MURDOCH UNIVERSITY

Assignment 2 – Family Tree Application

ICT373

Gareth Griffiths - 32577671

5/9/2019

All source files can be found in the project directory in the folder ‘source files’. FamilyTree files are located in the same directory.

A family tree application to create, save, load and view family trees.

Contents

Requirements and Specifications

Requirements

Requirements

The aim of this program is to be able to view, create, edit, load and save a family tree. The family tree is to be viewed through a JavaFX GUI with appropriate options to modify the tree (search, edit etc..)

The family tree is always started with a root person. A person will have a first name, surname at birth, surname after marriage, gender, life description and address. The address contains a street number, street name, suburb and postcode.

A person has the following immediate relatives: 0, 1 or 2 parents, 0 or more children and 0 or 1 spouse. The relatives should be outputted to a GUI element to display the family hierarchy. The family hierarchy should be able to be saved, loaded, and edited as well as the option to create a new family.

Assumptions

- There can only be 2 parents to a Person
- The parents do not have to be mother and father e.g. could have two fathers
- Cannot edit a person's immediate relatives after they have already been added to the system. E.g. remove a parent or change a parent to a spouse.
- Cannot add relatives to spouses and parents as these items would not be immediate family.

User Guide

Compile and run the program

As this is an Eclipse project, this program will need to be imported into eclipse and then right click on the class GUI and select run as Java application. This will compile and then run the program.

If there is a class path error, this can be fixed by right clicking on the project and selecting properties. Then go to java build path then click the libraries tab. Click on the unbound JRE System Library and then click edit. From here select the JRE 1.8 version installed on the host computer.

How to use the program

Create a family

1. Open the program
2. Either select create new tree at the top of the GUI or select add root person. This will open the GUI to insert information about a new root person.
3. Enter the details into the add root GUI. All fields are required. Street number and postcode must be numerical.
4. Once the root person is added, relatives can now be added to the root person. To do this, select the root person in the GUI and click the 'Add relative' button.
5. A new GUI will appear like the GUI for creating a root person, this time you must select what type of relative is being added (child, spouse, parent).
6. Once the details are entered click 'Add relative' and the relative will be added to the root person and the GUI will be updated to reflect this.

View a family member

1. Open the program and then either create or load a family tree.
2. To view a family member, simply click on the family member you want to view and the details will be shown to the right in the GUI.

Edit a family member

1. Open the program and then either create or load a family tree.
2. To edit a family members details, simply click on the family member you want to edit.
3. Click the 'Edit details' button to the bottom right of the GUI.
4. The GUI will change to allow the user to change details about a person.
5. Once you have changed what you want, click the 'Save button'.
6. The changes will now reflect in the system.

Load a family

1. Open the program.
2. Click the 'Load tree' button at the top of the GUI.
3. A dialog box will appear in the current location of the project.
4. Select the file you want to load e.g. 'family 1'
5. The family will then be loaded and displayed onto the GUI, from here it can be modified, saved and viewed.

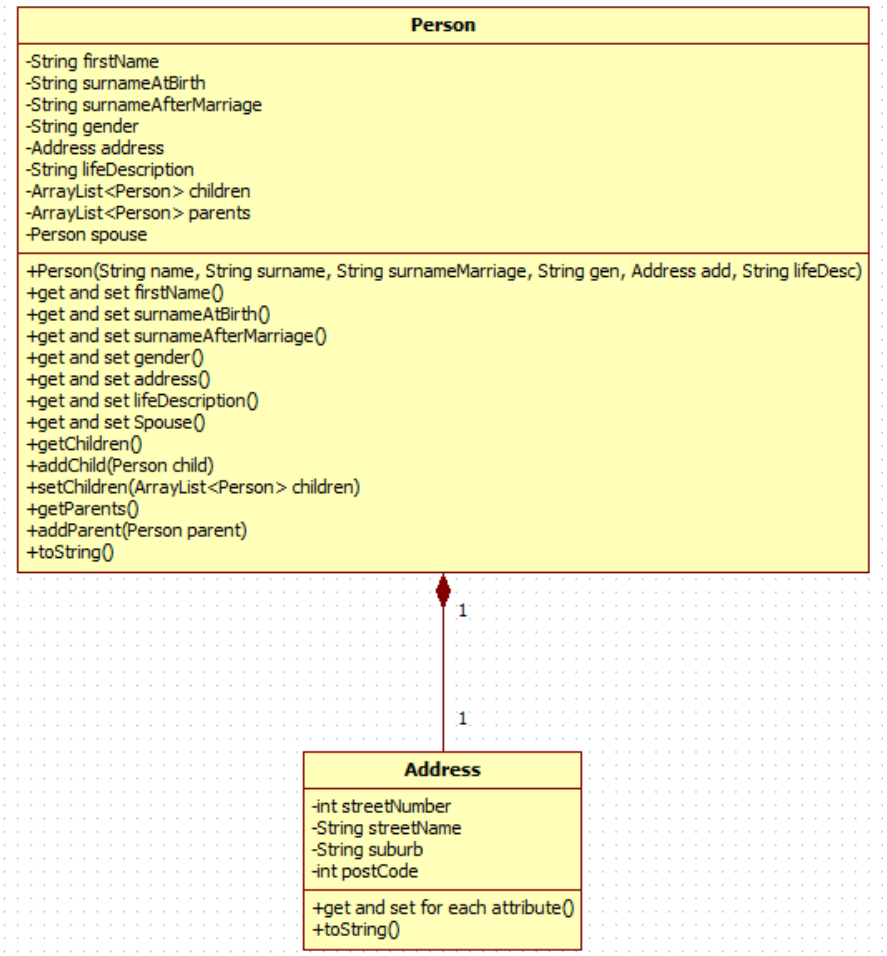
Save a family

1. To save a family, first create or load a family.
2. Click the 'Save tree' button at the top of the GUI.
3. A dialog box will appear where you can select a save location.
4. Once a location has been set the tree will now be saved.

Structure / Design

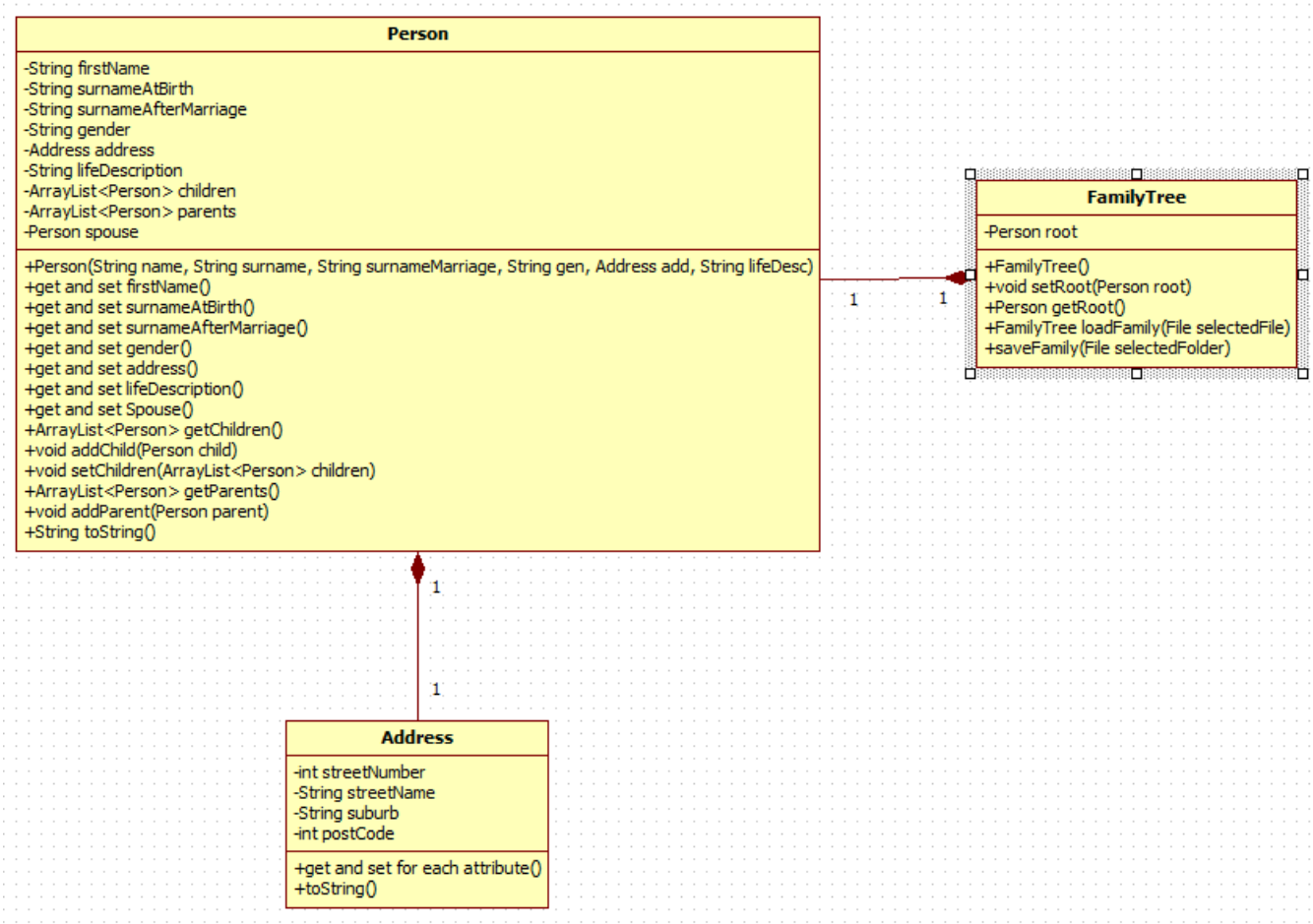
Person objects and family hierarchy

A person is defined with all the relevant details for a person as outlined in the specification. A spouse is just a Person contained in a person object, the children are an ArrayList contained in a Person and parents are an ArrayList contained in a Person. Each Person type will have a relevant add method e.g. addParent which will do some validation and then input the person to the list. A person is defined as follows:



With spouse, parents and children, Person objects can be linked together to create a hierarchy. We can construct a FamilyTree object of immediate relatives using an algorithm which will organize the hierarchy and display in a JavaFX TreeView. This is the main algorithm behind the program.

The FamilyTree class will encapsulate a tree and hold a root person with getters and setters for the root; this will be the starting item for the TreeView. Technically the FamilyTree will hold many Person objects that are linked to the root Person, but by definition there is only one person in the FamilyTree class. FamilyTree will also hold the methods needed to save and load FamilyTrees. FamilyTree is made to separate the data and business logic from the GUI i.e. a family tree can be outputted without the need of a GUI (the method just needs to be created). Having a FamilyTree class also makes saving a lot easier.



Algorithm for adding person's to hierarchy and tree view:

```
AddFamToTree(TreeItem treeRoot, Person root)
If(root != null)
TreeItem parentHeading
If(root.getParents <= 0 or root.getParents == null)
        ParentsHeading = null
Else
        ParentTitle = new treeItem
End if
For(parent in root.getParents)
        TreeItem parentData = TreeItem(parent)
        Check and assign gender graphic
        ParentTitle.getChildren().add(parentData)
End for
If(parentTitle != null)
        Root.getChildren().add(ParentTitle)
End if
End addFamToTree
```

This algorithm is also used for adding spouse and children, just modified to cater for different types of people. Every time a tree or person is edited the TreeView will clear and recall addFamToTree.

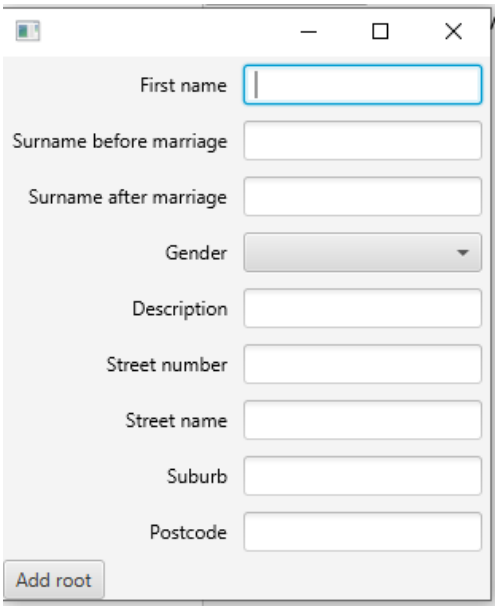
GUI

The main GUI class is called GUI. This is a singleton class that creates the main GUI and controls the state of the main GUI. The states are controlled by the user clicking on buttons e.g. if the user click edit when a person is selected the GUI changes to an edit state and EditPerson GUI is shown to the main GUI. The main GUI also holds the method with the algorithm to add Persons to the TreeView.

Below are the classes called when a state is changed by clicking a button:

Click ‘Create new tree’ or ‘Add root person’:

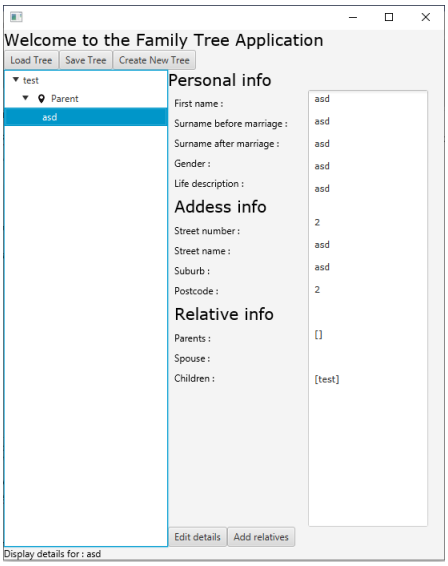
CreateRoot class is called which will construct to GUI for creating a new root Person. This will look like:



Once ‘Add root’ is clicked a Person will be created with the entered details and added as the root of the TreeView and then the main GUI will go back to its default state. *All options need to be inputted*

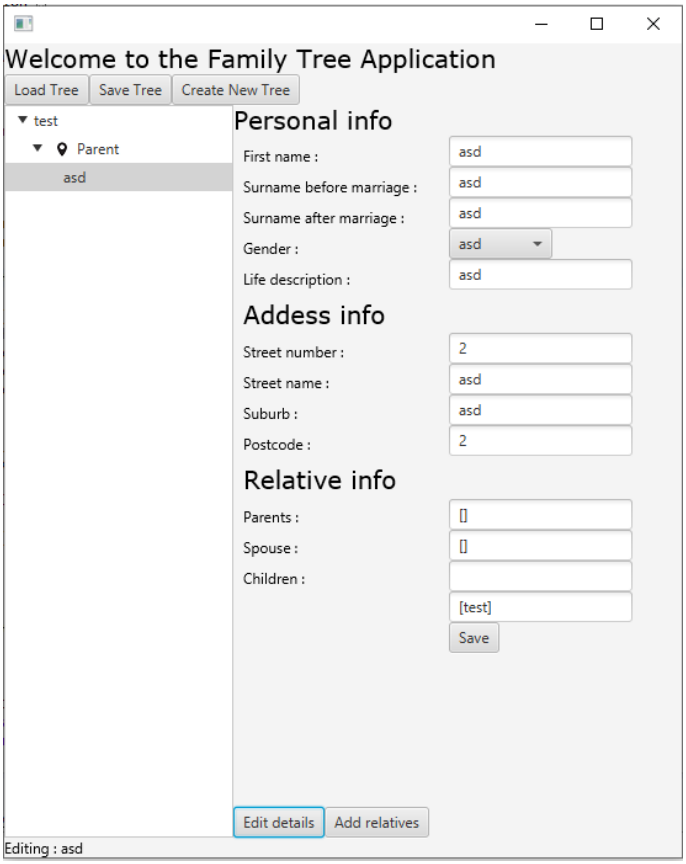
Click on person item in the TreeView:

After a person is selected in the tree view an action listener is called which calls the updateSelectedItem method, which will get the selected item of the TreeView and display the Person attributed to it. This is what viewing looks like:



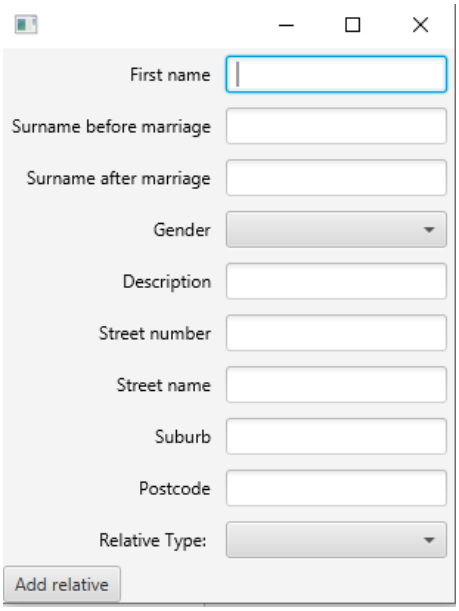
Click ‘Edit details’

When a person is selected a button called ‘Edit details’ will appear. When clicked this will call a class called EditNode. EditNode will create to GUI necessary to edit details of a person and will call the set method of the Person to change its details when the save button is pressed, pressing the save button will also change the GUI back to its default state. Here relative information cannot be changed but everything else about a person can (relatives can be added when clicking the ‘Add relative’ button). This is what the EditNode GUI looks like:



Click ‘Add relative’

When a person is selected in the TreeView, a button called ‘Add relatives’ will appear. When clicked this will call the class CreateRelative. This class will construct the necessary GUI to add a person and set its relation type (similar to add root). The GUI will look like this:



When ‘Add relative’ is clicked a person will be created with the inputted details and will set this Person as the relative type of the selected person in the TreeView. When the addFamToTreeView method is called in the main GUI again, the TreeView will display the updated family tree with the relative that was inputted. This will happen when GUI goes back to its default state. *All options need to be inputted*.

Click ‘Save tree’

When save tree is clicked a dialogbox will be shown to select a save location. Once selected each person object will be saved to a file. Each Person and Address is serializable and contained in a FamilyTree class which is also serializable. When this FamilyTree class is saved as an object, each Address and Person will be saved with it, thus saving the entire family tree to a file.

Click 'Load tree’

When load tree is clicked a dialogbox will be shown to select a file to load. Once selected, the FamilyTree class will load objects into Person objects in the exact same way they were saved, this will create a FamilyTree which will be outputted in the main GUI.

GUIFormatter

This class is used to create custom GUI items e.g. a Text object linked to a TextField with correct formatting which can be returned to other GUI’s for use. This is especially helpful for quickly creating GUI’s without having to do the formatting every time; this means a lot less code to write. For the purposes of this project I have a textAndCombo and textAndInput method.

Limitations

- When creating a new root or creating a new relative, all information has to be entered otherwise an error will occur. (Does not happen when editing a person).
- The way Person details are outputted does not wrap text, a scrollview is needed to see some details. (Especially description)

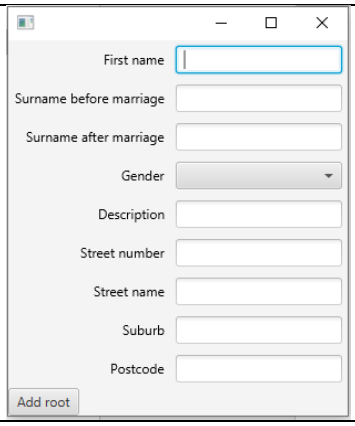
Testing

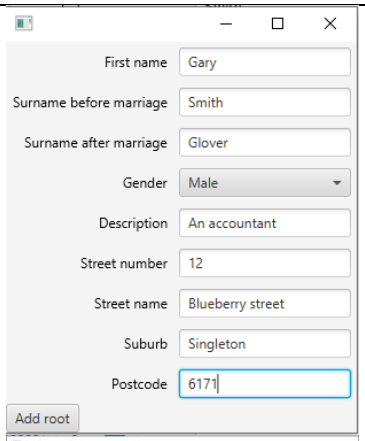
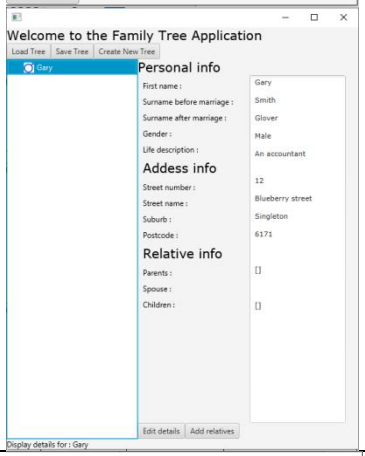
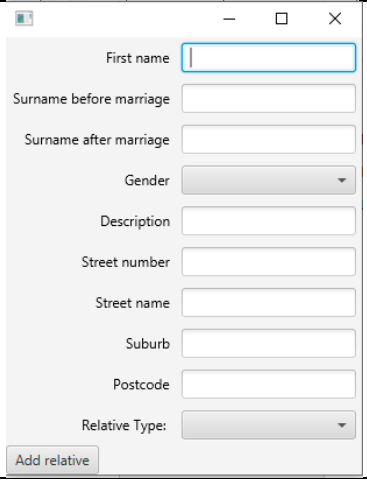
Strategy

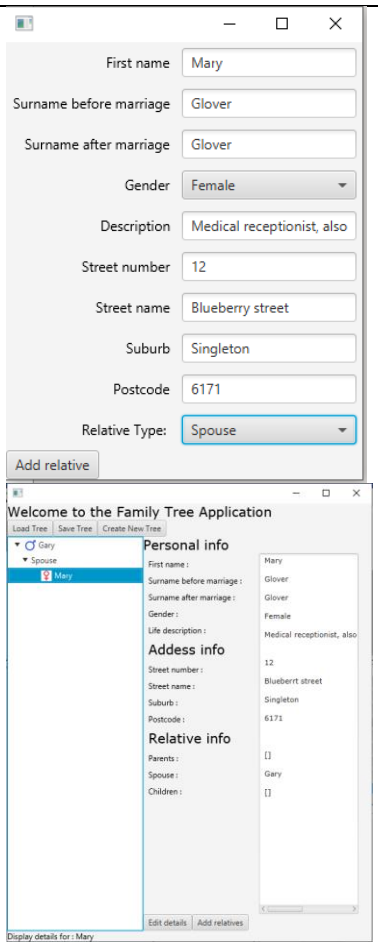
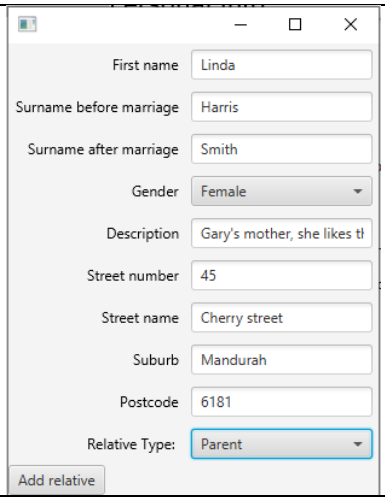
The way I am testing this program is to incorporate a new feature e.g. creating a new family and then testing its overall functionality, if anything goes wrong with the test (even minor details) it will have to be fixed until the test is passed (or test will be marked with a N if it still fails). This way all code needed to get the expected result will have to be working and do its job.

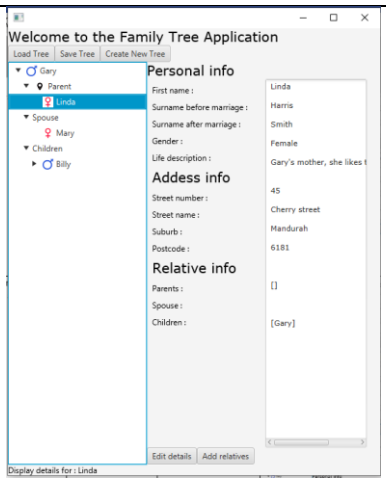
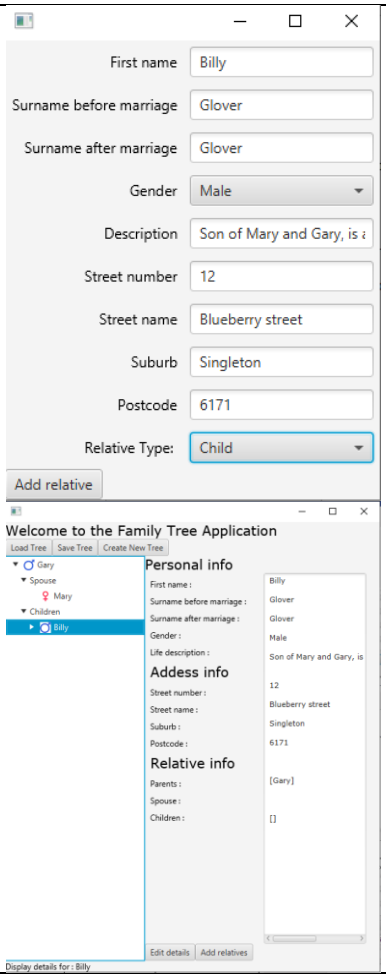
Tests

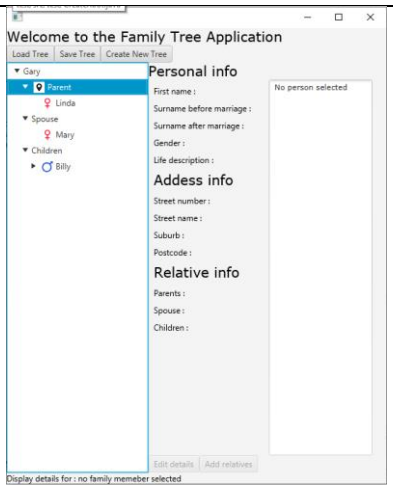
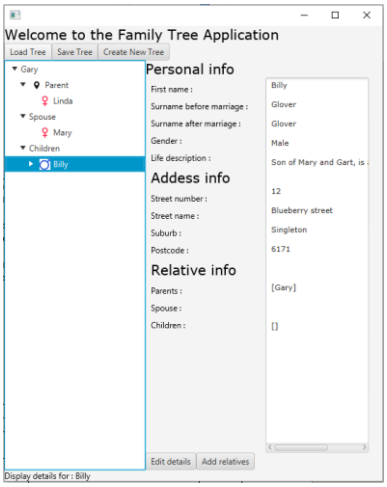
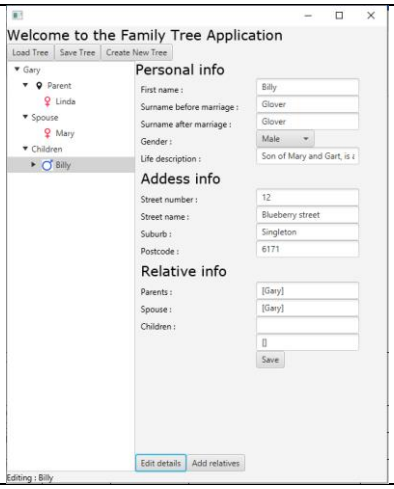
Y = passed, N = failed, P = partially working

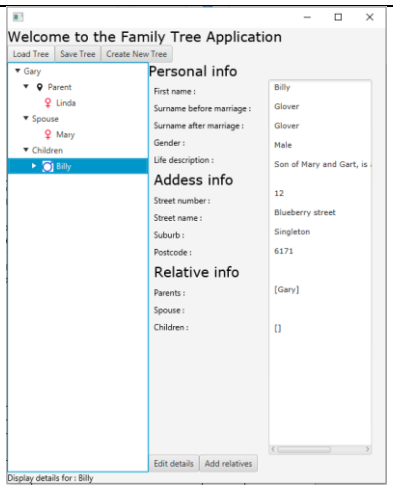
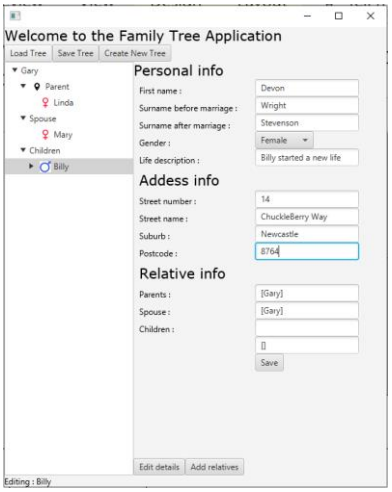
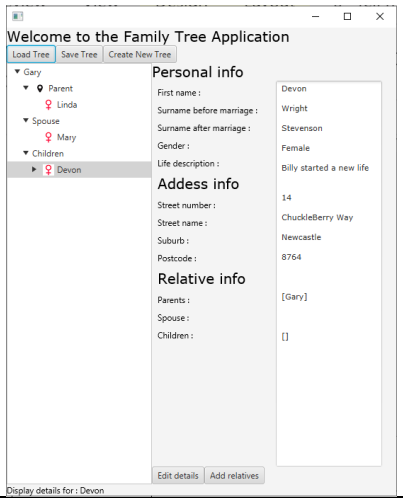
Test Name	Input	Expected output	Output	Passed
'Create new tree' button	Click 'Create new tree button'	CreateTree GUI will show		Y

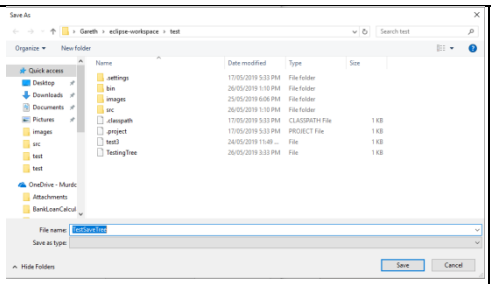
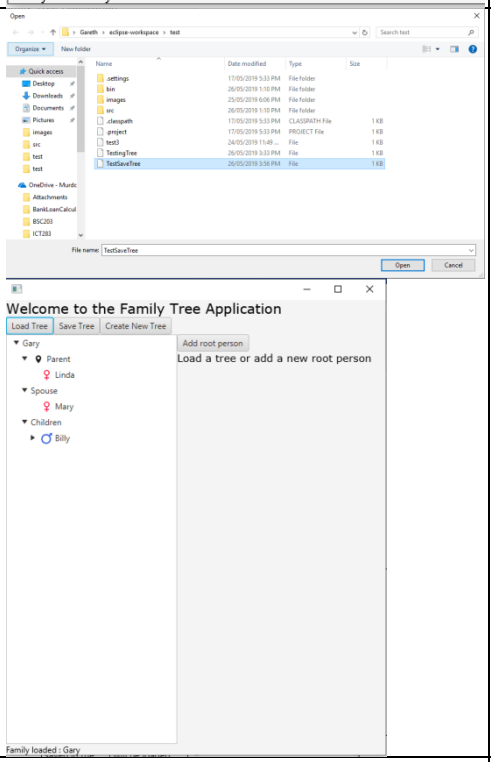
‘Add root’ button	Click ‘Add root’ button with all fields entered	CreateTree GUI will close and the treeView will be updated in the main GUI.	 		Y
'Add relatives’ button	After clicking on the root in the TreeView click the ‘Add relative’ button	CreateRelative GUI will be created and shown.			Y

‘Add a relative’ button as Spouse	Click ‘Add relative’ with all fields entered and spouse selected as the relative.	CreateRelative GUI will close and the TreeView will be updated with the new relative added. All details of the relative should be correct especially the Spouse detail.			Y
‘Add a relative’ button as Parent	Click ‘Add relative’ with all fields entered and parent selected as the relative.	CreateRelative GUI will close and the TreeView will be updated with the new relative added. All details of the relative should be correct especially the children detail.			Y

				
'Add a relative' button as Child	Click 'Add relative' with all fields entered and child selected as the relative.	CreateRelative GUI will close and the TreeView will be updated with the new relative added. All details of the relative should be correct especially the Parent detail.		Y
Select a relative in TreeView	Select all non person and person items in the TreeView without bugs	Whenever an item is clicked the correct details will be shown, if the non-person tree	Non person tree item:	Y

	or errors.	item is clicked no details will be shown and the edit and add relative button will be disabled.	 Person tree Item: 	
Click ‘Edit' button	Select a person tree item and click edit button	The EditNode GUI will be shown in the main GUI.		Y
Click ‘Save’ button	Open the edit menu on a person tree item, change all details apart from	The EditNode GUI will close and the updated item will be shown in the treeview	Before:	Y

	relative info (relative info cannot be changed in this program, only added) even if these values are change they will not reflect in the program.	with correct new details.	 Editted:  After: 	

Save a family tree	Save the tree created for these tests. Click save tree and select a save location	The FamilyTree will save to a file without any errors.		Y
Load a family tree	Load the tree saved in the 'Save a family tree' test.	The FamilyTree will be loaded from file and outputted to the TreeView. This test also includes going through the test plan again.		Y
Check infoText updates (the text at the bottom of the GUI)	Run through the entire test plan specifically looking at infoText	All info text will be relevant to the operations being done in the program.	If CreateRoot GUI or CreateRelative GUI is closed without saving, the infotext will not update.	P

Discussion

The tests I have included make sure the program functions as specified but more tests are needed (unit tests) for Person, FamilyTree and Address (test getters and setters). This will guarantee all functionality of these classes is concrete, and the classes are re-usable. The testing of this application only encompasses testing of the actual application, not every part of functionality of each class.