Project Summary -   
  
Architectural Patterns:  
MVC (Model-View-Controller): I used Java Swing to implement my UI for this application. It was chosen because I had previously developed an application in Java Swing/AWT for work, and I was hopeful that this would make life easier for me. Java Swing uses the Model-View-Controller Architecture Pattern. The majority of my code falls underneath the heading of Model, the data of the application. While there are many functions built in, most of them run in the background and are unrelated to the display system. The exceptions are the functions bound to the buttons, aka bound to the Controller system. The most important functions bound to the controller system are the “Add Appointment” button and the “Save Changes” button. The AWT (Abstract Window Toolkit) and GUI elements of Swing are the View elements of the application.

Layered Pattern: This project can also be described as fulfilling a Layered Pattern. The GUI elements, created using Swing and AWT, represent the Presentation layer. The direct functions called by the GUI, such as “Add Appointment” and “Save Changes”, could be categorized as being a Service Layer. The logic underneath, such as the functions to create appointments, search through them, and read in external files are an Application Layer. Finally, there is the internal ArrayList of appointments and ArrayList of employees that is the Persistent Data Layer. Separate from the application but still arguably part of the Persistent Data Layer is the Excel spreadsheet that holds the data.

Summary/Retrospective: This project began as an idea to help a friend out. She is stuck using Excel as a scheduling tool, and really wanted a system that would automatically generate a visual schedule given a table of appointments. After striking out with Excel Formulas, I realized that I would need to program something, using Visual Basic, SQL, or a full programming language. When I needed to come up with a software project for this class, it seemed like a fortuitous coincidence.

The project implements several libraries. Java Swing provided the graphics and the User Interface. Apache POI provides the tools to read/write Excel Files. JFreeChart provides an extension to Java Swing to draw Gantt Charts, as a way to quickly implement the visual schedule. iCal4j was intended to provide the ability to export appointments and full schedules as iCalendar .ics files, but that was not implemented.

The final version of the project only implements the three most important features – the Visual Schedule, the Add Appointment functionality, and the Save Appointment functionality. Moving forward, what will most likely happen is I will finish the Draw Visual Schedule in Excel functionality, remove the GUI elements, and create a headless application for my friend to run.

I bit off more than I could chew. Yes, as an adult part-time student it is my responsibility to balance school, work, and personal responsibilities however school has to be third in the priority list (the list being 1. Work, 2. Health, 3. School) and there were some… issues this past month. This is all I could get done. I am sorry.

Screenshots







