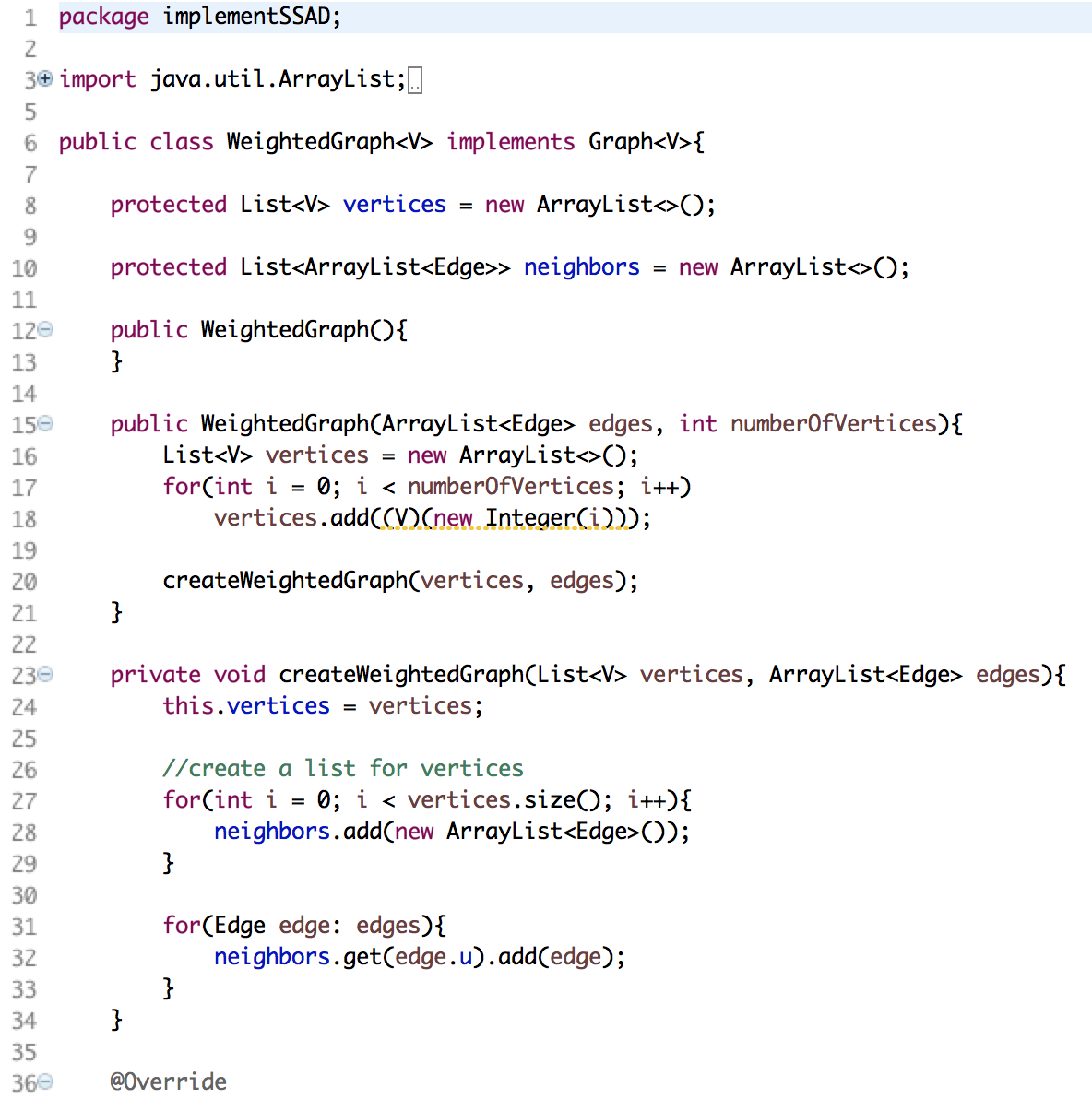
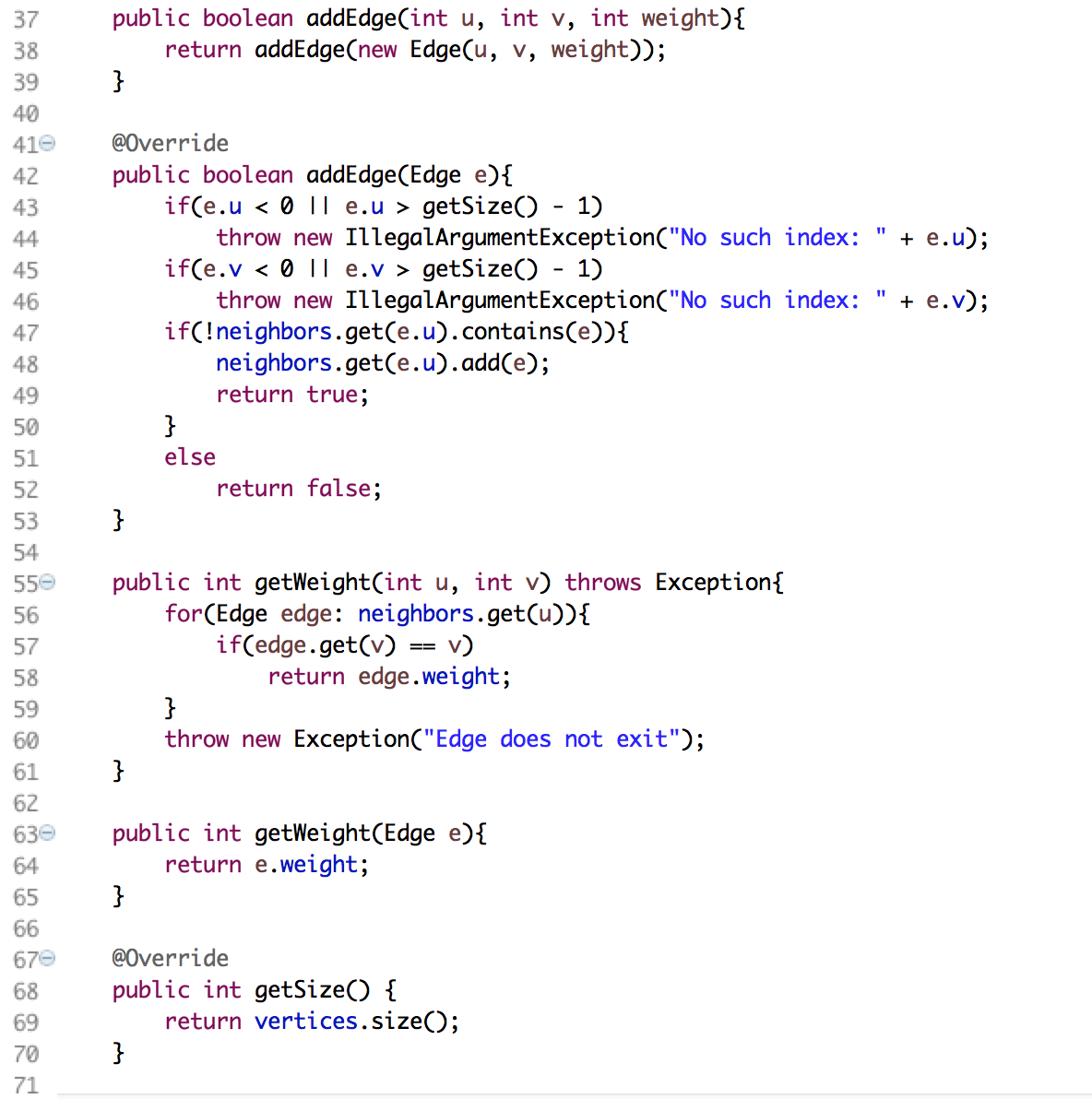


Build an interface of Graph, V is the generic type of graph



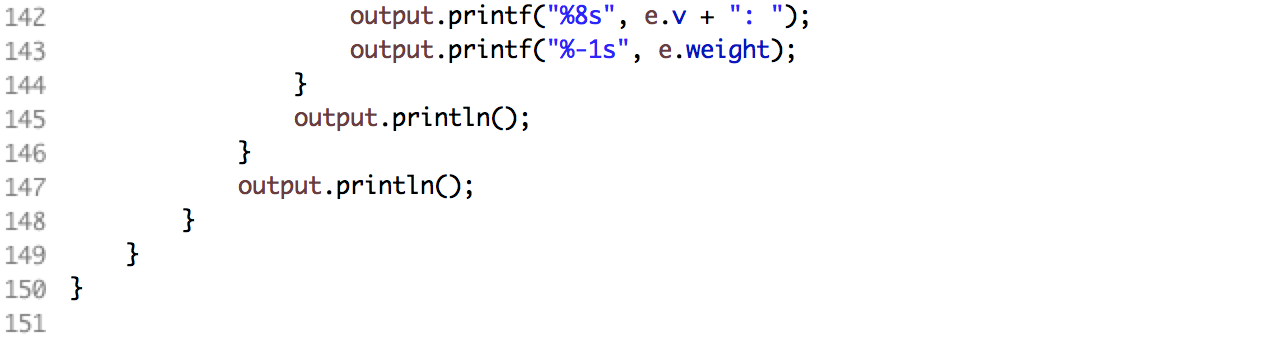
Concrete class WeightedGraph<V> implements all the methods defined in the Graph interface. Two protected data fields were defined: vertices and neighbors, neighbors is a list of type ArrayList<Edge>. Two constructors were created, one is empty, the other has two arguments: edges: ArrayList<Edge> and nuberOfVertices: int. A private method createWeightedGraph was used to created a graph with edges represented as adjacency list



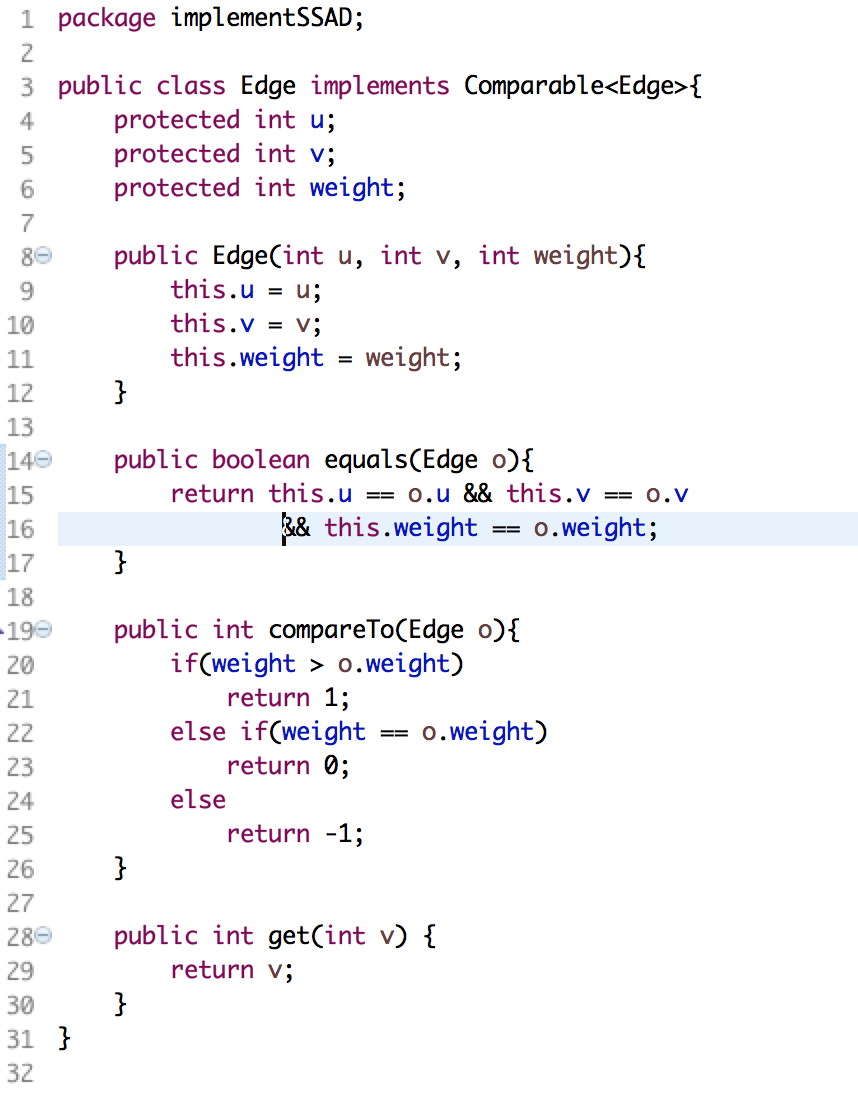
In consider of the unweightedgraph, the getWeight method was not defined in the graph interface



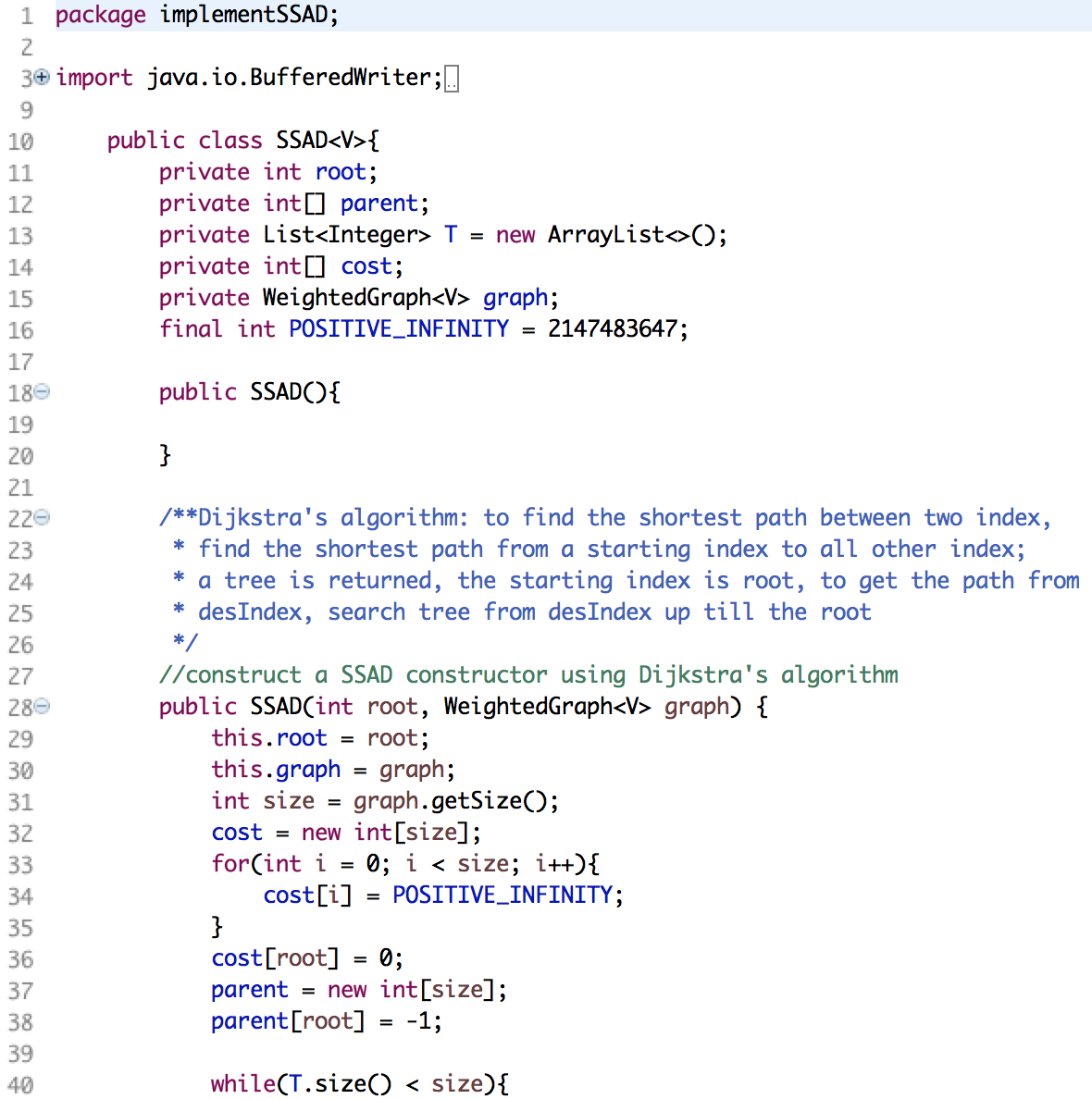




writeGraph() method takes graph: WeightedGraph<V> as parameter, the method output the header as well as the adjacency list of edges in the order of vertices. The method uses iterator to iterate all the elements in the neighbors, for each element neighbor, if its not empty, the method output the vertex index: neighbor.get(0).u, if it is empty, a helper variable i was written into txt as vertex index. For each edge in one neighbor, its second parameter e.v and third parameter e.weight was output following the corresponding vertex index.

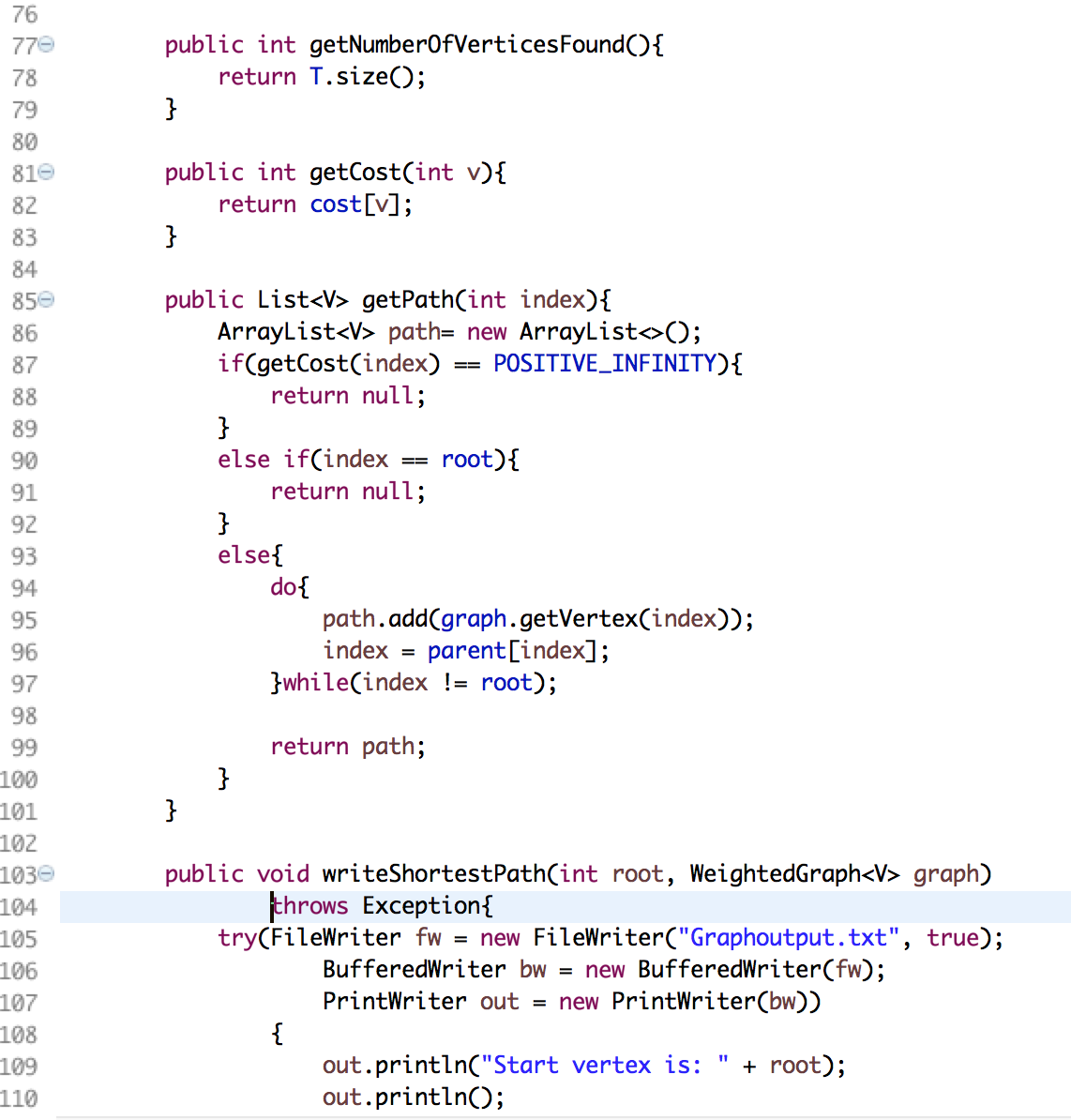


An Edge class was created, it has three datafields: two vertices u, v and one weight weight. Since we need to compare weights between different edges, the class implements Comparable<Edge> and has a method comparTo()



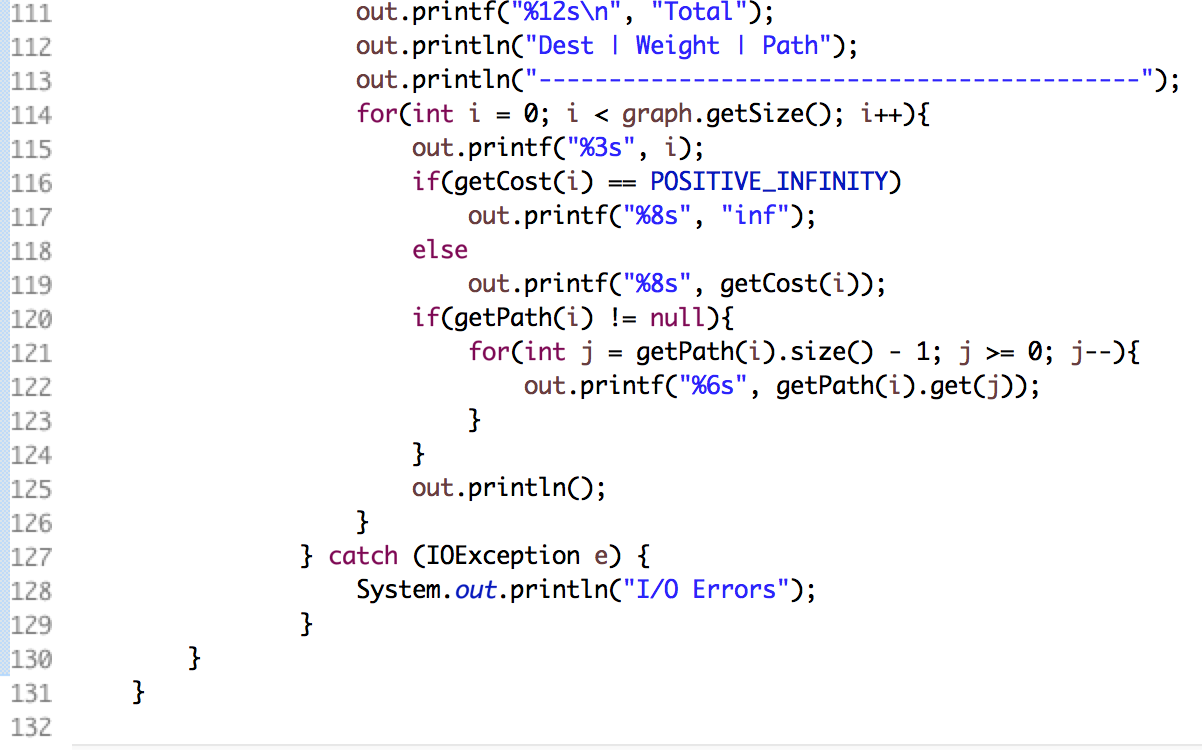
Class SSAD was used to get the info of shortest path Tree. Five data fields: root, parent, T, cost and graph was created. Root is the starting vertex, T is a List of type Integer to store the visited vertex, parent is an integer array define the parent-child relationship between two vertex, the child index is i and parent index is parent[i]. Cost is an integer array to store the smallest cost from the root the vertex i. Two constructors were created, one is empty and the other takes root and graph as parameters to update the datafields using Dijkstra’s algorithm.





A method getPath was created, it takes certain vertex as parameter and return a list of vertices it visited till the start vertex. The method uses parent[] to trace parent-child relation.

writeShortestPath() was created, it takes root and graph as arguments and output the cost and shortest path into the txt. Since the text file was written by the method in weightedGraph first, the FileWriter takes the filename and “true” to append contents rather than rewrite in the file. The method uses a for loop to iterate all the vertices in the graph to add destination vertex and cost(weight), for writing the path, it then invokes the getPath() method, uses another for loop to iterate from the last element in the path to the first one.

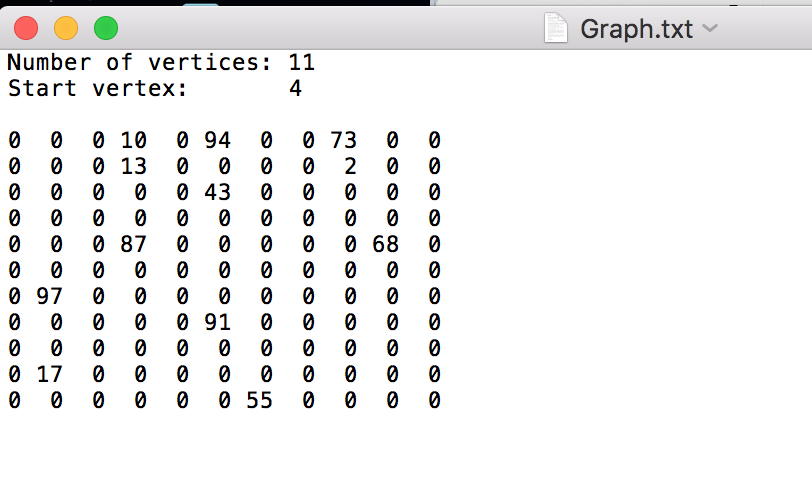






In the main method, the values of numberOfVertices and start vertex were obtained by following: replaceAll(“\\s+”) was used to strip all the white spaces in one line, then “:” was used as a delimiter to split the line into String[], and number was extracted. For reading the matrice in the text, if the number e != 0, which means there is an edge between vertices i and j, corresponding i, j and e was added to Edge(u, v, weight) object. A WeightedGraph<Integer> was constructed and graph info from the text was written into the file. A SSAD was constructed and the tree info was appended to the file.

input



output

