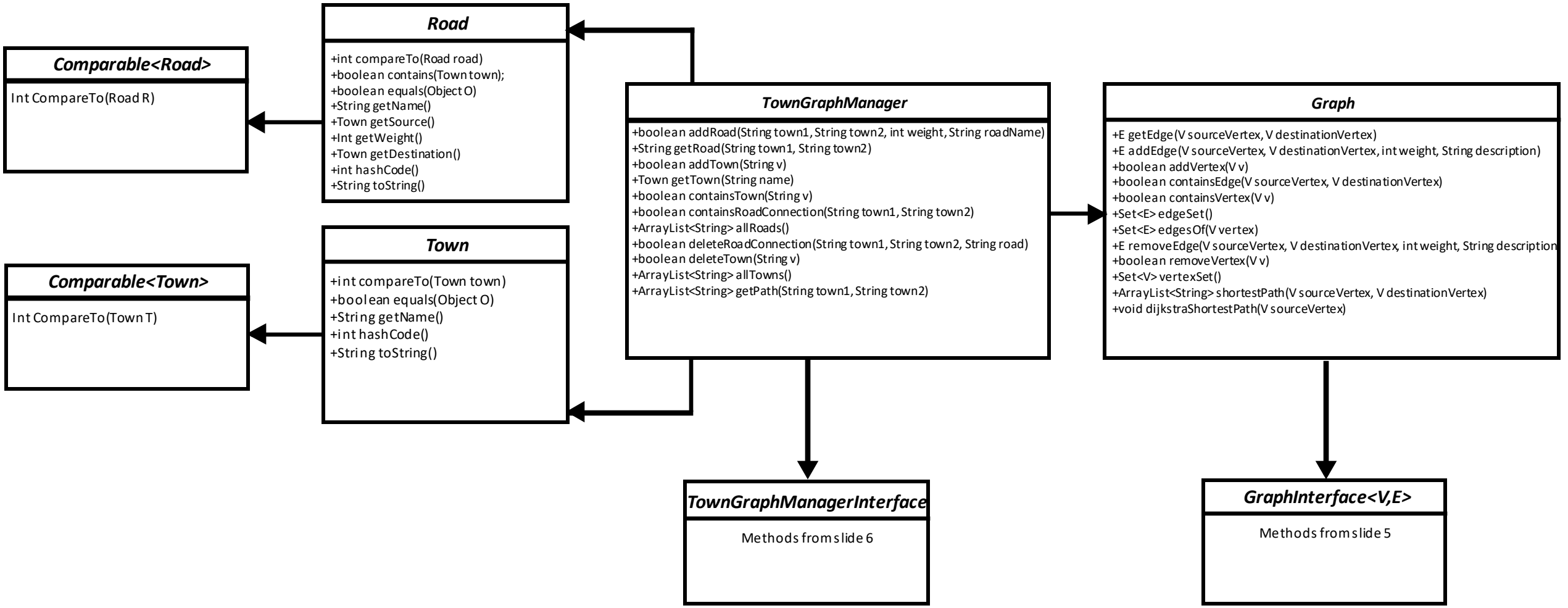
Assignment 6

Shortest path graph algorithm

UML


Comparable<Road>
Int CompareTo(Road R)
Road
+int compareTo(Road road)
+boolean contains(Town town);
+boolean equals(Object O)
+String getName()
+Town getSource()
+Int getWeight()
+Town getDestination()
+int hashCode()
+String toString()
TownGraphManager
+boolean addRoad(String town1, String town2, int weight, String roadName)
+String getRoad(String town1, String town2)
+boolean addTown(String v)
+Town getTown(String name)
+boolean containsTown(String v)
+boolean containsRoadConnection(String town1, String town2)
+ArrayList<String> allRoads()
+boolean deleteRoadConnection(String town1, String town2, String road)
+boolean deleteTown(String v)
+ArrayList<String> allTowns()
+ArrayList<String> getPath(String town1, String town2)
Graph
+E getEdge(V sourceVertex, V destinationVertex)
+E addEdge(V sourceVertex, V destinationVertex, int weight, String description)
+boolean addVertex(V v)
+boolean containsEdge(V sourceVertex, V destinationVertex)
+boolean containsVertex(V v)
+Set<E> edgeSet()
+Set<E> edgesOf(V vertex)
+E removeEdge(V sourceVertex, V destinationVertex, int weight, String description
+boolean removeVertex(V v)
+Set<V> vertexSet()
+ArrayList<String> shortestPath(V sourceVertex, V destinationVertex)
+void dijkstraShortestPath(V sourceVertex)
Comparable<Town>
Int CompareTo(Town T)
Town
+int compareTo(Town town)
+boolean equals(Object O)
+String getName()
+int hashCode()
+String toString()
TownGraphManagerInterface
Methods from slide 6
GraphInterface<V,E>
Methods from slide 5

Class: Town

- Towns are represented by the nodes/vertices of a graph which is defined by either a set of *V* of vertices or as the x/y of an adjacency matrix.
- Implements Comparable<Town>
- Methods
  - int compareTo(Town town)
  - boolean equals(Object O)
  - String getName()
  - int hashCode()
  - String toString()

Class: Road

- Roads are represented by edges on a graph, these can be implemented as either a set of edges or as the values within the adjacency matrix of the graph.
- Implements Comparable<Road>
- Methods
  - int compareTo(Road road)
  - boolean contains(Town town);
  - boolean equals(Object O)
  - String getName()
  - Town getSource()
  - int getWeight()
  - Town getDestination()
  - int hashCode()
  - String toString()

Class: Graph

- The graph is the main data structure of this assignment. A graph is defined as a double, a set of vertices, and a set of edges. Implementations of this can be done traditionally using a double type parameter generic or as an adjacency matrix.
- Implements GraphInterface<V,E>
- Methods
  - E getEdge(V sourceVertex, V destinationVertex)
  - E addEdge(V sourceVertex, V destinationVertex, int weight, String description)
  - boolean addVertex(V v)
  - boolean containsEdge(V sourceVertex, V destinationVertex)
  - boolean containsVertex(V v)
  - Set<E> edgeSet()
  - Set<E> edgesOf(V vertex)
  - E removeEdge(V sourceVertex, V destinationVertex, int weight, String description)
  - boolean removeVertex(V v)
  - Set<V> vertexSet()
  - ArrayList<String> shortestPath(V sourceVertex, V destinationVertex)
  - void dijkstraShortestPath(V sourceVertex)

Class: TownGraphManager

- Puts together the whole town graph. Has methods to populate the town and find the shortest path between two towns.
- Implements TownGraphManagerInterface
- Methods
  - boolean addRoad(String town1, String town2, int weight, String roadName)
  - String getRoad(String town1, String town2)
  - boolean addTown(String v)
  - Town getTown(String name)
  - boolean containsTown(String v)
  - boolean containsRoadConnection(String town1, String town2)
  - ArrayList<String> allRoads()
  - boolean deleteRoadConnection(String town1, String town2, String road)
  - boolean deleteTown(String v)
  - ArrayList<String> allTowns()
  - ArrayList<String> getPath(String town1, String town2)