atau add(int index, E element) mungkin memerlukan waktu lebih lama untuk mengeksekusi, tergantung pada posisi elemen yang ditambahkan. Misalnya, addLast() juga O(1) dalam LinkedList, tetapi add(int index, E element) bisa sampai O(n) dalam kasus terburuk karena mungkin perlu melewati beberapa elemen untuk mencapai posisi yang ditentukan.

   Dalam konteks graf, urutan simpul yang terhubung biasanya tidak penting, jadi menggunakan addFirst() adalah pilihan yang efisien dan praktis.

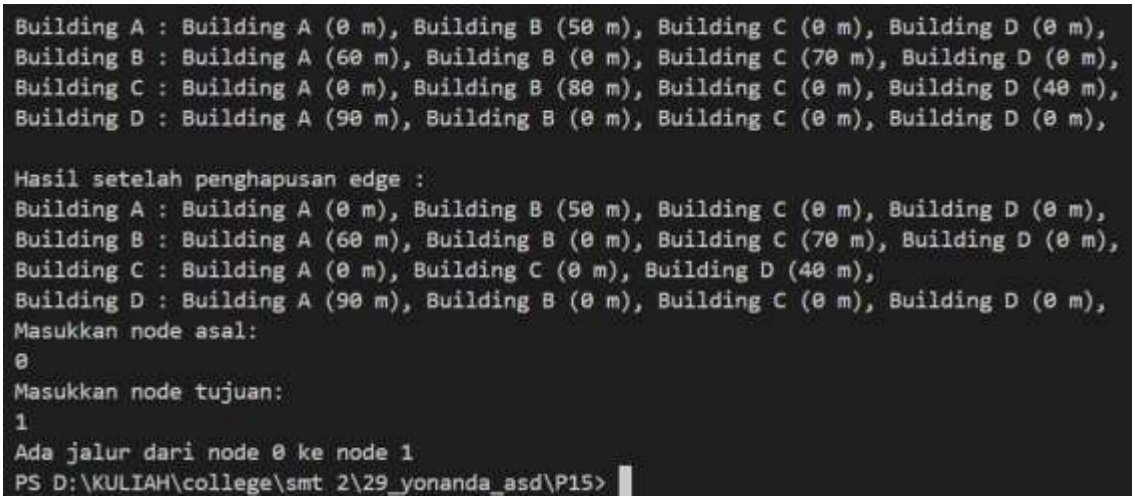
5. Modifikasi kode program sehingga dapat dilakukan pengecekan apakah terdapat jalur antara suatu node dengan node lainnya, seperti contoh berikut (Anda dapat manfaatkan Scanner).
   Jawab : Berikut adalah hasil modifikasinya :
   - Kode program class Graph29 :

```
// modif no 5 praktikum 1
public boolean isConnected(int source, int destination) {
    return list[source].stream().anyMatch(edge -> edge.destination == destination);
}
```

   - Kode program class GraphMain :

```
Scanner scanner = new Scanner(System.in);    Resource leak: 'scanner' is never closed
System.out.println(x:"Masukkan node asal: ");
int source = scanner.nextInt();
System.out.println(x:"Masukkan node tujuan: ");
int destination = scanner.nextInt();
if (building.isConnected(source, destination)) {
    System.out.println("Ada jalur dari node " + source + " ke node " + destination);
} else {
    System.out.println("Tidak ada jalur dari node " + source + " ke node " + destination);
}
```

   - Output :


Building A : Building A (0 m), Building B (50 m), Building C (0 m), Building D (0 m),
Building B : Building A (60 m), Building B (0 m), Building C (70 m), Building D (0 m),
Building C : Building A (0 m), Building B (80 m), Building C (0 m), Building D (40 m),
Building D : Building A (90 m), Building B (0 m), Building C (0 m), Building D (0 m),
Hasil setelah penghapusan edge :
Building A : Building A (0 m), Building B (50 m), Building C (0 m), Building D (0 m),
Building B : Building A (60 m), Building B (0 m), Building C (70 m), Building D (0 m),
Building C : Building A (0 m), Building C (0 m), Building D (40 m),
Building D : Building A (90 m), Building B (0 m), Building C (0 m), Building D (0 m),
Masukkan node asal:
0
Masukkan node tujuan:
1
Ada jalur dari node 0 ke node 1
PS D:\KULIAH\college\smt 2\29_yonanda_asd\P15>

PRAKTIKUM 2 : Implementasi Graph Menggunakan Matriks

Kode program class GraphMatriks29 :

```
public class GraphMatriks29 {
    int vertex;
    int[][] matrix;

    public GraphMatriks29(int vertex) {
        this.vertex = vertex;
        matrix = new int[vertex][vertex];
    }

    public void makeEdge(int source, int destination, int jarak) {
        matrix[source][destination] = jarak;
    }

    public void removeEdge(int source, int destination) {
        matrix[source][destination] = -1;
    }

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

Kode program class GraphMain29 :

```java
public class GraphMain29 {
    Run | Debug
    public static void main(String[] args) {
        Graph29 building = new Graph29(vertex:6);
        building.addEdge(source:0, destination:1, distance:50);
        building.addEdge(source:0, destination:2, distance:100);
        building.addEdge(source:1, destination:3, distance:70);
        building.addEdge(source:2, destination:3, distance:40);
        building.addEdge(source:3, destination:4, distance:60);
        building.addEdge(source:4, destination:5, distance:80);
        try {
            building.degree(source:0);
        } catch (Exception e) {
            System.out.println("Exception in degree method: " + e.getMessage());
        }
        try {
            building.printGraph();
        } catch (Exception e) {
            System.out.println("Exception in printGraph method: " + e.getMessage());
        }

        try {
            building.removeEdge(source:1, destination:3);
        } catch (Exception e) {
            System.out.println("Exception in removeEdge method: " + e.getMessage());
        }
        try {
            building.printGraph();
        } catch (Exception e) {
            System.out.println("Exception in printGraph method: " + e.getMessage());
        }

        System.out.println(x:"====================================");
        GraphMatriks29 bdg = new GraphMatriks29(vertex:4);
        bdg.makeEdge(source:0, destination:1, jarak:50);
        bdg.makeEdge(source:1, destination:0, jarak:60);
        bdg.makeEdge(source:1, destination:2, jarak:70);
        bdg.makeEdge(source:2, destination:1, jarak:80);
        bdg.makeEdge(source:2, destination:3, jarak:40);
        bdg.makeEdge(source:3, destination:0, jarak:90);
        bdg.printGraph();
        System.out.println();
        System.out.println(x:"Hasil setelah penghapusan edge : ");
        bdg.removeEdge(source:2, destination:1);
        bdg.printGraph();
    }
}
```

Output :

```
Building A : Building A (0 m), Building B (50 m), Building C (0 m), Building D (0 m),
Building B : Building A (60 m), Building B (0 m), Building C (70 m), Building D (0 m),
Building C : Building A (0 m), Building B (80 m), Building C (0 m), Building D (40 m),
Building D : Building A (90 m), Building B (0 m), Building C (0 m), Building D (0 m),

Hasil setelah penghapusan edge :
Building A : Building A (0 m), Building B (50 m), Building C (0 m), Building D (0 m),
Building B : Building A (60 m), Building B (0 m), Building C (70 m), Building D (0 m),
Building C : Building A (0 m), Building C (0 m), Building D (40 m),
Building D : Building A (90 m), Building B (0 m), Building C (0 m), Building D (0 m),
PS D:\KULIAH\college\smt 2\29_yonanda_asd\P15>
```

Pertanyaan :

1. Perbaiki kode program Anda apabila terdapat error atau hasil kompilasi kode tidak sesuai!
   Jawab : Kode program tidak ada yang error dan kompilasi sesuai

2. Apa jenis graph yang digunakan pada Percobaan 2?
   Jawab : Pada Percobaan 2, jenis graf yang digunakan adalah graf berarah dengan bobot (Directed Weighted Graph).

3. Apa maksud dari dua baris kode berikut?
   ```
   gdg.makeEdge(1, 2, 70);
   gdg.makeEdge(2, 1, 80);
   ```
   Jawab : Dua baris kode tersebut digunakan untuk membuat edge (sisi) antara dua simpul dalam graf.
   - gdg.makeEdge(1, 2, 70); membuat edge dari simpul 1 ke simpul 2 dengan bobot 70. Dalam konteks ini, bobot bisa diartikan sebagai jarak antara dua bangunan.
   - gdg.makeEdge(2, 1, 80); membuat edge dari simpul 2 ke simpul 1 dengan bobot 80.

4. Modifikasi kode program sehingga terdapat method untuk menghitung degree, termasuk inDegree dan outDegree!
   Jawab :
   - Kode program class Graph29 :

```java
public int calculateDegree(int vertex) {
    int degree = 0;
    for (int i = 0; i < this.vertex; i++) {
        if (matrix[vertex][i] != -1) {
            degree++;
        }
        if (matrix[i][vertex] != -1) {
            degree++;
        }
    }
    return degree;
}

public int calculateInDegree(int vertex) {
    int inDegree = 0;
    for (int i = 0; i < this.vertex; i++) {
        if (matrix[i][vertex] != -1) {
            inDegree++;
        }
    }
    return inDegree;
}

public int calculateOutDegree(int vertex) {
    int outDegree = 0;
    for (int i = 0; i < this.vertex; i++) {
        if (matrix[vertex][i] != -1) {
            outDegree++;
        }
    }
    return outDegree;
}
```

- Kode program class GraphMain29 :

```java
// Calculate and print degree, inDegree, and outDegree of each vertex
for (int i = 0; i < 4; i++) {
    System.out.println("Degree of vertex " + i + ": " + bdg.calculateDegree(i));
    System.out.println("InDegree of vertex " + i + ": " + bdg.calculateInDegree(i));
    System.out.println("OutDegree of vertex " + i + ": " + bdg.calculateOutDegree(i));
}
```

- Output :

```
Degree of vertex 0: 8
InDegree of vertex 0: 4
OutDegree of vertex 0: 4
Degree of vertex 1: 7
InDegree of vertex 1: 3
OutDegree of vertex 1: 4
Degree of vertex 2: 7
InDegree of vertex 2: 4
OutDegree of vertex 2: 3
Degree of vertex 3: 8
InDegree of vertex 3: 4
OutDegree of vertex 3: 4
PS D:\KULIAH\college\smt 2\29_yonanda_asd\P15>
```

TUGAS

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

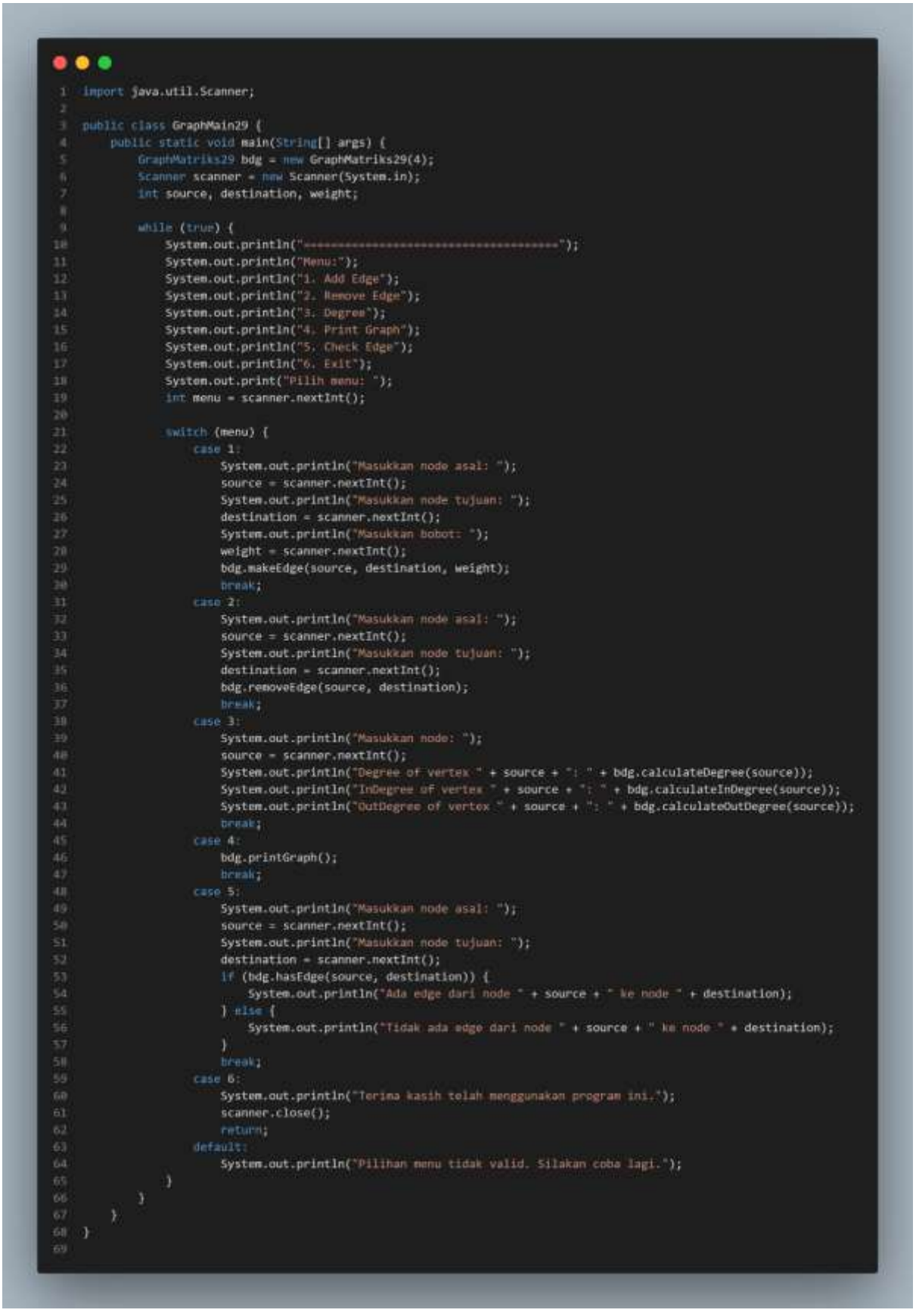
   Pengguna dapat memilih menu program melalui input Scanner

   Jawab :

   - Kode program class GraphMatriks29 :

```
public boolean hasEdge(int source, int destination) {
    // Periksa apakah bobot antara source dan destination tidak nol
    return matrix[source][destination] != -1;
}
```

   - Kode program class GraphMain29 :



```java
import java.util.Scanner;

public class GraphMain29 {
    public static void main(String[] args) {
        GraphMatriks29 bdg = new GraphMatriks29(4);
        Scanner scanner = new Scanner(System.in);
        int source, destination, weight;

        while (true) {
            System.out.println("==================================");
            System.out.println("Menu:");
            System.out.println("1. Add Edge");
            System.out.println("2. Remove Edge");
            System.out.println("3. Degree");
            System.out.println("4. Print Graph");
            System.out.println("5. Check Edge");
            System.out.println("6. Exit");
            System.out.print("Pilih menu: ");
            int menu = scanner.nextInt();

            switch (menu) {
                case 1:
                    System.out.println("Masukkan node asal: ");
                    source = scanner.nextInt();
                    System.out.println("Masukkan node tujuan: ");
                    destination = scanner.nextInt();
                    System.out.println("Masukkan bobot: ");
                    weight = scanner.nextInt();
                    bdg.makeEdge(source, destination, weight);
                    break;
                case 2:
                    System.out.println("Masukkan node asal: ");
                    source = scanner.nextInt();
                    System.out.println("Masukkan node tujuan: ");
                    destination = scanner.nextInt();
                    bdg.removeEdge(source, destination);
                    break;
                case 3:
                    System.out.println("Masukkan node: ");
                    source = scanner.nextInt();
                    System.out.println("Degree of vertex " + source + ": " + bdg.calculateDegree(source));
                    System.out.println("InDegree of vertex " + source + ": " + bdg.calculateInDegree(source));
                    System.out.println("OutDegree of vertex " + source + ": " + bdg.calculateOutDegree(source));
                    break;
                case 4:
                    bdg.printGraph();
                    break;
                case 5:
                    System.out.println("Masukkan node asal: ");
                    source = scanner.nextInt();
                    System.out.println("Masukkan node tujuan: ");
                    destination = scanner.nextInt();
                    if (bdg.hasEdge(source, destination)) {
                        System.out.println("Ada edge dari node " + source + " ke node " + destination);
                    } else {
                        System.out.println("Tidak ada edge dari node " + source + " ke node " + destination);
                    }
                    break;
                case 6:
                    System.out.println("Terima kasih telah menggunakan program ini.");
                    scanner.close();
                    return;
                default:
                    System.out.println("Pilihan menu tidak valid. Silakan coba lagi.");
            }
        }
    }
}
```

- Output :

```
Menu:
1. Add Edge
2. Remove Edge
3. Degree
4. Print Graph
5. Check Edge
6. Exit
Pilih menu: 1
Masukkan node asal:
0
Masukkan node tujuan:
1
Masukkan bobot:
2
===================================
Menu:
1. Add Edge
2. Remove Edge
3. Degree
4. Print Graph
5. Check Edge
6. Exit
Pilih menu: 4
Building A : Building A (0 m), Building B (2 m), Building C (0 m), Building D (0 m),
Building B : Building A (0 m), Building B (0 m), Building C (0 m), Building D (0 m),
Building C : Building A (0 m), Building B (0 m), Building C (0 m), Building D (0 m),
Building D : Building A (0 m), Building B (0 m), Building C (0 m), Building D (0 m),
===================================
```

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

   Jawab :

   - Kode program class GraphMatriks29 :

```
public void updateJarak(int source, int destination, int newWeight) {
    if (source >= 0 && source < vertex && destination >= 0 && destination < vertex) {
        matrix[source][destination] = newWeight;
    } else {
        System.out.println(x:"Invalid source or destination");
    }
}
```

   - Kode program class GraphMain29:

```java
case 7:
    System.out.println(x:"Masukkan node asal: ");
    source = scanner.nextInt();
    System.out.println(x:"Masukkan node tujuan: ");
    destination = scanner.nextInt();
    System.out.println(x:"Masukkan jarak baru: ");
    int newWeight = scanner.nextInt();
    bdg.updateJarak(source, destination, newWeight);
    break;
default:
    System.out.println(x:"Pilihan menu tidak valid. Silakan coba lagi.");
```

- Output :

```
Menu:
1. Add Edge
2. Remove Edge
3. Degree
4. Print Graph
5. Check Edge
6. Exit
7. Update Jarak
Pilih menu: 1
Masukkan node asal:
0
Masukkan node tujuan:
1
Masukkan bobot:
2
===================================
Menu:
1. Add Edge
2. Remove Edge
3. Degree
4. Print Graph
5. Check Edge
6. Exit
7. Update Jarak
Pilih menu: 7
Masukkan node asal:
0
Masukkan node tujuan:
1
Masukkan jarak baru:
4
```

```
Menu:
1. Add Edge
2. Remove Edge
3. Degree
4. Print Graph
5. Check Edge
6. Exit
7. Update Jarak
Pilih menu: 4
Building A : Building A (0 m), Building B (4 m), Building C (0 m), Building D (0 m),
Building B : Building A (0 m), Building B (0 m), Building C (0 m), Building D (0 m),
Building C : Building A (0 m), Building B (0 m), Building C (0 m), Building D (0 m),
Building D : Building A (0 m), Building B (0 m), Building C (0 m), Building D (0 m),
```

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

   Jawab :

   - Kode program class GraphMatriks29 :

```
public int hitungEdge() {
    int count = 0;
    for (int i = 0; i < vertex; i++) {
        for (int j = 0; j < vertex; j++) {
            if (matrix[i][j] != -1) {
                count++;
            }
        }
    }
    return count;
}
```

   - Kode program class GraphMain29 :

```
case 8:
    System.out.println("Jumlah edge dalam graf: " + bdg.hitungEdge());
    break;
```

   - Output :

```
Menu:
1. Add Edge
2. Remove Edge
3. Degree
4. Print Graph
5. Check Edge
6. Exit
7. Update Jarak
8. Hitung Edge
Pilih menu: 1
Masukkan node asal:
0
Masukkan node tujuan:
1
Masukkan bobot:
3
==================================
Menu:
1. Add Edge
2. Remove Edge
3. Degree
4. Print Graph
5. Check Edge
6. Exit
7. Update Jarak
8. Hitung Edge
Pilih menu: 1
Masukkan node asal:
1
Masukkan node tujuan:
2
Masukkan bobot:
4
```

```
Menu:
1. Add Edge
2. Remove Edge
3. Degree
4. Print Graph
5. Check Edge
6. Exit
7. Update Jarak
8. Hitung Edge
Pilih menu: 8
Jumlah edge dalam graf: 16
==================================
```