Requirements for Genealogy App

The genealogy app functions by reading a provided genealogy document, and then creating a family tree based on this data. Once the family tree has been created, it should provide a number of quality of life features to help the user read and navigate the tree, such as the ability to search for members of the tree or add new members.

Functional Requirements

1. Upon running the app, the user is given options to “Search”, “Add”, and “Exit”. Here, it allows users to add any person to the app and view their relationships to other people.
2. The user can read in a text file of current people, their family relationships, and their partnering history. Upon request they can save any changes made back to a text file. The user will be provided with an example text file to work with, including a default format. The public class GeneDataBase reads in files and creates a HashMap to store and relate each person added to the database.
3. The user can determine if a given person or people are known to the app. The user can look up a person or people by their family name, given name, or full name (names do not need to be unique as they often repeat in families across generations). The app will tell them if the name or names are not in the database and therefore if the search cannot return results.
4. If a user searches up a family name, given name, or full name contained in the database, the search methods will find and return a list of IDs that match that name.
5. The user can search for a person in the database but also access data about that person, such as their children or their birthday. The user can find any person’s parents, children, grandparents, or siblings, as long as that person is present in the database. If any requested relative(s) cannot be found, the user will be notified. If these relatives are found, an ArrayList is returned with their IDs.
6. For a person present in the database, the user can search for a person’s spouse whose database ID is returned if they are present in the database.
7. Since children result from partnering between the child’s biological parents, the app includes records of partnership and any offspring associated with that partnering. Children are related to both their biological parents and parents may be in more than one partnership at the same or different times. It is required to trace all children of each person across all partnerships. The user should be able to access any relationship via one person.
8. It is likely that some members of the tree will be unknown so there will be placeholders for them (“N/a”).
9. Using a GUI, the user should be able to add a person to the database and provide information about them that is then entered into the database. They can select the person’s parents from men and women already present in the database.
10. The createPartnership method creates a new partnership between a woman and a man. A String array containing information about a marriage is passed as a parameter and iterated through. For each element in the array, if it is not blank, the user can retrieve the names of the members of the partnership via the geneMap and add them into a new partnership or set their marriage details depending on which index of the array is currently being read.
11. The user can record a new child(ren)of an existing or new partnership. Given a String array, parents are set for a particular child if applicable. Children are also set for parents.

Nonfunctional Requirements

1. Generality
   1. A Person added to the database can be added without requiring knowledge of if this person is a parent, married or not, etc.
2. Usability
   1. Implementing a GUI so that the user does not have to type in the command line, all formatting for input is explicit and clear, and allows for a more aesthetic design in data entry.
3. Consistency
   1. All people have the same attributes available to be accessed, but unknowns will be kept as “N/a” to prevent any data gaps.
4. Testability
   1. Test set of data allow for testing, which can be used to look for bugs or other defects
5. Reusability
   1. If a different family were to have a file in the same format for read in, the same system could be used to create a genealogy tree without relying specifically on Johnson family data.