Test Driven development log –

Name –

|  |  |  |
| --- | --- | --- |
| Class - Member | | |
| Function – What it does | **Test – What is being tested** | **Result – expected result** |
| Set\_mother – it checks to see if there’s a value in the list, if there isn’t one, it raises an error. It then checks if the mother’s gender is female, if it isn’t, it raises an error, but if it is there, it sets that as the mother variable. | It tests to see if there is a mother in the member list.  It also tests for the gender of the mother once it is found in the list | Raising an error if there isn’t a mother in the member’s list as well as raising an error if the mother’s gender is incorrect. |
| Get\_paternal\_grandmother – it gets the father’s mother | Tests if there is a father in the member’s list.  Tests if the father has a mother in the member’s list. | Return empty if either one of those tests fail.  If it finds the member, it sets her as the paternal grandmother |
| Get\_paternal\_aunt – set a paternal aunt as a member by looking at the paternal grandmother and if she has other children | Tests if the grandmother variable has a value inside.  Tests if the grandmother has other children. | Return empty if either one of those tests fail.  If it finds her other children, it gets the one that is female and set her as the aunt. |
| Get\_brother-in\_law - Sets a brother-in-law | Tests if there is a value inside spouse\_mother.  Tests if spouse\_mother has children | Return empty if either one of those tests fail.  If the value is found, it filters the value to only male and for it to be under the spouse’s mother other children. |
| Get\_son – finds the son in the member list | Tests if there is a child under the selected member in the code and he is male. | Return empty if there is no child under the selected member.  If the child is found, it will set a filter to only male, so it becomes the son. |
| Set\_father – sets a father | Tests if a mother is in the member list.  Tests if the father’s gender is male. | Raising an error if there isn’t a father in the member’s list as well as raising an error if the father’s gender is incorrect. |
| Get\_maternal\_grandmother – it gets the mother’s mother | Tests if there is a mother in the member’s list.  Tests if the mother has a mother in the member’s list. | Return empty if either one of those tests fail.  If it finds the member, it sets her as the maternal grandmother |
|  |  |  |
| Get\_sister\_in\_law – it finds the daughter of the spouse’s mother and set it as the sister-in-law | Tests if there is a value inside spouse\_mother.  Tests if spouse\_mother has children | Return empty if either one of those tests fail.  If the value is found, it filters the value to only female and for it to be under the spouse’s mother other children. |
| Get\_siblings – gets all siblings of a selected member | Tests if the selected member has a mother.  Tests if the mother has other children. | Returns with an empty list if there is no mother found or no other children.  Then filtering the member to make sure it doesn’t have the exact name as the specified member.  Further setting that child to a siblings list. |
| Get\_daughter – finds the daughter in the member list | Tests if there is a child under the selected member in the code and she is female | Return empty if there is no child under the selected member.  If the child is found, it will set a filter to only female, so it becomes the son. |
| Get\_spouse\_sibling – getting the sibling of the spouse | Tests if the selected member has a spouse. | If the member doesn’t have a spouse, it returns empty, but if it does have a spouse, it gets the spouse’s siblings and returns it. |

|  |  |  |
| --- | --- | --- |
| Class - FamilyTree | | |
| Function – What it does | **Test – What is being tested** | **Result – expected result** |
| Get\_relationship – gets the relationships of a specified member | Tests if the specified member exists in the Member list.  Tests for the members’ relationship type. | If the member doesn’t exist, it will return with a message to tell them that.  If the member doesn’t have any relationships, it returns with None.  If the tests are successful, it will return back with a sorted id and relationships on the member |
| Add\_spouse – add a spouse to a specified member | Tests if the specified member exists in the family tree.  Tests if the person you are trying to add already exists.  Tests to find the spouse of the specified member.  Tests for the gender of the spouse.  Tests if the spouse already has a spouse. | If the member doesn’t exist, it returns a message.  If the person already exists, it will return a message.  If the member doesn’t exist, it returns a message.  If the spouse has the same gender as the member that it fails.  If the spouse already has a spouse than the test fails.  If all tests succeed, it sets the spouse to the specified member and return with an appropriate message. |
| Add\_child – adding a child in the family tree | Tests if the member exists in the family tree.  Tests to find a mother in the family tree.  Tests the mother’s gender.  Testing if the mother has a spouse, the father. | If it doesn’t, it will add the child to the family tree.  If it already exists, then the addition of the child will fail.  If there is no mother, it returns with an appropriate message.  We expect the mother to be female for it to work.  If the father doesn’t exist, the addition of the child fails.  After all tests are successful, it sets a mother and a father to the child that is being added, with a name, as well as returning a message saying the addition was successful. |
| \_\_init\_\_ – makes the family tree a dictionary for all members to go inside. |  |  |