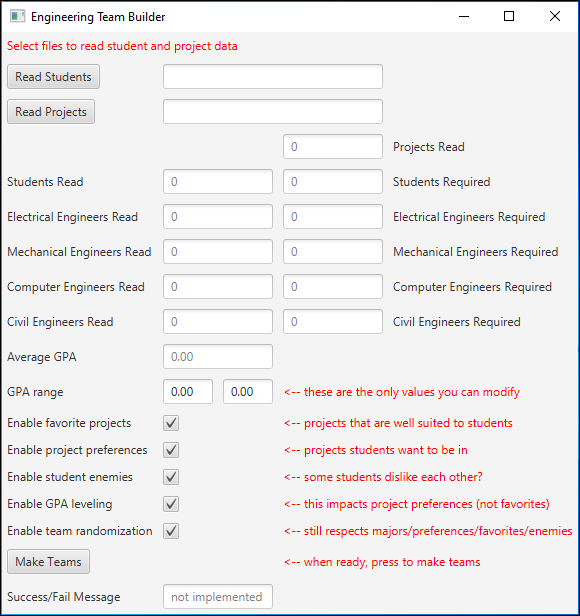
1. How to use the program!
   1. Execute code from Eclipse or JAR file (JAR file found in the “program” directory)

Hopefully you see the GUI (the red text is there to help you, but I will go into a little more detail here):



* 1. Press the “Read Students” button to read a VALID students file. The provided “Student Input File.xls” should work and is the expected format.
  2. Press the “Read Projects” button to read a VALID projects file. The provided “Project Input File.xls” should work and is the expected format.
  3. The next 7 rows of information **cannot** be directly modified by the GUI. These values are drawn from the input files you chose. Ideal scenario is that each number of “Read” is equal to it’s “Required” pair. Any differences **should not** prevent the teams from being built. But some students might be left out (if there are too many students) or some projects might not be completely filled (if there are too few students)
  4. GPA range can be adjusted within the GUI. Its values are defaulted to +/- 0.2 from the student body’s average GPA. There is no guarantee that teams will fall within this range. Please read code comments about the GPA leveling mechanic and its deficiencies.
  5. Checkboxes to enable/disable various preferences.
     1. Favorite projects: these are projects students are well suited towards, there is an associated weight in the students file. This has the highest priority in assigning students to projects and nothing can override it.
     2. Project preferences: when students list projects they want to do. This is the 2nd highest priority sorting decision.
     3. Student enemies: if a student has an enemy, or the student is the enemy of another student, those students will never be placed on the same team. This has the least testing of all sorting mechanisms (but has worked so far). It appears we only support 1 enemy per student (that is all that has been tested so far). ExcelIO.readNextRow() would need updating to support multiple enemies. The input file might also need some change. The rest of the code should already support multiple enemies per student.
     4. GPA leveling: this attempts to bring up the teams with low GPAs by trading their lowest GPA student with a high team’s highest GPA student. The algorithm is described in detail inside TeamBuilder.java (line 167ish). This will probably break some student project preferences (not major, enemies, or favorites) by trading a student off a project they wanted to be on. A fancier algorithm could certainly prevent that. This currently does not care about high GPA teams, it only tries to fix low GPA teams by stealing high GPA students from high GPA teams. The algorithm will give up eventually, so there is no guarantee that all teams will fall within the GPA window (its like 50% of the time right now, but very few teams squeak by with a low GPA…1 or 2 tops).
     5. Team randomization: it respects all other rules. It just randomizes the rest of the students. Turn this off, and the output will always be the same.
  6. Press “Make Teams” when you are ready to go. Save as a .xlsx please (.xls works, but you will be warned about the file type when you try to open it).
  7. The Success/Fail message was never implemented and will always show that
  8. There are many ways that this code can fail to produce good results. We have only written code and tested to ideal circumstances. The files we’ve provided should produce good output.

1. How to get this working in Eclipse
   1. Install JavaFX
      1. Help->Install New Software
      2. Add
         1. Name = Oxygen (or whatever you want to call it)
         2. Location = <http://download.eclipse.org/releases/oxygen>
      3. Select that repository from the “Work with” dropdown menu
      4. General Purpose Tools
      5. Select and install: e(fx)lipse
         1. I’m working from memory here, it might be called something else. But its name is pretty close to this
   2. FIX YOUR BUILD PATH
      1. The build path is currently set to my machine, and the paths are explicit. Unless you happen to have the same directory structure as me, you must change your build path.
      2. Right click on the project->Build Path->Configure Build Path
      3. delete everything with an explicit path (be sure not to remove JRE system library and Junit 5)
      4. Add External JARs
      5. Select everything in the “All needed libraries” directory
      6. Apply & close
   3. Run/debug main.java
   4. Run any test/\*Test.java you’d like (unfortunately, we could not get a test suite to work). Note there are no test files for Main.java, or the GUI MainWindow.java
   5. Let us know if there are problems, we will work with you to solve them
2. Excel File Instructions.

Input file instructions:

The program can interpret excel files with the .xls or .xlsx extensions.

The program currently requires that the input files be formatted as expected. For precise details, see the instructions below or the example input file included in the documentation.

Student Input File

Row 1 should contain the headers for each column. Rows 2 onward should each contain the data for an individual student.

Column A: This column must not be blank. Contains the name, first and last, of the student.

Column B: This column must not be blank. Contains the ID Number of the student.

Column C: This column must not be blank. Contains the Major of the student.

Column D: This column must not be blank. Contains the GPA of the student.

Column E: Contains the name, first and last, of the student’s incompatible partner. This column may be left blank if the student does not have an incompatible partner.

NOTE: The program can currently only handle one incompatible partner per student.

Column F: Contains the name of the student’s one favored project. This column may be left blank if the student does not have a favorite project.

NOTE: The name must exactly match the name of the project as written in the project input file.

Column G: Contains the weight value, on a scale of 1-5, of the weight assigned by the user to a student’s project preference, with higher numbers referring to a more valid reasoning for their preferred project. This column may be left blank if the student does not have a favorite project.

Column F onward: These columns contain the student’s project preferences as displayed in the file resulting from the qualtrics survey. These values can be directly copied from that file into the input file. This file can handle as many projects as are needed, and allows an individual student to have as few or as many preferred projects as they like.

NOTE: The name in each cell must exactly match the name of the project as written in the project input file.

Project Input File

Row 1 should contain the headers for each column. Rows 2 onwards should each contain the data for an individual project.

Column A: This column must not be blank. Contains the name of the project.

NOTE: This name must be matched at several places throughout the input files.

Column B onward: These columns contain the types of engineer required for the project. The program can handle as many or as few required students as the sponsor wants.

Output File:

The program will attempt to ouptut a modern Excel file, with the .xlsx extension. If the user specifically requests a .xls file, it will be readable, but Excel will throw an error when the results file is first opened.