

## JOBSHEET 12-GRAPH

### 2. Praktikum

#### 2.1 Implementasi Graph menggunakan Linked List

##### 2.1.1 Tahapan Percobaan

Waktu percobaan (30 menit)

Pada percobaan ini akan diimplementasikan Graph menggunakan Linked Lists untuk merepresentasikan graph adjacency. Silakan lakukan langkah-langkah praktikum sebagai berikut.

1. Buatlah class Node, dan class Linked Lists sesuai dengan praktikum Double Linked Lists.
2. Tambahkan class Graph yang akan menyimpan method-method dalam graph dan juga method main().
3. Di dalam class Graph, tambahkan atribut vertex bertipe integer dan list[] bertipe LinkedList.
4. Tambahkan konstruktor default untuk menginisialisasi variabel vertex dan menambahkan perulangan untuk jumlah vertex sesuai dengan jumlah length array yang telah ditentukan.
5. Tambahkan method addEdge(). Jika yang akan dibuat adalah graph berarah, maka yang dijalankan hanya baris pertama saja. Jika graph tidak berarah yang dijalankan semua baris pada method addEdge().
6. Tambahkan method degree() untuk menampilkan jumlah derajat lintasan pada suatu vertex. Di dalam metode ini juga dibedakan manakah statement yang digunakan untuk graph berarah atau graph tidak berarah. Eksekusi hanya sesuai kebutuhan saja.
7. Tambahkan method removeEdge(). Method ini akan menghapus lintasan ada suatu graph. Oleh karena itu, dibutuhkan 2 parameter untuk menghapus lintasan yaitu source dan destination.
8. Tambahkan method removeAllEdges() untuk menghapus semua vertex yang ada di dalam graph.
9. Tambahkan method printGraph() untuk mencatat graph ter-update.
10. Compile dan jalankan method main() dalam class Graph untuk menambahkan beberapa edge pada graph, kemudian tampilkan. Setelah itu keluarkan hasilnya menggunakan pemanggilan method main(). Keterangan: degree harus disesuaikan dengan jenis graph yang telah dibuat (directed/undirected).

## Class Node

```
...java | Node.java | DoubleLinkedListsMain.java | AntriVaksin.java | NodeVaksin.java | AntriVaksinMain.java |
Source | History |
1 | 1 | /*
2 | 2 |  * To change this license header, choose License Headers in Project Properties.
3 | 3 |  * To change this template file, choose Tools | Templates
4 | 4 |  * and open the template in the editor.
5 | 5 |  */
6 | 6 |
7 | 7 | package doublelinkedlists;
8 | 8 |
9 | 9 |
10 | 10 | /**
11 | 11 |  *
12 | 12 |  * @author Hp
13 | 13 |  */
14 | 14 | class Node {
15 | 15 |     int data;
16 | 16 |     Node prev, next;
17 | 17 |
18 | 18 |     Node(Node prev, int data, Node next){
19 | 19 |         this.prev=prev;
20 | 20 |         this.data=data;
21 | 21 |         this.next=next;
22 | 22 |     }
23 | 23 | }
24 | 24 |
```

## Class DoubleLinkedLists

```
...java | DoubleLinkedLists.java | Node.java | DoubleLinkedListsMain.java | AntriVaksin.java | NodeVaksin.java |
Source | History |
1 | 1 | /*
2 | 2 |  * To change this license header, choose License Headers in Project Properties.
3 | 3 |  * To change this template file, choose Tools | Templates
4 | 4 |  * and open the template in the editor.
5 | 5 |  */
6 | 6 |
7 | 7 | package doublelinkedlists;
8 | 8 |
9 | 9 |
10 | 10 | public class DoubleLinkedLists {
11 | 11 |     Node head;
12 | 12 |     int size;
13 | 13 |
14 | 14 |     public DoubleLinkedLists(){
15 | 15 |         head = null;
16 | 16 |         size = 0;
17 | 17 |     }
18 | 18 |     public boolean isEmpty(){
19 | 19 |         return head == null;
20 | 20 |     }
21 | 21 |     public void addFirst (int item){
22 | 22 |         if (isEmpty()){
23 | 23 |             head = new Node (null, item, null);
24 | 24 |         }else {
25 | 25 |             Node newNode = new Node (null, item, head);
26 | 26 |             head.prev = newNode;
27 | 27 |             head = newNode;
28 | 28 |         }
29 | 29 |         size++;
30 | 30 |     }
31 | 31 |     public void addLast (int item){
32 | 32 |         if (isEmpty()){
33 | 33 |             addFirst(item);
34 | 34 |         }else{
35 | 35 |             Node current = head;
36 | 36 |             while (current.next != null){
37 | 37 |                 current = current.next;
38 | 38 |             }
39 | 39 |             Node newNode = new Node (current, item, null);
40 | 40 |             current.next = newNode;
41 | 41 |             size++;
42 | 42 |         }
43 | 43 |     }
44 | 44 |     public void add (int item, int index) throws Exception{
45 | 45 |         if (isEmpty()){
46 | 46 |             addFirst (item);
47 | 47 |         }else if (index < 0 || index > size){
48 | 48 |             throw new Exception("Nilai indeks di luar batas");
49 | 49 |         } else {
50 | 50 |             Node current = head;
51 | 51 |             int i = 0;
52 | 52 |             while (i < index) {
53 | 53 |                 current = current.next;
54 | 54 |                 i++;
55 | 55 |             }
56 | 56 |             if (current.prev == null){
57 | 57 |                 Node newNode = new Node (null, item, current);
58 | 58 |                 current.prev = newNode;
59 | 59 |                 head = newNode;
60 | 60 |             }else{
61 | 61 |                 Node newNode = new Node (current.prev, item, current);
62 | 62 |                 newNode.prev = current.prev;
63 | 63 |                 newNode.next = current;
64 | 64 |                 current.prev.next = newNode;
65 | 65 |                 current.prev = newNode;
66 | 66 |             }
67 | 67 |             size++;
68 | 68 |         }
69 | 69 |     }
70 | 70 | }
```

```

69 public int size(){
70     return size;
71 }
72 public void clear(){
73     head = null;
74     size = 0;
75 }
76 public void print(){
77     if (!isEmpty()){
78         Node tmp = head;
79         while (tmp != null){
80             System.out.print (tmp.data + "\t");
81             tmp = tmp.next;
82         }
83         System.out.println("\nberhasil diisi");
84     }else {
85         System.out.println("Linked Lists Kosong");
86     }
87 }
88

```

```

90 public void removeFirst() throws Exception{
91     if (isEmpty()){
92         throw new Exception ("Linked List masih kosong, tidak dapat dihapus!");
93     }else if (size == 1){
94         removeLast();
95     }else {
96         head = head.next;
97         head.prev = null;
98         size--;
99     }
100 }
101 public void removeLast () throws Exception{
102     if (isEmpty()){
103         throw new Exception("Linked List masih kosong, tidak dapat dihapus!");
104     }else if (head.next == null){
105         head = null;
106         size--;
107         return;
108     }
109     Node current = head;
110     while (current.next.next != null){
111         current = current.next;
112     }
113     current.next = null;
114     size--;
115 }
116 public void remove (int index) throws Exception{
117     if (isEmpty() || index >= size) {
118         throw new Exception ("Nilai indeks di luar batas");
119     }else if (index == 0){
120         removeFirst();
121     }else{
122         Node current = head;
123         int i = 0;
124         while (i < index){
125             current = current.next;
126             i++;
127         }
128         if (current.next == null){
129             current.prev.next = null;
130         }else if (current.prev == null){
131             current = current.next;
132             current.prev = null;
133             head = current;
134         }else {
135             current.prev.next = current.next;
136             current.next.prev = current.prev;
137         }
138         size--;
139     }
140 }
141 }
142
143 public int getFirst() throws Exception{
144     if (isEmpty()){
145         throw new Exception ("Linked List Kosong");
146     }
147     return head.data;
148 }
149 public int getLast () throws Exception{
150     if (isEmpty()){
151         throw new Exception ("Linked List Kosong");
152     }
153     Node tmp = head;
154     while (tmp.next != null){
155         tmp = tmp.next;
156     }
157     return tmp.data;
158 }
159 public int get (int index) throws Exception {
160     if (isEmpty () || index >= size){
161         throw new Exception ("Nilai indeks di luar batas.");
162     }
163     Node tmp = head;
164     for (int i = 0; i < index; i++){
165         tmp = tmp.next;
166     }
167     return tmp.data;
168 }
169 }
170

```

## Class Graph

```
7 package doublelinkedlists;
8
9 /**
10  *
11  * @author Hp
12  */
13 public class Graph {
14     int vertex;
15     DoubleLinkedLists list [];
16
17     public Graph (int vertex){
18         this.vertex = vertex;
19         list = new DoubleLinkedLists[vertex];
20         for (int i = 0; i < vertex; i++){
21             list [i] = new DoubleLinkedLists();
22         }
23     }
24     public void addEdge (int source, int destination){
25         //add edge
26         list[source].addFirst(destination);
27
28         //add back edge (for undirected)
29         list [destination].addFirst(source);
30     }
31     public void degree(int source) throws Exception{
32
33         //degree undirected graph
34         System.out.println("degree vertex" + source + " : " +list[source].size());
35
36         //degree directed graph
37         //inDegree
38         int k, totalIn = 0, totalOut = 0;
39         for (int i = 0; i< vertex; i++){
40             for (int j = 0; j< list [i].size(); j++){
41                 if(list [i].get(j)==source)
42                     ++totalIn;
43             }
44             //outDegree
45             for (k = 0; k < list [source].size(); k++){
46                 list [source].get (k);
47             }
48             totalOut = k;
49         }
50         System.out.println("Indegree dari vertex"+ source + " : " + totalIn);
51         System.out.println("Outdegree dari vertex"+ source + " : " + totalOut);
52         System.out.println("degree dari vertex"+ source + " : " + (totalIn+totalOut));
53     }
54     public void removeEdge (int source, int destination) throws Exception{
55         for (int i = 0; i < vertex; i++){
56             if(i ==destination){
57                 list[source].remove(destination);
58             }
59         }
60     }
61
62     public void removeAllEdges(){
63         for (int i = 0; i < vertex; i++){
64             list[i].clear();
65         }
66         System.out.println("Graph Berhasil Dikosongkan");
67     }
68     public void printGraph() throws Exception{
69         for (int i = 0; i <vertex; i++){
70             if(list[i].size()>0){
71                 System.out.print("Vertex" + i + " terhubung dengan ");
72                 for(int j = 0; j < list[i].size(); j++){
73                     System.out.print(list[i].get(j) + " ");
74                 }
75                 System.out.println("");
76             }
77         }
78         System.out.println(" ");
79     }
80 }
```

## Class Main

```

1  /*
2   * To change this license header, choose License Headers in Project Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6
7   package doublelinkedlists;
8   /**
9    *
10   * @author Hp
11   */
12   public class DoubleLinkedListsMain {
13       public static void main(String [] args) throws Exception{
14
15
16           Graph graph = new Graph(6);
17           graph.addEdge(0, 1);
18           graph.addEdge(0, 4);
19           graph.addEdge(1, 2);
20           graph.addEdge(1, 3);
21           graph.addEdge(1, 4);
22           graph.addEdge(2, 3);
23           graph.addEdge(3, 4);
24           graph.addEdge(3, 0);
25           graph.printGraph();
26           graph.degree(2);
27

```

11. Amati hasil running tersebut.

```

DoubleLinkedLists - NetBeans IDE 7.4
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Output - DoubleLinkedLists (run)

run:
Vertex0 terhubung dengan 3 4 1
Vertex1 terhubung dengan 4 2 2 0
Vertex2 terhubung dengan 3 1
Vertex3 terhubung dengan 0 4 2 1
Vertex4 terhubung dengan 2 1 0

degree vertex2 : 2
Indegree dari vertex2 : 2
Outdegree dari vertex2 : 2
degree dari vertex2 : 4
BUILD SUCCESSFUL (total time: 0 seconds)

```

12. Tambahkan pemanggilan method removeEdge() sesuai potongan code di bawah ini pada method main(). Kemudian tampilkan graph tersebut.

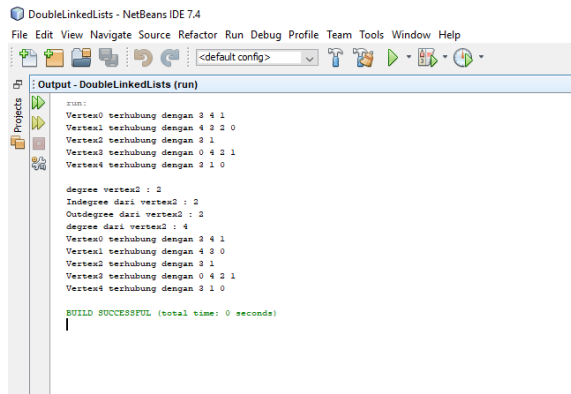
13. Amati hasil running tersebut.

14. Uji coba penghapusan lintasan yang lain! Amati hasilnya!

```

1  /*
2   * To change this license header, choose License Headers in Project Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6
7   package doublelinkedlists;
8   /**
9    *
10   * @author Hp
11   */
12   public class DoubleLinkedListsMain {
13       public static void main(String [] args) throws Exception{
14
15
16           Graph graph = new Graph(6);
17           graph.addEdge(0, 1);
18           graph.addEdge(0, 4);
19           graph.addEdge(1, 2);
20           graph.addEdge(1, 3);
21           graph.addEdge(1, 4);
22           graph.addEdge(2, 3);
23           graph.addEdge(3, 4);
24           graph.addEdge(3, 0);
25           graph.printGraph();
26           graph.degree(2);
27           graph.removeEdge(1,2);
28           graph.printGraph();
29
30

```



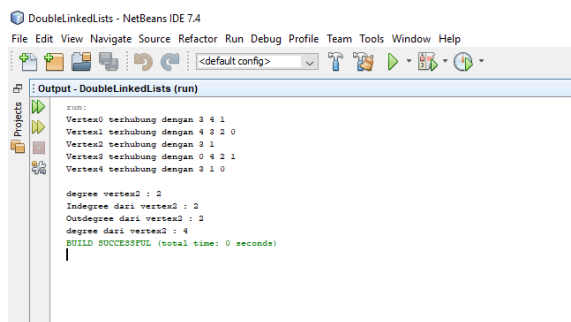
```
DoubleLinkedLists - NetBeans IDE 7.4
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config>
Output - DoubleLinkedLists (run)
run:
Vertex0 terhubung dengan 3 4 1
Vertex1 terhubung dengan 4 2 0
Vertex2 terhubung dengan 3 1
Vertex3 terhubung dengan 0 4 2 1
Vertex4 terhubung dengan 2 1 0

degree vertex2 : 2
Indegree dari vertex2 : 2
Outdegree dari vertex2 : 2
degree dari vertex0 : 4
Vertex0 terhubung dengan 3 4 1
Vertex1 terhubung dengan 4 2 0
Vertex2 terhubung dengan 3 1
Vertex3 terhubung dengan 0 4 2 1
Vertex4 terhubung dengan 2 1 0

BUILD SUCCESSFUL (total time: 0 seconds)
```

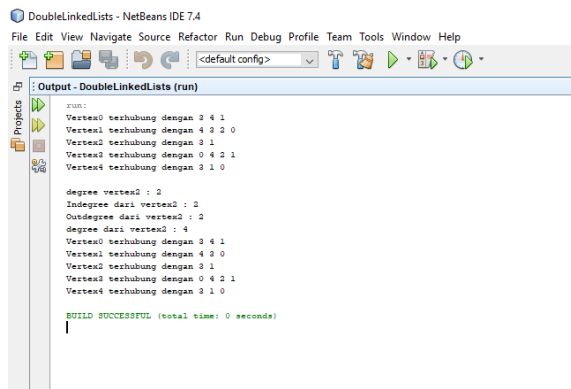
## 2.1.2 Verifikasi Hasil Percobaan

Verifikasi hasil kompilasi kode program Anda dengan gambar berikut ini.



```
DoubleLinkedLists - NetBeans IDE 7.4
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config>
Output - DoubleLinkedLists (run)
run:
Vertex0 terhubung dengan 3 4 1
Vertex1 terhubung dengan 4 2 0
Vertex2 terhubung dengan 3 1
Vertex3 terhubung dengan 0 4 2 1
Vertex4 terhubung dengan 2 1 0

degree vertex2 : 2
Indegree dari vertex2 : 2
Outdegree dari vertex2 : 2
degree dari vertex0 : 4
BUILD SUCCESSFUL (total time: 0 seconds)
```



```
DoubleLinkedLists - NetBeans IDE 7.4
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config>
Output - DoubleLinkedLists (run)
run:
Vertex0 terhubung dengan 3 4 1
Vertex1 terhubung dengan 4 2 0
Vertex2 terhubung dengan 3 1
Vertex3 terhubung dengan 0 4 2 1
Vertex4 terhubung dengan 2 1 0

degree vertex2 : 2
Indegree dari vertex2 : 2
Outdegree dari vertex2 : 2
degree dari vertex0 : 4
Vertex0 terhubung dengan 3 4 1
Vertex1 terhubung dengan 4 2 0
Vertex2 terhubung dengan 3 1
Vertex3 terhubung dengan 0 4 2 1
Vertex4 terhubung dengan 2 1 0

BUILD SUCCESSFUL (total time: 0 seconds)
```

## 2.1.3 Pertanyaan Percobaan

1. Sebutkan beberapa jenis (minimal 3) algoritma yang menggunakan dasar Graph, dan apakah kegunaan algoritma-algoritma tersebut?  
beberapa jenis algoritma yang menggunakan dasar Graph yang pertama adalah Algoritma Brent yang digunakan untuk menentukan adanya jalur pada graph yang kedua adalah Algoritma Floyd yang digunakan untuk menentukan adanya jalur pada graph dan yang ketiga adakah Algoritma Hungaria yang digunakan untuk penjadwalan yang sempurna.

2. Pada class Graph terdapat array bertipe LinkedList, yaitu LinkedList list[]. Apakah tujuan pembuatan variabel tersebut ?  
 Pada class Graph terdapat array bertipe LinkedList, yaitu LinkedList list[] dengan tujuan untuk membuat variable LinkedList[] yang nantinya akan digunakan untuk memanggil fungsi linked list dan diisi oleh vertex pada linked list.
3. Apakah alasan pemanggilan method addFirst() untuk menambahkan data, bukan method add jenis lain pada linked list ketika digunakan pada method addEdge pada class Graph?  
 alasan pemanggilan method addFirst() untuk menambahkan data, bukan method add jenis lain pada linked list ketika digunakan pada method addEdge pada class Graph adalah untuk mengenalkan vertex tersebut dan koneksinya.
4. Bagaimana cara mendeteksi prev pointer pada saat akan melakukan penghapusan suatu edge pada graph ?  
 cara mendeteksi prev pointer pada saat akan melakukan penghapusan suatu edge pada graph adalah dengan cara melakukan looping vertex pada saat melakukan penghapusan.
5. Kenapa pada praktikum 2.1.1 langkah ke-12 untuk menghapus path yang bukan merupakan lintasan pertama kali menghasilkan output yang salah ? Bagaimana solusinya ?

### **2.2.1 Tahapan Percobaan**

Waktu percobaan: 30 menit

Pada praktikum 2.2 ini akan diimplementasikan Graph menggunakan matriks untuk merepresentasikan graph adjacency. Silakan lakukan langkah-langkah praktikum sebagai berikut.

1. Uji coba graph bagian 2.2 menggunakan array 2 dimensi sebagai representasi graph. Buatlah class graphArray yang didalamnya terdapat variabel vertices dan array twoD\_array!
2. Buatlah konstruktor graphArray sebagai berikut!
3. Untuk membuat suatu lintasan maka dibuat method makeEdge() sebagai berikut. Untuk menampilkan suatu lintasan diperlukan pembuatan method getEdge() berikut.
4. Kemudian buatlah method main() seperti berikut ini.
5. Jalankan class graphArray dan amati hasilnya!

## Class GraphArray

```
package graph;

public class GraphArray {
    private final int vertices;
    private final int[][] twoD_array;

    public GraphArray(int v) {
        vertices = v;
        twoD_array = new int [vertices + 1][vertices + 1];
    }

    public void makeEdge (int to, int from, int edge){
        try{
            twoD_array[to][from] = edge;
        }catch (ArrayIndexOutOfBoundsException index){
            System.out.println("Vertex tidak ada");
        }
    }

    public int getEdge (int to, int from){
        try{
            return twoD_array[to][from];
        }
        catch (ArrayIndexOutOfBoundsException index){
            System.out.println("vertex tidak ada");
        }
        return -1;
    }
}
```

## Class Main

```
package graph;

import java.util.Scanner;

/**
 * @author Hp
 */
public class GraphMain {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        int v, e, count =1, to=0, from=0;
        Scanner sc= new Scanner(System.in);
        GraphArray graph;
        try{
            System.out.println("Masukkan jumlah vertices: ");
            v = sc.nextInt();
            System.out.println("Masukkan jumlah edge: ");
            e = sc.nextInt();

            graph = new GraphArray(v);

            System.out.println("Masukkan edge: <to> <from>");
            while(count <= e){
                to = sc.nextInt();
                from = sc.nextInt();

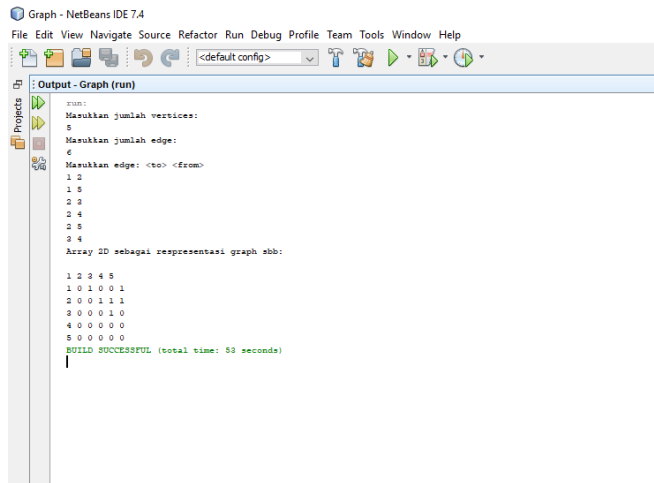
                graph.makeEdge(to, from, 1);
                count++;
            }

            System.out.println("Array 2D sebagai representasi graph sbb: ");
            System.out.println(" ");
            for(int i=1; i<=v; i++){
                System.out.print(i + " ");
            }
            System.out.println();

            for(int i=1; i<=v; i++){
                System.out.print(i + " ");
                for(int j=1; j<=v; j++){
                    System.out.print(graph.getEdge(i, j) + " ");
                }
                System.out.println();
            }
        }
        catch (Exception E){
            System.out.println("Error. Silahkan cek kembali\n" + E.getMessage());
        }
        sc.close();
    }
}
```

## Output





```
Graph - NetBeans IDE 7.4
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Output - Graph (run)
run:
Masukkan jumlah vertices:
5
Masukkan jumlah edge:
6
Masukkan edge: <to> <from>
1 2
1 5
2 2
2 4
2 5
2 4
Array 2D sebagai representasi graph abb:
1 2 2 4 5
1 0 1 0 0 1
2 0 0 1 1 1
3 0 0 0 1 0
4 0 0 0 0 0
5 0 0 0 0 0
BUILD SUCCESSFUL (total time: 53 seconds)
```

### 2.2.3 Pertanyaan Percobaan

1. Apakah perbedaan degree/derajat pada directed dan undirected graph?  
perbedaan degree/derajat pada directed dan undirected graph terdapat pada directed degreeIn dengan DegreeOut yang memiliki perbedaan namun pada undirected degreeIn dengan DegreeOut nya sama.
2. Pada implementasi graph menggunakan adjacency matriks. Kenapa jumlah vertices harus ditambahkan dengan 1 pada indeks array berikut?  
Pada implementasi graph menggunakan adjacency matriks. Kenapa jumlah vertices harus ditambahkan dengan 1 pada indeks array berikut dikarenakan index dimulai dari 0, sehingga perlu ditambahkan dengan 1
3. Apakah kegunaan method getEdge() ?  
kegunaan method getEdge() adalah untuk menampilkan suatu lintasan yang dibutuhkan
4. Termasuk jenis graph apakah uji coba pada praktikum 2.2?  
uji coba pada praktikum 2.2 termasuk ke dalam Graph directed
5. Mengapa pada method main harus menggunakan try-catch Exception ?  
pada method main harus menggunakan try-catch Exception dikarenakan pada method main harus memakai try-catch Exeception dengan tujuan penanganan proses error.

## 3. Tugas Praktikum

1. Ubahlah lintasan pada praktikum 2.1 menjadi inputan!

```

6
7 package doublelinkedlists;
8
9 import java.util.Scanner;
10
11 /**
12  *
13  * @author Hp
14  */
15 public class DoubleLinkedListsMain {
16     public static void main(String [] args) throws Exception{
17         Scanner sc = new Scanner (System.in);
18         System.out.print("Masukkan vertex graf : ");
19         int vertex = sc.nextInt();
20
21         Graph graph = new Graph (vertex);
22
23         System.out.print("Masukkan Banyak edge : ");
24         int edge = sc.nextInt();
25
26         for (int i = 0; i < edge; i++){
27             System.out.print("Masukkan Source : ");
28             int source = sc.nextInt();
29             System.out.print("Masukkan Destination : ");
30             int des = sc.nextInt();
31
32             graph.addEdge(source, des);
33         }
34         graph.printGraph();
35     }
36 }

```

DoubleLinkedLists - NetBeans IDE 7.4

File Edit View Navigate Source Refactor Run Debug Profile 1

<default config>

Output - DoubleLinkedLists (run)

```

run
Masukkan vertex graf : 6
Masukkan Banyak edge : 8
Masukkan Source : 0
Masukkan Destination : 1
Masukkan Source : 0
Masukkan Destination : 4
Masukkan Source : 1
Masukkan Destination : 3
Masukkan Source : 1
Masukkan Destination : 4
Masukkan Source : 2
Masukkan Destination : 3
Masukkan Source : 2
Masukkan Destination : 3
Masukkan Source : 1
Masukkan Destination : 3
Masukkan Source : 4
Masukkan Destination : 1
Vertex0 terhubung dengan 4 1
Vertex1 terhubung dengan 4 3 4 3 0
Vertex2 terhubung dengan 2 3
Vertex3 terhubung dengan 1 2 2 1
Vertex4 terhubung dengan 1 1 0
BUILD SUCCESSFUL (total time: 32 seconds)

```

2. Tambahkan method graphType dengan tipe boolean yang akan membedakan graph termasuk directed atau undirected graph. Kemudian update seluruh method yang berelasi dengan method graphType tersebut (hanya menjalankan statement sesuai dengan jenis graph) pada praktikum 2.1

```

79     public boolean graphType(int source, int destination){
80         list[source].addFirst(destination);
81         return true;
82     }
83 }
84

```

3. Modifikasi method removeEdge() pada praktikum 2.1 agar tidak menghasilkan output yang salah untuk path selain path pertama kali!

```

54     public void removeEdge (int source, int destination) throws Exception{
55         for (int i = 0; i < vertex; i++){
56             if(i ==destination){
57                 // list[source].remove(destination);
58                 list[source].remove(source);
59             }
60         }
61     }

```

4. Ubahlah tipe data vertex pada seluruh graph pada praktikum 2.1 dan 2.2 dari Integer menjadi tipe generic agar dapat menerima semua tipe data dasar Java! Misalnya setiap vertex yang awalnya berupa angka 0,1,2,3, dst. selanjutnya ubah menjadi suatu nama daerah seperti Gresik, Bandung, Yogya, Malang, dst.

## Class Graph

```

68 public void printGraph() throws Exception{
69     String Kota = " ";
70     for (int i = 0; i < vertex; i++){
71         if(list[i].size()>0){
72             if (i == 0 ){
73                 Kota = "Malang";
74             }else if (i == 1){
75                 Kota = "Surabaya";
76             }else if (i == 2){
77                 Kota = "Gresik";
78             }else if (i == 3){
79                 Kota = "Pandaan";
80             }else if (i == 4){
81                 Kota = "Bandung";
82             }else{
83                 Kota = "Jakarta";
84             }
85             System.out.print(" Vertex " + Kota + " terhubung dengan ");
86             for(int j = 0; j < list[i].size(); j++){
87                 System.out.print(list[i].get(j) + " ");
88             }
89             System.out.println("");
90         }
91     }
92     System.out.println(" ");
93 }

```

## Class Main

```

1  /*
2   * To change this license header, choose License Headers in Project Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6
7  package doublelinkedlists;
8
9  import java.util.Scanner;
10
11 public class DoubleLinkedListMain {
12     public static void main(String [] args) throws Exception{
13         Graph graph = new Graph(6);
14         graph.addEdge(0, 1);
15         graph.addEdge(0, 4);
16         graph.addEdge(1, 2);
17         graph.addEdge(1, 3);
18         graph.addEdge(1, 4);
19         graph.addEdge(2, 3);
20         graph.addEdge(3, 4);
21         graph.addEdge(3, 0);
22         graph.printGraph();
23         graph.degree(2);
24         graph.removeEdge(1,2);
25         graph.printGraph();
26         // Scanner sc = new Scanner (System.in);
27         // System.out.print("Masukkan vertex graf : ");

```

## Output

```

DoubleLinkedLists - NetBeans IDE 7.4
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config>
Output - DoubleLinkedLists (run)
run:
Vertex Malang terhubung dengan 3 4 1
Vertex Surabaya terhubung dengan 4 2 0
Vertex Gresik terhubung dengan 2 1
Vertex Pandaan terhubung dengan 0 4 2 1
Vertex Bandung terhubung dengan 2 1 0

degree vertex2 : 2
Indegree dari vertex2 : 2
Outdegree dari vertex2 : 2
degree dari vertex2 : 4
Vertex Malang terhubung dengan 2 4 1
Vertex Surabaya terhubung dengan 4 2 0
Vertex Gresik terhubung dengan 2 1
Vertex Pandaan terhubung dengan 0 4 2 1
Vertex Bandung terhubung dengan 2 1 0

BUILD SUCCESSFUL (total time: 0 seconds)

```