ICT373 - Assignment 2

Author: *Matthew Boan*
Student ID: *33276203*
Date: *31/05/2020*
File Names:

- main.java
- Controller.java
- Person.java
- Address.java
- main.fxml
- newRelative.fxml

Purpose: *To design and develop a family tree application with a graphical user interface using JavaFX.*

Requirements

Overview
The assignment asks for a graphical user interface (GUI) application for recording information about an abstract family tree. The family tree will consist of a root person that will expand out with immediate relatives and also a spouse. The root person can have zero to many children, zero to 2 parents, zero or one spouse. Any of these relations can have their own relation creating the tree structure.

Assumptions

Assumption	Description
Selecting a relative	When selecting a relative I assume that the program just allows you to edit that person, and add relatives to that person. Not load that person as the new root person.

User Guide

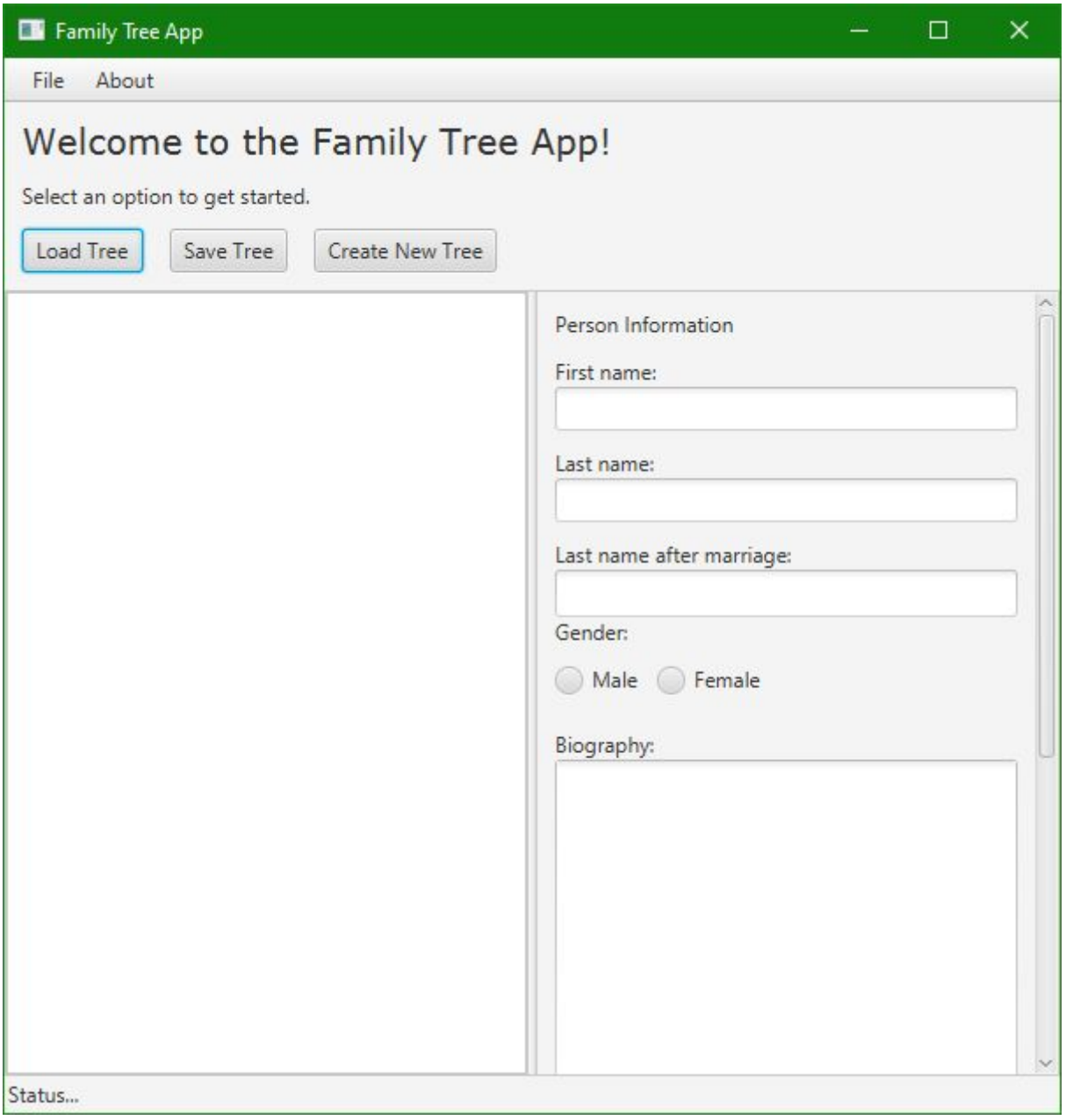
Compile
To compile the source code run this command:

```
javac -classpath "WHERE JAVAFX.JAR IS" *.java
```

Run
To run the compiled class run this command:

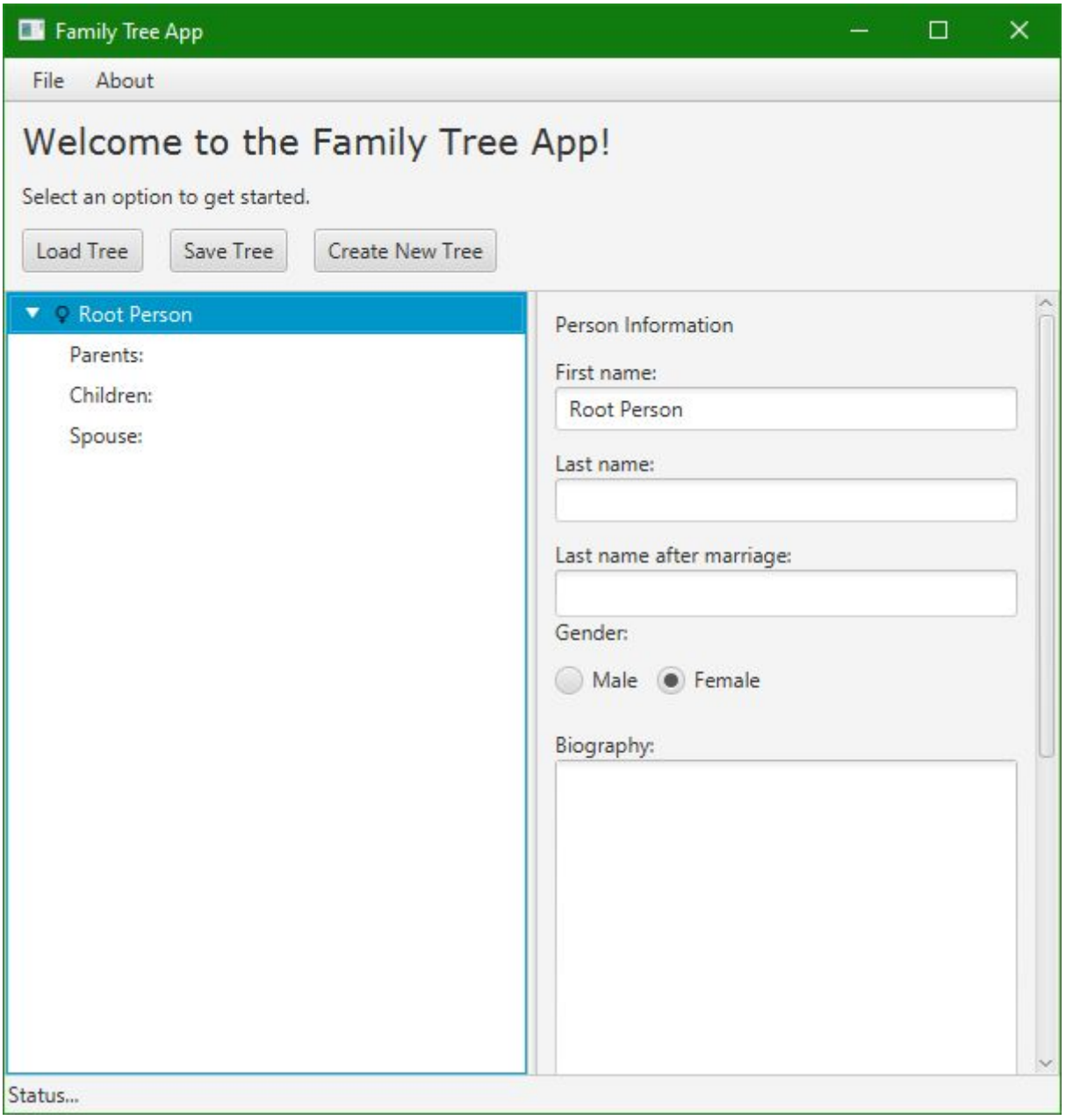
```
java -classpath "WHERE JAVAFX.JAR IS" main
```

How to Use the Program



Above is what you will see once you have launched the application, you can do the following actions:

1. To create a new tree either click the create new tree button, or go to the file menu and select new tree.
2. To load a tree click the load tree button, or go to the file menu to select load tree
3. To save a tree click the save tree button, or go the the file menu to select save tree



Once a new tree is created you can start editing the root person, and adding relatives to the tree.

Structure

Model View Controller Approach

I decided to use the model view controller (MVC) as I was developing a graphical user interface (GUI). A MVC is an extremely well documenting development method for developing anything that requires a visual representation. It is really good for GUIs because it separates the business logic and the view/GUI logic. The model is the business logic, the “objects”. The View is the GUI, the controller is the component that sits between the models and the View.

The advantage of using the MVC is that each component can be changed without affecting the others too much. So if we can reuse the Person class in other applications, and other GUI renders such as Swing. This makes developing and expanding the application a lot easier. It also isolates the errors to one particular aspect of the MVC.

In my case this worked perfectly, I could model People for the tree view and display them to the user using the view, and the controller will handle actions and manipulate the models and the view.

Classes

Class Name	Description
Person.java	The Person class is for representing a person in the family tree. It has a first name, last name, last name after marriage, gender, biography, an Address object to store the Persons address, a Parent array for storing parents of type Person, an ArrayList of children to store children of type Perso, and a spouse of type Person.
Address.java	The Address class represents an address, it encapsulates a street number, a street name, a postcode, and a suburb.
Controller.java	The controller class handles events from the GUI such as buttons being clicked, etc.
main.java	The main class is where the static main function lives and where the controller, and main window are created to start the application.
main.fxml	This is an FXML document for the layout of the main GUI window.
newRelative.fxml	This is an FXML document for the layout of a new relative window.

Limitations

Due to time restrictions caused by COVID-19, and one unit ICT283 not marking an assignment that was submitted months ago that is a continuation for the second assignment I have been unable to fully complete this assignment. I would have liked to include some more try catch blocks to catch errors during runtime, such as inputs not being validated, changed, etc. However, I think with more time it is clear from my code that I could have implemented them, my apologies.

Also some further testing is definitely needed to ensure that more bugs and errors are found and fixed.

The documentation in the code is also really bad, my apologies, it needs some javadoc comments and more explanations, especially for the recursive methods that could confuse other programs and even myself later if I come back to this code.

Testing

To test the application I chose to test to see if event listeners were firing when certain GUI elements were interacted with e.g. clicking a button. Then I set a list of unit tests to be certain that everything was working okay.

TID	Function Name	Expected Result	Pass
1	handleNew()	Handle creating a new tree, create a new treeview and populate the tree with one root person.	Yes
2	handleOpen()	Handle opening a tree, open a file chooser and run the function loadPerson() passing the file chosen by the user to it.	Yes
3	handleSave()	Handle saving a file, open a file chooser and run the function saveTree() with the rootPerson and the file chosen by the file chooser.	Yes
4	handleEditDetails()	Handle editing a Persons details, get all the input text from all the inputs and set the Persons details to these inputs	Yes
5	handleAddRelative()	Create a new window to allow a user to add a relative of their choice to the selected person.	Yes
6	saveTree()	Save a serializable object to a file that is passed into the function	Yes
7	hadnTreeItemClick()	Handle click events for the treeview. If a person is selected populate the person info panel.	Yes
8	printPerson()	Print a person and all of their relatives, for debugging, and possible other uses.	Yes
9	repopulateTree()	Repopulate a tree from the root person only	Yes
10	populateTree()	Recursive function that populates the treeview with treeitems of persons	Yes
11	populatePersonInfo()	Populate the person info panel with the selected persons info	Yes

Conclusion

Overall I know this assignment could have been a lot better including more error checks, try and catch blocks, smoother code and GUI. However, one aspect which I thought I did well at was how the family tree GUI was populated. Because it is a tree I thought the best approach would be to implement a recursive function to populate the tree and then just call that

function every time a change happens. Also I think I did a good job at making the GUI look neat and intuitive.