Daniel Santillan

Mrs. Terri Kelly

DE Discrete Structures and Object-Oriented Programming

3 June 2025

222 Final Project Final Description

For my 222 Final Project, I built a functional education portal. The portal is a rudimentary technology allowing schools to maintain their grades, students, teachers, classes, and more under one database. The project involves school systems. This consists of a school with administrators, teachers, and students. The school is registered in the database by an administrator, who can add, remove, or edit users. These users each have a name and login, either set by the initial administrator or themselves. Each school has several blocks, or periods, set by the administrator. The range of blocks is locked between six and eight. Administrators create the classes and assign teachers and students to the classes. Students and teachers cannot be in multiple classes during the same block. Teachers, meanwhile, set the grading method for their classes. This can be a points value grading system, a weight system, or a traditional percent system. They also grade assignments they create based on self-established grading conventions. Teachers can also create announcements on a class homepage. These announcements can serve as a homework page for students. Students can view their grades and calculate their grades on a grading simulator. They can also view their infractions, which are imposed on them by teachers or administrators. These infractions come with reasons and names that the staff can modify. In general, this education portal digitizes many of the basics when it comes to school data, saving a lot of time and making the process for faculty and students more efficient.

My program was coded in Java. The program was separated into several packages: calculations, files, graphics, objects, and run. First, let us examine the objects. The objects package includes a User class. This has methods determining the role of the user as well as fields for the user's name, login, and unique ID. The roles of Administrator, Student, and Teacher are subclasses that extend the User superclass. The use of overridden methods to determine the user's role demonstrates the foundational object-oriented programming principles of inheritance and polymorphism. There are also School, SchoolClass, Assignment, and Infraction objects. Each have their own special characteristics, fields, and methods that are important to the program.

Next, let us examine the files. The files package contains the crucial DataManagement.java file. The DataManagement.java file uses the FileWorker class, which contains methods for reading and writing files, to read the schoolList.csv file. The .csv, or comma-separated value, file is the backbone of my project. Most of my data is stored in .csv files. Due to the use of commas, the code ensures that all user input does not include commas to prevent ruining data reading. The DataManagement.java file has methods to create User, Assignment, Infraction, and other instances based on what it reads in the files. The School objects created by the DataManagement.java file can look for their own files and create objects as well. The data structure of my project is very intricate. Each school is in the schoolList.csv file. Every school has a file for users, infractions, and classes. Each class has a separate file for class announcements, assignments, rosters, and grading methods. Lastly, each assignment has a file for grades. Each object has a unique ID that connects assignments to classes and classes to schools.

Since the data is moved to objects, it is mostly protected by encapsulation. Therefore, it is difficult to access and modify the data directly. This makes DataManagement.java even more important because it serves as a crossroads between the encapsulated data, which it accesses through methods from the object classes, and the static graphics classes. The graphics classes build the graphics of the project. They include text boxes and drop-down lists that take and give data from and to DataManagement.java. The graphics classes are connected by a card layout in Frame.java, which establishes an ActionListener to determine how the buttons move a user between pages. The graphics methods are also important because they not only have code to position graphics (using the SpringLayout) but also have some methods to check data to ensure it can be sent to DataManagement.java.

Moving from the data, the calculations class is only used for the student grading simulator and was placed in a separate package for organizational purposes. The graphipcs package also had some constants files to assist with frequently-used fields and methods. Lastly, the run package has the Run class that runs the project. Besides raw code and files, the program also uses images made by me in Photoshop for the headers of every page in graphics. The graphics classes use the ImageIcon class to obtain the image.

There were no major setbacks for me when writing my code and finishing my program. The largest setback was making sure that the program correctly wrote and read data from the .csv files. It often took me several tries to develop the correct code to properly handle the data. I also had small issues with the JTables that allowed teachers to grade assignments because I did not want every column to be editable. I had to figure out how to override a JTable method from the Internet to solve that problem. The only other setback I experienced was determining why my Javadocs did not fully load at first. I realized through trial and error that it was an error in how I

was formatting the export. Besides those issues, I had barely any issues writing the code. With better time management compared to the last project, I was more comfortable with the deadline and the project workload.

If I had more time with the project, I would enhance my use of file reading to build a messaging system between users in the system. These messages would not be exactly live, but with instant file writing and reading, it might be close. I would also use an API to add a calendar to allow for teachers to make deadlines and administrators to make special schedules. I initially chose to pursue this project because of the constant complaints I heard with Blackbaud. I learned that a software like Schoology or Blackbaud is very intricate and difficult to replicate, which makes me less angry with their issues upon understanding how advanced their proprietary work is compared to my project, which was already a slight challenge in making. All in all, however, I had a lot of fun with this final project. It helped me learn data management better and may be helpful in preparing for the Data Structures class next year.

Works Cited

Career & Tech HQ. "Java GUI Tutorial #65 - Create a ScrollPane Using JScrollPane Class In Java GUI Swing." *YouTube*, Feb. 2022, www.youtube.com/watch?v=OJSAnlzXRDk.

Coding with John. "Java File Input/Output - It's Way Easier Than You Think." *YouTube*, Apr. 2021, www.youtube.com/watch?v=ScUJx4aWRi0.

"How to Use SpringLayout." *Oracle*, docs.oracle.com/javase/tutorial/uiswing/layout/spring.html. Accessed 30 Jan. 2025.

Kapse, Smita. "Set JTextArea to Wrap by Word in Java." *TutorialsPoint*, Nov. 2024, www.tutorialspoint.com/java-program-to-set-jtextarea-to-wrap-by-word-in-java.

Mahajan, Aayush. "round up to 2 decimal places in java? [duplicate]." *Stack Overflow*, 28 July 2012, stackoverflow.com/questions/11701399/round-up-to-2-decimal-places-in-java.

martinbshp. "how to capture ok button pressed from JOptionPane." *Stack Overflow*, Dec. 2015, stackoverflow.com/questions/34108645/how-to-capture-ok-button-pressed-from-joptionpane.

Raina, Siddharth. "How to make a JTable non-editable." *Stack Overflow*, 2 Jan. 2012, stackoverflow.com/questions/1990817/how-to-make-a-jtable-non-editable.

user2855405. "How to format a JTextField so words don't cutoff?" *Stack Overflow*, Dec. 2014, stackoverflow.com/questions/27238902/how-to-format-a-jtextfield-so-words-dont-cutoff.