**Program Architecture**

The initial program reads a text file and interprets it as a graph.

**Input File**

All the input files in this course will look something like this. Note that all columns of data are separated by 1 or more spaces (not tabs):

~ val A B C D E

Alpha ~ ~ 3 2 ~ 5

Bravo S 4 ~ 5 3 ~

Charlie G 9 ~ ~ ~ ~

Delta ~ 6 ~ 5 ~ 1

Echo ~ ~ 6 4 2 ~

* The first row is a header row.
* The first “~” is merely a placeholder. All files will have this at the beginning of the header row.
* “val” is a column label for the column of node values.
* The rest of the strings are *mnemonics* for the nodes of the graph. They are alphanumeric and do not contain spaces.
* Each subsequent row describes one node and its outgoing edges.
* The first column contains the *name* of the node (e.g., “Alfa”)
* The string in the “val” column is the *value* of the node.
* For the rest of the columns, if there is no edge going from the node to another node, there is a “~“ in the appropriate column. Otherwise, there will be a label for the edge from the row node to the column node. This label will be an integer, often a positive integer denoting the distance from the row node to the column node.

**Program Classes and Methods**

There are 4 classes in the program given to you (in addition to a stub class for each deliverable, A-D):

*Prog340* is the main class, which handles the I/O.

*Graph* is a class whose objects represent graphs.

*Node* is a class whose objects represent nodes (a.k.a., vertices) in the graph.

*Edge* is a class whose objects represent edges (a.k.a., arcs) between nodes of the graph. Note that each edge has a direction. It goes from one specific node to another specific node.

Here is a basic UML class diagram for these four classes. All methods and attributes are listed explicitly rather than using composition connectors, to be easily understood by people with little UML background.

A screenshot of a computer program

Description automatically generated

