NewGui Individual Studies Application

Thomas Norman

**Introduction**

The original scope of this project was solely to work on Python code, as opposed to the typical Java ventures that we are more often exposed to. Due to some complications with Jython and getting python and java to interact in a favorable fashion, the final scope was changed to ensure the most stable operation of the app itself. The GUI was built with JavaFX, using FXML files for the design of each scene with a respective attached controller.java file. The package itself contains a main java class and several object classes to define the objects that are used to fill observable lists that populate TableView modules as well. While the code was uploaded to github with a similar yet very simplified version of the app in Python, the major version of the app is based around JavaFX and SQL, as all functions, handlers and tables are built around SQL functions and database tables and schema.

**The IDE**

The project was built first in PyCharms and IntelliJ (for java), later using Sublime Text for code editing, and Netbeans for final coding and deployment. In order to run a JAR from this program or to rebuild it as it is deployed, the User would need to install the JDBC Sqlite Jar, JavaFx and the updated Java JDK 8 with Lambda capabilities. In order to run the similar python project, one would also need to install the Jython Jar.

**Python Vs. Java**

While the familiarity of Java is hard to compete with, Python is quite easy to learn as it’s very simplified and meant especially for string queries, lists and text editing. Java, being a static language, requires declared variables, where no declaration is required in python’s dynamic language instruction. If the user is looking to quickly look at testing some code, python is great in that no class is needed to achieve instant code creation. Even a somewhat cumbersome process of file reading/writing, for which java requires several imports and exceptions in order to run, is very simplified in python requiring only the object to be defined in order to print the contents. Python, quite simply, requires only what is absolutely necessary and is therefore much more reliable, concise and resource light.

**JavaFX**

Though my past code creation has been mostly in Java, giving me the most familiarity with that language, I ultimately used JavaFX to create a more modern and graphically centered GUI with a fairly linear operation in terms of attaching controllers and actions. Due to my unfamiliarity with JavaFX and the fact that there are quite a few differences in language, large and small, I consider my usage of JavaFX to be a slight venture into another language as well. Using FXML sheets to design the physical components draws a lot from my experience in web design, but the idea of having one main class with subsequent controllers attached to each scene/FXML sheet is very helpful in terms of planning and editing. Being able to declare and define all necessary variables and objects and then go through the application to attach them to the respective graphical element helped to visualize every single component of this application. The most utilized component of JavaFX in this case has been the TableView model, which creates a simple table object that can hold an observable list of Items or database info, which is what I used in this case.

**SQL/Database**

The third main component of this application, along with JavaFX and Python, is SQL. I created a simple user table with usernames and passwords to create a match in our login screen. Next, I created a table for both faculty courses and student schedule courses. By creating a database connection and statement controller in my java code, I was able to access the database, print a successful connection message in the output terminal, and subsequently preform SQL statement queries to populate, add and delete table info built from the schema.

Though the initial Connection proved slightly problematic, all problems were eventually solved and the database proved to be a much more efficient and time manageable way to fill the tables, rather than trying to create hard lists. At one point I had even attempted to use a txt file to read from and write to but I’m pleased with my final decision .

**Integration of multiple languages**

In the field of software programming, I feel the one thing more beneficial than learning a new language is learning to integrate several languages, familiar or new, in a multifaceted application. My experience in SQL was not nearly as beneficial as I’d hoped, as it had mostly been work regarding table/schema creation and editing database permissions, but I’d never had much experience with it directly through Java code. Taking the weaknesses of each language out and utilizing each one’s strength in order to preform where it’s most needed helps the coder, I believe, to do a better job of planning , developing, and critical thinking when it comes to scratch-made code and applications.

Though I encountered many errors along the way, and ended up putting a great deal of time toward simply finishing the java components in order to have a complete project that at least saw itself through from point a to b. The mistakes and errors, as they should, served as great learning experiences and ultimately a better project overall. I’m pleased with the end result, though it’s not the exact product I had originally intended, and I feel that it is at least a resource friendly, intuitive application that runs correctly, passes operational tests and pulls from many facets of programming in general. I’m glad it ended up going the way it did as I feel it was more of a challenge than anticipated and the learning was much better versed.