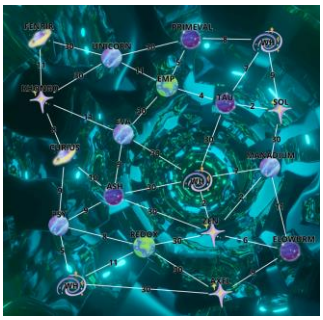
MODUL V

GRAPH

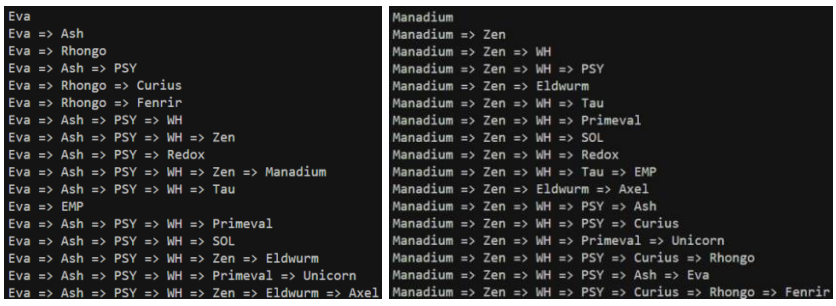
5.1 PERMASALAHAN

4.1.1 *Graph Hard* Level Verda Kalanta



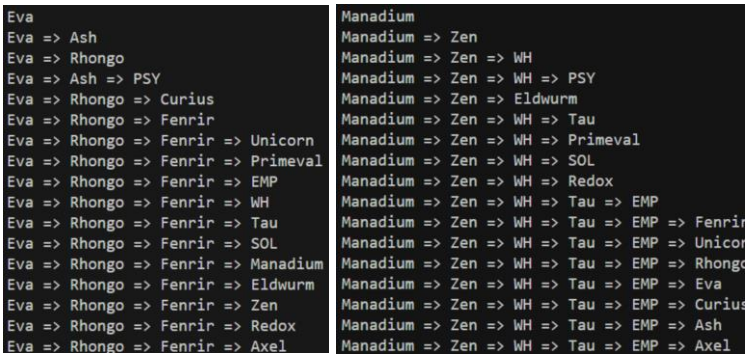
Gambar 5.1 *Graph* Permasalahan

Verda Kalanta adalah sesosok mahluk inter-stelar yang dimana memiliki kekuatan yang sangat dahsyat sehingga dengan mudahmenempuh perjalanan antar bintang. Disebuah galaxy verda sedang melakukan expedisi dalam rangkamengenali galaxy yang baru saja ia temukan, galaxy ini beberapahal unik dan baru pertama kali ia temukan. Gambar di samping adalah sebuah white hole yang menyambung antara white hole lain sehingga bisa membuat verda langsung berpindah secara instan antar white hole lainnya hal ini dapat memperpenda waktu perjalanan verda dalam mengaruhi alam semesta Bantulah verda melakukan mapping terhadap galaxy baru, dia menginginkan sebuah list dari satu tempat ke tempat lainnya untuk semua tempat



Gambar 4.2 Rute Perjalanan

Setelah berdiam beberapa juta tahun di galaxy ini verda menemukan sebuah trobosan baru yaitu sebuah teleportation device yang dia taruh di sebuah planet yang bernama fenrir, teleportation device ini memungkinkan fenrir untuk berpindah tempat di seluruh galxy secara instan, dengan teknologi baru ini bantukan kembali verda kalanta untuk melakukan mapping.



Gambar 4.3 Rute Setelah Ada Portal

5.2 HASIL PERCOBAAN

5.2.1 Soal Medium Modul IV Pohon

1. Algoritma
   a. Inisialisasi kelas untuk menyimpan graph, vertex, dan edge.
   b. Tambahkan vertex untuk merepresentasikan planet dalam soal.
   c. Tambahkan edge yang menghubungkan vertex.
   d. Melakukan pencarian rute terpendek menggunakan algoritma Dijkstra.
   e. Memasangkan perangkat teleportasi pada planet tertentu.
   f. Melakukan pencarian rute terpendek setelah penambahan perangkat teleportasi
2. *Source Code*

```
public class Vertex {
    int vertex;
    int weight;
    Vertex next;

    public Vertex(int vertex, int weight) {
        this.vertex = vertex;
        this.weight = weight;
        this.next = null;
    }
}

class Edge {
    Vertex head;
    public void add(int vertex, int weight) {
        Vertex newNode = new Vertex(vertex, weight);
        if (head == null) {
            head = newNode;
        } else {
            Vertex temp = head;
            while (temp.next != null) {
                temp = temp.next;
            }
            temp.next = newNode;
        }
    }
}

public class Graph {
    int vertices;
    Edge[] adjMatrix;
    String[] vertexNames;
    int[] teleportation;

    public Graph(int vertices) {
        this.vertices = vertices;
        adjMatrix = new Edge[vertices];
        vertexNames = new String[vertices];
        teleportation = new int[vertices];
        for (int i = 0; i < vertices; i++) {
            adjMatrix[i] = new Edge();
            teleportation[i] = -1;
        }
```

```
    }

    public void setVertexName(int index, String name) {
        vertexNames[index] = name;
    }

    public int getVertexIndex(String name) {
        for (int i = 0; i < vertices; i++) {
            if (vertexNames[i].equals(name)) {
                return i;
            }
        }
        return -1;
    }

    public  void  addEdge(String  source,  String  destination,  int
weight) {
        int sourceIndex = getVertexIndex(source);
        int destinationIndex = getVertexIndex(destination);
        if (sourceIndex == -1 || destinationIndex == -1) {
            System.out.println("Invalid vertex name.");
            return;
        }
        adjMatrix[sourceIndex].add(destinationIndex, weight);
        adjMatrix[destinationIndex].add(sourceIndex, weight);
    }

    public void shortestPath(String startName, String endName, String
ignorePlanet) {
        int start = getVertexIndex(startName);
        int end = getVertexIndex(endName);
        int     ignore     =     ignorePlanet     !=     null     ?
getVertexIndex(ignorePlanet) : -1;

        if (start == -1 || end == -1) {
            System.out.println("Invalid vertex name.");
            return;
        }

        int[] distance = new int[vertices];
        boolean[] visited = new boolean[vertices];
        int[] previous = new int[vertices];

        for (int i = 0; i < vertices; i++) {
            distance[i] = Integer.MAX_VALUE;
            visited[i] = false;
            previous[i] = -1;
        }

        distance[start] = 0;

        for (int count = 0; count < vertices - 1; count++) {
            int u = selectMinVertex(distance, visited);
            if (u == ignore) continue;
            visited[u] = true;

            if (teleportation[u] == end) {
                distance[end] = distance[u];
                previous[end] = u;
                break;
            }
```

```
            if (teleportation[u] != -1 && !visited[teleportation[u]])
{
                int teleportVertex = teleportation[u];
                if (distance[u] < distance[teleportVertex]) {
                    distance[teleportVertex] = distance[u];
                    previous[teleportVertex] = u;
                }
            }

            Vertex temp = adjMatrix[u].head;
            while (temp != null) {
                if (!visited[temp.vertex] && temp.vertex != ignore
&& distance[u] != Integer.MAX_VALUE &&
                        distance[u]      +      temp.weight      <
distance[temp.vertex]) {
                    distance[temp.vertex]    =    distance[u]    +
temp.weight;
                    previous[temp.vertex] = u;
                }
                temp = temp.next;
            }
        }

        if (distance[end] == Integer.MAX_VALUE) {
            System.out.println("No path found.");
        } else {
            printPath(previous, end);
            System.out.println();
        }
    }

    public void setTeleportation(String from, String to) {
        int fromIndex = getVertexIndex(from);
        int toIndex = getVertexIndex(to);
        if (fromIndex == -1 || toIndex == -1) {
            System.out.println("Invalid     vertex     name     for
teleportation.");
            return;
        }
        teleportation[fromIndex] = toIndex;
        addEdge(from, to, 0);
    }

    private int selectMinVertex(int[] distance, boolean[] visited) {
        int min = Integer.MAX_VALUE, index = -1;
        for (int i = 0; i < vertices; i++) {
            if (!visited[i] && distance[i] < min) {
                min = distance[i];
                index = i;
            }
        }
        return index;
    }

    private void printPath(int[] previous, int vertex) {
        if (previous[vertex] == -1) {
            System.out.print(vertexNames[vertex] + " ");
            return;
        }
        printPath(previous, previous[vertex]);
```

```
        System.out.print("=> " + vertexNames[vertex] + " ");
    }

    public void removeTeleport() {
        for (int i = 0; i < vertices; i++) {
            teleportation[i] = -1;
        }
        for (int i = 0; i < vertices; i++) {
            Vertex current = adjMatrix[i].head;
            Vertex prev = null;
            while (current != null) {
                if (current.weight == 0) {
                    if (prev == null) {
                        adjMatrix[i].head = current.next;
                    } else {
                        prev.next = current.next;
                    }
                } else {
                    prev = current;
                }
                current = current.next;
            }
        }
    }
}

public class Main{

    public static void main(String[] args) {
        Graph graph = new Graph(17);
        String[] names = {"wh", "zen", "manadium", "tau","emp",
"primeval", "psy",
        "eldwurm", "redox", "rhongo", "curius", "ash", "eva",
"unicorn", "Fenrir", "axel", "sol"};
for (int i = 0; i < names.length; i++) {
    graph.setVertexName(i, names[i]);
}
        graph.addEdge("wh", "zen", 2);
        graph.addEdge("zen", "manadium", 2);
        graph.addEdge("tau", "sol", 2);
        graph.addEdge("tau", "emp", 4);
        graph.addEdge("emp", "primeval", 5);
        graph.addEdge("psy", "wh", 5);
        graph.addEdge("zen", "eldwurm", 6);
        graph.addEdge("wh", "manadium", 7);
        graph.addEdge("wh", "tau", 7);
        graph.addEdge("psy", "redox", 8);
        graph.addEdge("rhongo", "curius", 8);
        graph.addEdge("wh", "primeval", 8);
        graph.addEdge("ash", "eva", 9);
        graph.addEdge("psy", "curius", 9);
        graph.addEdge("psy", "ash", 9);
        graph.addEdge("axel", "eldwurm", 9);
        graph.addEdge("wh", "sol", 9);
        graph.addEdge("curius", "ash", 10);
        graph.addEdge("unicorn", "primeval", 10);
        graph.addEdge("wh", "redox", 11);
        graph.addEdge("unicorn", "emp", 11);
        graph.addEdge("rhongo", "eva", 11);
        graph.addEdge("Fenrir", "rhongo", 11);
        graph.addEdge("manadium", "eldwurm", 11);
```

```
        graph.addEdge("Fenrir", "unicorn", 30);
        graph.addEdge("rhongo", "unicorn", 30);
        graph.addEdge("eva", "emp", 30);
        graph.addEdge("eva", "wh", 30);
        graph.addEdge("ash", "wh", 30);
        graph.addEdge("redox", "zen", 30);
        graph.addEdge("redox", "axel", 30);
        graph.addEdge("sol", "manadium", 30);
        graph.addEdge("wh", "tau", 30);
        graph.addEdge("wh", "axel", 30);
        graph.addEdge("ash", "zen", 30);

        System.out.println("eva");
        graph.shortestPath("eva", "ash",null);
        graph.shortestPath("eva", "rhongo",null);
        graph.shortestPath("eva", "psy",null);
        graph.shortestPath("eva", "curius",null);
        graph.shortestPath("eva", "Fenrir",null);
        graph.shortestPath("eva", "wh",null);
        graph.shortestPath("eva", "zen",null);
        graph.shortestPath("eva", "redox",null);
        graph.shortestPath("eva", "manadium",null);
        graph.shortestPath("eva", "tau",null);
        graph.shortestPath("eva", "emp",null);
        graph.shortestPath("eva", "primeval",null);
        graph.shortestPath("eva", "sol",null);
        graph.shortestPath("eva", "eldwurm",null);
        graph.shortestPath("eva", "unicorn",null);
        graph.shortestPath("eva", "axel",null);
        System.out.println();
        System.out.println("\nmanadium");
        graph.shortestPath("manadium", "zen",null);
        graph.shortestPath("manadium", "wh",null);
        graph.shortestPath("manadium", "psy",null);
        graph.shortestPath("manadium", "eldwurm",null);
        graph.shortestPath("manadium", "tau",null);
        graph.shortestPath("manadium", "primeval",null);
        graph.shortestPath("manadium", "sol",null);
        graph.shortestPath("manadium", "redox",null);
        graph.shortestPath("manadium", "emp",null);
        graph.shortestPath("manadium", "axel",null);
        graph.shortestPath("manadium", "ash",null);
        graph.shortestPath("manadium", "curius",null);
        graph.shortestPath("manadium", "unicorn",null);
        graph.shortestPath("manadium", "rhongo",null);
        graph.shortestPath("manadium", "eva",null);
        graph.shortestPath("manadium", "Fenrir",null);

        graph.setTeleportation("Fenrir", "unicorn");
        graph.setTeleportation("Fenrir", "primeval");
        graph.setTeleportation("Fenrir", "emp");
        graph.setTeleportation("Fenrir", "wh");
        graph.setTeleportation("Fenrir", "tau");
        graph.setTeleportation("Fenrir", "sol");
        graph.setTeleportation("Fenrir", "manadium");
        graph.setTeleportation("Fenrir", "eldwurm");
        graph.setTeleportation("Fenrir", "zen");
        graph.setTeleportation("Fenrir", "redox");
        graph.setTeleportation("Fenrir", "axel");
```

```
        System.out.println("\nSetelah kendaraan teleportasi ajaib
ditaruh di Fenrir\neva");
        graph.shortestPath("eva", "ash",null);
        graph.shortestPath("eva", "rhongo",null);
        graph.shortestPath("eva", "psy",null);
        graph.shortestPath("eva", "curius",null);
        graph.shortestPath("eva", "unicorn",null);
        graph.shortestPath("eva", "primeval",null);
        graph.shortestPath("eva", "emp",null);
        graph.shortestPath("eva", "wh",null);
        graph.shortestPath("eva", "tau",null);
        graph.shortestPath("eva", "sol",null);
        graph.shortestPath("eva", "manadium",null);
        graph.shortestPath("eva", "eldwurm",null);
        graph.shortestPath("eva", "zen",null);
        graph.shortestPath("eva", "redox",null);
        graph.shortestPath("eva", "axel",null);
        graph.removeTeleport();
        System.out.println("\nmanadium\nkendaraan       teleportasi
dipindahkan ke emp");
        graph.shortestPath("manadium", "zen",null);
        graph.shortestPath("manadium", "wh",null);
        graph.shortestPath("manadium", "psy",null);
        graph.shortestPath("manadium", "eldwurm",null);
        graph.shortestPath("manadium", "tau",null);
        graph.shortestPath("manadium", "primeval",null);
        graph.shortestPath("manadium", "sol",null);
        graph.shortestPath("manadium", "redox",null);
        graph.shortestPath("manadium", "emp",null);
        graph.shortestPath("manadium", "Fenrir",null);
        graph.setTeleportation("emp", "Fenrir");
        graph.setTeleportation("emp", "unicorn");
        graph.setTeleportation("emp", "rhongo");
        graph.setTeleportation("emp", "eva");
        graph.setTeleportation("emp", "curius");
        graph.setTeleportation("emp", "ash");
        graph.setTeleportation("emp", "axel");
        graph.shortestPath("manadium", "Fenrir",null);
        graph.shortestPath("manadium", "unicorn",null);
        graph.shortestPath("manadium", "rhongo",null);
        graph.shortestPath("manadium", "eva",null);
        graph.shortestPath("manadium", "curius",null);
        graph.shortestPath("manadium", "ash",null);
        graph.shortestPath("manadium", "axel",null);
        graph.removeTeleport();
    }
}
```

5.3 ANALISIS DATA

5.3.1 Soal Medium Modul IV Pohon

```
public class Vertex {
    int vertex;
    int weight;
    Vertex next;

    public Vertex(int vertex, int weight) {
        this.vertex = vertex;
        this.weight = weight;
        this.next = null;
    }
}
```

Script di atas merupakan kelas "`Vertex`" yang digunakan untuk menyimpan nilai dari vertex beserta bobot dari jalur yang telah ditempuh.

```
class Edge {
    Vertex head;
    public void add(int vertex, int weight) {
        Vertex newNode = new Vertex(vertex, weight);
        if (head == null) {
            head = newNode;
        } else {
            Vertex temp = head;
            while (temp.next != null) {
                temp = temp.next;
            }
            temp.next = newNode;
        }
    }
}
```

Script di atas merupakan kelas "`Edge`" yang digunakan untuk menambahan bobot kepada vertex yang telah melalui edge tersebut.

```
public class Graph {...}
```

Script di atas merupakan kelas "`Graph`" yang digunakan untuk menyimpan semua method yang digunakan dalam program.

```
public Graph(int vertices) {
        this.vertices = vertices;
        adjMatrix = new Edge[vertices];
        vertexNames = new String[vertices];
        teleportation = new int[vertices];
        for (int i = 0; i < vertices; i++) {
            adjMatrix[i] = new Edge();
            teleportation[i] = -1;
        }
    }
```

Script di atas merupakan sebuah method "`Graph`" merupakan sebuah konstruktor yang digunakan untuk mendeklarasikan nama, edge, dan perangkat teleportasi pada suatu vertex kemudian membuat adjesensi matrix yang digunakan untuk menentukan jalur terpendek.

```
    public void setVertexName(int index, String name) {
        vertexNames[index] = name;
    }

    public int getVertexIndex(String name) {
        for (int i = 0; i < vertices; i++) {
            if (vertexNames[i].equals(name)) {
                return i;
            }
        }
        return -1;
    }
```

Script di atas merupakan sebuah method "`setVertexName`" yang digunakan untuk memberikan nama pada sebuah vertex.

```
    public void shortestPath(String startName, String endName, String
ignorePlanet) {
        int start = getVertexIndex(startName);
        int end = getVertexIndex(endName);
        int ignore = ignorePlanet != null ? getVertexIndex(ignorePlanet)
: -1;

        if (start == -1 || end == -1) {
            System.out.println("Invalid vertex name.");
            return;
        }

        int[] distance = new int[vertices];
        boolean[] visited = new boolean[vertices];
        int[] previous = new int[vertices];

        for (int i = 0; i < vertices; i++) {
            distance[i] = Integer.MAX_VALUE;
            visited[i] = false;
            previous[i] = -1;
        }

        distance[start] = 0;

        for (int count = 0; count < vertices - 1; count++) {
            int u = selectMinVertex(distance, visited);
            if (u == ignore) continue;
            visited[u] = true;

            if (teleportation[u] == end) {
                distance[end] = distance[u];
                previous[end] = u;
                break;
            }

            if (teleportation[u] != -1 && !visited[teleportation[u]]) {
                int teleportVertex = teleportation[u];
                if (distance[u] < distance[teleportVertex]) {
                    distance[teleportVertex] = distance[u];
                    previous[teleportVertex] = u;
                }
            }

            Vertex temp = adjMatrix[u].head;
```

```
            while (temp != null) {
                if (!visited[temp.vertex] && temp.vertex != ignore &&
distance[u] != Integer.MAX_VALUE &&
                        distance[u] + temp.weight <
distance[temp.vertex]) {
                    distance[temp.vertex] = distance[u] + temp.weight;
                    previous[temp.vertex] = u;
                }
                temp = temp.next;
            }
        }

        if (distance[end] == Integer.MAX_VALUE) {
            System.out.println("No path found.");
        } else {
            printPath(previous, end);
            System.out.println();
        }
    }
```

Script di atas merupakan sebuah method "`shortestPath`" yang digunakan untuk menentukan jalur terpendek menggunakan algoritma Dijkstra. Algoritma ini akan menentukan jalur terpendek menggunakan sebuah adjesensi matrix.

```
    public void setTeleportation(String from, String to) {
        int fromIndex = getVertexIndex(from);
        int toIndex = getVertexIndex(to);
        if (fromIndex == -1 || toIndex == -1) {
            System.out.println("Invalid vertex name for
teleportation.");
            return;
        }teleportation[fromIndex] = toIndex;
        addEdge(from, to, 0);
    }
```

Script di atas merupakan sebuah method "`setTeleportation`" yang digunakan untuk menambahkan sebuah perangkat teleportasi pada planet yang diinginkan. Perangkat ini akan membantu memperpendek traversal data antar vertex.

```
    private void printPath(int[] previous, int vertex) {
        if (previous[vertex] == -1) {
            System.out.print(vertexNames[vertex] + " ");
            return;
        }
        printPath(previous, previous[vertex]);
        System.out.print("=> " + vertexNames[vertex] + " ");
    }
```

Script di atas merupakan sebuah method "`printPath`" yang digunakan untuk menampilkan planet mana saja yang dilalui oleh program pada proses traversal data.

```
    public void removeTeleport() {
        for (int i = 0; i < vertices; i++) {
            teleportation[i] = -1;
        }
        for (int i = 0; i < vertices; i++) {
            Vertex current = adjMatrix[i].head;
            Vertex prev = null;
            while (current != null) {
```

```
                if (current.weight == 0) {
                    if (prev == null) {
                        adjMatrix[i].head = current.next;
                    } else {
                        prev.next = current.next;
                    }
                } else {
                    prev = current;
                }
                current = current.next;
            }
        }
    }
```

Script di atas merupakan sebuah method "`removeTeleport`" yang digunakan untuk menghapus perangkat teleportasi dari planet ketika sudah tidak digunakan untuk mencegah penempatan perangkat teleportasi duplikat.

```
public class Main{

    public static void main(String[] args) {
        Graph graph = new Graph(17);
        String[] names = {"wh", "zen", "manadium", "tau","emp",
"primeval", "psy",
        "eldwurm", "redox", "rhongo", "curius", "ash", "eva", "unicorn",
"Fenrir", "axel", "sol"};
for (int i = 0; i < names.length; i++) {
    graph.setVertexName(i, names[i]);
}
        graph.addEdge("wh", "zen", 2);
        graph.addEdge("zen", "manadium", 2);
        graph.addEdge("tau", "sol", 2);
        graph.addEdge("tau", "emp", 4);
        graph.addEdge("emp", "primeval", 5);
        graph.addEdge("psy", "wh", 5);
        graph.addEdge("zen", "eldwurm", 6);
        graph.addEdge("wh", "manadium", 7);
        graph.addEdge("wh", "tau", 7);
        graph.addEdge("psy", "redox", 8);
        graph.addEdge("rhongo", "curius", 8);
        graph.addEdge("wh", "primeval", 8);
        graph.addEdge("ash", "eva", 9);
        graph.addEdge("psy", "curius", 9);
        graph.addEdge("psy", "ash", 9);
        graph.addEdge("axel", "eldwurm", 9);
        graph.addEdge("wh", "sol", 9);
        graph.addEdge("curius", "ash", 10);
        graph.addEdge("unicorn", "primeval", 10);
        graph.addEdge("wh", "redox", 11);
        graph.addEdge("unicorn", "emp", 11);
        graph.addEdge("rhongo", "eva", 11);
        graph.addEdge("Fenrir", "rhongo", 11);
        graph.addEdge("manadium", "eldwurm", 11);
        graph.addEdge("Fenrir", "unicorn", 30);
        graph.addEdge("rhongo", "unicorn", 30);
        graph.addEdge("eva", "emp", 30);
        graph.addEdge("eva", "wh", 30);
        graph.addEdge("ash", "wh", 30);
        graph.addEdge("redox", "zen", 30);
        graph.addEdge("redox", "axel", 30);
```

```
        graph.addEdge("sol", "manadium", 30);
        graph.addEdge("wh", "tau", 30);
        graph.addEdge("wh", "axel", 30);
        graph.addEdge("ash", "zen", 30);

        System.out.println("eva");
        graph.shortestPath("eva", "ash",null);
        graph.shortestPath("eva", "rhongo",null);
        graph.shortestPath("eva", "psy",null);
        graph.shortestPath("eva", "curius",null);
        graph.shortestPath("eva", "Fenrir",null);
        graph.shortestPath("eva", "wh",null);
        graph.shortestPath("eva", "zen",null);
        graph.shortestPath("eva", "redox",null);
        graph.shortestPath("eva", "manadium",null);
        graph.shortestPath("eva", "tau",null);
        graph.shortestPath("eva", "emp",null);
        graph.shortestPath("eva", "primeval",null);
        graph.shortestPath("eva", "sol",null);
        graph.shortestPath("eva", "eldwurm",null);
        graph.shortestPath("eva", "unicorn",null);
        graph.shortestPath("eva", "axel",null);
        System.out.println();
        System.out.println("\nmanadium");
        graph.shortestPath("manadium", "zen",null);
        graph.shortestPath("manadium", "wh",null);
        graph.shortestPath("manadium", "psy",null);
        graph.shortestPath("manadium", "eldwurm",null);
        graph.shortestPath("manadium", "tau",null);
        graph.shortestPath("manadium", "primeval",null);
        graph.shortestPath("manadium", "sol",null);
        graph.shortestPath("manadium", "redox",null);
        graph.shortestPath("manadium", "emp",null);
        graph.shortestPath("manadium", "axel",null);
        graph.shortestPath("manadium", "ash",null);
        graph.shortestPath("manadium", "curius",null);
        graph.shortestPath("manadium", "unicorn",null);
        graph.shortestPath("manadium", "rhongo",null);
        graph.shortestPath("manadium", "eva",null);
        graph.shortestPath("manadium", "Fenrir",null);

        graph.setTeleportation("Fenrir", "unicorn");
        graph.setTeleportation("Fenrir", "primeval");
        graph.setTeleportation("Fenrir", "emp");
        graph.setTeleportation("Fenrir", "wh");
        graph.setTeleportation("Fenrir", "tau");
        graph.setTeleportation("Fenrir", "sol");
        graph.setTeleportation("Fenrir", "manadium");
        graph.setTeleportation("Fenrir", "eldwurm");
        graph.setTeleportation("Fenrir", "zen");
        graph.setTeleportation("Fenrir", "redox");
        graph.setTeleportation("Fenrir", "axel");

        System.out.println("\nSetelah   kendaraan   teleportasi   ajaib
ditaruh di Fenrir\neva");
        graph.shortestPath("eva", "ash",null);
        graph.shortestPath("eva", "rhongo",null);
        graph.shortestPath("eva", "psy",null);
        graph.shortestPath("eva", "curius",null);
        graph.shortestPath("eva", "unicorn",null);
        graph.shortestPath("eva", "primeval",null);
```

```
        graph.shortestPath("eva", "emp",null);
        graph.shortestPath("eva", "wh",null);
        graph.shortestPath("eva", "tau",null);
        graph.shortestPath("eva", "sol",null);
        graph.shortestPath("eva", "manadium",null);
        graph.shortestPath("eva", "eldwurm",null);
        graph.shortestPath("eva", "zen",null);
        graph.shortestPath("eva", "redox",null);
        graph.shortestPath("eva", "axel",null);
        graph.removeTeleport();
        System.out.println("\nmanadium\nkendaraan           teleportasi
dipindahkan ke emp");
        graph.shortestPath("manadium", "zen",null);
        graph.shortestPath("manadium", "wh",null);
        graph.shortestPath("manadium", "psy",null);
        graph.shortestPath("manadium", "eldwurm",null);
        graph.shortestPath("manadium", "tau",null);
        graph.shortestPath("manadium", "primeval",null);
        graph.shortestPath("manadium", "sol",null);
        graph.shortestPath("manadium", "redox",null);
        graph.shortestPath("manadium", "emp",null);
        graph.shortestPath("manadium", "Fenrir",null);
        graph.setTeleportation("emp", "Fenrir");
        graph.setTeleportation("emp", "unicorn");
        graph.setTeleportation("emp", "rhongo");
        graph.setTeleportation("emp", "eva");
        graph.setTeleportation("emp", "curius");
        graph.setTeleportation("emp", "ash");
        graph.setTeleportation("emp", "axel");
        graph.shortestPath("manadium", "Fenrir",null);
        graph.shortestPath("manadium", "unicorn",null);
        graph.shortestPath("manadium", "rhongo",null);
        graph.shortestPath("manadium", "eva",null);
        graph.shortestPath("manadium", "curius",null);
        graph.shortestPath("manadium", "ash",null);
        graph.shortestPath("manadium", "axel",null);
        graph.removeTeleport();
```

Script di atas merupakan sebuah kelas "`Main`" yang menjadi kelas utama dalam program. Pada kelas ini terdapat perintah-perintah yang akan memanggil masing-masing method yang ada dalam program untuk menjalankan program sesuai dengan alur permasalahan.