ArrayList<>

- vertices: ArrayList<VertexL<V, E>

- adjacencyList: ArrayList<VertexL<V, E>>

- edges: ArrayList<EdgeL<V, E>>

1

1

1

<<Interface>>
IGraph<V,E>

+insertVertex(v:V):
+insertEdge(position1:int , position2:int, conection:E):
+deleteVertex(positionVertex:int):
+deleteAllEdge(position1:int , position2:int):
+deleteEdge(position1:int , position2:int , conection:E):
+getVertex():Arraylist<Vertex<V>>
+DFS(startPosition:int): Arraylist<Integer>
+BFS(startPosition:int): Arraylist<Integer>
+DFS(): ArrayList<Arraylist<Integer>>
+Prim(int: startPosition):ArrayList<EdgeL<V, E>
+Kruskal():ArrayList<Edge<V, E>>
+ Dijkstra(int: startPosition): Object[]
+ Floyd_Warshall(): double[][]

AdjacencyListGraph<V, E>

- directed: boolean

- weighted: boolean

- time: int

- nVertices: int

- nEdges: int

+ AdjacencyListGraph(boolean, boolean)

+ AdjacencyListGraph(boolean, boolean, Vertex<V, E>)

+ insertVertex(V): void

+ insertEdge(int, int, E): void

- changeOrderOfTheEdge(EdgeL<V, E>): void

- deleteFromAdjacencyList(VertexL<V, E>): void

- deleteFromEdgesList(ArrayList<EdgeL<V, E>>): void

+ deleteVertex(int): void

+ getEdges(int, int): ArrayList<EdgeL<V, E>>

+ getEdge(int, int, E): int

+ deleteEdge(int, int, E): void

+ isThereAnEdges(VertexL<V, E>, VertexL<V, E>): boolean

+ deleteAllEdge(int, int): void

+ getVerticesL(): ArrayList<VertexL<V, E>>

+ getVerticesM(): ArrayList<VertexM<V, E>>

+ getEdges(): ArrayList<EdgeL<V, E>>

+ BFS(int): ArrayList<Integer>

- BFS_algorithm(int): ArrayList<Integer>

+ BFS(): ArrayList<ArrayList<Integer>>

+ DFS(int): ArrayList<Integer>

+ DFS(): ArrayList<ArrayList<Integer>>

- DFS_restart(): void

- DFS_Visit(ArrayList<Integer>, int): void

+ Prim(int): ArrayList<Integer>

+ Kruskal(): ArrayList<EdgeL<V, E>>

+ Dijkstra(int): Object[]

- getMinimumVertex(double[], boolean[]): int

+ Floyd_Warshall(): double[][]

+ isDirected(): boolean

+ isWeighted(): boolean

+ getNumberOfVertices(): int

+ getNumberOfEdges(): int

VertexL<V, E>

- value: V

- key: E

- degree: int

- distance: double

- color: byte

- predecessor: int

- initialTime: int

- finalTime: int

+ Vertex(V)

+ setValue(V): void

+ getValue(): V

+ getKey(): E

+ setKey(E): void

+ getDegree(): int

+ setDistance(double): void

+ getDistance(): double

+ setColor(byte): void

+ getColor(): byte

+ setPredecessor(int): void

+ getPredecessor(): int

+ getInitialTime(): int

+ setInitialTime(int): void

+ getFinalTime(): int

+ setFinalTime(int): void

+ addAdjacentVertex(int): VertexL<V, E>

+ getAdjacencyList(): ArrayList<VertexL<V, E>>

+ addAdjacentVertex(VertexL<V, E>>): void

+ addAdjacencyList(ArrayList<VertexL<V, E>>): void

+ deleteFromAdjacencyList(int): void

+ deleteAdjacencyList(): void

+ compareTo(VertexL<V, E>): int

EdgeL<V, E>

- weight: E

+ Edge(Vertex<V>, int)

+ getOriginVertex(): VertexL<V>

+ setOriginVertex(VertexL<V>): void

+ getDestinationVertex(): VertexL<V>

+ setDestinationVertex(VertexL<V>): void

+ getWeight(): E

+ setWeight(E): void

+ compareTo(EdgeL<V, E>): int

- u: VertexL<V, E

1

- v: VertexL<V, E:

1