Team Z Genealogy App

Kelvin Bui, Haris Kureishy, Kaylin Luck, Kendall Phillips, Kieran Taylor

Intro

Genealogy is complicated. Surely the average individual could name their parents, children, and siblings. Perhaps most individuals could also name all their grandparents, grandchildren, cousins, aunts, uncles, nieces, and nephews. Still, depending on how many relatives a person had, all those names can add up very quickly. What if you go even farther, and ask a person to name all their second cousins? Their parents' cousins? Their great aunts and uncles, or any other distant relationship? You're asking individuals to keep track of information that no person could realistically be expected to have a thorough knowledge of.

The solution? The Team Z Genealogy App. A single unified program that compiles and stores all information of your family tree, and provides you with several tools to easily search this material, and update it yourself.

Part A: What Our App Does

The Team Z Genealogy App is structured to logically follow the way a user would approach reading their genealogy information. The very first thing the system does is read in all information about the family that the user provided them. It then creates a family tree based on this information, and provides the user with a summary of all parsed information, specifically all of the people and marriages that are now in the family tree. Behind the scenes, this information is stored in a HashMap, which means this information is stored securely and can later be easily retrieved using unique personID values as keys to each Person object.

Following this introductory step, the user is taken to a GUI menu where they can select one of four options: add a new person to the family tree, add a new relationship to the family tree, search the family tree to find all of a specific type of relative for an individual in the family tree, and to edit the information of an individual in the tree.

Option 1: Add New Person: If the user needs to add a new individual to the family tree, our app allows them to do so directly, in order to avoid the hassle of adding a new person to their read-in data and then feed this information back into the system again. When "Add Person" is selected from the main menu, the user is taken to an "Add Person" screen, where they are prompted to add in all information for their new individual, including their name, and date and place of birth. The user can also note that the individual being added is deceased, in which case they are also asked to provide the user's date and place of death. Finally, the user is asked to provide the mother and father of the person being added, which can be selected from drop down menus of all men and women in the system. Buttons at the bottom allow the user to submit all information once complete, or otherwise clear all information given so far, or close the add person window all together.

Option 2: Add New Relationship: The user is able to add new relationships for marriages through this class. The new marriage and details will be added to a separate data attribute than the original marriage attribute, as any person in the HashMap has the ability to get divorced and remarried. This allows for viewing not only past relationships, but also new ones.

Note about relationships The addition of parent/child relationships is implemented while adding a new person, thus only a new parental relationship can be added for new people.

Option 3: Search Family Tree: Should the user desire to scan their generated family tree, the Team Z Genealogy App allows them to easily do this. Upon selecting "search family tree", the user is allowed to choose a person in the tree, and what kind of relationship they want to check for: children, parents, grandparents, or siblings. Once this choice is made, the system returns a list of all relatives of the specified person with the given type

Option 4: Edit Person: If the user makes a mistake while making a new entry or finds an error in existing entries, they have the ability to make edits to various attributes about any individual. Thos include: DOB, DOD, birthplace, death place, suffix, given name, or family name. We did not pursue editing existing marriage/parent-child relationships because of the way our data is stored.

After each option is completed, the user is taken back to the main menu, where they are able to either choose a new option, select the same option again (to add multiple people to the tree, for example), or indicate that they are done and close the program. Once the user indicates they are finished, the system will provide them with an output file composing all information that was given to the system in the first place, along with any updates made while using the system.

Part B: The Fine Details, and Comments on Quality

Each member of Team Z dedicated an incredible amount of time to our Genealogy App, to ensure it would run effectively and with the highest degree of quality that we could achieve. We had multiple team members with the job to routinely review our code for places to improve it, and even our primary coders were going through and reviewing their code for quality as it was created. Quality assurance was a big part of the design process for us: We didn't just want to have an app that worked, we wanted an app that would work well, and not cause multiple other performance issues for the user when implemented.

One of the recurring aspects of our design process was asking ourselves "If I was a user, what would I expect to happen at this point? Does this make sense for the user?" This is one of the primary reasons we pushed so early to incorporate GUI elements into our program. There is certainly nothing wrong with a terminal-level UI, but the fact of the matter is that with the level computer programs have advanced in recent years, most users are expecting a program that "does a lot of the work for them", so to say. The use of GUI elements made with Java's Swing toolkit reflects this idea that much of the "heavy work" of coding should be abstracted to make the final program more approachable and manageable for the user.

The behind the scenes architecture of our app reflects this commitment to logical sense, and it too was routinely checked in order to ensure we removed "code smells". The app is primarily run out of a "GenealogyApp.java" file, but makes extensive use of supporting classes to ensure we didn't have any "God objects" or "Super classes" that were charged with doing too much in the system. We have a separate class for each of the four GUIs listed above, as well as behind the scenes classes that define logically what a person is, and a class that controls the gene database storing all these people. This architecture is also reflected in the Class Diagram our team created, shared, and updated multiple times, thus providing us with a visual model of the architecture of our system that saved us from many headaches.

Through the process of code review we described, we were able to catch many of the bugs present in our system, which have now been dealt with, and which the customer should not even worry about. That said, it is often impossible to make a perfectly airtight system (and in fact, doing so can sometimes create more problems than it solves), so we are sure some bugs still exist in our code. However, these are minor bugs, mostly relating to the specific challenges that arise from GUIs: a frame may occasionally appear too early, or a button may be a bit too far left on the page. Bugs are an inevitable part of the coding experience, but in our case the major ones have already been dealt with.

Part C: What We Learned

Prior to creating this app, none of us had any experience creating GUIs using Java Swing, so that was a large takeaway and new skill developed. Additionally, the role of clear communication became very important due to not only the online environment but also the fact that when we had good communication, our project came together much more smoothly. Our group is very happy with the way we were all able to bond and gel while creating this project, and the fact that we often had fun in the process. While there were many individual components and assignments, we were able to coordinate well with each other to ensure all expectations were met in the final product, which is absolutely a team effort. If we were to do something different, we may refactor how relationships are stored to make editing them easier. Additionally, we may have spent more time debugging to ensure all data is stored properly and attributes are easily modifiable.