**public class NodeADT**

NodeADT(*int* cityIndex, *String* city,

*String* color, *double* index,

*double[]* location) Construct a node

*int* getCityIndex() Get city index

*String* getCity() Get city name

*String* getColor() Get color representation of the city

*double* getIndex() Get average air quality index of the city

*double[]* getLocation() Get latitude and longitude of the city

*int* compareTo() Compare the city name to other city

*String* toString() String representation of the NodeADT

**public class Load**

**private:**

*ArrayList<String[]>* loadCGNDB() Load the CGNDB list file

*String* color(*double* index) Color representation of the index

*double* averageCalculator(

*ArrayList<Double>* index) Calculate the average index

*ArrayList<Double>* indexArray(*ArrayList<String>* s) Convert a list of string values to a list

of double values

**public:**

*NodeADT[]* toNodeADT() Store information to an array of

NodeADTs

**public class BinarySearch**

**public:**

*int* searchCity(*NodeADT[]* x, *String* key) Binary search on city name in

an array of NodeADTs

*int* searchCityIndex(*NodeADT[]* x, *int* key) Binary search on city index in

an array of NodeADTs

**public class Heapsort**

**public:**

*void* sortCity(*Comparable[]* x, *int* n) Heap sort an array of compare x

*void* sortCityIndex(*NodeADT[]* x, *int* n) Heap sort an array of NodeADTs by

cities index

*void* sortIndex(*NodeADT[]* x, *int* n) Heap sort an array of NodeADTs by

air qualities index

**private:**

*void* sinkCity(*Comparable[]* x, *int* k, *int* n) Method “sink down” for sortCity

*void* sinkCityIndex(*NodeADT[]* x, *int* k, *int* n) Method “sink down” for

sortCityIndex

*void* sinkIndex(*NodeADT[]* x, *int* k, *int* n) Method “sink down” for sortIndex

**public class Graph**

Graph(*int* V) Construct a graph with V vertices

*void* addEdge(*int* v, *int* w) Add an edge between two vertices

*Iterable<Integer>* adj(*int* v) Get the connected vertices from v

*Int*  degree(*Graph* G, *int* v) Get the number of degree of v from G

*Int*  V() Get the number of vertices

*int* E() Get the number of edges

*String* toString() String representation of the Graph

**public class BFS**

BFS(*Graph* G, *int* s) Constructor of BFS algorithm

*private void* bfs(*Graph* G, *int* s) BFS from single source s

*boolean* hasPathTo(*int* v) Check if it is a path to vertices v

*Int*  distTo(*int* v) The number of edges from s to v

*Stack<Integer>*  pathTo(*int* v) The path from s to v

**public class FindPath**

**public:**

*int* cityToCityIndex(*NodeADT[]* nodes,

*String* city) Convert city name to its index

*String* cityIndexToCity(*NodeADT[]* nodes,

*int* key) Convert city index to city name

*Graph* toGraph(*NodeADT[]* nodes) Graph representation of the NodeADTs

**Other classes:**

**References:**

Bag.java (authors: Robert Sedgewick and Kevin Wayne)

Queue.java (authors: Robert Sedgewick and Kevin Wayne)

Stack.java (authors: Robert Sedgewick and Kevin Wayne)

BFS

BinarySearch

Graph

Heapsort

Stack

NodeADT

Load

FindPath

Queue

Bag