ICT 373
Assignment 2
Family Tree Application
2020

Student: Faiz Syed

ID: 33243485

Professor: Ferdous Sohel

Table of Contents

Title

This report documents the Family Tree Application project, presenting its requirements, specs, assumption, user guide, limitations, UML diagram, design description, State transition diagram, Activity diagrams, and testing strategy including sample test results.

Author: Faiz Syed

Date: 31/05/2020

File names: *FamilyTree.java*, *FormHandler.java*, *TreeHandler.java*, *Person.java*, *Spouse.java* and *Child.java*

Purpose: To present a document that provides thorough description of the Family Tree application

Requirements and Specifications

The project aims to design and implement, in Java, a basic graphical user interface (GUI) program for recording information about a family tree. The family tree will have some family member, and each family member has a first name, surname, maiden name, description, gender and address. The address will have a street name, street number, suburb and postcode.

The family tree will contain the information of a member's immediate relatives like father, mother, zero or one spouse, zero or more children. The application should allow adding a new person, editing an existing person, adding spouse or children, saving the family tree, and loading an existing family tree.

The GUI will display details for one person at a time. The family tree will have a specified starting person called the root person. If a person has parents, children, and any spouse then the GUI should display the family in the form of a hierarchy which can be easily navigated to view the details of any specific family member. The GUI will have a separate mode for editing which protects the family tree from any unintentional changes. In editing mode, the user is allowed to make changes to the details of a chosen member.

Some of the key Java features implemented in this project are – *JavaFX*, *Object Serialization*, *FileChooser*, *Exception handlings*, *TreeView*, and *TreeMap*.

Assumptions

1) The root person type can only be of child person type
2) All input fields will be required in the person details form
3) A person can have only zero or one spouse
4) A person can have zero or more children
5) The form user input will be of appropriate type
6) The member type cannot be changed even in the editing mode
7) Relatives cannot be added to a spouse person
8) To start a new tree, the user may choose to restart the application after saving the current family tree

Limitations

1) Does not represent relationships like grand children or grandparent among family members
2) No validation of user input has been integrated
3) GUI is not designed to be used in full screen mode
4) Functionality to remove members should ideally be present, but it could be implemented in future iterations on this same project
5) FXML for GUI design was explored but could not be implemented due to time constraints, hence, basic CSS has been used instead.
6) Only accepts family tree files in .TXT format
7) To empty the current tree and make a new one, the application needs to be restarted. Although this functionality does not have any purpose other than giving user the ability to start a new tree without restarting the application, but it is handy to have. However, it could not be implemented in this iteration due to serious time constraints and focus being on making core functionality work efficiently and documenting them.

User Guide

System requirements:

- ✓ Windows 7 (or later version) will be ideal to support this program

Compiler requirements:

- ✓ Java Compiler and utilities which can be download from the internet free of cost will be required.
- ✓ Java SE Development Kit 8 Update 22 (64-bit) need to be installed on the machine to run the program.
- ✓ Kindly follow this software installation instructions for the JDK: http://www.java.com/en/download/help/windows_manual_download.xml
- ✓ The following packages need to be installed:
  - jdk-8u202-windows-x64.exe
  - Java SE Runtime Environment 8u202 32-bit & 64-bit
  - jre-8u202-windows-i586.exe
  - jre-8u202-windows-x64.exe

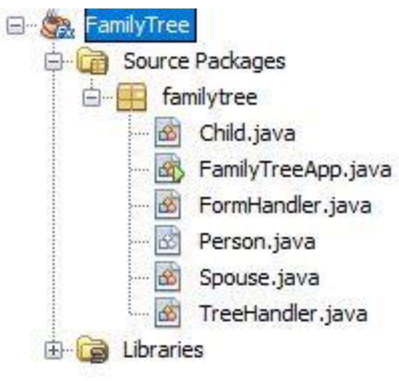
Install NetBeans IDE:

- ✓ NetBeans IDE 8.0.2 can be used to run the program
- ✓ Install NetBeans 8.0.2 from https://netbeans.org/downloads/old/8.2/

Running the JavaFX application:

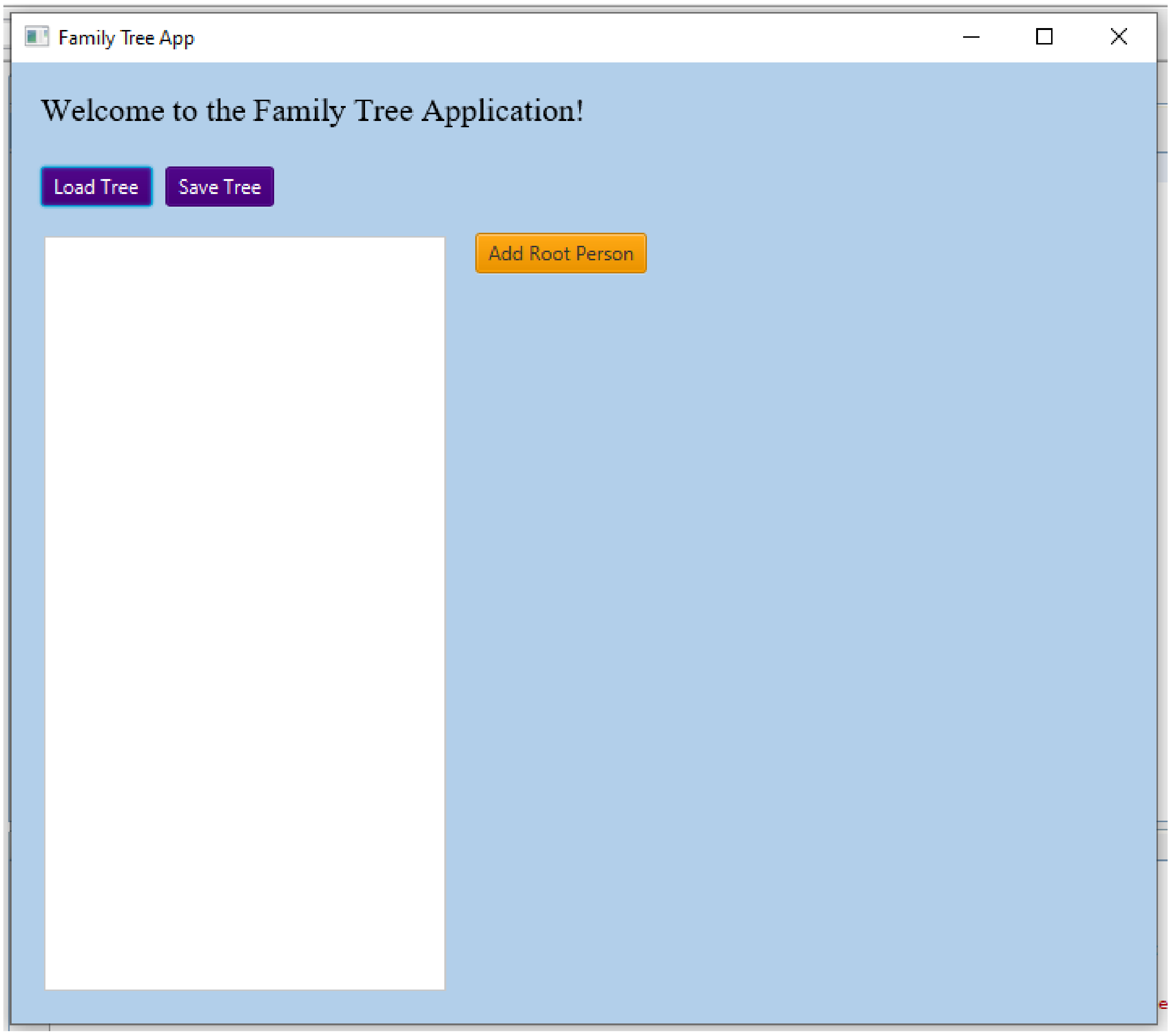
1. Open NetBeans IDE

2. Load the FamilyTree project in NetBeans (Make sure you set all the system paths before this)



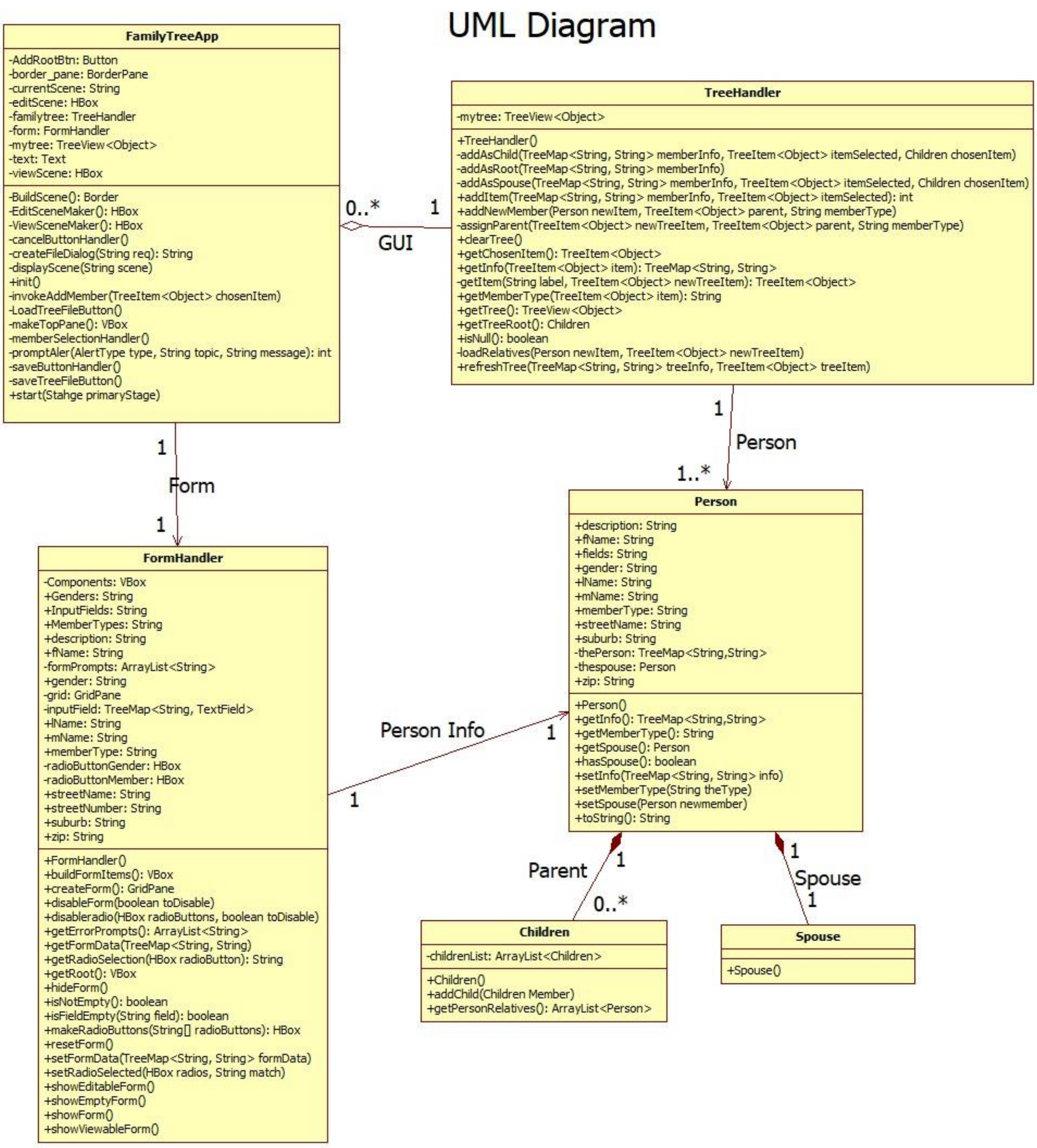
3. Click Run Project button (F6) to start the application

4. The application will start the GUI for family tree



5. The user may choose to create a new tree or load an existing tree into the application

UML Diagram


UML Diagram
FamilyTreeApp
-AddRootBtn: Button
-border_pane: BorderPane
-currentScene: String
-editScene: HBox
-familytree: TreeHandler
-form: FormHandler
-mytree: TreeView<Object>
-text: Text
-viewScene: HBox
-BuildScene(): Border
-EditSceneMaker(): HBox
-ViewSceneMaker(): HBox
-cancelButtonHandler()
-createFileDialog(String req): String
-displayScene(String scene)
+init()
-invokeAddMember(TreeItem<Object> chosenItem)
-LoadTreeFileButton()
-makeTopPane(): VBox
-memberSelectionHandler()
-promptAler(AlertType type, String topic, String message): int
-saveButtonHandler()
-saveTreeFileButton()
+start(Stahge primaryStage)
0..*
1
GUI
TreeHandler
-mytree: TreeView<Object>
+TreeHandler()
-addAsChild(TreeMap<String, String> memberInfo, TreeItem<Object> itemSelected, Children chosenItem)
-addAsRoot(TreeMap<String, String> memberInfo)
-addAsSpouse(TreeMap<String, String> memberInfo, TreeItem<Object> itemSelected, Children chosenItem)
+addItem(TreeMap<String, String> memberInfo, TreeItem<Object> itemSelected): int
+addNewMember(Person newItem, TreeItem<Object> parent, String memberType)
-assignParent(TreeItem<Object> newTreeItem, TreeItem<Object> parent, String memberType)
+clearTree()
+getChosenItem(): TreeItem<Object>
+getInfo(TreeItem<Object> item): TreeMap<String, String>
-getItem(String label, TreeItem<Object> newTreeItem): TreeItem<Object>
+getMemberType(TreeItem<Object> item): String
+getTree(): TreeView<Object>
+getTreeRoot(): Children
+isNull(): boolean
-loadRelatives(Person newItem, TreeItem<Object> newTreeItem)
+refreshTree(TreeMap<String, String> treeInfo, TreeItem<Object> treeItem)
1
Person
1..*
1
Form
1
FormHandler
-Components: VBox
+Genders: String
+InputFields: String
+MemberTypes: String
+description: String
+fName: String
-formPrompts: ArrayList<String>
+gender: String
-grid: GridPane
-inputField: TreeMap<String, TextField>
+lName: String
+mName: String
+memberType: String
-radioButtonGender: HBox
-radioButtonMember: HBox
+streetName: String
+streetNumber: String
+suburb: String
+zip: String
+FormHandler()
+buildFormItems(): VBox
+createForm(): GridPane
+disableForm(boolean toDisable)
+disableradio(HBox radioButtons, boolean toDisable)
+getErrorPrompts(): ArrayList<String>
+getFormData(TreeMap<String, String>)
+getRadioSelection(HBox radioButton): String
+getRoot(): VBox
+hideForm()
+isNotEmpty(): boolean
+isFieldEmpty(String field): boolean
+makeRadioButtons(String[] radioButtons): HBox
+resetForm()
+setFormData(TreeMap<String, String> formData)
+setRadioSelected(HBox radios, String match)
+showEditableForm()
+showEmptyForm()
+showForm()
+showViewableForm()
Person Info
1
1
Person
+description: String
+fName: String
+fields: String
+gender: String
+lName: String
+mName: String
+memberType: String
+streetName: String
+suburb: String
-thePerson: TreeMap<String,String>
-thespouse: Person
+zip: String
+Person()
+getInfo(): TreeMap<String,String>
+getMemberType(): String
+getSpouse(): Person
+hasSpouse(): boolean
+setInfo(TreeMap<String, String> info)
+setMemberType(String theType)
+setSpouse(Person newmember)
+toString(): String
Parent
1
0..*
1
Spouse
1
Children
-childrenList: ArrayList<Children>
+Children()
+addChild(Children Member)
+getPersonRelatives(): ArrayList<Person>
Spouse
+Spouse()

Design Description

The program code is object oriented and modular which separates the GUI from the backend logic. Separate classes like *TreeHandler.java*, *Person.java*, *Spouse.java* and *Child.java* were developed to achieve the solution. Following this approach makes the solution modular, reusable, easy to maintain and test. Separating the GUI from backend logic makes it easier to keep improving the frontend without disturbing the logic, and also reaffirms the Single Responsibility Principle.

FamilyTreeApp.java
This class creates the graphical user interface using JavaFX. It creates the main stage and handles navigation between different scenes with the use of buttons and event handlers. JavaFX is a rich Java library for creation of desktop applications. It offers a set of graphics and media packages that enables developers to design, create, test, debug, and deploy rich client applications that operate consistently across diverse platforms.

This class has been designed to just handle the GUI side of things, and does not involve anything related to the business logic.

FormHandler.java
One of the design choices faced during the early development stages of this application was to include the form components also into the *FamilyTreeApp* class, this would have made sure that all GUI related functionalities were in a single class. But, later on this plan was abandoned because it was felt that this application could definitely be further developed, and for that, the form could be changed according to the needs. The form could be expanded by including phone numbers, nationality, age, or even education details. So, necessary to keep the form components separate and not over burden the *FamilyTreeApp* class.

This class now acts as a helper class and can be assumed to be an extension for the *FamilyTreeApp* class. The main duty of this class is to build the different components of the form that is needed to receive details of a new member. It builds items like text fields, labels, radio buttons etc. This class is invoked by the *FamilyTreeApp* class to perform different operations with the form like checking for empty fields, refreshing the form, or hiding the form when required.

TreeHandler.java
This class is responsible for handling all the tree related operations like adding a new member. This class makes use of *TreeView* which provides control for displaying data in hierarchical tree like structure. Each family member item in the *TreeView* is an instance of the *TreeItem* class. The *TreeMap* data structure has been used to store the family data, it provides an efficient means of storing key-value pairs in a sorted manner.

Another most widely used type of map is called the HashMap which offers a performance of *O(1)* for most operations like *add()* and *remove()*. HashMap is significantly faster than a

TreeMap which offers a performance of *O(log(n))*. But, TreeMap offers better memory saving when compared to HashMap.

Person.java

This class has setters and getters to store information of a family member in a TreeMap. The TreeMap is used by the *FormHandler* class to set text into the form when required. The getters and setters were extremely useful for testing purpose during the program development. This class will also be the super class to two other classes, *the Spouse.java* and the *Children.java.*

Also, this class has been made *Serializable* so that objects can be represented as a sequence of bytes. Serialized objects can be written into a file, and can be read from the file after making them deserialized. The Java libraries *ObjectInputStream* and *ObjectOutputStream* were used to serialize and deserialize objects when reading and writing from a file.

Spouse.java

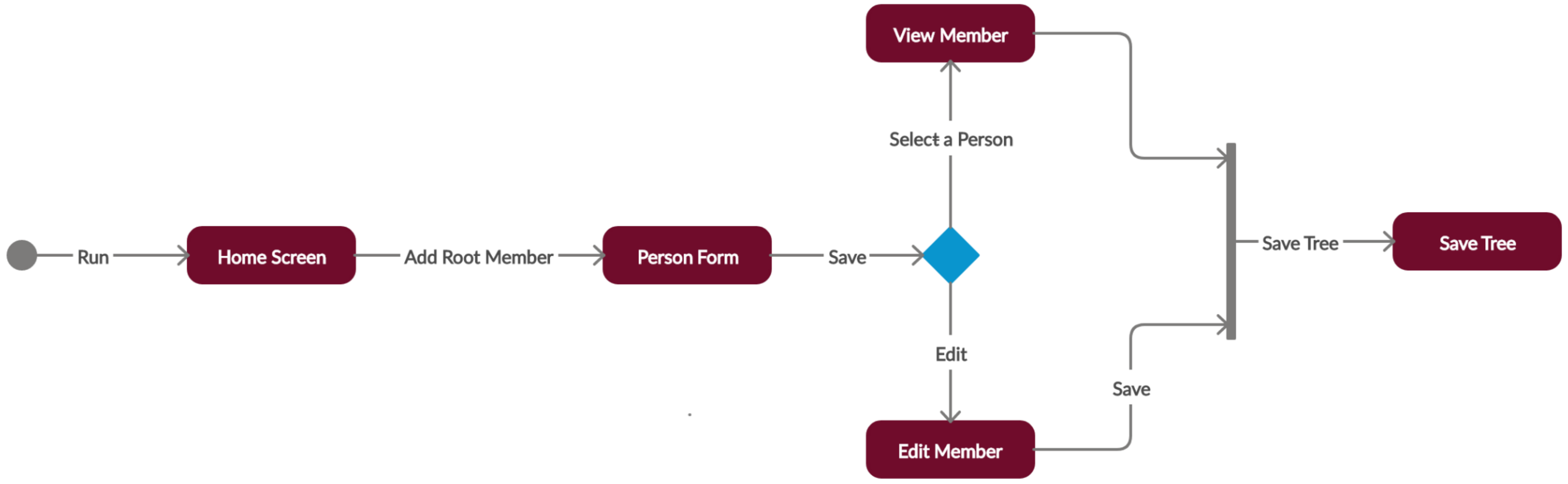
This class is for spouse member type, derived from the *Person* class as it has similar properties. A class in Java is derived using the *extends* keyword, this way the *Spouse* class inherits all the properties from its super class which is *Person* class. All properties like first name, maiden name and description will also be inherited.

Children.java

This class is for child member type, derived from the *Person* class as it has similar properties. Much like *Spouse.java,* this class is also derived from *Person.java* which becomes its super class. And will inherit all the properties from the *Person* class.

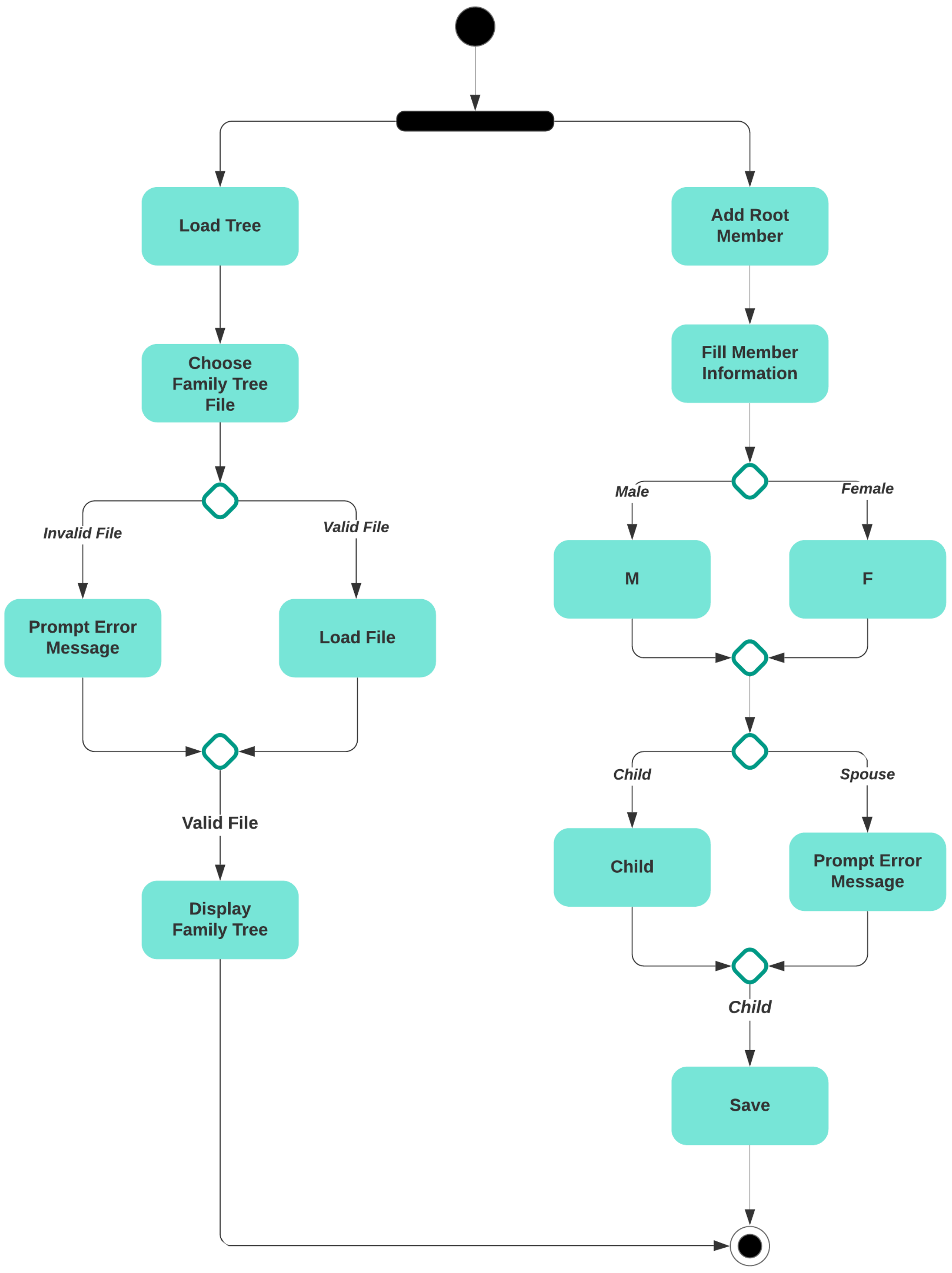
State Transition Diagram

This is a State Transition Diagram illustrating transition of states from creating a member till saving the family tree

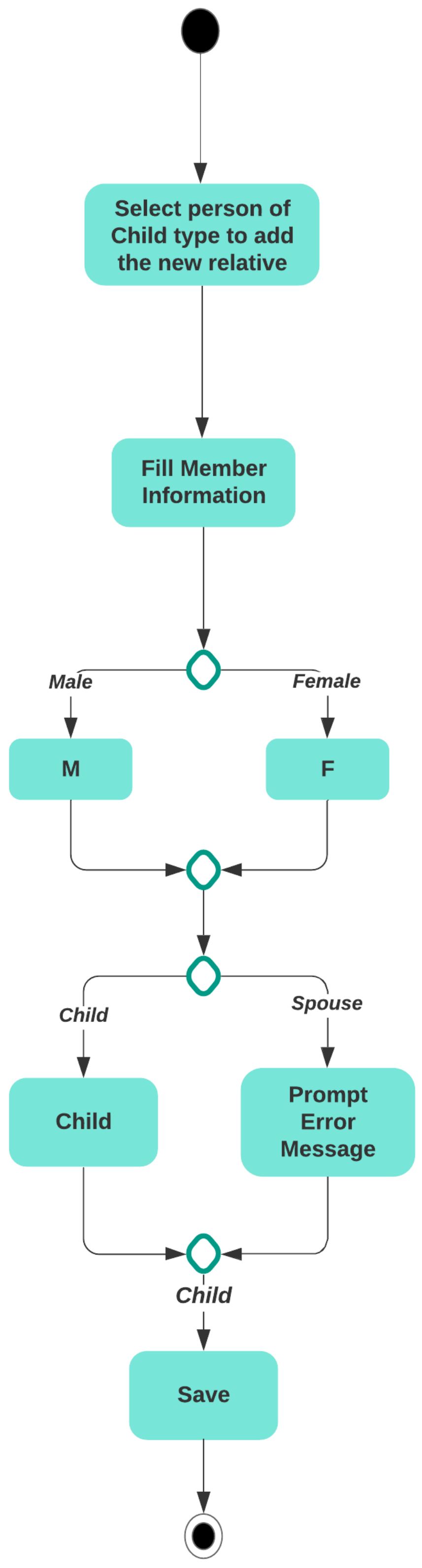


Activity Diagram

1) Loading a Tree/Adding Root Member



2) Adding a Relative



Testing

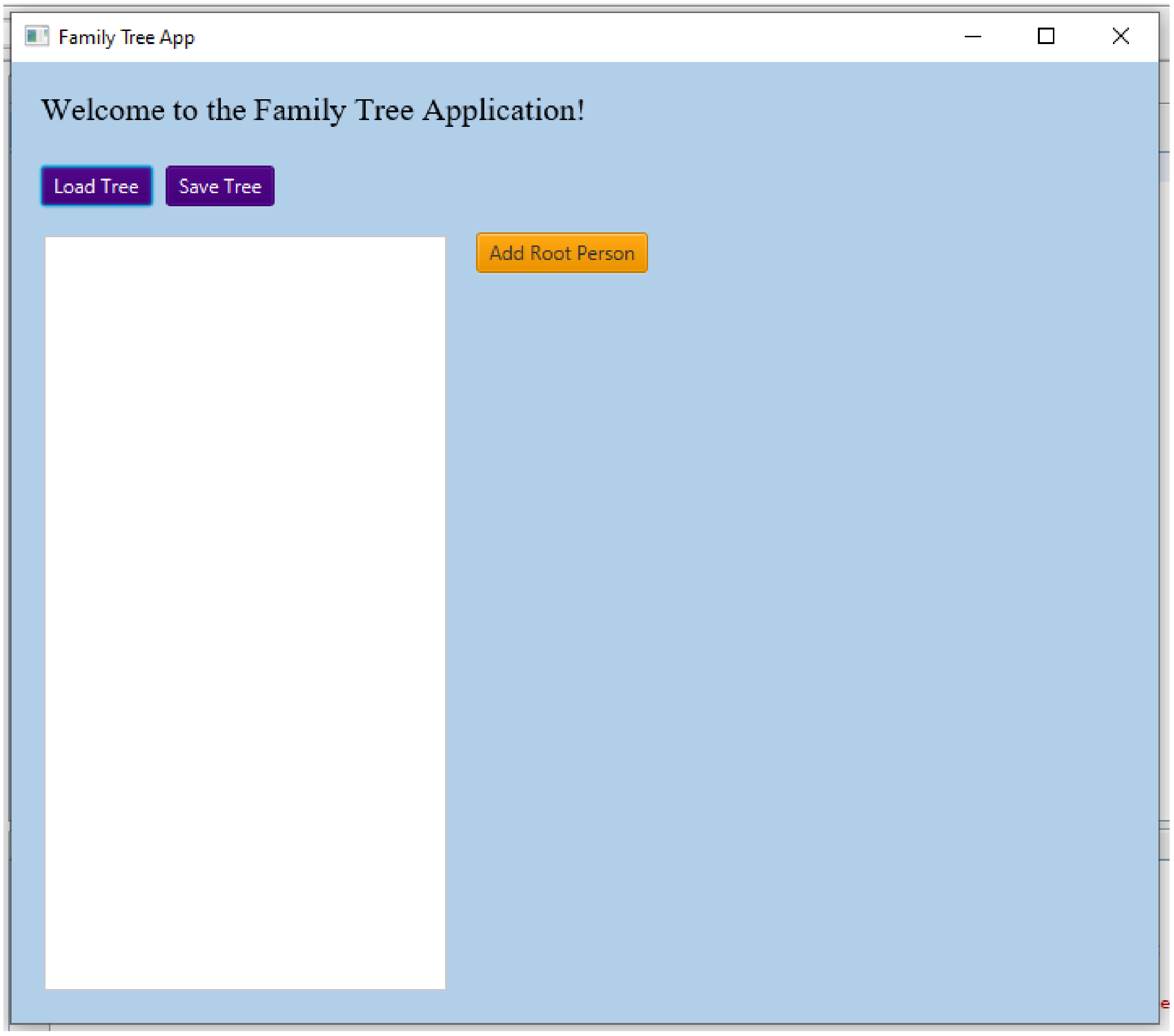
Test Strategy

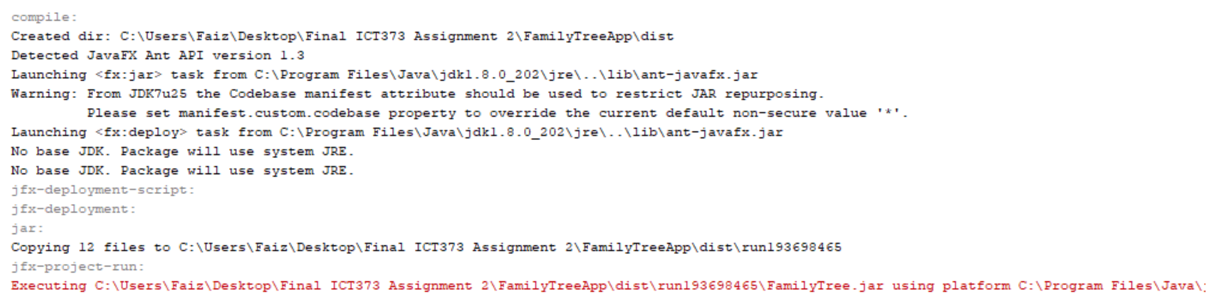
Test No.	**Test Description**	**Expected Result**	**Actual Result**	**Pass/ Fail**
1	Program complies on pressing Run and application starts	Application should compile and runs	Application compiles and runs successfully	Pass
2	GUI works and scenes are logically organized	The home screen should be well organized	The home screen is well organized	Pass
3	Pressing button to add root person	Person details form being displayed, and form elements organized properly	The form is displayed all elements are properly organized	Pass
4	Can fill in person details form	Allows entering details	Allows entering details	Pass
5	Radio buttons working as per choice	Radio buttons should be working fine	Radio are working fine	Pass
6	Saving the form	The form should save	The form gets saved	Pass
7	Viewing person details	Viewing details should work	The form is viewable after saving	Pass
8	Editing person details	Should be editable	The form is editable	Pass
9	Adding a spouse	Should allow adding spouse	Spouse can be added	Pass
10	Trying to add a second spouse	Should stop the user from adding 2nd spouse	A message is displayed and only 1 spouse can be added	Pass
11	Adding a child	Should allow adding children	Children can be added	Pass
12	Adding more children	Should allow multiple	Multiple children can be added	Pass
13	Adding relatives to the new child	More relatives can be added	Allows adding more descendants	Pass
14	Saving the family tree	Tree can be saved	The family tree can be saved in a .txt file	Pass
15	Loading the saved tree	Present tree should be overwritten	The present tree can be overwritten	Pass
16	Creating a new tree	A new tree can be started	The application needs to be restarted	Fail
17	Aesthetic look	Good aesthetic look	Has neat aesthetic look	Pass
18	Evidence of using FXML	FXML should ideally be used	FXML was not used, CSS was used instead	Fail

19	Trying to load an incorrect file	A warning message should be presented	An error message is displayed	Pass
20	Tring to load file of different format than .TXT	An error message should be displayed	An error message is prompted	Pass
21	Input validation	Form input validation should ideally be there	No, form input validation done	Fail
22	Not entering all input fields while saving the form	A warning message should be displayed	A warning message is prompted	Pass
23	Ability to view grandparent/grandchildren information	Relations should be represented	Relationships are not represented	Fail
24	Full screen GUI experience	Full screen feature should work	Full screen feature does not work well	Fail
25	Scrolling experience	Scrolling should be allowed	No ability to scroll up/down or left/right	Fail
26	Use of Serializable for objects	Serializable should be used	Serializable for objects is used	Pass
27	Provides exceptions	Exceptions should be present	Yes, provides most required exceptions but can be expanded further	Pass

Sample Tests

1) Program complies and starts on pressing Run

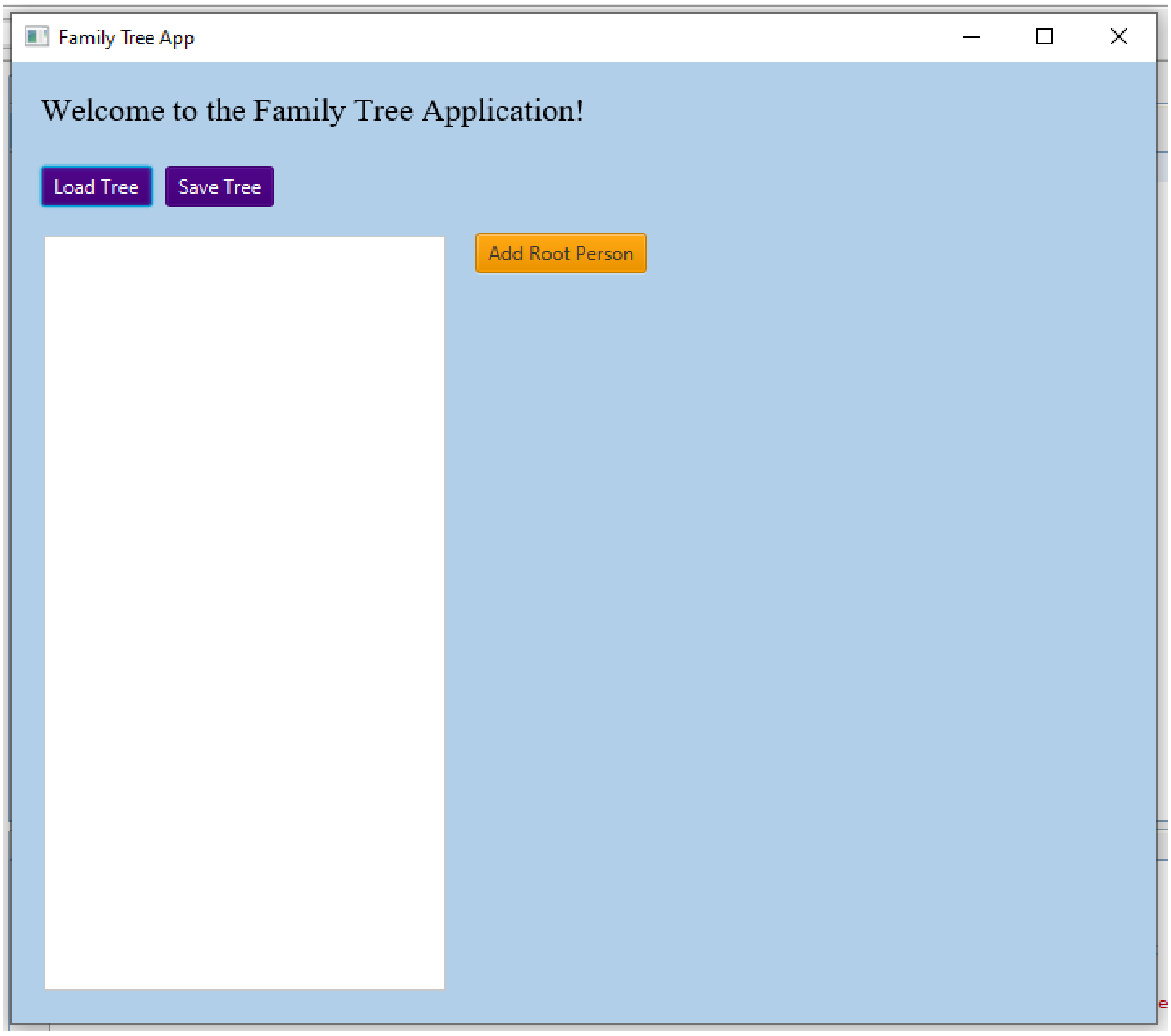




```
compile:
Created dir: C:\Users\Faiz\Desktop\Final ICT373 Assignment 2\FamilyTreeApp\dist
Detected JavaFX Ant API version 1.3
Launching <fx:jar> task from C:\Program Files\Java\jdk1.8.0_202\jre\..\lib\ant-javafx.jar
Warning: From JDK7u25 the Codebase manifest attribute should be used to restrict JAR repurposing.
         Please set manifest.custom.codebase property to override the current default non-secure value '*'.
Launching <fx:deploy> task from C:\Program Files\Java\jdk1.8.0_202\jre\..\lib\ant-javafx.jar
No base JDK. Package will use system JRE.
No base JDK. Package will use system JRE.
jfx-deployment-script:
jfx-deployment:
jar:
Copying 12 files to C:\Users\Faiz\Desktop\Final ICT373 Assignment 2\FamilyTreeApp\dist\run193698465
jfx-project-run:
Executing C:\Users\Faiz\Desktop\Final ICT373 Assignment 2\FamilyTreeApp\dist\run193698465\FamilyTree.jar using platform C:\Program Files\Java\
```

Although there are certain warnings given by the compiler, they do not hinder the application from working well.

2) GUI works and scenes are logically organized



Welcome to the Family Tree Application!

Scene Description:

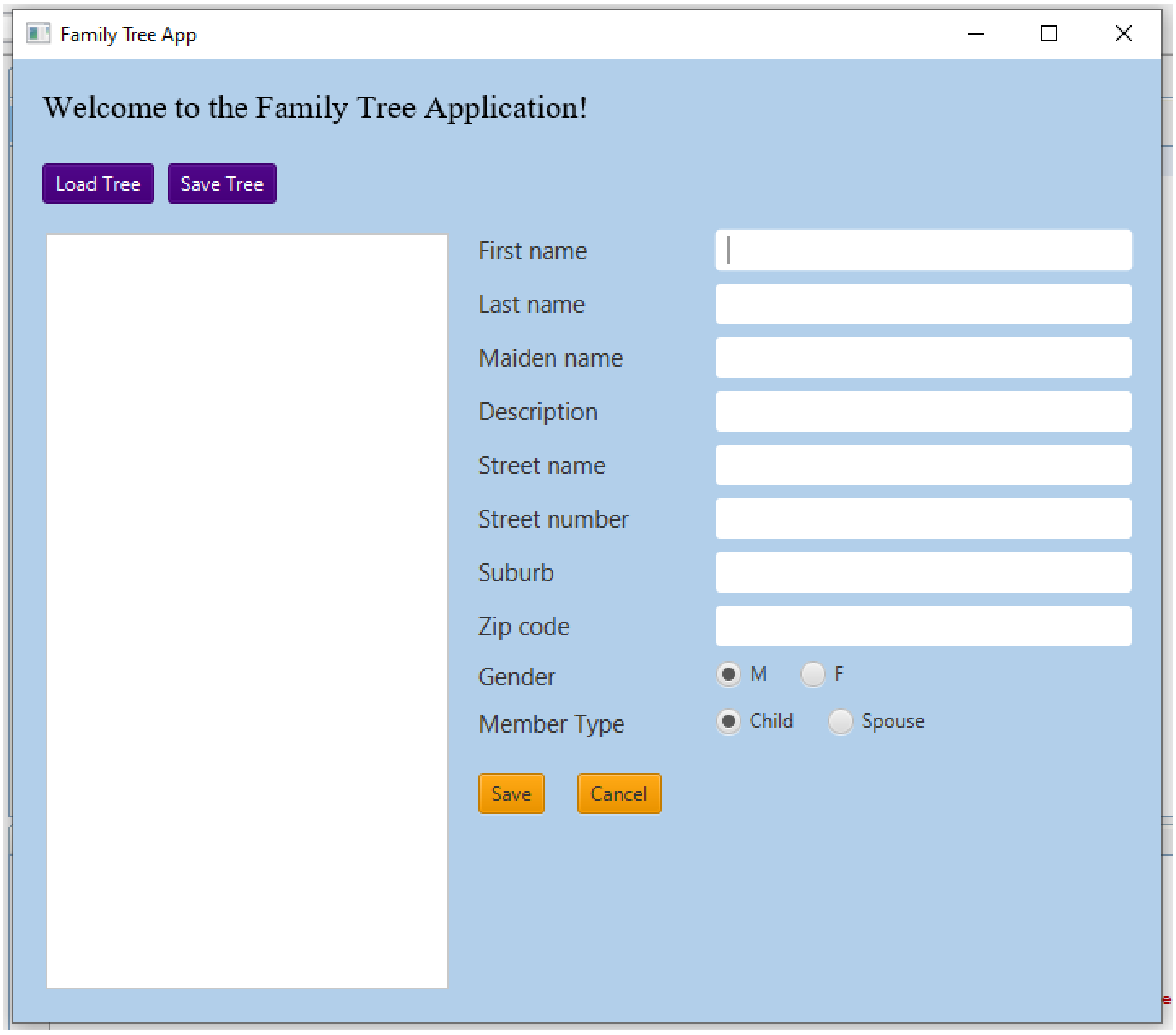
Load Tree Button - Allows loading an already created tree into the application

Add Root Member Button – Is the starting point for creating a new family tree

Save Tree Button – Allows saving a family tree after creating it

The white vertical panel on the left – This is where the family tree hierarchy is displayed

3) Pressing button to add root person


Family Tree App

Welcome to the Family Tree Application!

First name	
Last name	
Maiden name	
Description	
Street name	
Street number	
Suburb	
Zip code	
Gender	[✓ M] F
Member Type	[✓ Child] Spouse

Scene Description:

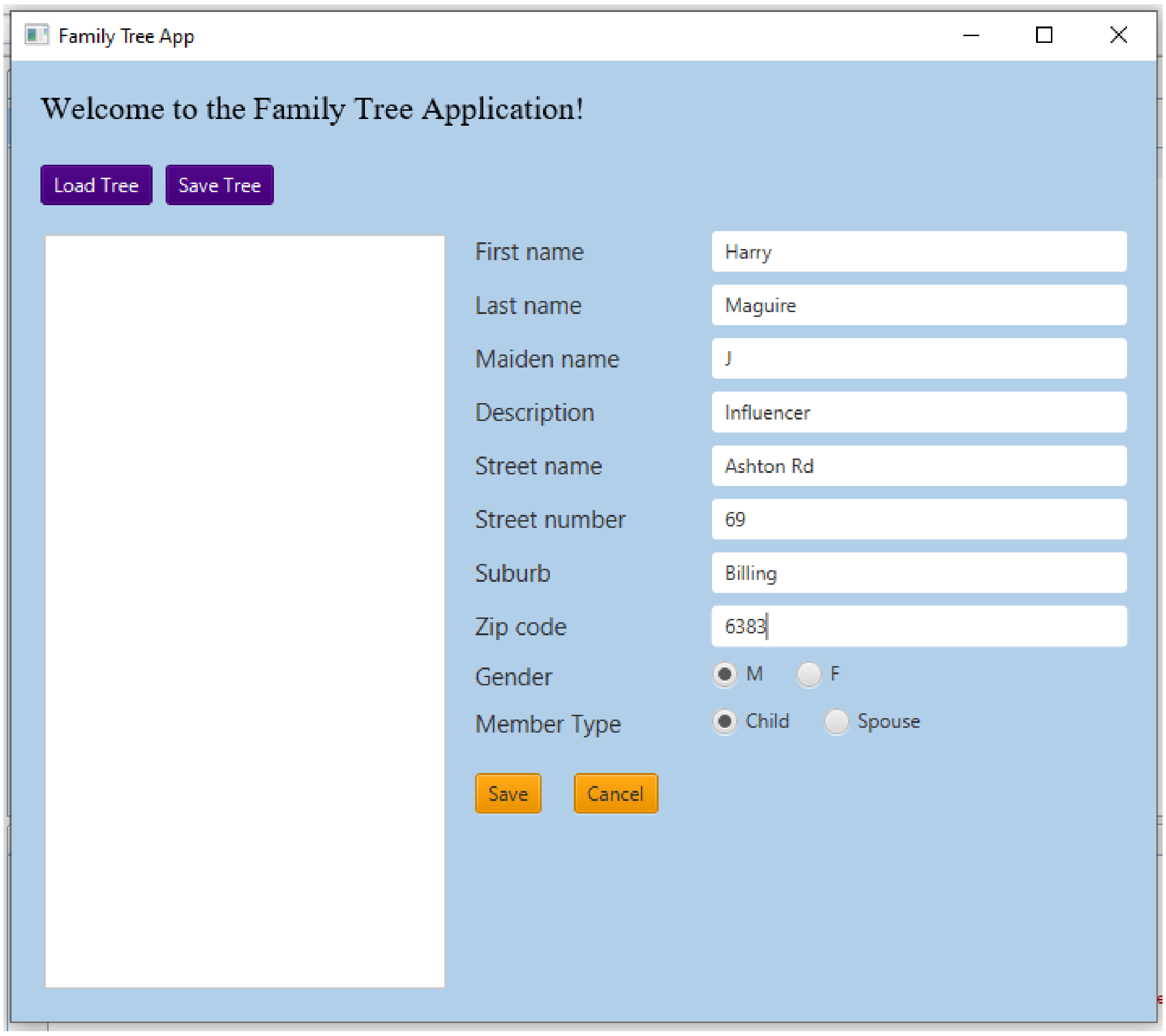
Text fields – To type necessary information about the person

Radio button – Allowing user to choose between male or female, and child or spouse relationship

Save Button – To save the person form, and the displaying the tree simultaneously in the left pane

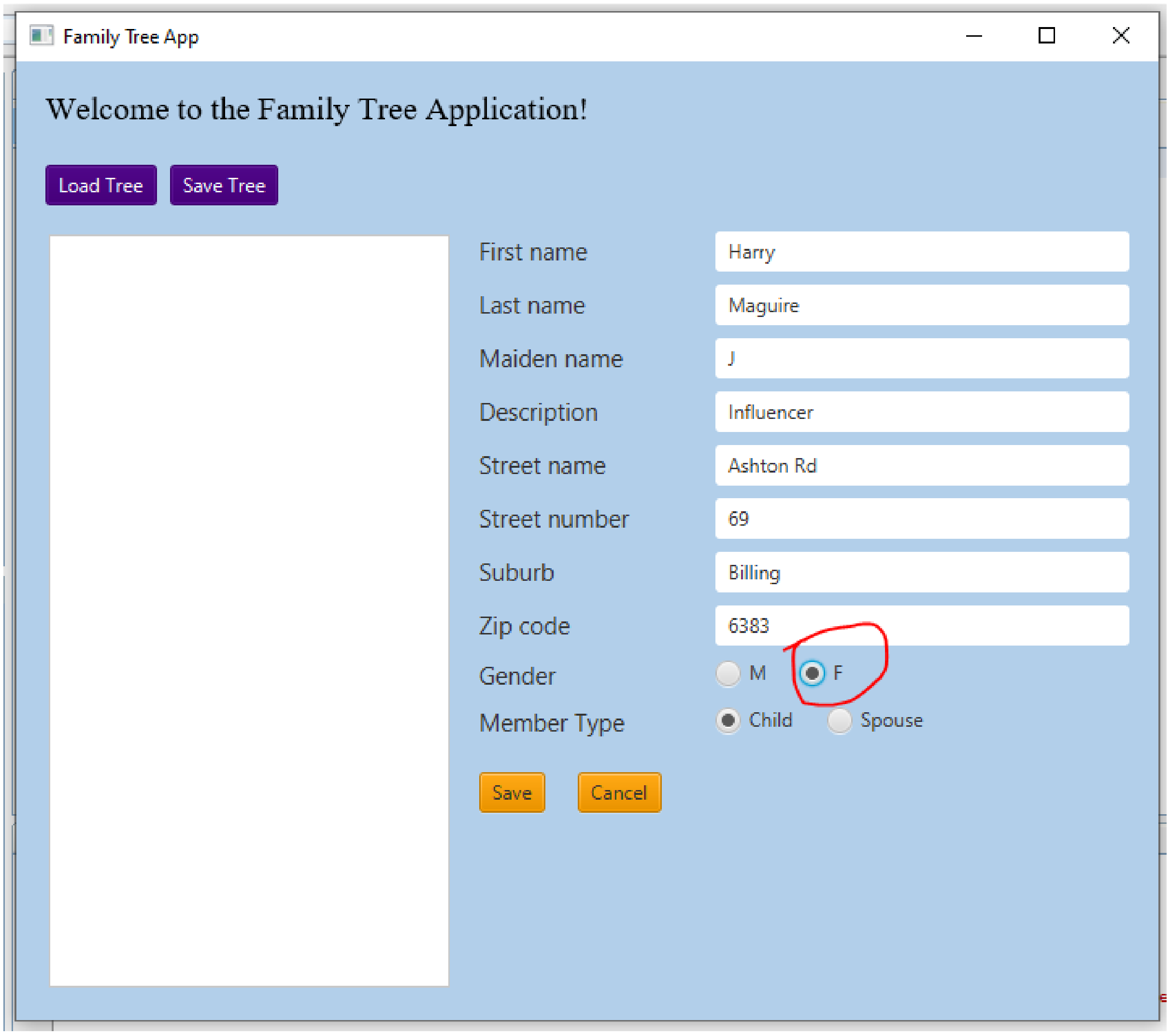
Cancel Button – To cancel the present person creation process and go back to the home screen

4) Can fill in person details form



First name	Harry
Last name	Maguire
Maiden name	J
Description	Influencer
Street name	Ashton Rd
Street number	69
Suburb	Billing
Zip code	6383
Gender	[✓ M] F
Member Type	[✓ Child] Spouse

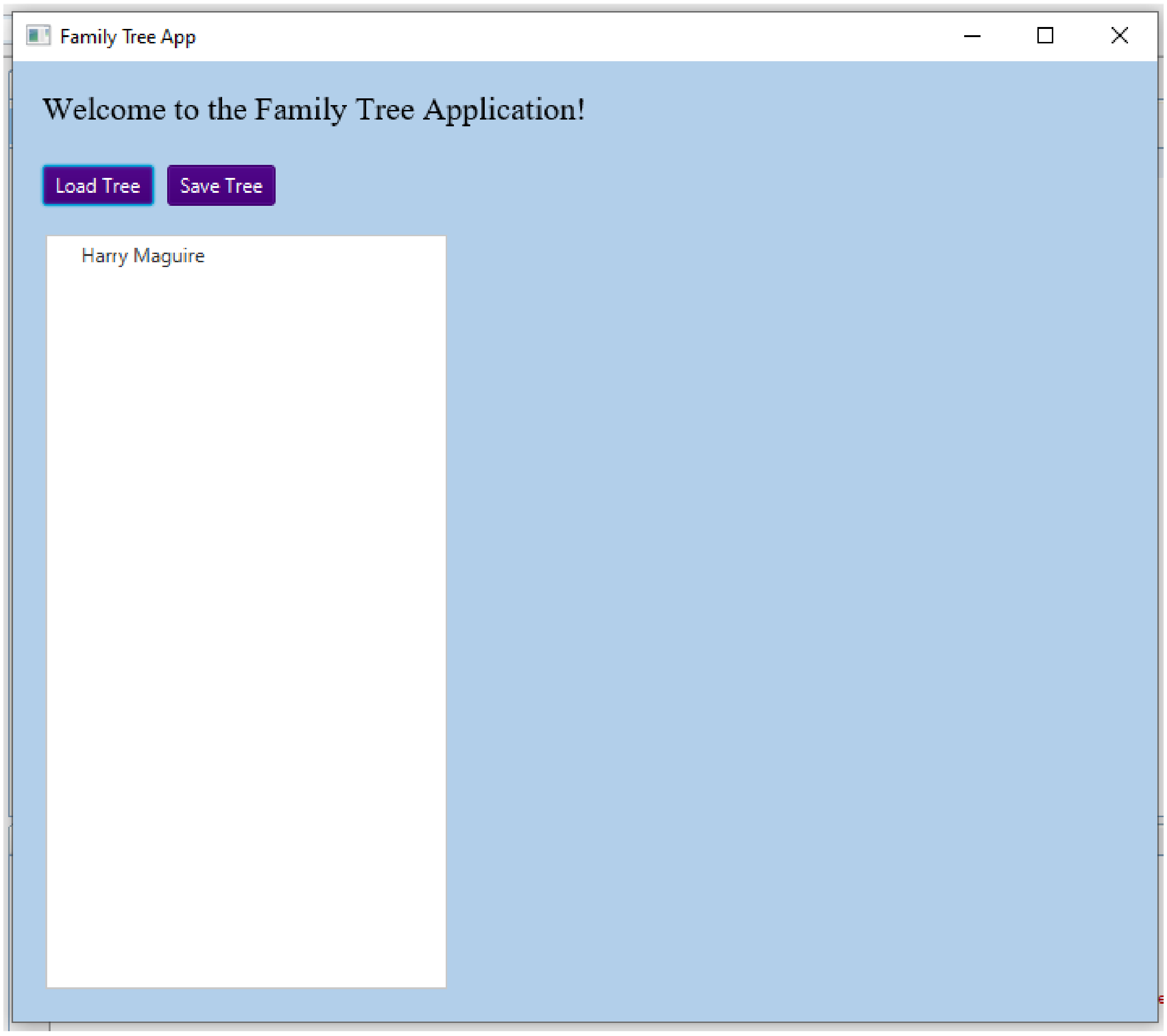
5) Radio button working as per choice


Family Tree App

Welcome to the Family Tree Application!

First name	Harry
Last name	Maguire
Maiden name	J
Description	Influencer
Street name	Ashton Rd
Street number	69
Suburb	Billing
Zip code	6383
Gender	M [✓ F]
Member Type	[✓ Child] Spouse

6) Saving the form

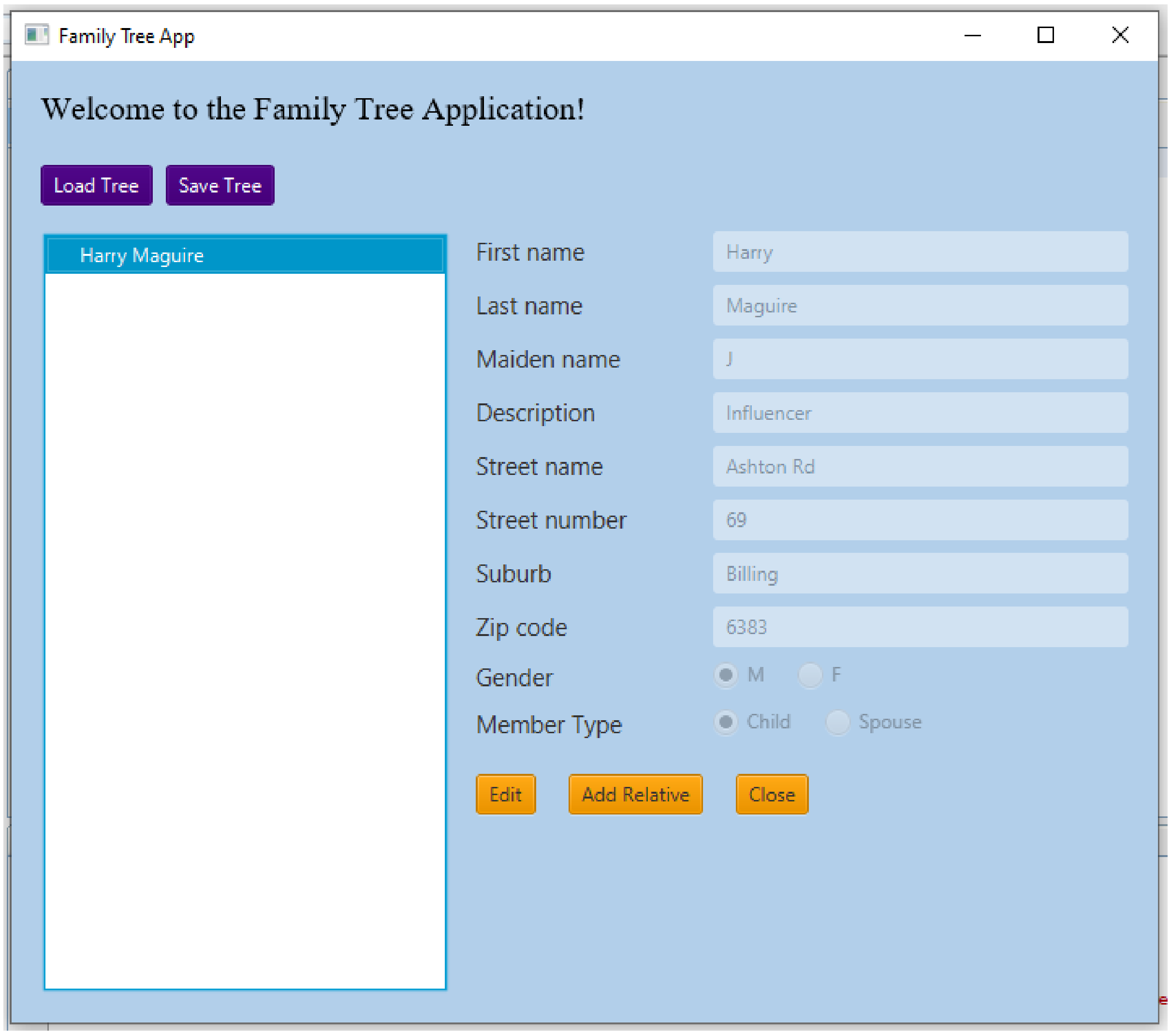

Family Tree App

Welcome to the Family Tree Application!

Harry Maguire

The family members are displayed in the white pane on the left as per the core requirements

7) Viewing person details



Welcome to the Family Tree Application!

Harry Maguire

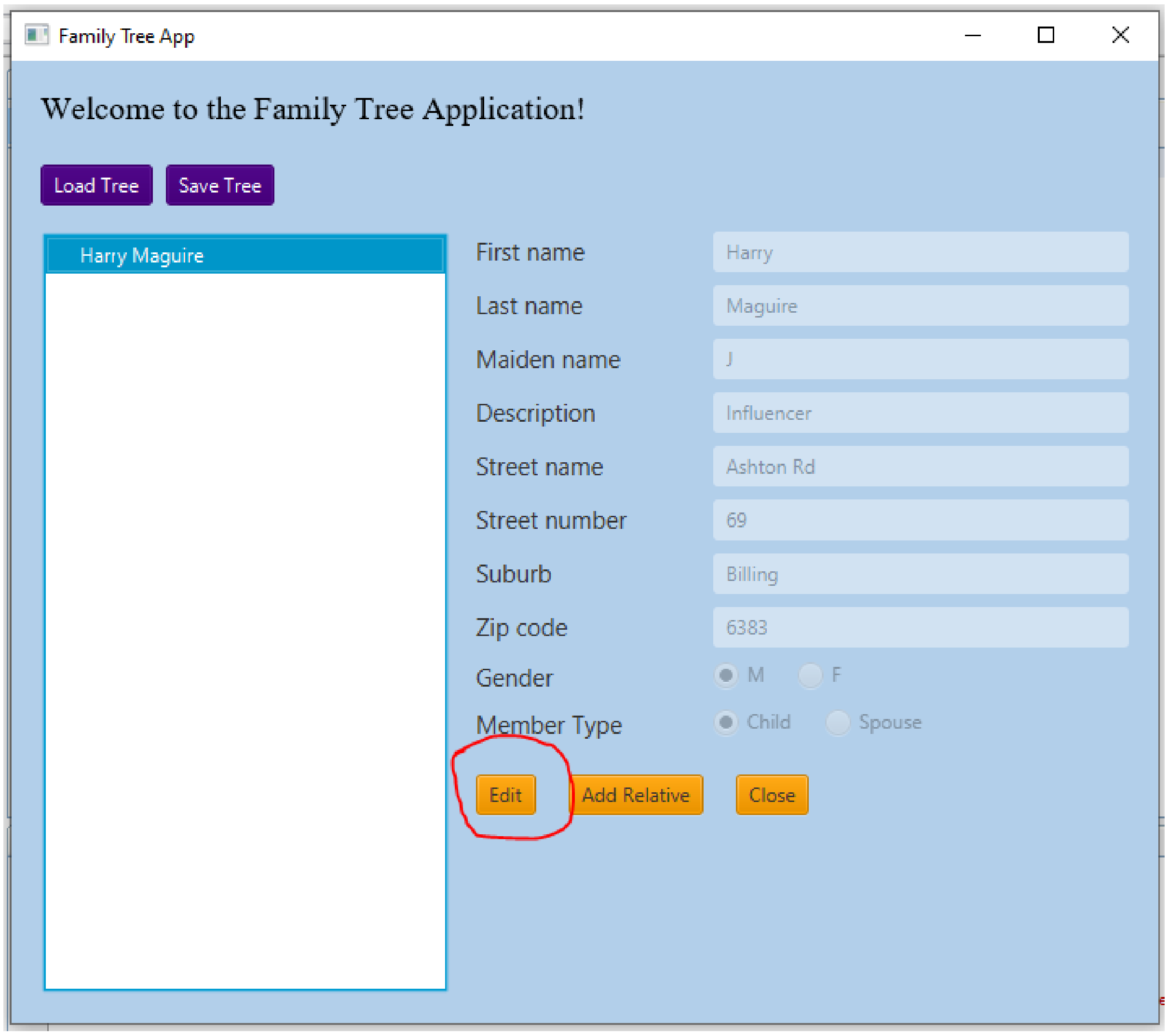
First name	Harry
Last name	Maguire
Maiden name	J
Description	Influencer
Street name	Ashton Rd
Street number	69
Suburb	Billing
Zip code	6383
Gender	[✓ M] F
Member Type	[✓ Child] Spouse

Scene Description:

Viewing Information - Member information can be viewed simply by clicking on a person.

Edit Button – The edit button allows changing person information, although, the child or spouse radio button is kept locked

8) Editing person details



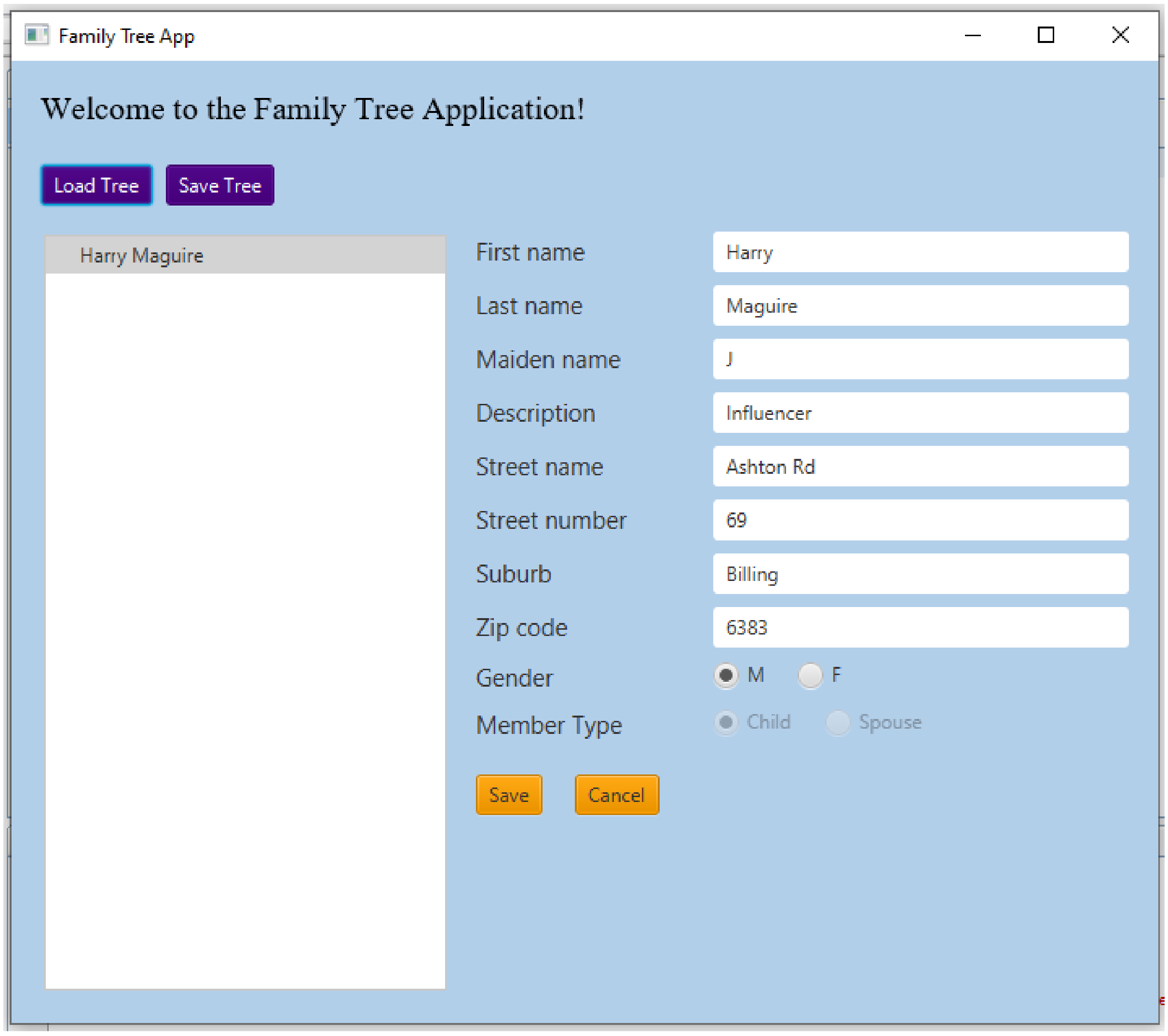
Family Tree App

Welcome to the Family Tree Application!

Harry Maguire

First name	Harry
Last name	Maguire
Maiden name	J
Description	Influencer
Street name	Ashton Rd
Street number	69
Suburb	Billing
Zip code	6383
Gender	[✓ M] F
Member Type	[✓ Child] Spouse

Description gets updated

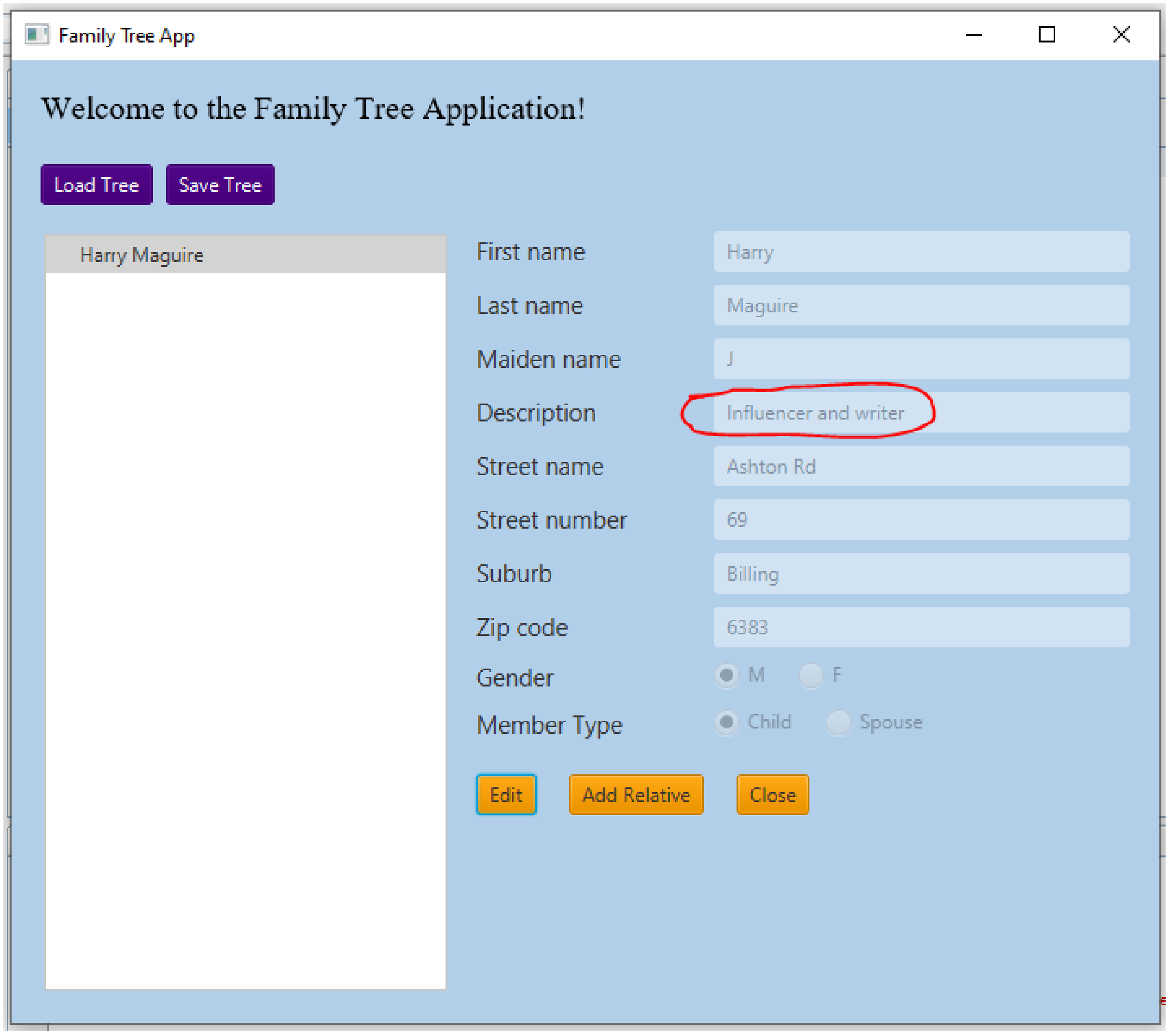

Family Tree App

Welcome to the Family Tree Application!

Harry Maguire

First name	Harry
Last name	Maguire
Maiden name	J
Description	Influencer
Street name	Ashton Rd
Street number	69
Suburb	Billing
Zip code	6383
Gender	[✓ M] F
Member Type	[✓ Child] Spouse

Viewing form after saving the changes



Family Tree App

Welcome to the Family Tree Application!

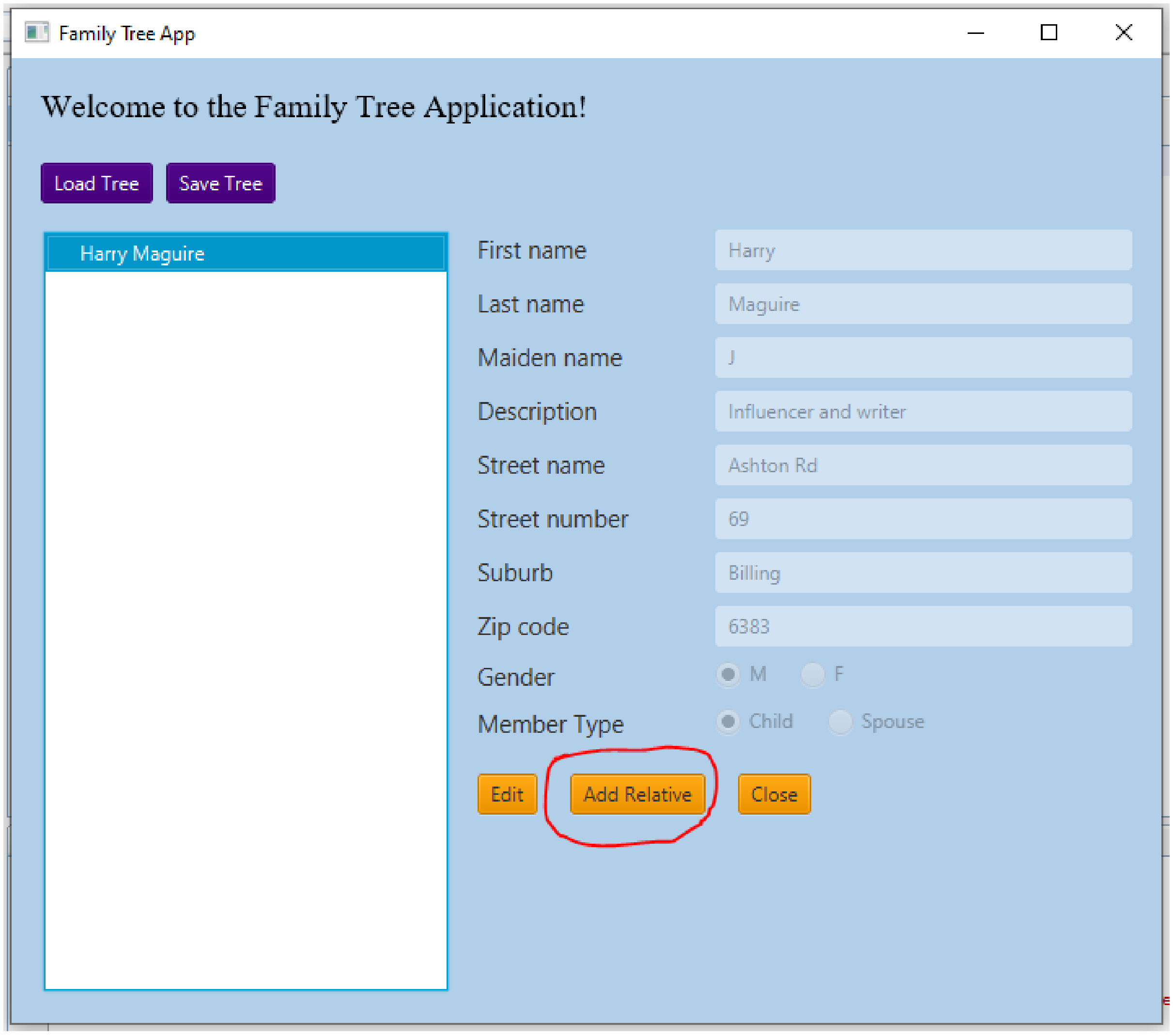
Load Tree | Save Tree

Harry Maguire

First name	Harry
Last name	Maguire
Maiden name	J
Description	Influencer and writer
Street name	Ashton Rd
Street number	69
Suburb	Billing
Zip code	6383
Gender	[✓ M] F
Member Type	[✓ Child] Spouse

Edit | Add Relative | Close

9) Adding a spouse

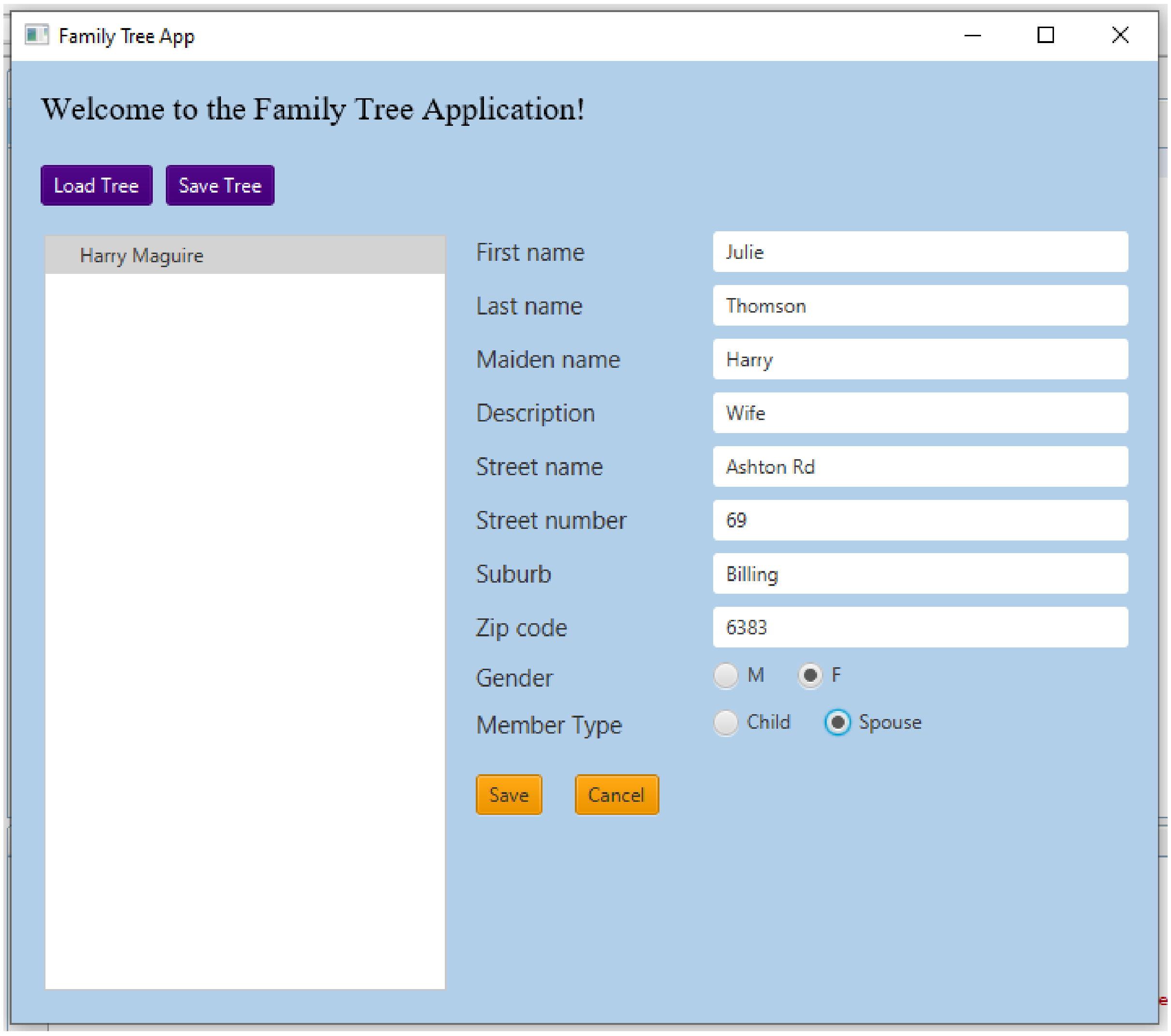


Welcome to the Family Tree Application!

Harry Maguire

First name	Harry
Last name	Maguire
Maiden name	J
Description	Influencer and writer
Street name	Ashton Rd
Street number	69
Suburb	Billing
Zip code	6383
Gender	[✓ M] F
Member Type	[✓ Child] Spouse

Filling the details and selecting member type as “Spouse” and pressing “Save” button

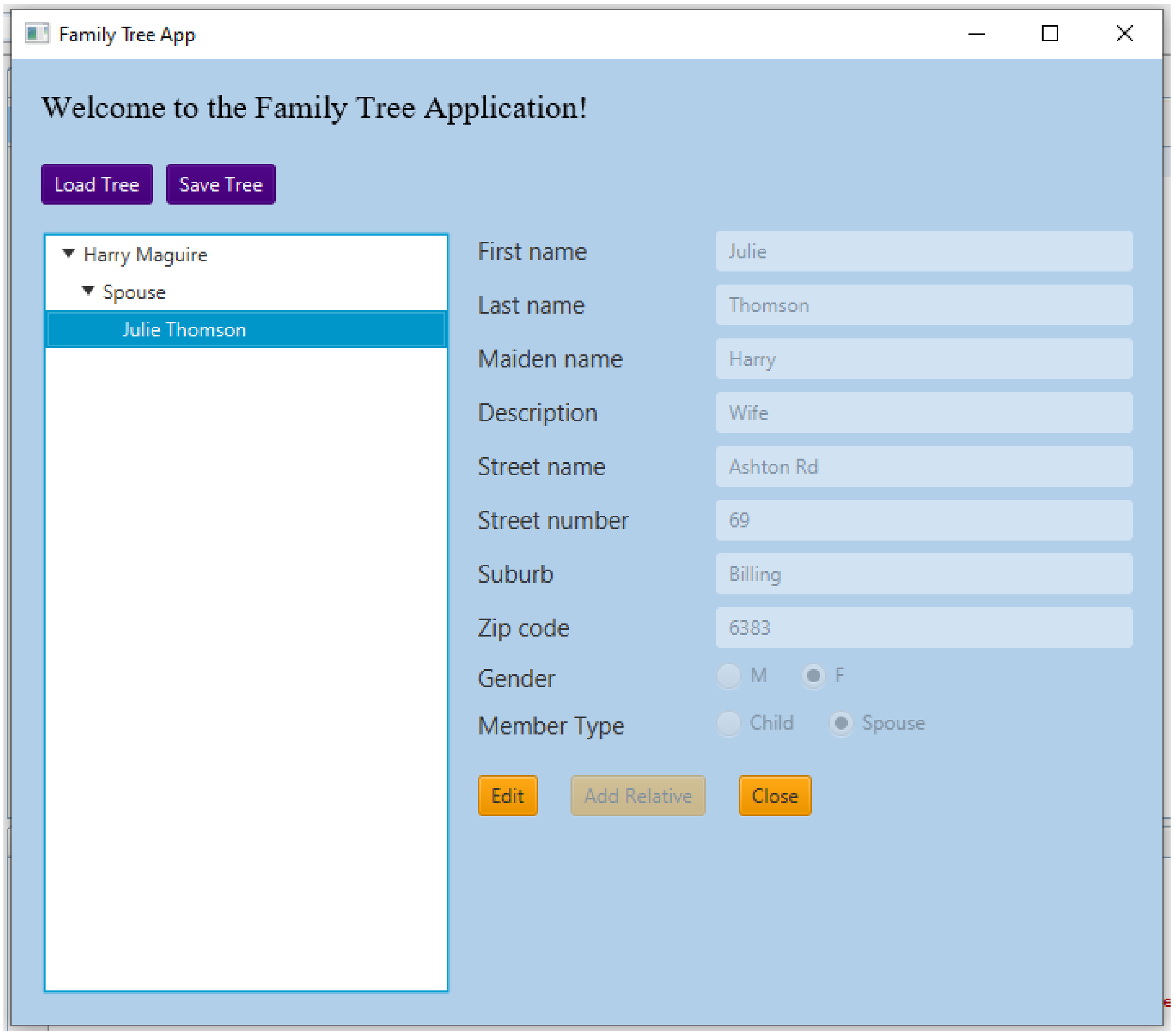


Welcome to the Family Tree Application!

Harry Maguire

First name	Julie
Last name	Thomson
Maiden name	Harry
Description	Wife
Street name	Ashton Rd
Street number	69
Suburb	Billing
Zip code	6383
Gender	M [✓ F]
Member Type	Child [✓ Spouse]

Viewing details



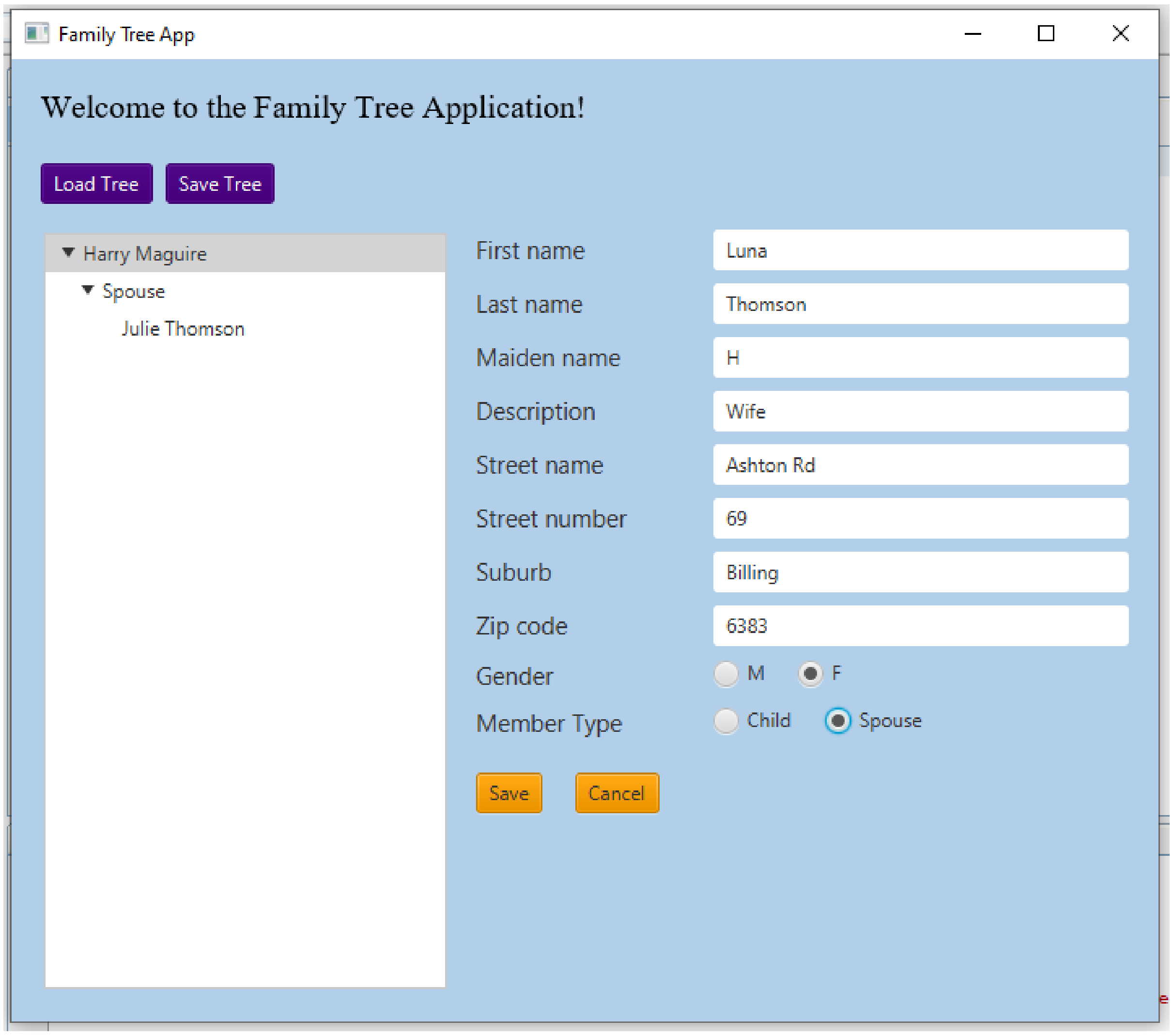
Family Tree App

Welcome to the Family Tree Application!

- Harry Maguire
  - Spouse
    - Julie Thomson

First name	Julie
Last name	Thomson
Maiden name	Harry
Description	Wife
Street name	Ashton Rd
Street number	69
Suburb	Billing
Zip code	6383
Gender	M [✓ F]
Member Type	Child [✓ Spouse]

10) Trying to add a second spouse



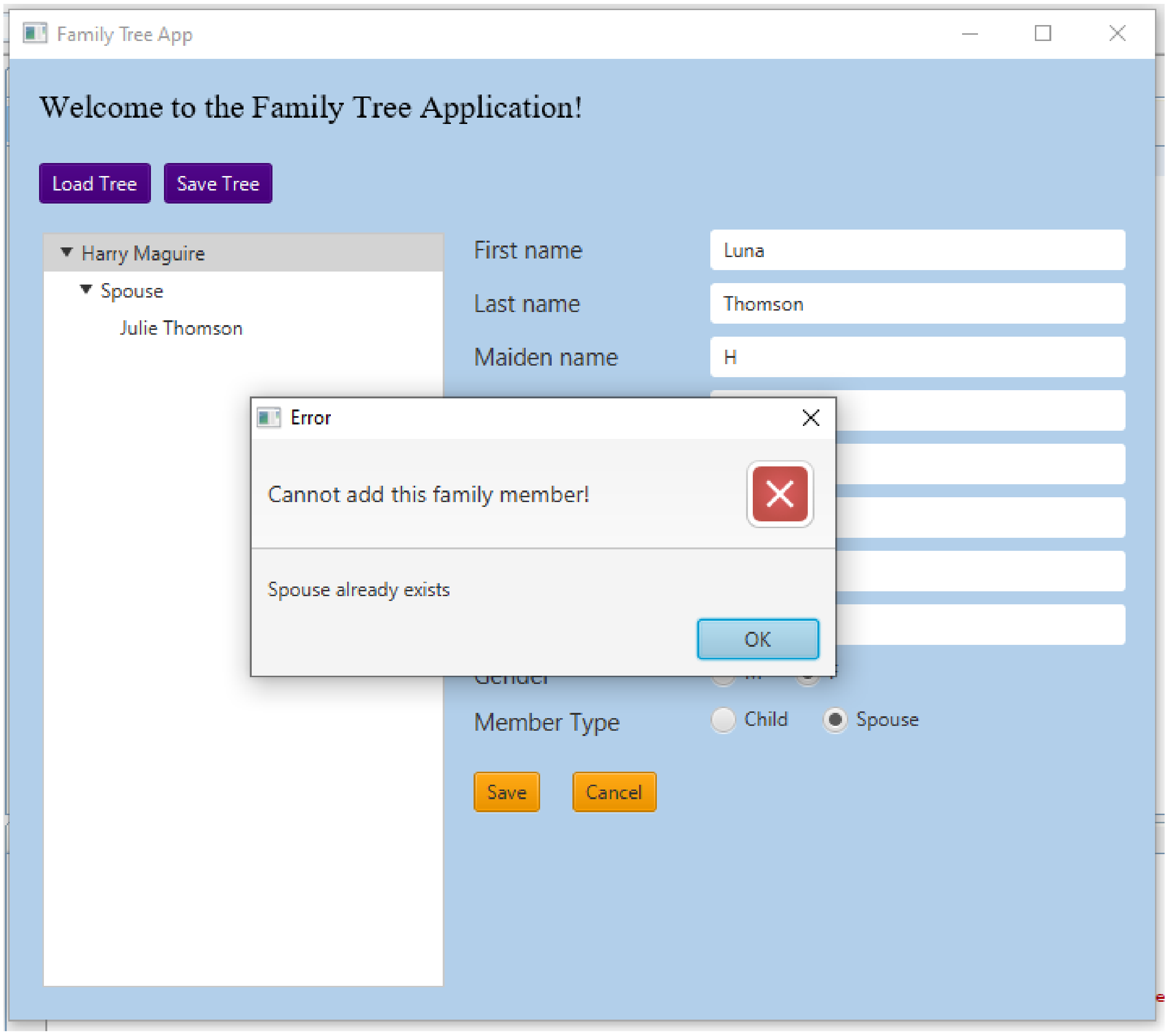
Family Tree App

Welcome to the Family Tree Application!

- Harry Maguire
  - Spouse
    - Julie Thomson

First name	Luna
Last name	Thomson
Maiden name	H
Description	Wife
Street name	Ashton Rd
Street number	69
Suburb	Billing
Zip code	6383
Gender	M [✓ F]
Member Type	Child [✓ Spouse]

Trying to save this form



Family Tree App

Welcome to the Family Tree Application!

- Harry Maguire
  - Spouse
    - Julie Thomson

First name	Luna
Last name	Thomson
Maiden name	H

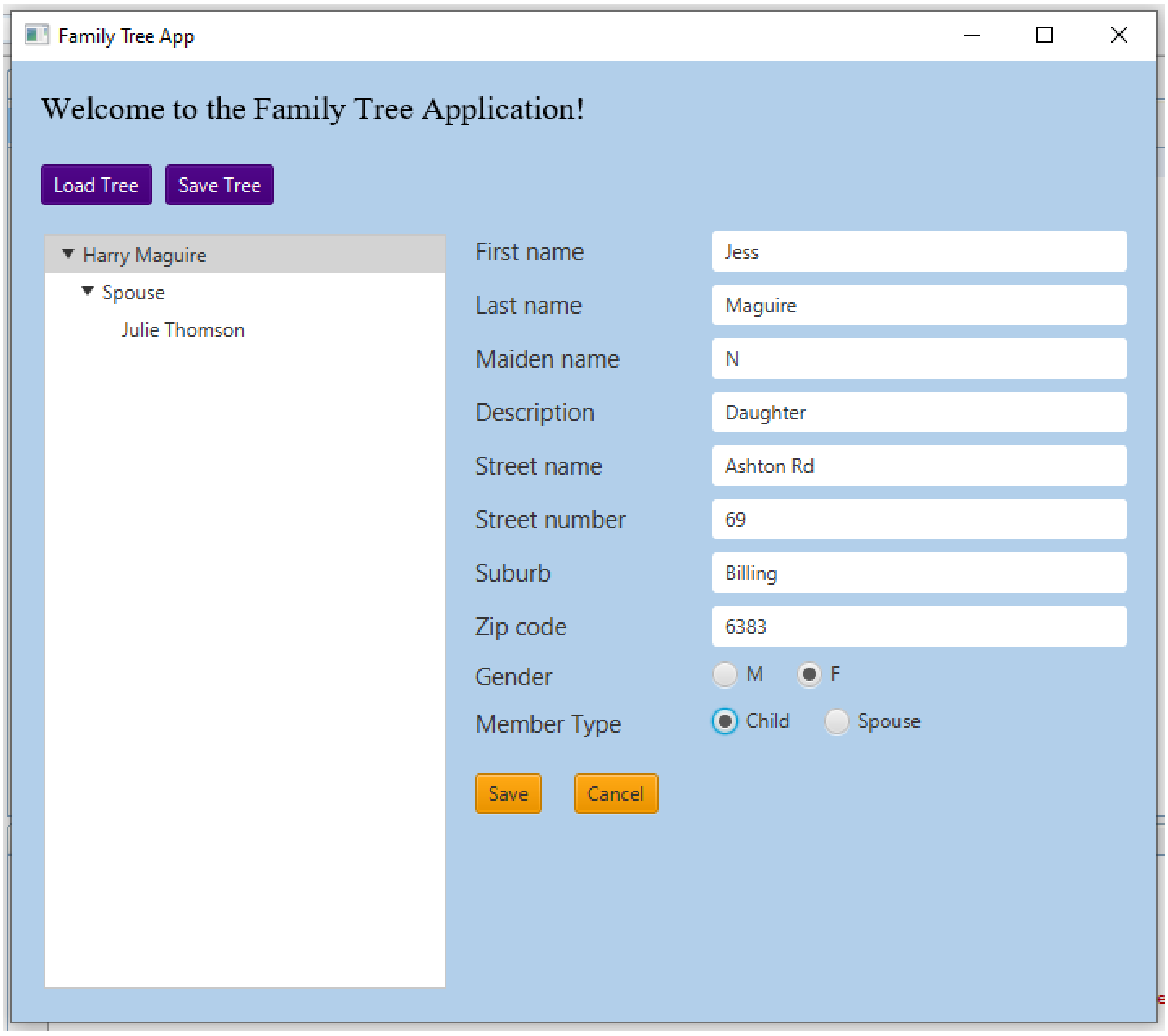
Member Type: Child [✓ Spouse]

Error

Cannot add this family member!

Spouse already exists

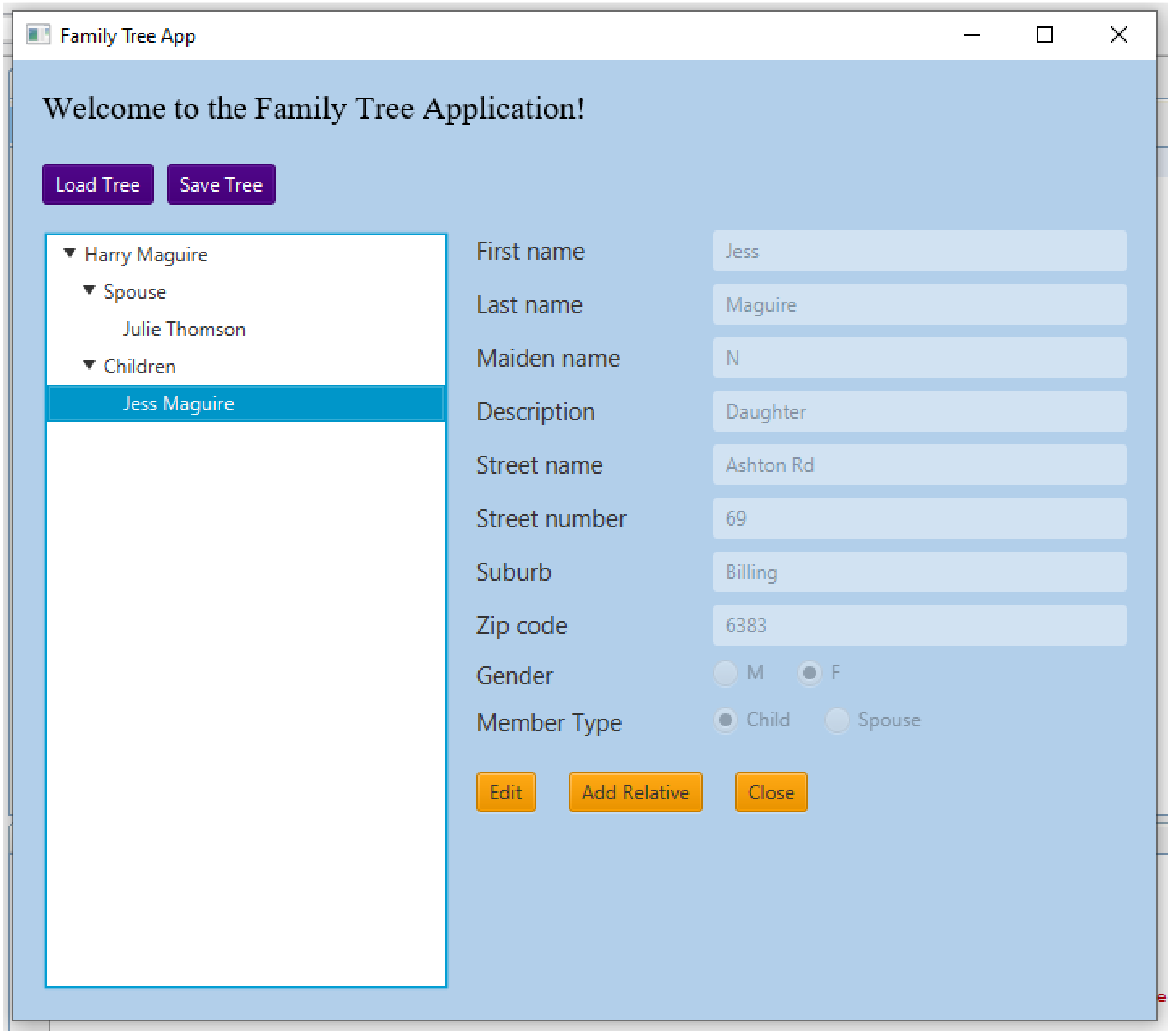
11) Adding a child



Welcome to the Family Tree Application!

First name	Jess
Last name	Maguire
Maiden name	N
Description	Daughter
Street name	Ashton Rd
Street number	69
Suburb	Billing
Zip code	6383
Gender	M [✓ F]
Member Type	[✓ Child] Spouse

Saving the form

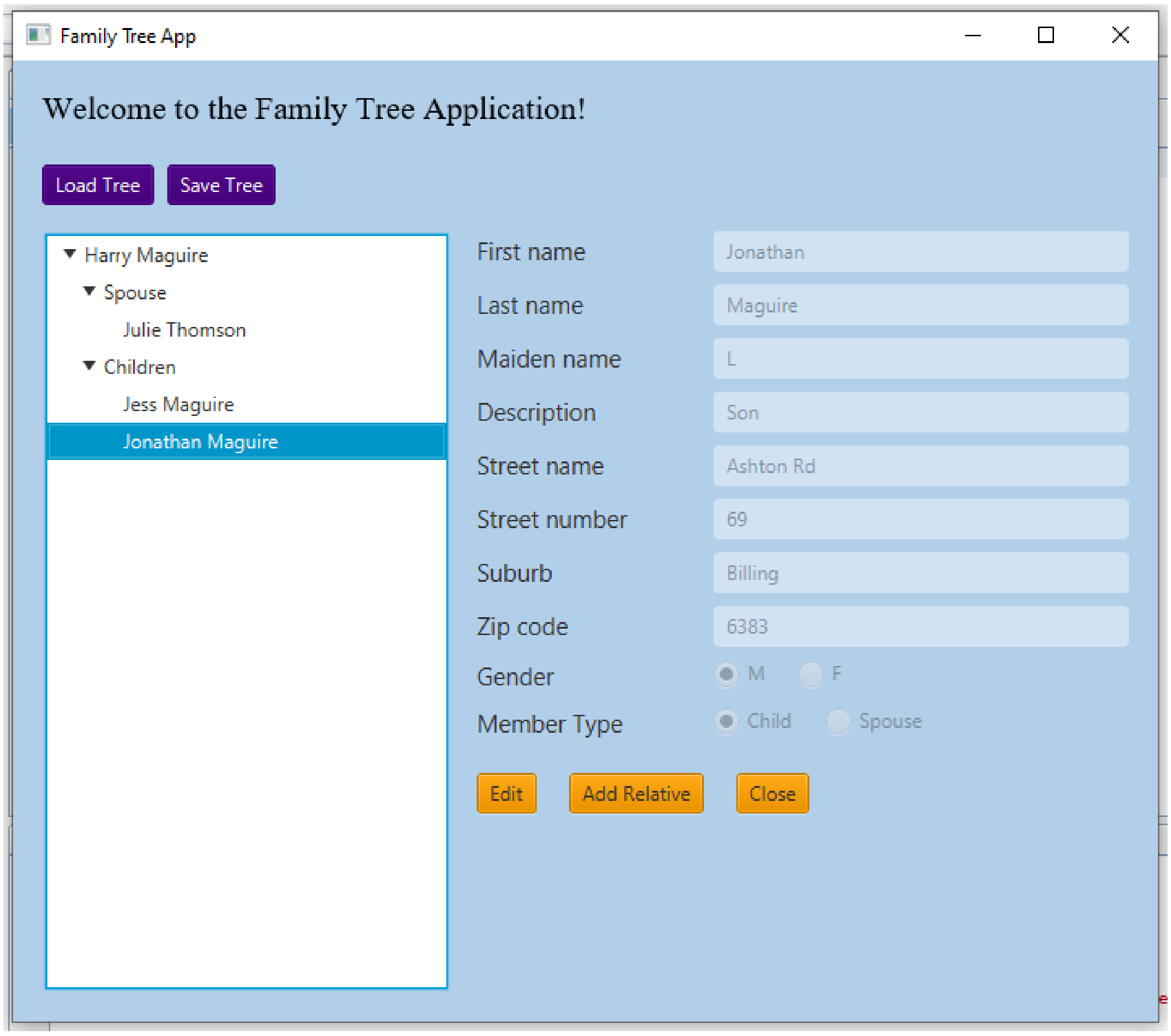


Welcome to the Family Tree Application!

- Harry Maguire
  - Spouse
    - Julie Thomson
  - Children
    - Jess Maguire

First name	Jess
Last name	Maguire
Maiden name	N
Description	Daughter
Street name	Ashton Rd
Street number	69
Suburb	Billing
Zip code	6383
Gender	M [✓ F]
Member Type	[✓ Child] Spouse

12) Adding more children

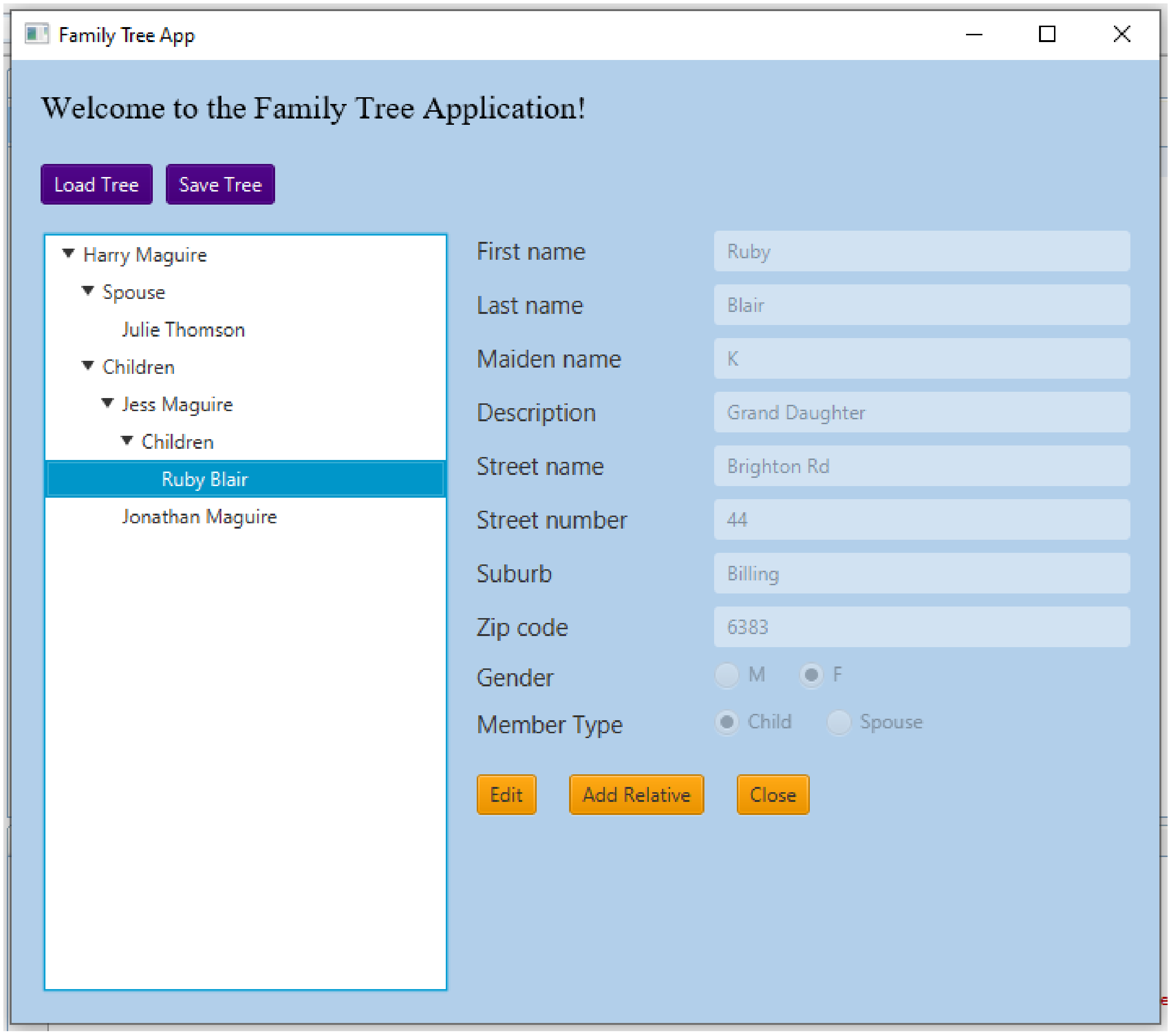

Family Tree App

Welcome to the Family Tree Application!

- Harry Maguire
  - Spouse
    - Julie Thomson
  - Children
    - Jess Maguire
    - Jonathan Maguire

First name	Jonathan
Last name	Maguire
Maiden name	L
Description	Son
Street name	Ashton Rd
Street number	69
Suburb	Billing
Zip code	6383
Gender	[✓ M] F
Member Type	[✓ Child] Spouse

13) Adding grandchildren

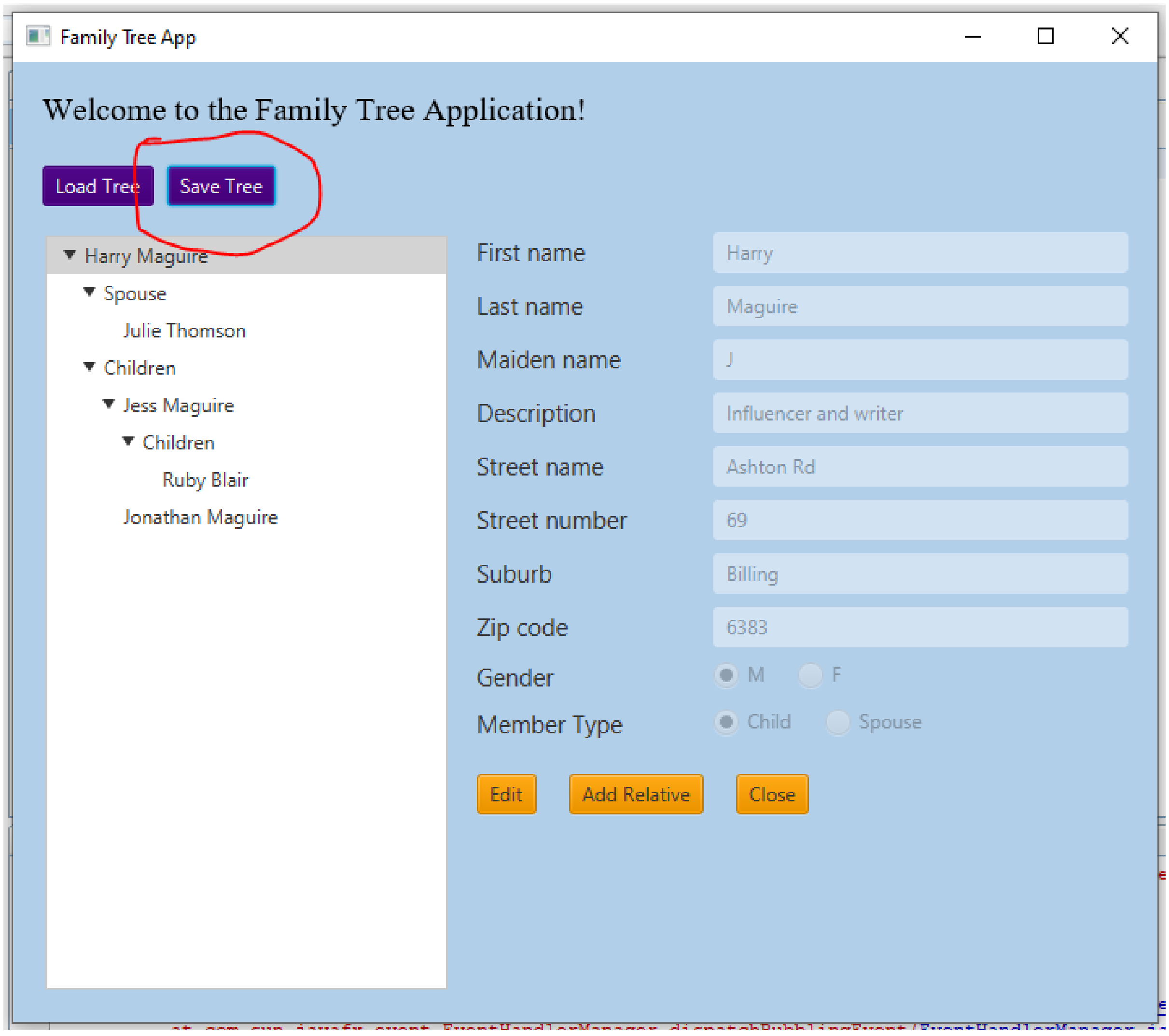


Welcome to the Family Tree Application!

- Harry Maguire
  - Spouse
    - Julie Thomson
  - Children
    - Jess Maguire
      - Children
        - Ruby Blair
    - Jonathan Maguire

First name	Ruby
Last name	Blair
Maiden name	K
Description	Grand Daughter
Street name	Brighton Rd
Street number	44
Suburb	Billing
Zip code	6383
Gender	M [✓ F]
Member Type	[✓ Child] Spouse

14) Saving the family tree

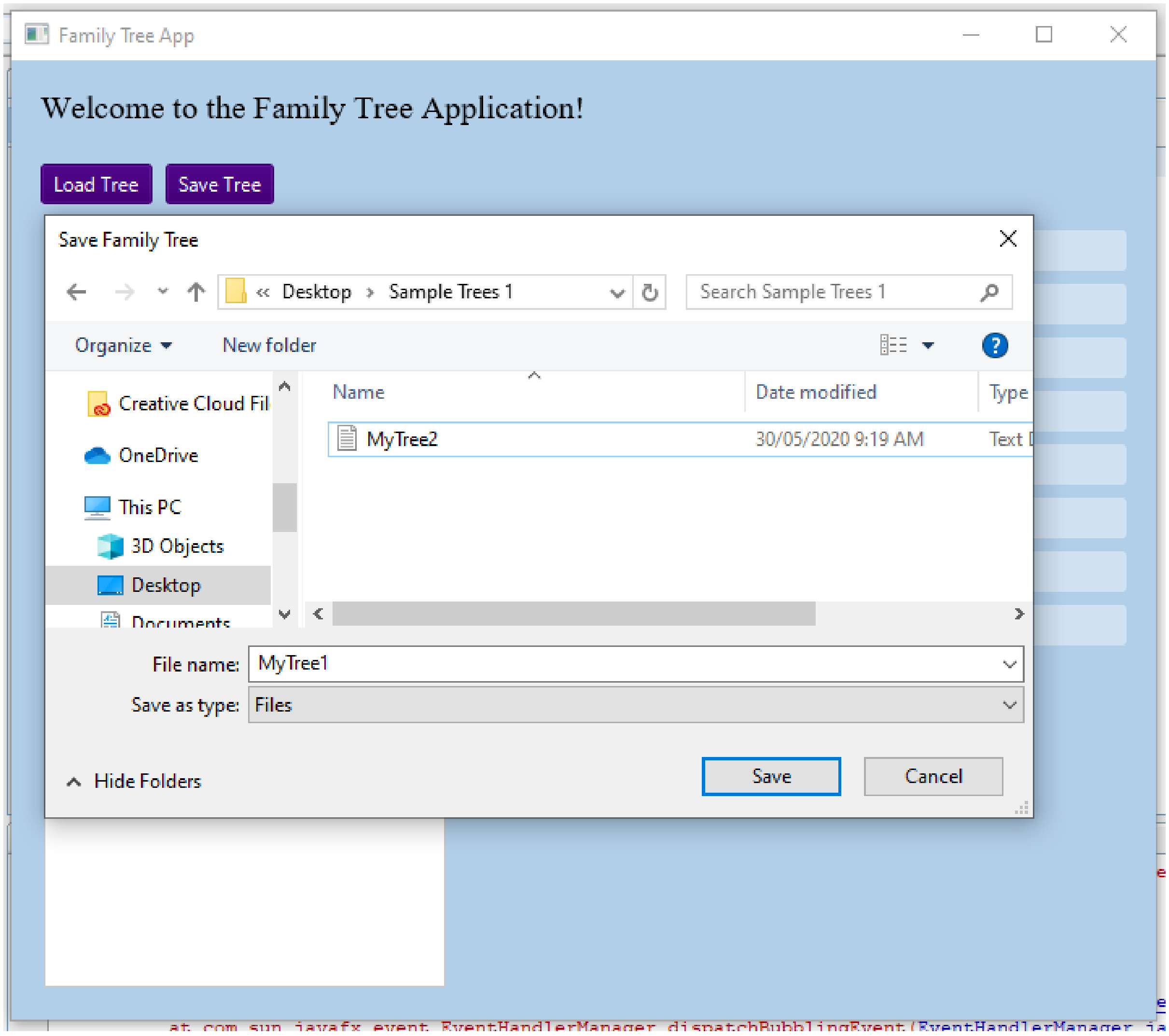


Welcome to the Family Tree Application!

- Harry Maguire
  - Spouse
    - Julie Thomson
  - Children
    - Jess Maguire
      - Children
        - Ruby Blair
    - Jonathan Maguire

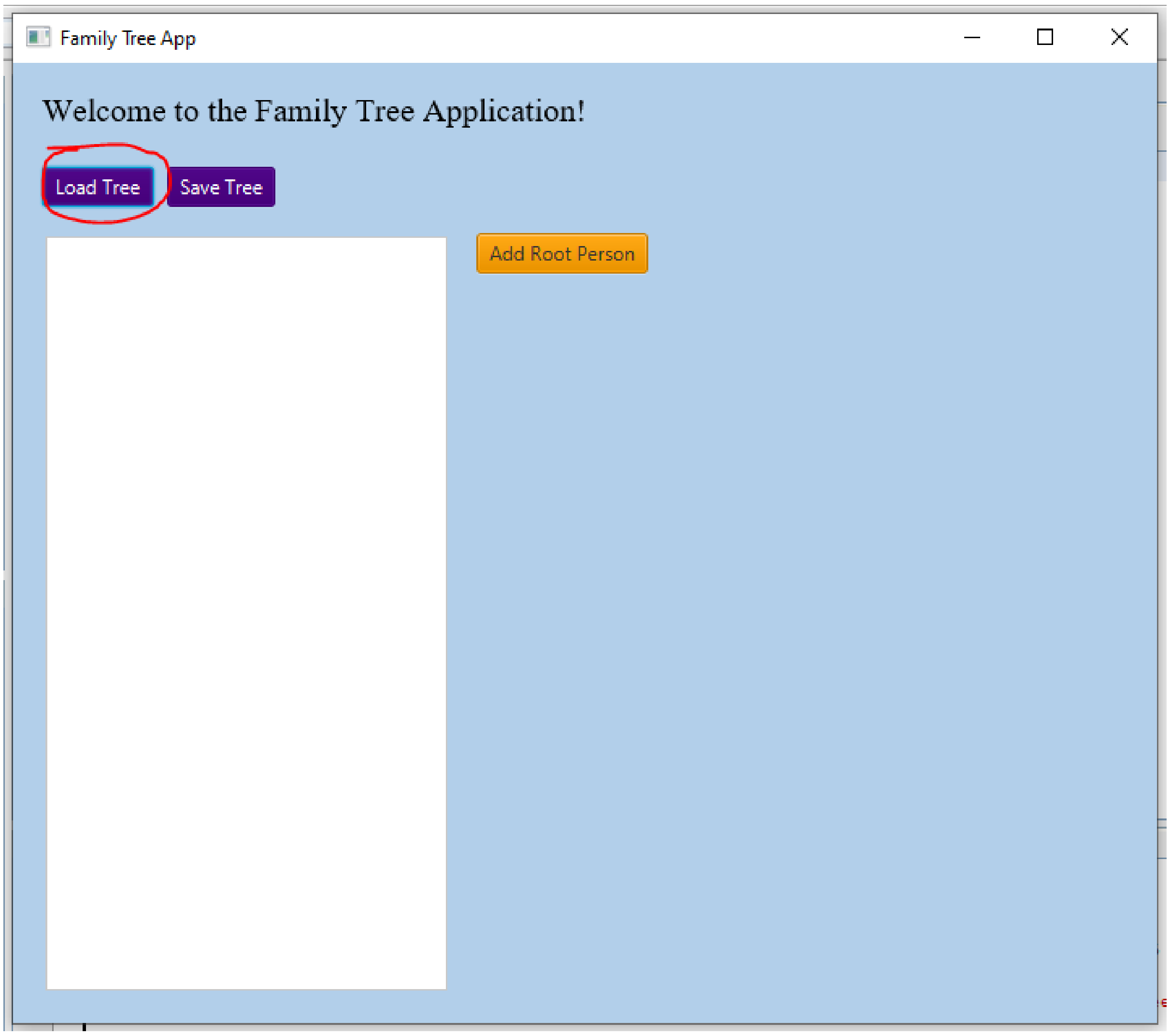
First name	Harry
Last name	Maguire
Maiden name	J
Description	Influencer and writer
Street name	Ashton Rd
Street number	69
Suburb	Billing
Zip code	6383
Gender	[✓ M] F
Member Type	[✓ Child] Spouse

File Chooser Dialog

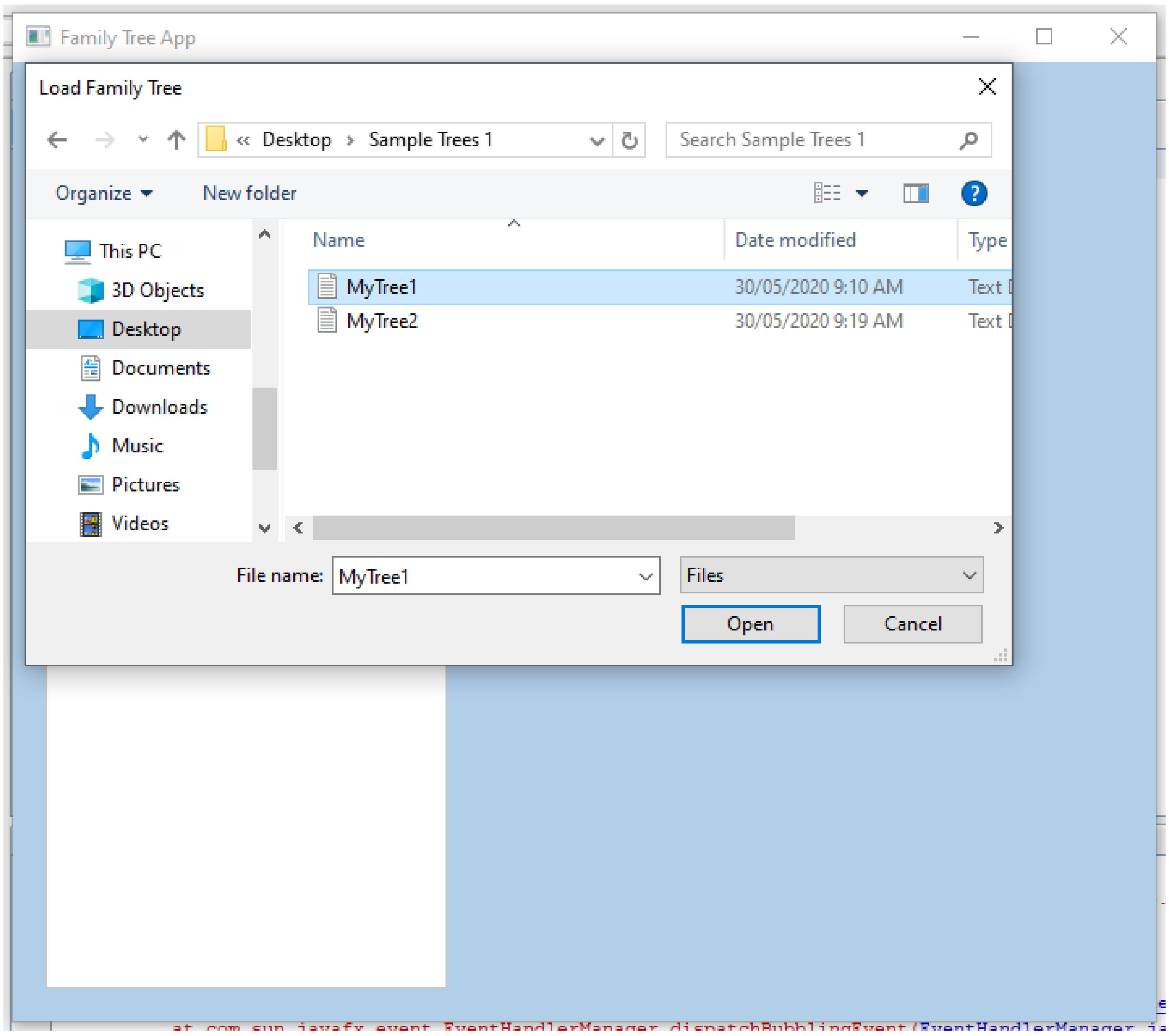


```
at com.sun.javafx.event.EventHandlerManager.dispatchBubblingEvent(EventHandlerManager.ja
```

15) Loading a family tree file



Select the .TXT file

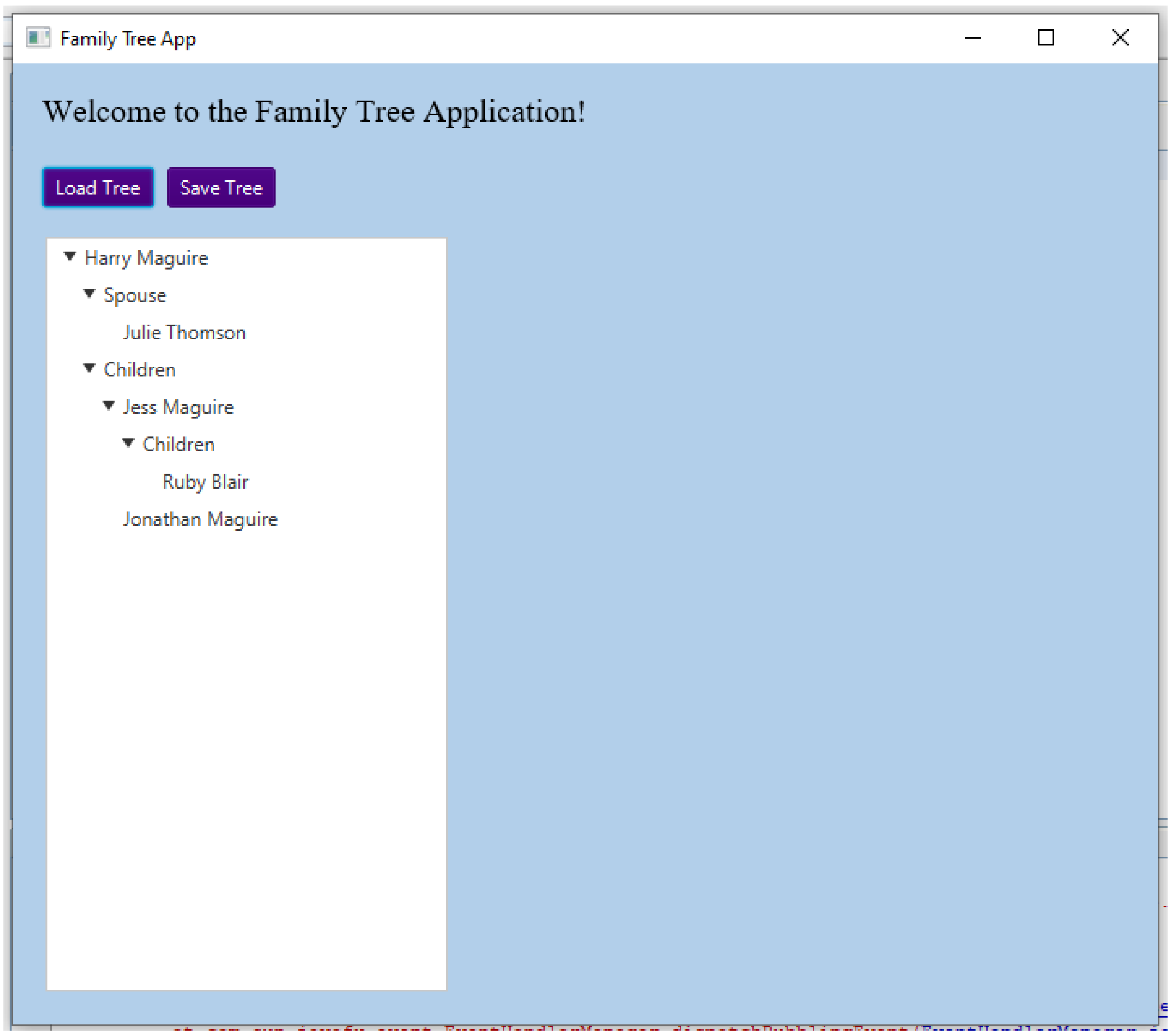


Name	Date modified	Type
MyTree1	30/05/2020 9:10 AM	Text
MyTree2	30/05/2020 9:19 AM	Text

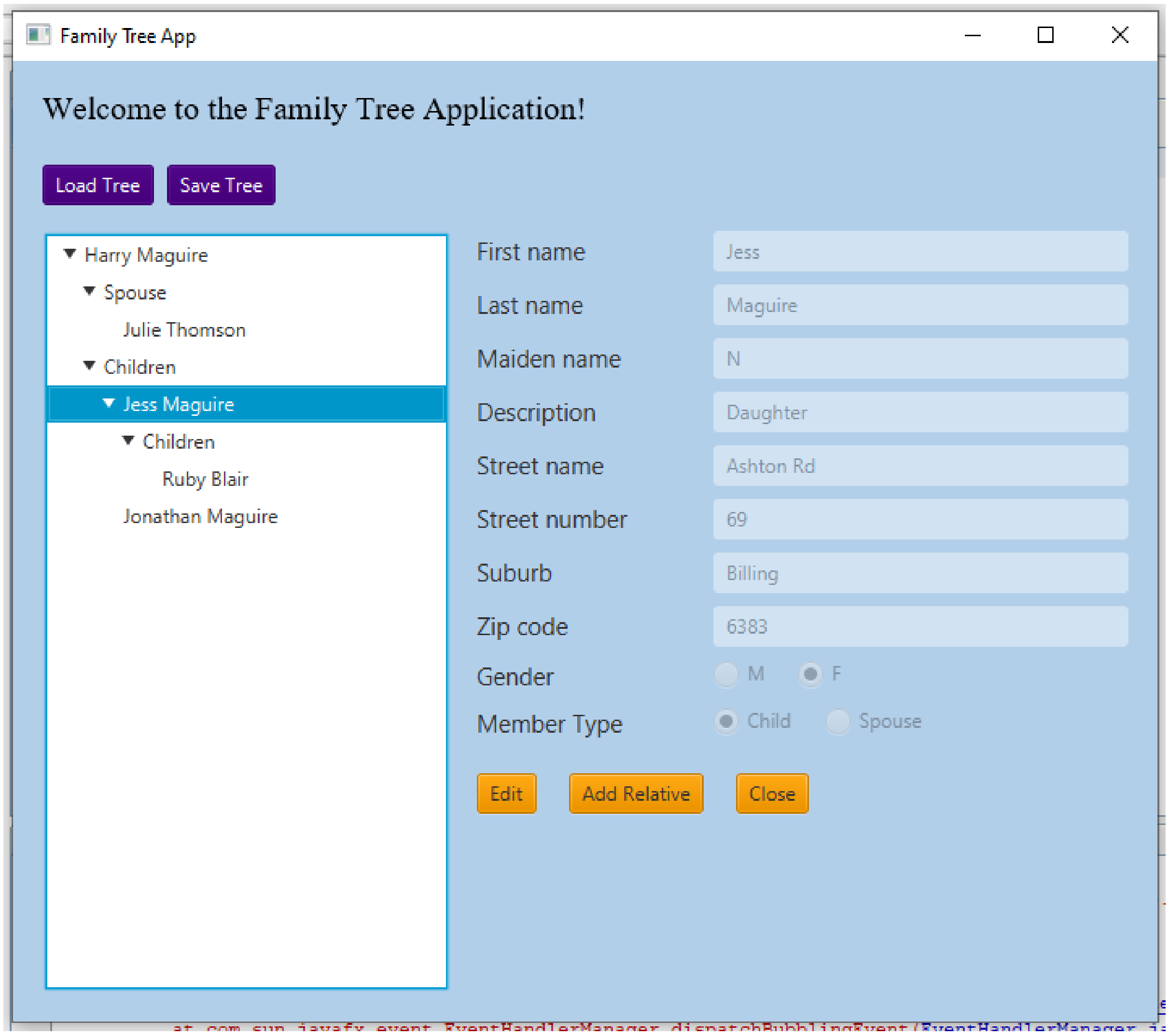
File name: MyTree1

Files

New tree gets loaded, and the user can click on individuals from the family to view their details.



Viewing details of person Jess Maguire –

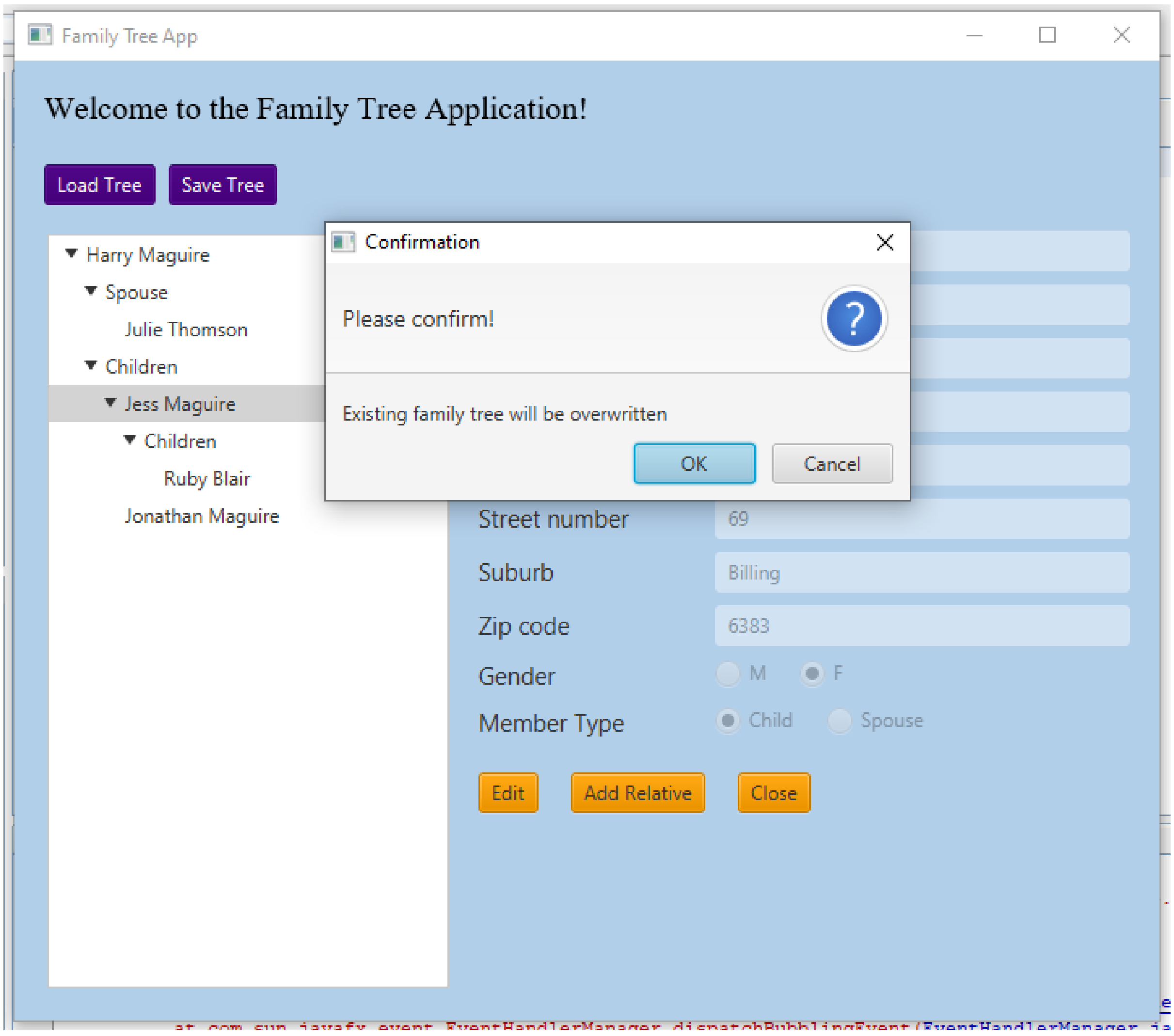


- Harry Maguire
  - Spouse
    - Julie Thomson
  - Children
    - Jess Maguire
      - Children
        - Ruby Blair
    - Jonathan Maguire

First name	Jess
Last name	Maguire
Maiden name	N
Description	Daughter
Street name	Ashton Rd
Street number	69
Suburb	Billing
Zip code	6383
Gender	M [✓ F]
Member Type	[✓ Child] Spouse

```
at com.sun.javafx.event.EventHandlerManager.dispatchBubblingEvent(EventHandlerManager.ja
```

16) Trying to load another tree, when there already one family tree that is loaded



Welcome to the Family Tree Application!

Confirmation

Please confirm!

Existing family tree will be overwritten

Street number 69

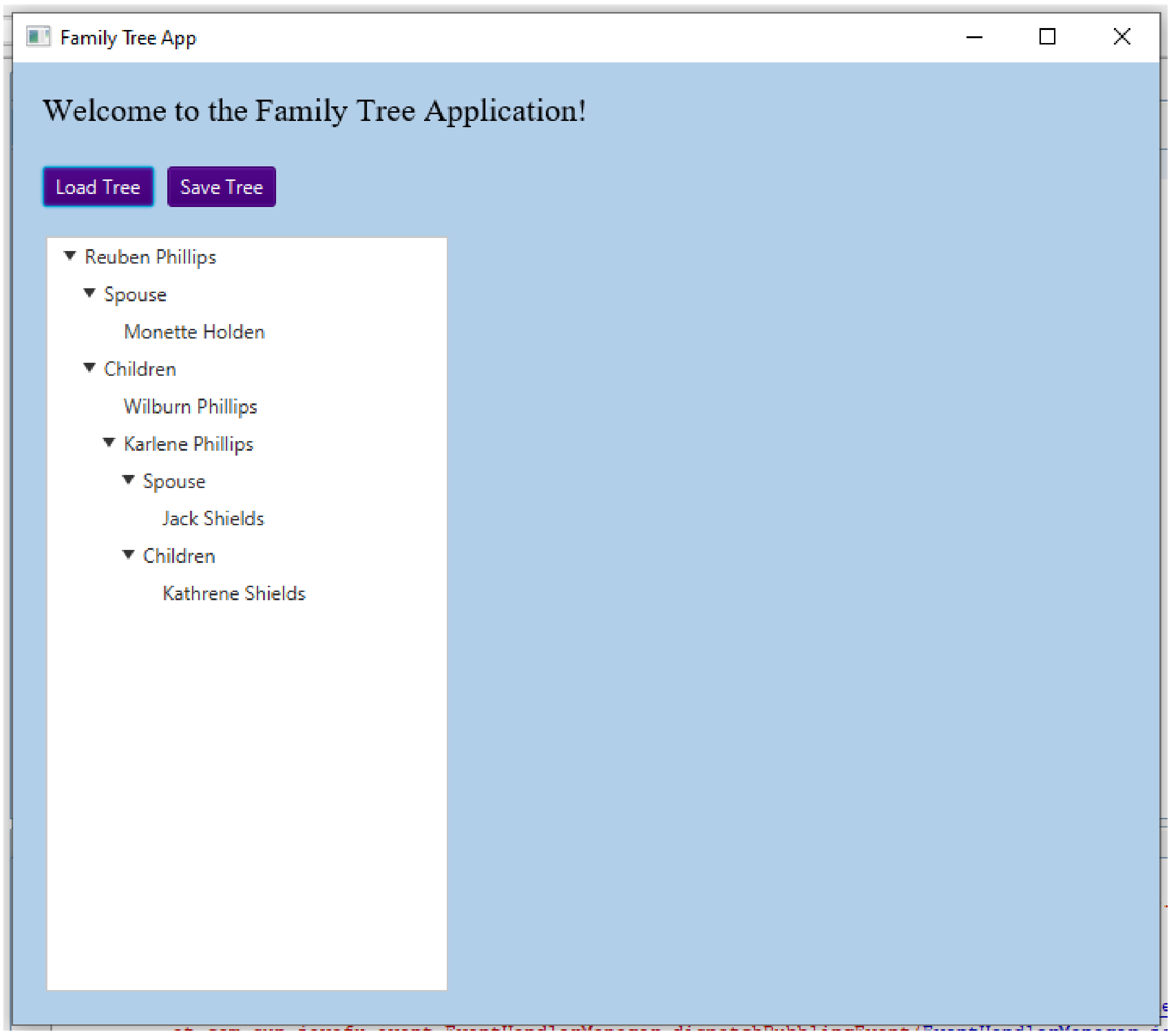
Suburb Billing

Zip code 6383

Gender M [✓ F]

Member Type [✓ Child] Spouse

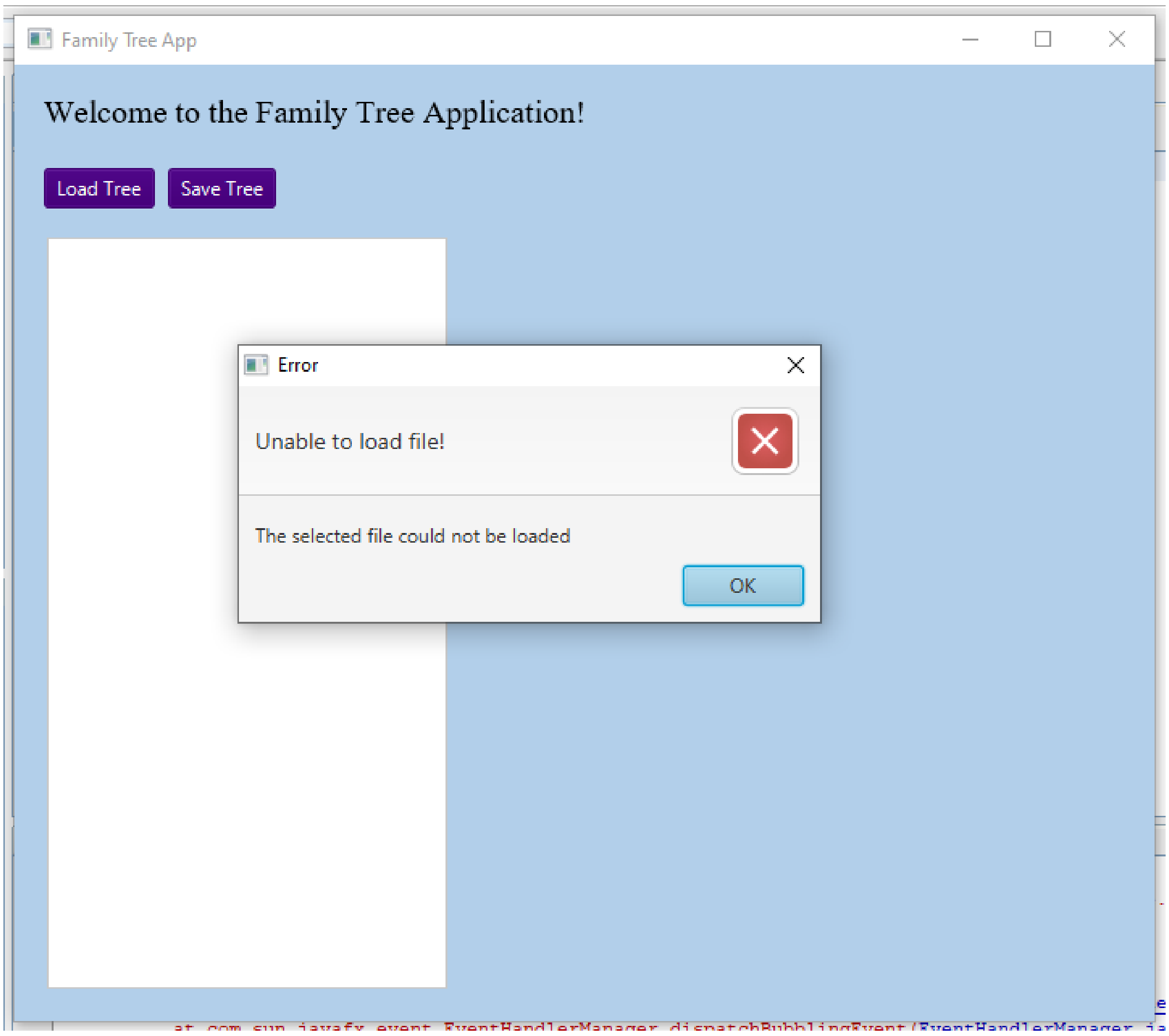
After pressing OK –



Welcome to the Family Tree Application!

```
▼ Reuben Phillips
  ▼ Spouse
      Monette Holden
  ▼ Children
      Wilburn Phillips
    ▼ Karlene Phillips
      ▼ Spouse
          Jack Shields
      ▼ Children
          Kathrene Shields
```

17) Loading incorrect file for the family tree



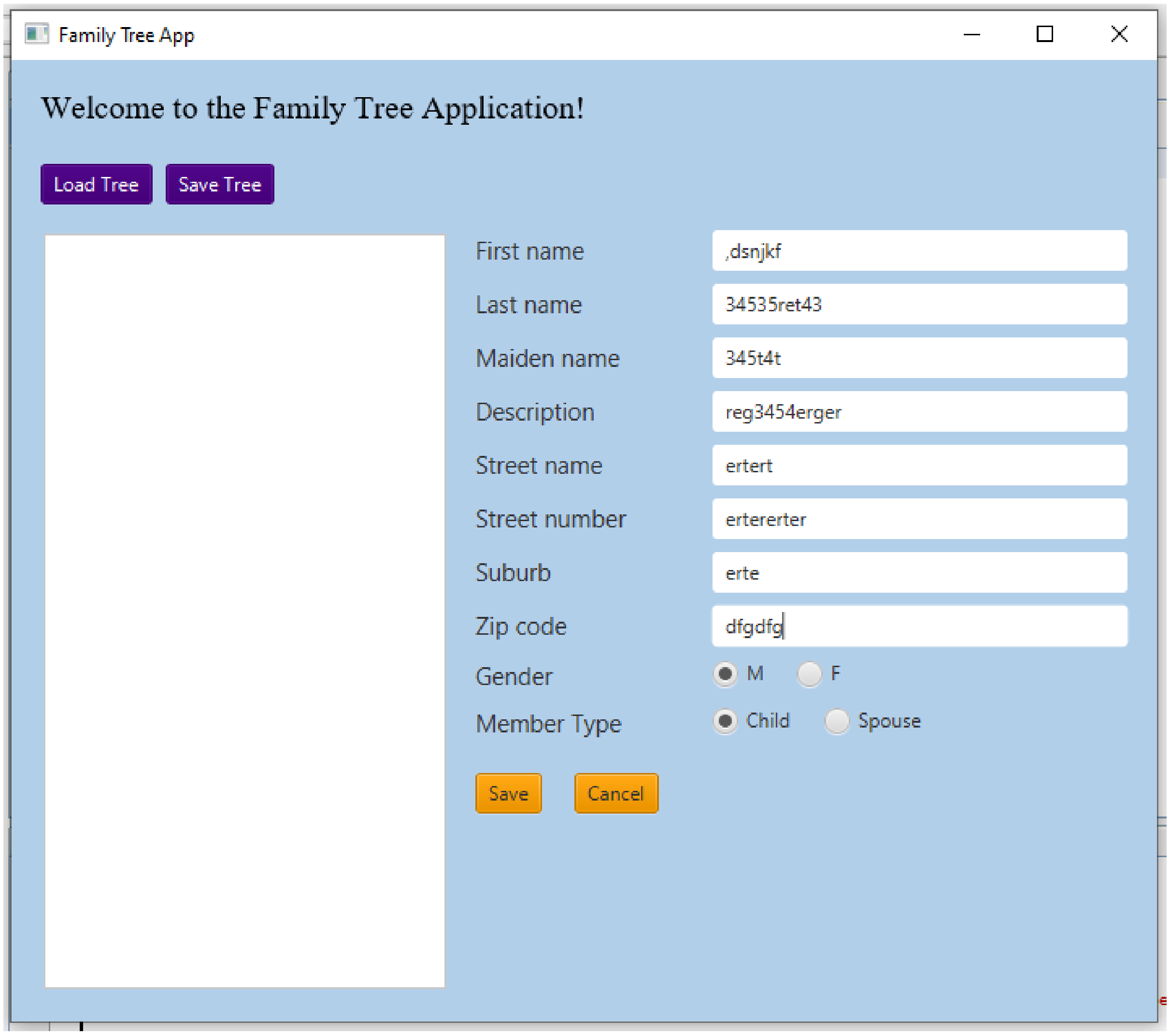
Welcome to the Family Tree Application!

Unable to load file!

The selected file could not be loaded

```
at com.sun.javafx.event.EventHandlerManager.dispatchBubblingEvent(EventHandlerManager.ja
```

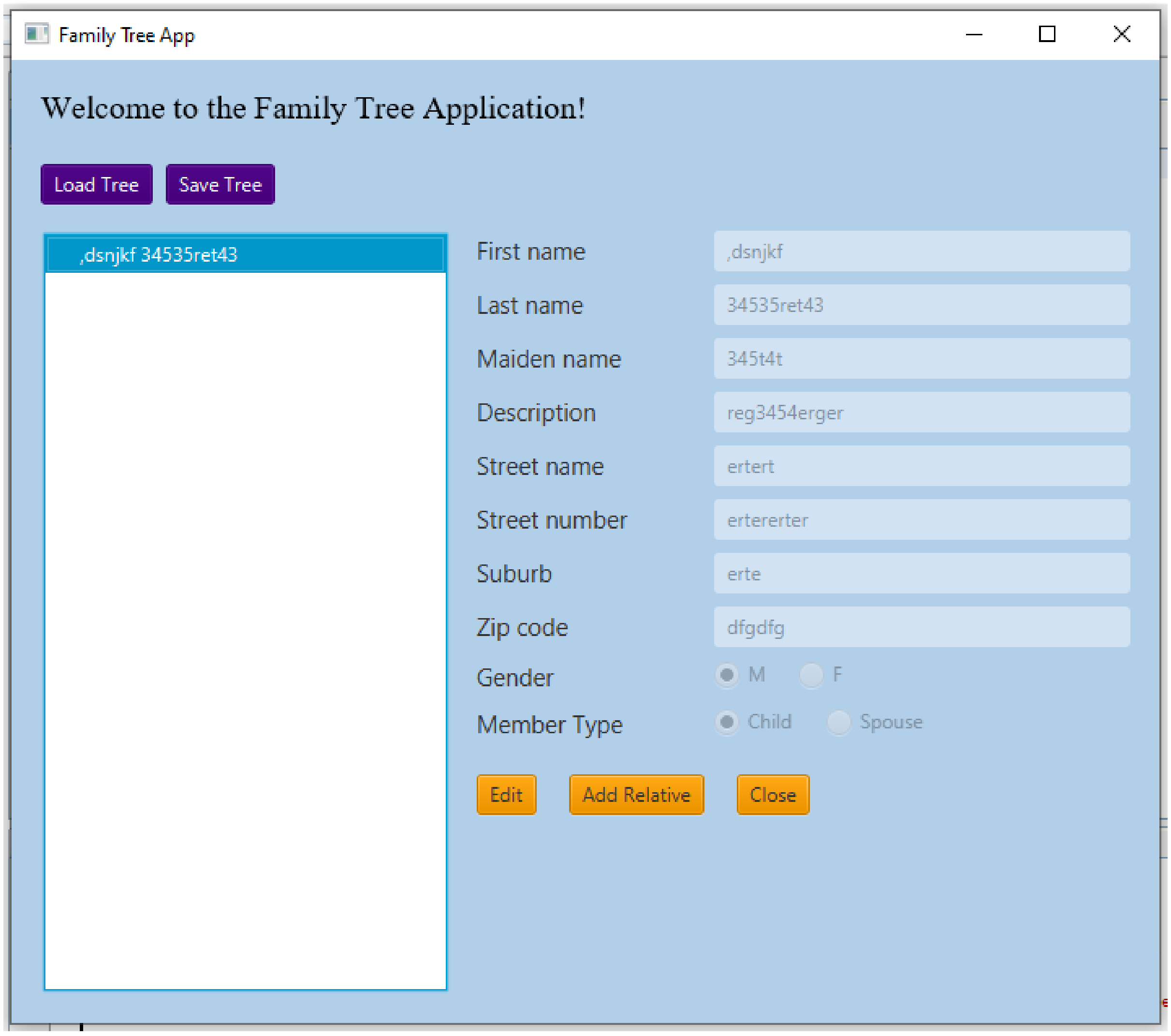
18) Testing input validation



Welcome to the Family Tree Application!

First name	,dsnjkf
Last name	34535ret43
Maiden name	345t4t
Description	reg3454erger
Street name	ertert
Street number	erterterter
Suburb	erte
Zip code	dfgdfg
Gender	[✓ M] F
Member Type	[✓ Child] Spouse

No input validation is supported in this application



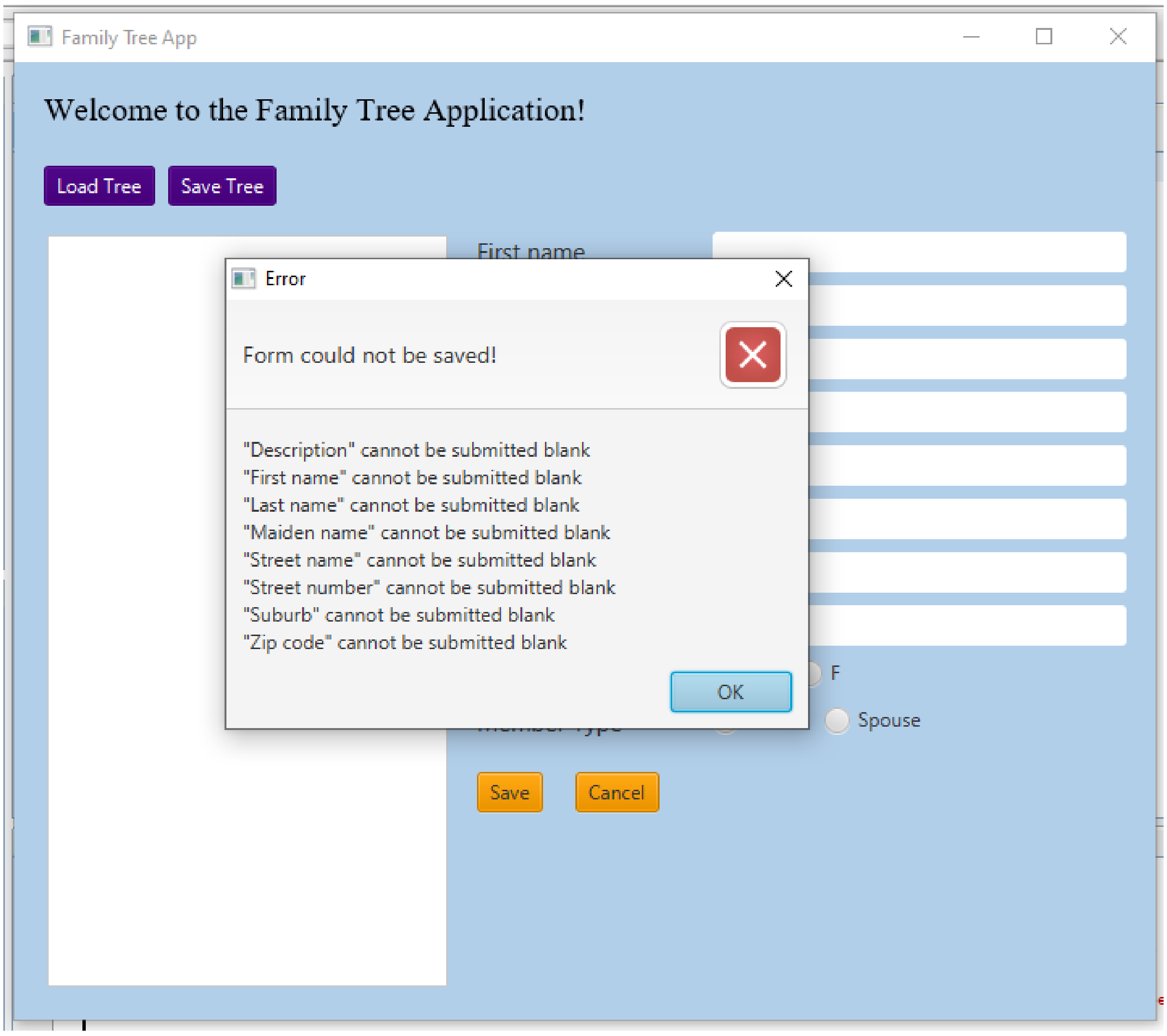
Family Tree App

Welcome to the Family Tree Application!

,dsnjkf 34535ret43

First name	,dsnjkf
Last name	34535ret43
Maiden name	345t4t
Description	reg3454erger
Street name	ertert
Street number	erterert er
Suburb	erte
Zip code	dfgdfg
Gender	[● M] F
Member Type	[● Child] Spouse

19) Submitting an incomplete form



Form could not be saved!

"Description" cannot be submitted blank
"First name" cannot be submitted blank
"Last name" cannot be submitted blank
"Maiden name" cannot be submitted blank
"Street name" cannot be submitted blank
"Street number" cannot be submitted blank
"Suburb" cannot be submitted blank
"Zip code" cannot be submitted blank

All input fields need to be filled to submit the form

Sources

https://docs.oracle.com/javafx/2/overview/jfxpub-overview.htm

https://docs.oracle.com/javafx/2/get_started/fxml_tutorial.htm

https://www.youtube.com/watch?v=FLkOX4Eez6o&list=PL6gx4Cwl9DGBzfXLWLSYVy8EbTdpGbUIG

https://www.youtube.com/watch?v=SvmSNbXQSnQ

https://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/TreeView.html

http://tutorials.jenkov.com/javafx/treeview.html

https://docs.oracle.com/javafx/2/css_tutorial/jfxpub-css_tutorial.htm

https://www.tutorialspoint.com/javafx/javafx_event_handling.htm

https://www.geeksforgeeks.org/javafx-alert-with-examples/

https://docs.oracle.com/javafx/2/ui_controls/file-chooser.htm

https://docs.oracle.com/javase/7/docs/api/java/io/ObjectOutputStream.html

https://docs.oracle.com/javase/7/docs/api/java/io/ObjectInputStream.html

https://stackoverflow.com/questions/26361559/general-exception-handling-in-javafx-8

https://docs.oracle.com/javafx/2/get_started/form.htm

https://www.youtube.com/watch?v=MAiKpkQqb6Q&t=64s

https://www.youtube.com/watch?v=YtKF1JKtRWM

https://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/TextField.html

https://docs.oracle.com/javafx/2/ui_controls/radio-button.htm

https://docs.oracle.com/javase/8/docs/api/java/util/TreeMap.html

https://stackoverflow.com/questions/14149984/for-each-loop-on-java-hashmap

https://docs.oracle.com/javafx/2/api/javafx/scene/control/TreeItem.html

https://www.youtube.com/watch?v=suLCy6tysJQ