Java Independent Project Narrative - Aliya Petranik - Rate My Class

The Rate My Class application that I ended up choosing wasn't my first idea for the project. My first idea was a organization app for students, but I realized that it wasn't a very original idea. I was discussing ideas with my dad and we started talking about the Rate My Teacher website. I realized that I could take that idea and extend it to a rating system for classes instead of teachers. I didn't want to do it all within the Dr. Java console, so I searched online for a simple way to create a GUI. I found a java plugin called Swing that allows the creation of GUIs. Lucky for me, it can be very dulled down for simple users like me. I then proceeded to do a lot of research on Oracle about Swing. I read through documentation and compiled all the information into a View class file. In my 3rd commit, I learned how to create editable text fields in swing, meaning it would allow the user to type into the field. The next big step I made in Swing can be seen in my 4th commit -- Action Listener, the part of swing that detects and runs when the user types something into the text field. As seen in my first Record of Thinking, I was very excited when I saw a window pop out of Dr. Java instead of everything displaying in the console. This was a big step for me in my programming life, since I had never before coded something that wasn't console based. In my 5th commit, I started work on the Driver, which at the beginning did nothing but call the view class. I started adding all the courses into arrays by subject. Once I had gotten the base subject selection into the Driver, I attempted to display it. It looked a little awkward, so David gave me a few tips on Swing layouts. Seen in my 7th commit, I added margins to the screen so the text didn't look so awkward and squished against the edge. Once I had fixed the View class, I transferred the subject selection from the driver to a new class, which can be seen in my 8th commit. I had some challenges with displaying text from a long string without the application crashing or just displaying a blank screen. One of the main problems I had at this time was that blank screens kept coming up because the view class had nothing to display. I again went to David again for some help because he had experience using Swing. With his help I overcame some of the problems with the blank screens in commit #9. For the next couple of commits I focused on the survey, possibly the hardest part of my project. As seen in my 2nd and 3rd Record of Thinking, I originally had 2D arrays of strings as seen in commit #12 that held the information for each course and survey question. Then I talked to Mr. Kiang and upon his advice briefly changed to ArrayLists that held course objects. As I talked about in my 3rd record of thinking, the idea for a course object was great because it could hold the String ID, the number of entries and the actual averaged value for that course. The problem that I ran into was trying to display a course object. Since Swing only accepts strings, I was having a lot of trouble getting past that barrier. I decided that I should just go with what I know and so I went back to using 5 ArrayLists in my 17th commit, one for each survey question. I was still hitting a wall on getting the course string ID to correspond to an index in the survey question ArrayLists. Basically, if algebra 1 had to correspond to index 0. So in my 21st commit, I decided to label each subject section w/ index numbers. That way, when a user typed in a String ID, I could easily match it to a set of index numbers in the survey question ArrayLists. After that, it was easier to figure out how to take input and save it to the correct survey question ArrayList at the correct course index. After that, the rest was just trouble shooting and fixing errors. One of my goals was to make the GUI easy to understand and therefore not crash when the user types in the wrong return type (String vs. int). This was one of my main trouble-shooting problems. The program was so fragile and could easily crash if the user typed in the wrong thing or the order of things was typed in wrong. While I was going through and trouble-shooting crashes, I realized that every time the user closes and reopens the program the values for the survey question ArrayLists reset. This meant that the averages weren't getting calculated properly. This led me to one of the biggest challenges I had to overcome, created a "back" command. The intended purpose of the "back" command was to allow the user to go back to the beginning at any time without having to restart the application. I spent a couple hours trying to figure out how to restart the program if the user typed "back" without the program crashing. After I had finally figured that out, the user could successfully start the program over at the end and the program rarely crashed or showed blank screens. I felt so accomplished and relieved, so I commented the code that I hadn't already commented and cleaned up some code I wasn't using. I fixed up what crashes and blank screens I could find, but because Swing is such a new concept to me, I think my program is still a bit fragile and susceptible to random blank screens and malfunctions. I have confidence in the program, but I would love to continue my work in GUIs. Overall, this was such a fun project because I got to chose what I wanted to do, so I actually cared about what I was creating. I really did learn a lot and I now feel a lot more comfortable with Java and my programming skills.