

Requirements for Team Assignment 2

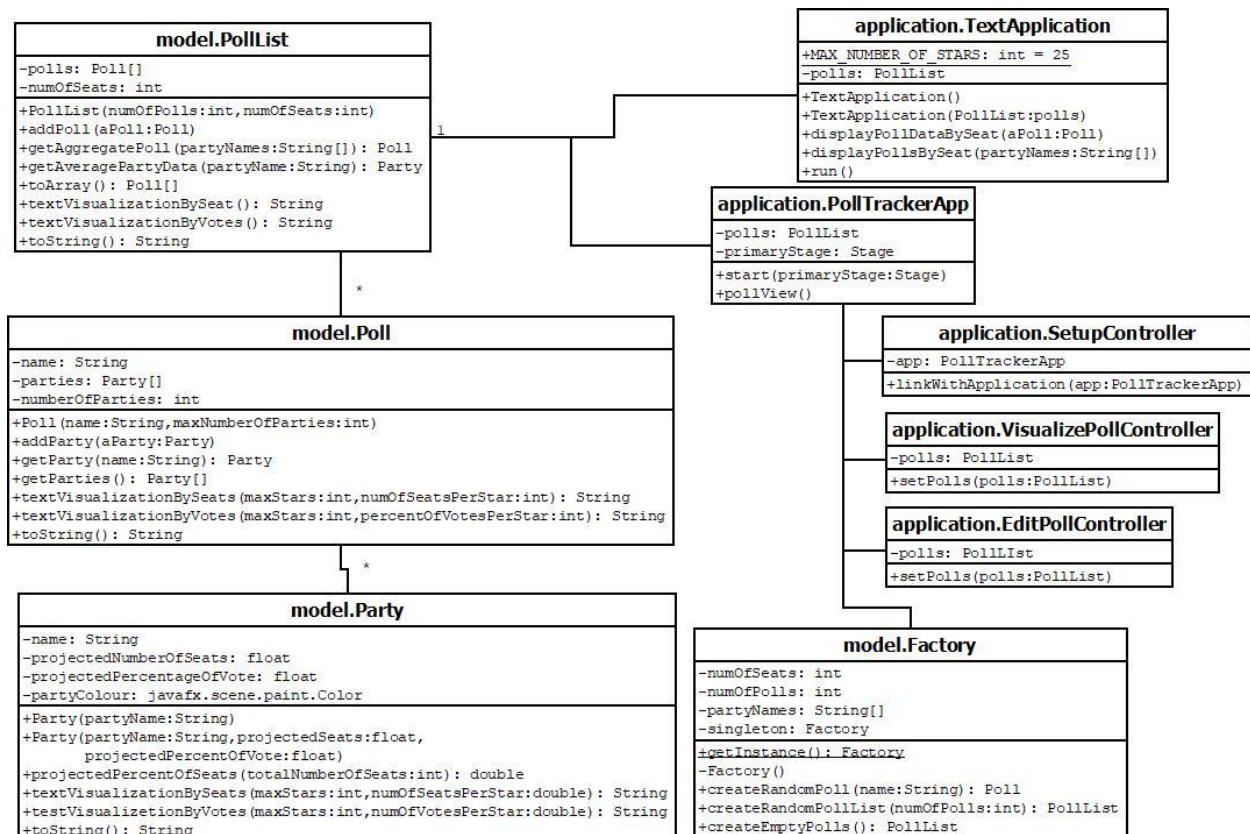
You'll now combine the work from individual and team assignment 1 and the work that each team member completed for individual assignment 2. This means you should have working versions of the following classes.

- In the model package: Party, Poll, PollList, Factory
- In the view folder (these can be fxml files or java files): SetupView, EditPollView, VisualizePollView (there may be additional files if some team members broke their view over multiple files).
- In the application package (you don't need to include the Tester classes: SetupController, EditPollController, VisualizePollController, TextApplication)

Add the provided PollTrackerApp class to the application package. This class uses the three views and the three controllers to pull them all in a single application that allows the user to setup a new poll tracker, update information in the polls that are being tracked, and visualize the data stored in the poll tracker. **This addition of the provided class will likely cause some compile/syntax errors.**

Overview of Requirements

By the end of this iteration, all the below classes should be created. They should all compile and you should be able to run both the TextApplication class and the PollTrackerApp class.



(Note that the view classes/files are not included in this diagram.)

Grading

Team grading breakdown is as follows:

- **Functionality:**
 - (1 point) PollTrackerApp class is included such that it loads all three views.
 - (1 point) PollTrackerApp starts with setup view.
 - (1 point) When setup is complete, view is changed to a tabbed pane that contains one tab to edit polls and one tab that visualizes data.
 - (1 point) All views contain up to date information: information is carried from one view to the next. (Through the Factory for the setup information and by using the shared PollList instance for the edit and visualize tabs.)
- **Code quality:**
 - (1 points) Throughout the application, variable names and IDs are self-documenting and code is easy to read and understand. Good use of white space and all methods fit on a single (small) screen (both height and width).
 - (1 point) Throughout the application, code is well organized and easy to maintain: nesting deeper than three levels is avoided; multiple return statements are avoided; break and continue are never used.
 - (1 point) All classes in the application use the same coding conventions and have similar naming conventions, spacing conventions, placement of braces, and documentation conventions
 - (1 point) In the application as a whole, code duplication is avoided and minimized; code is well organized over classes and methods; no unnecessary instance variables.
- **Documentation:**
 - (1 point) PollTrackerApp class and public methods are fully javadoc'd with appropriate tags and content.
 - (1 points) Good use of in-line documentation to explain blocks of code.

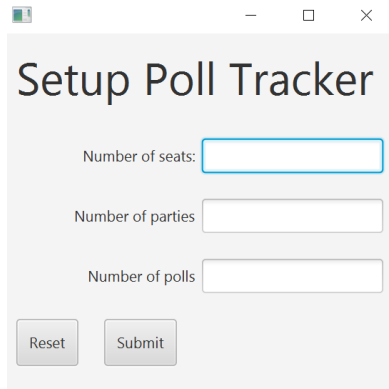
Detailed Requirements

As a team, work on the PollTrackerApp class, which combines the Setup, PollEdit, and VisualizePoll views. An implementation is provided but the following edits are likely needed. (You may code PollTrackerApp from scratch or edit it in some other way if you prefer.):

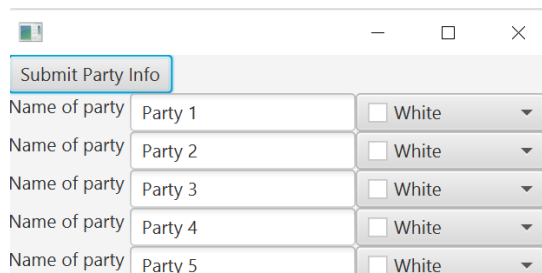
1. Add a method to the SetupController called *linkWithApplication* that takes an instance of PollTrackerApp as an argument.
 - a. This method should initialize an instance variable of type PollTrackerApp using this argument. (You'll have to add this instance variable to the controller.)
 - b. SetupController also needs to be updated to call the method *pollView* in PollTrackerApp when the *Submit Party Info* button is pressed. This will open the tabbed view that contains the other two windows that your team created.
2. Update EditPollController to add the method *setPolls*. This will be called by PollTrackerApp to provide the controller with the list of polls to edit in the view. So the controller will have to be updated such that it uses this provided poll rather than the one from the Factory that you used for testing.

3. Update VisualizePollsController to also add the method setPolls. This will also be called by PollTrackerApp to provide your controller with the list of polls to display. It will be the same list of polls that was provided to the EditPollController. So this controller should also be updated to use this PollList in the view instead of the random one it got from the Factory.

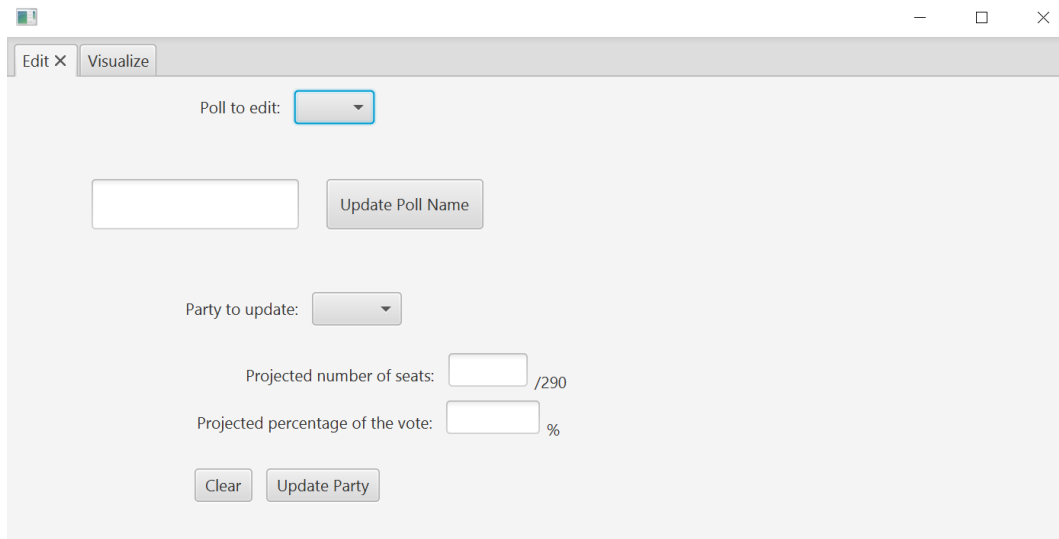
Make sure to edit this class to add documentation and to ensure it conforms to the team's coding conventions. Once it runs, it should look as follows when you run the application:



After pressing submit, the party name view should appear (as designed for the setup view):



When the user presses the *Submit Party Names* button, the following view should appear:



You should be able to switch between tabs at will and the correct, updated data will be displayed.