Programming Project

There will be one multi-deliverable Java programming project this semester, which will be divided into three separately-graded program deliverables (DelivA through DelivC). A specification for each deliverable will be uploaded to D2L. Each deliverable will build on past deliverables.

Start from a Working Program

Some Java programs work on student computers but not on University lab (Windows) computers or grading computer. So this semester, you are given an initial codebase, "Prog340", that seems to compile and run on all platforms.

- It opens a menu for you to select an action. It has a number of predefined menu items describing the various actions (which you will implement during the semester).
    - When you start the program the first action you will want to take will be to read a file. The input files for all deliverables have the same, specified format.
- It allows you to continuously select actions as long as you wish
    - Including read another file.
- It allows you to exit the program.

Each Deliverable will involve your modifying your existing program (DelivA involves modifying the initial program given to you). This minimizes any duplicate effort. But it means that if you screw up a deliverable, you may have screwed up another deliverable later in the course.

Grading

Deliverable A is worth 25 points. It is pretty simple if you understand OO programming in Java.

For each of Deliverables B (50 pts) and C (55 pts):

- Correct functioning of the program is worth 40 and points (DelivB and DelivC). You will receive a set of test files which will be used to evaluate the program. Some other test files may also be used that are not given to you (in real life, you design to the specification, you can't test everything). You will receive the desired output for at least one test file. Part of learning the material in the class is figuring out the correct answer to the test files yourself. The correctness points are divided among the tests. For instance, if there are 8 tests, each test is worth 10 points. Normally, a test is either all or nothing, 0 or max points, but you can get substantial partial credit for output whose content is correct, but whose formatting is off.
- For Deliverable C, there will be a report worth 5 points.

Program Architecture

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

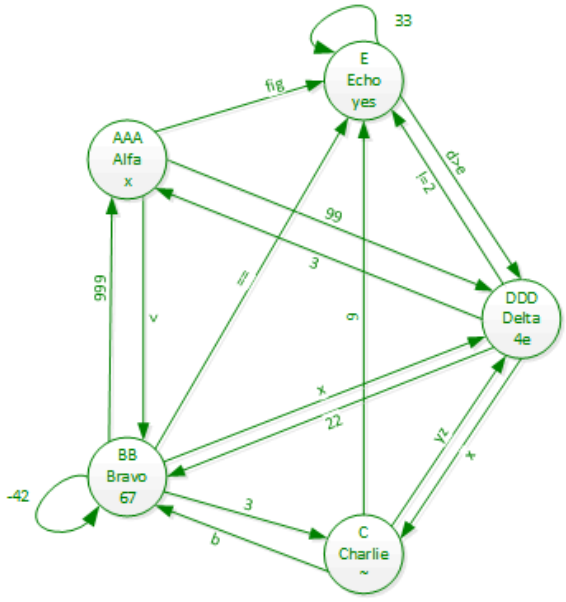
Input File

All the input files in this course will look something like this, file *AB0.txt*. Note that all columns of data are separated by 1 or more spaces (not tabs):

```
~          val   AAA    BB     C   DDD     E
Alfa         x     ~     >     ~    99   fig
Bravo       67   999   -42     3     x    ==
Charlie      ~     ~     4     ~    yz     9
Delta       4e     3    22     x     ~   !=2
Echo       yes     ~     ~     ~   d>e    33
```

- 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, which may be numeric, but which will certainly always consist of a string of printable characters. The character "~" by itself stands for "no value". (e.g., node "Charlie" has no value.)
  - 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. (e.g., there is no edge from node "Charlie" to node "Alfa".)
  - If there is anything else in that column, that is the *label* of the edge. For some deliverables, this will be numeric, for others it may not be. Note that self-edges are allowed.

If you like a picture of what this is, here's what the above table looks like in graphical form:



Program Classes and Methods

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

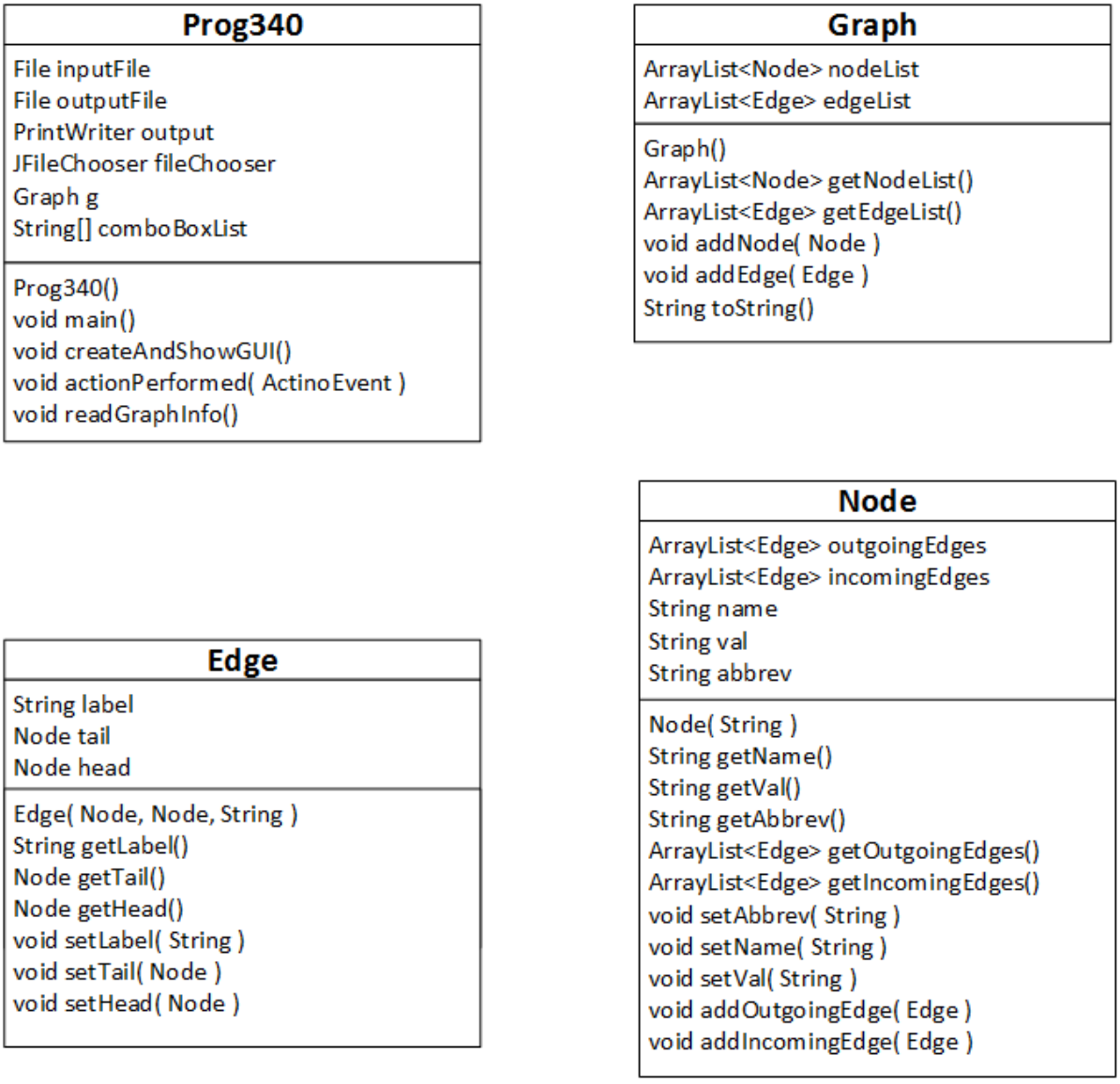
Prog340 is the main class, which handles the I/O.

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.



Rules

These rules will apply to all the program deliverables for this course:

1. You may use any code you can find on the publicly accessible internet (or any other published source) as part of your program. However, if you take code from the internet or published source, you must acknowledge your source. You may not use code from your classmates or other unpublished sources. Two programs or program fragments that are unduly similar and are not acknowledged as having come from a published source will be considered as being incidences of plagiarism, with a 0 for the assignment and a report sent to the Academic Integrity committee. A "published source" is *not* somebody else's program unless it's on a publicly accessible, free site (thus, for instance, stackoverflow.com is fine, but Chegg or CourseHero (which require signup) are not).
2. Do not publish any of your solutions to GitHub or any other public place before the end of the semester. If you do, and if someone uses your code, this will be considered academic misconduct and result in your getting a 0 for that deliverable.
3. Your program must run on the university's Windows computers "as is" – without having to import any packages that are not part of the Oracle Java SE installed on the university systems. If it does so, it should run on the instructors' system.

   Two common problems in the past have been:

   a) People who write their programs on Macs sometimes have trouble, especially with I/O. This should not be a problem because you are given working I/O code. If your Deliverable A runs properly on the grading computers, it's very likely all the other deliverables will, too, as long as you don't violate rule (b) below.

   b) People who import other packages, most commonly import from eclipse.org or netbeans.org. Please ***do not import anything outside of Java SE*** to do problems in this class, and if you do so, this may make your program not compile and/or run properly on the university system. This will severely impact your grade, since 80-90% of the program grade is based on test results from a running program. There is no reason for you to make the mistake of importing any package not in the Java SE. If you want to use code from the internet that uses such packages, you'll need to strip out those imports and rewrite the equivalent functionality yourself. (Remember to reference your source, please.)

Deliverable Submissions

Note that most deliverables are due at 2 pm on the due date on D2L.