NOTE: For whatever reason NetBeans refuses to run my project. So, to run mine, use IntelliJ.

Introduction

For my coursework I have been assigned the task of creating a family tree application that will allow for the addition of a prime ancestor, a partner, and children. To do this I will implement a binary search tree data structure.

I chose to develop this coursework on my own, so everything created was created by myself. I also made use of a GitHub repository to store the work each step of the way.

This repository can be found at <https://github.com/brandonrobson/Coursework>

Step 1

To begin with I created the skeleton of the tree. This includes the main method, the tree node class, the constructor, and some basic methods that add and partners. I added a minor amount of exception handling.

Throughout the development I added comments to each method or wherever I felt was appropriate to do so.

You can either display the entire family tree, or the details of a specific family member.

Difficulties

I wasn’t able to implement a constructor that set up both the ancestor and their partner.

Step 2

For the second half of the project, things got a little more complex with the addition of identifiers for each node in the tree. Partners and children could only be created under certain conditions. When these conditions aren’t met, further exception handling had to be introduced.

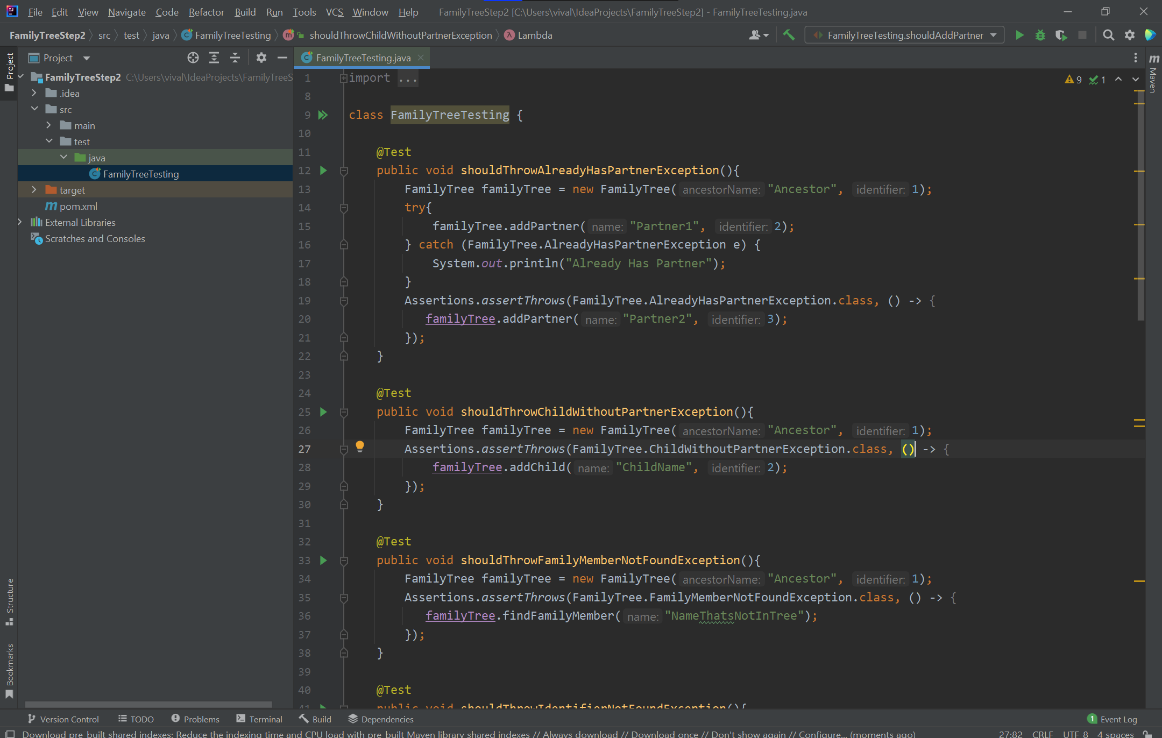
A partner can be added to a ancestor or child as long as that specific node is made the current node using the find or get family member methods.

Difficulties

Couldn’t restrict the “no children” message to just children with partners.

Testing

Junit Testing



Alongside the use of Junit Testing, I opted to manually test the program with the test tables below.

|  |  |  |  |
| --- | --- | --- | --- |
| Test | Description | Expected | Result |
| Add child (without partner) | The first ancestor has been added and we try to create a child before assigning a partner | A child cannot be added, an exception will be raised and the user will be alerted of this. | The user can enter a name, but the child isn’t actually created. An exception is raised and the user is alerted. |
| Add child (with partner) | Same as before but this time the ancestor has a partner | The user will enter a name. If the name isn’t a duplicate, the child is added to the tree. | The child is added, assuming the name isn’t a duplicate. |
| Add child (duplicate name) | The user attempts to add a child, but the name is a duplicate. | The user will enter a name, but the system will throw an exception and the user is alerted. No child is created. | The user enters the name, and the system correctly raises a duplicate name exception. |
| Add partner (No partner) | The user attempts to add a partner to a single family member. | The user will enter a name for the partner and the partner will be added to the current family member. | A partner is created using the inputted name. |
| Add partner (Already has partner) | The user attempts to add a partner but the family member is already taken. | The user will attempt to create a new partner, but the system detects that the family member is already taken. An exception is raised | The system detects that the family member the partner is to be assigned to is already taken and then raises an exception. |
| Search for family member by name | The user enters a name and the system searches the tree for the family member with that name. | If the name is in the tree, the system will return that family member’s details. Otherwise, an exception will be raised. | Given that the name is actually in the system, that family member is made the current. Otherwise, an exception is raised |
| Search for family member by ID | The user enters an ID and the system searches for the family member with that ID. | If the ID is in the tree, the system will return that family member’s details. Otherwise, an exception will be raised. | Given that the ID is actually in the system, that family member is made the current, and their details are displayed. |
| Display family tree | A method that displays the entire family tree is called. | The system displays the family tree in its entirety. Including partners, and children’s partners. | The system displays the family tree in its entirety. Including partners, and children’s partners. |