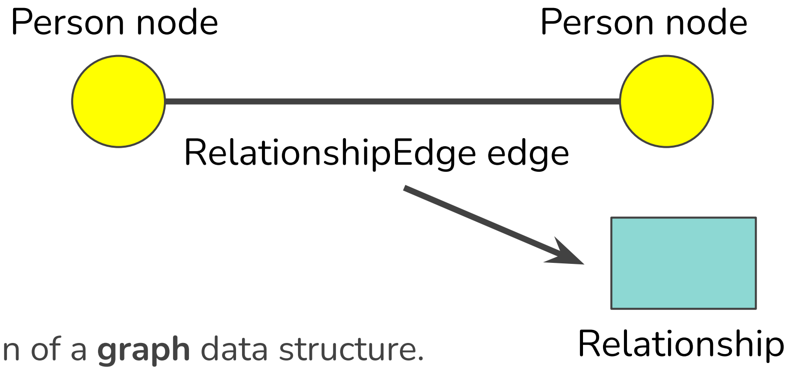
Team Q Customer Report

Collin Jones, Jordan Nazemi, Michael Galvan, Robert Bulai

System Architecture

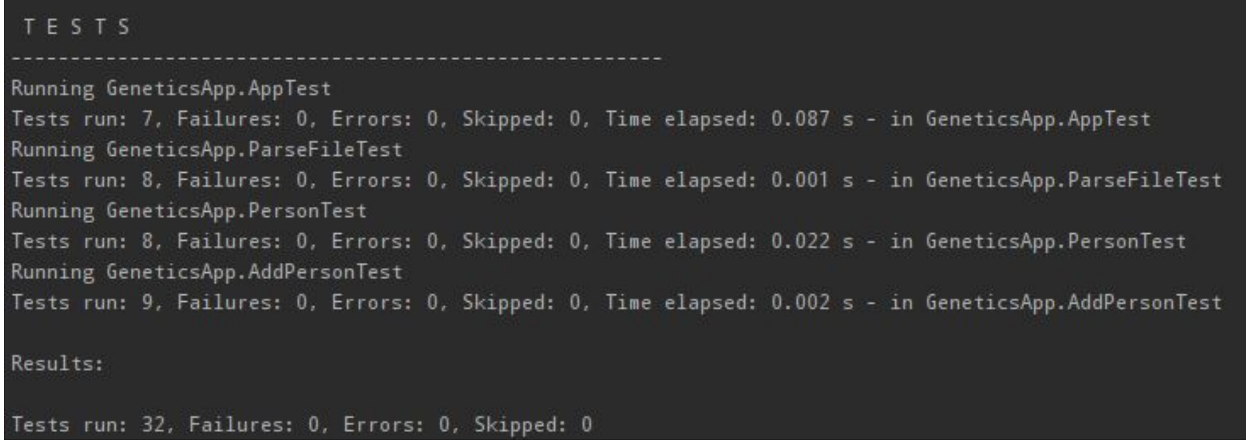


- The basis for our family tree is the JGrapht implementation of a **graph** data structure.
  - In a graph **nodes** are connected to one-another via **edges.**
  - **Nodes:** The nodes of a graph can be any type of object, in our case being a **Person** object which holds the biographical information for that person
    - **Person** objects hold the biographical information for a person including their ID, name, DOB, DOD, etc. as well as all the prerequisite get and set functions for those fields
  - **Edges:** Our implementation uses a custom edge-class we call **RelationshipEdge** which besides connecting two nodes also has a "label" object in the form of a **Relationship.**
    - **RelationshipEdge** objects, besides having standard edge functionality, also has a getLabel() function that provides the relationship it represents
    - **Relationship** objects hold all the biographical data for a given relationship including participant ID's, relationship ID, start date, end date, and short location description as well as all the necessary get and set functions.

Known/Unknown Bugs

- During cycle two there was a bug that affected the output of grandparents/grandchild. It resulted in random person's getting labeled as there grandparent or grandchild. Now for cycle 3 we changed the format and now it's fixed.
- There was a bug that left a person out of the output, when added a new person before outputting. Fixed for cycle 3
- No unknown Bugs were left after cycle 3

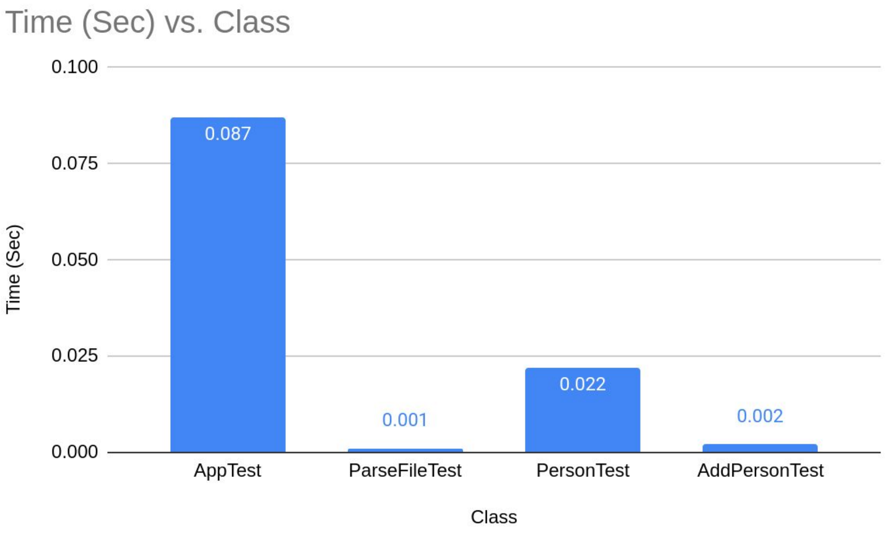
Test Report

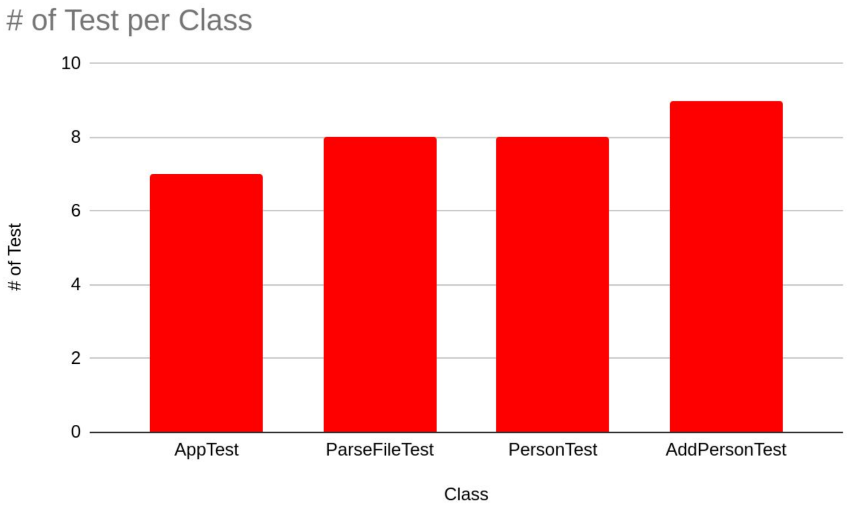


```
 T E S T S
-------------------------------------------------------
Running GeneticsApp.AppTest
Tests run: 7, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.087 s - in GeneticsApp.AppTest
Running GeneticsApp.ParseFileTest
Tests run: 8, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.001 s - in GeneticsApp.ParseFileTest
Running GeneticsApp.PersonTest
Tests run: 8, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.022 s - in GeneticsApp.PersonTest
Running GeneticsApp.AddPersonTest
Tests run: 9, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.002 s - in GeneticsApp.AddPersonTest

Results:

Tests run: 32, Failures: 0, Errors: 0, Skipped: 0
```





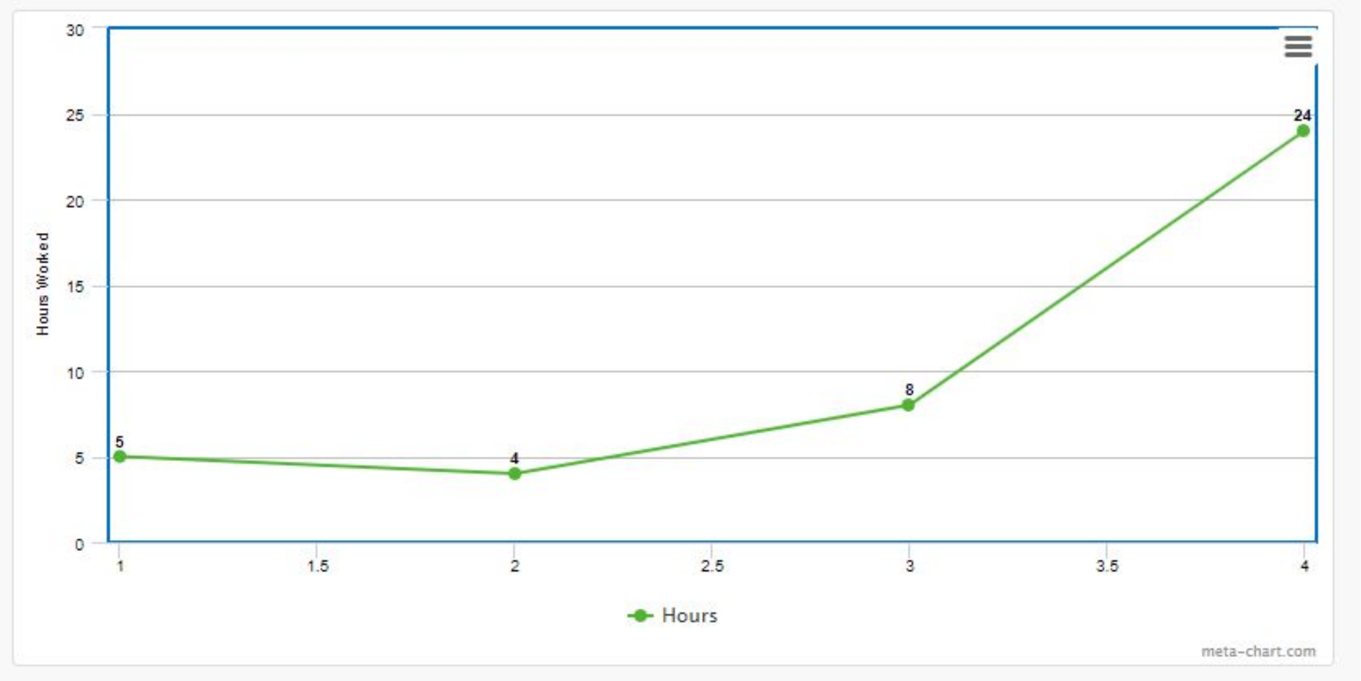
Product Quality

- Obviously of the highest quality… we wouldn’t have it any other way ***/s***
- Functionality of the application is high
    - Able to parse any text file
    - Graph is built automatically upon importing file
    - Able to add as many desired new people and as many desired new relationships
        - Can add people or relationships without providing any information (ID still required)
    - Able to search entire graph structure for people
        - Full name
        - First name
        - Last name
    - Able to output file
- Only thing missing really is refactoring & presentational quality (no GUI, still a very large “primary” class that does the bulk of the work)

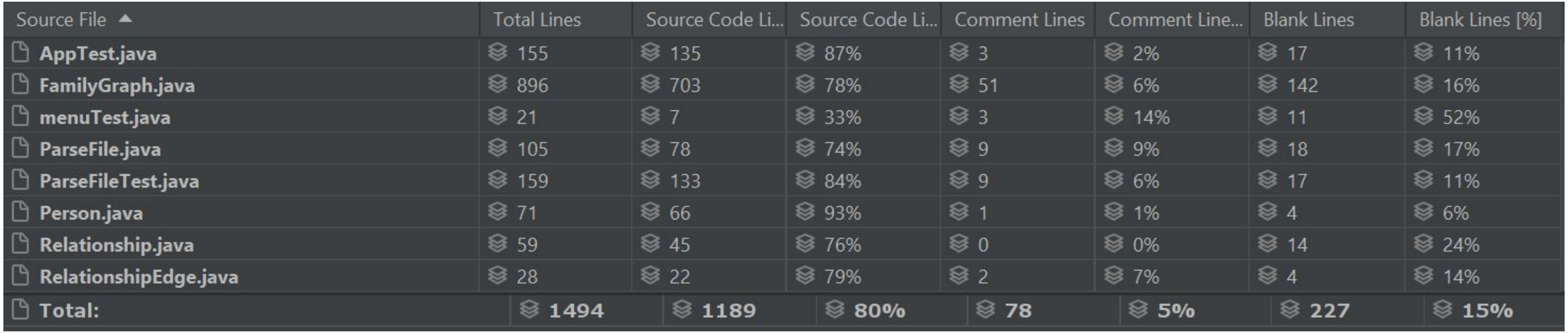
Demo

Metrics

Weekly Hours Worked



Product Size

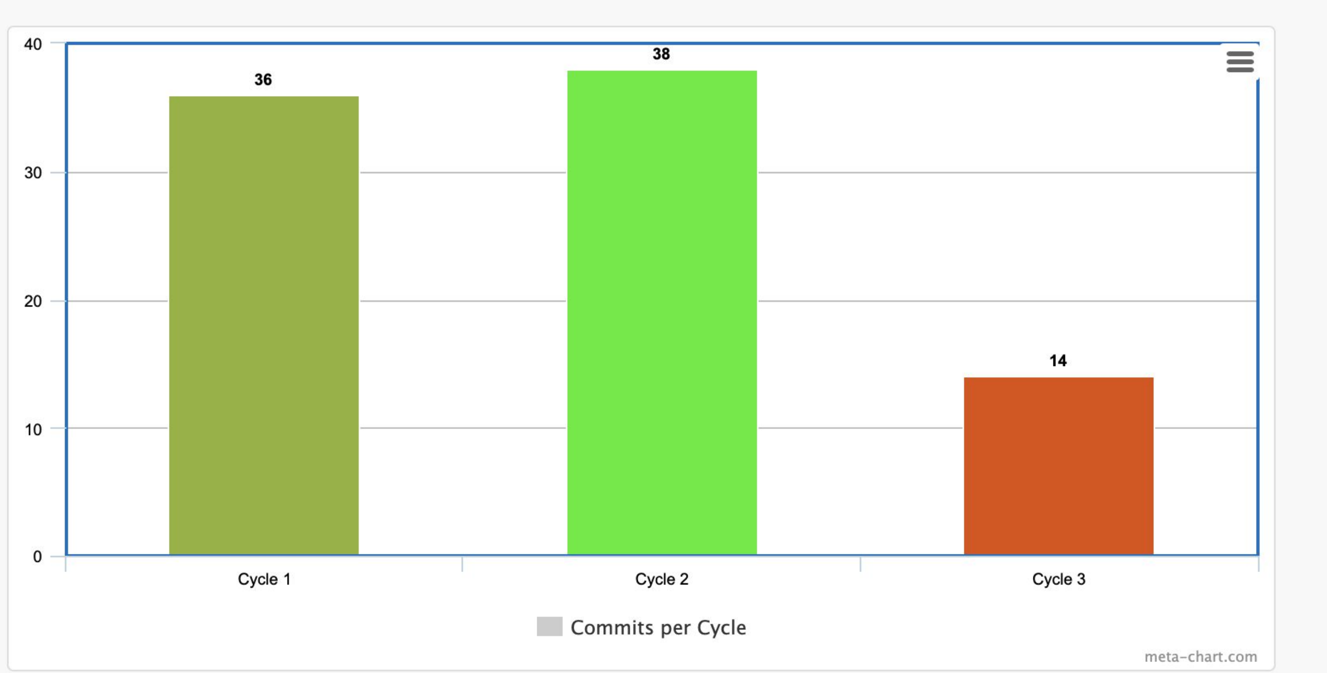


Source File ▲	Total Lines	Source Code Li...	Source Code Li...	Comment Lines	Comment Line...	Blank Lines	Blank Lines [%]
AppTest.java	155	135	87%	3	2%	17	11%
FamilyGraph.java	896	703	78%	51	6%	142	16%
menuTest.java	21	7	33%	3	14%	11	52%
ParseFile.java	105	78	74%	9	9%	18	17%
ParseFileTest.java	159	133	84%	9	6%	17	11%
Person.java	71	66	93%	1	1%	4	6%
Relationship.java	59	45	76%	0	0%	14	24%
RelationshipEdge.java	28	22	79%	2	7%	4	14%
Total:	**1494**	**1189**	**80%**	**78**	**5%**	**227**	**15%**

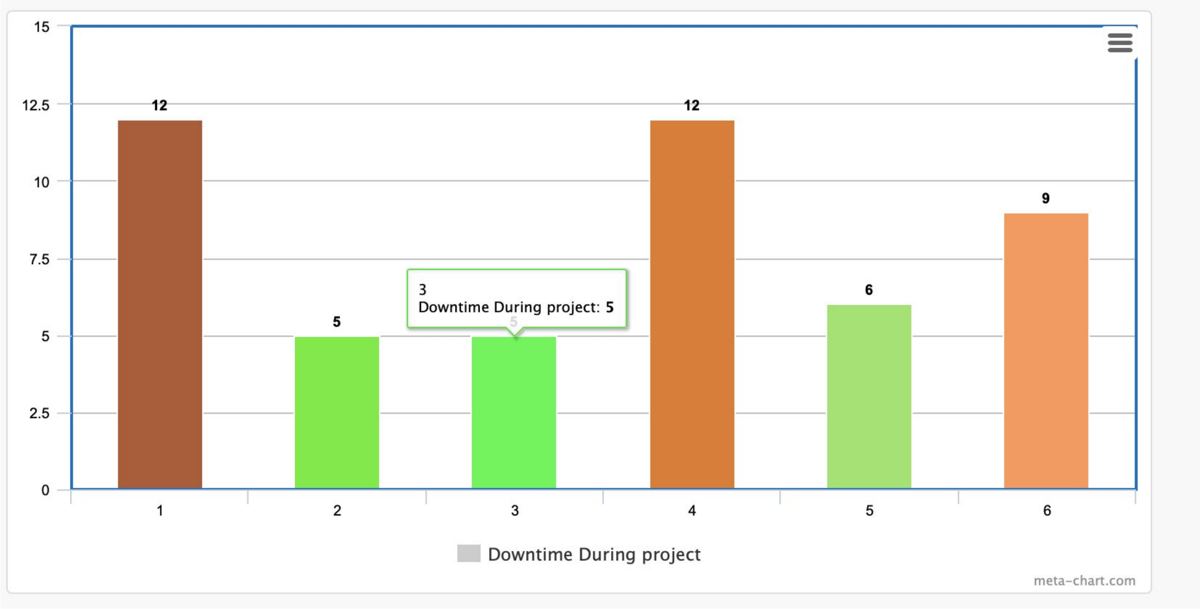
Important Notes

- ~1500 lines of code
- ~70% of code is development, 30% test
- 15% blank lines

Number of Commits

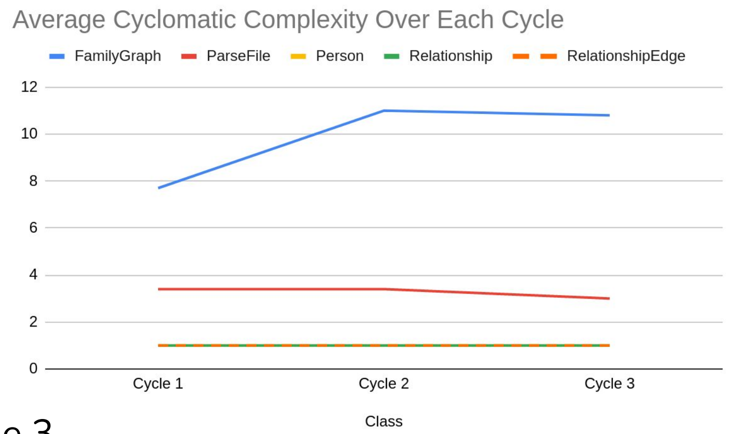


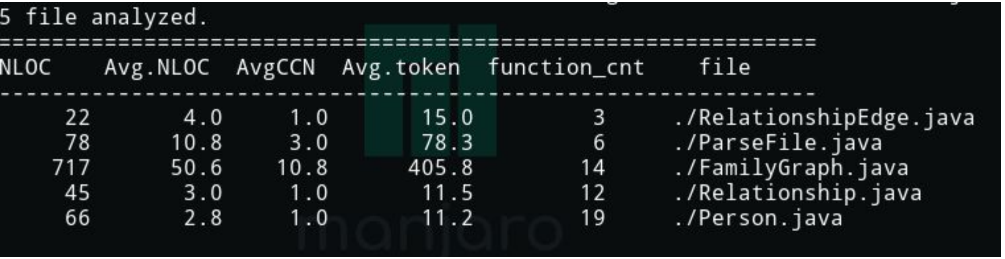
Development Downtime



49/98 days = 50% total downtime

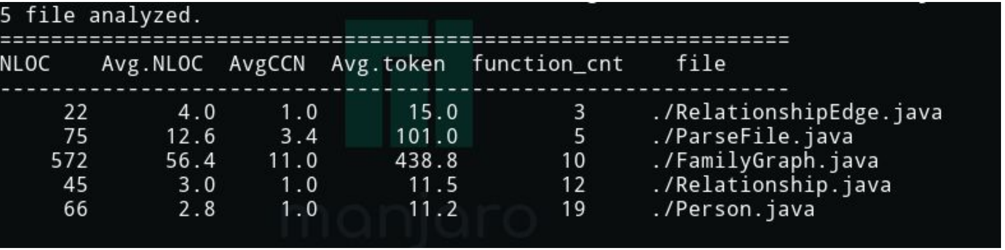
Cyclomatic Complexity





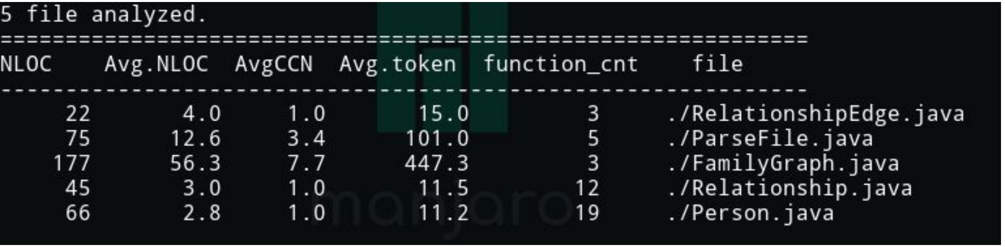
```
5 file analyzed.
==============================================================
NLOC    Avg.NLOC  AvgCCN  Avg.token  function_cnt    file
--------------------------------------------------------------
    22       4.0     1.0       15.0         3     ./RelationshipEdge.java
    78      10.8     3.0       78.3         6     ./ParseFile.java
   717      50.6    10.8      405.8        14     ./FamilyGraph.java
    45       3.0     1.0       11.5        12     ./Relationship.java
    66       2.8     1.0       11.2        19     ./Person.java
```

Cycle 3



```
5 file analyzed.
==============================================================
NLOC    Avg.NLOC  AvgCCN  Avg.token  function_cnt    file
--------------------------------------------------------------
    22       4.0     1.0       15.0         3     ./RelationshipEdge.java
    75      12.6     3.4      101.0         5     ./ParseFile.java
   572      56.4    11.0      438.8        10     ./FamilyGraph.java
    45       3.0     1.0       11.5        12     ./Relationship.java
    66       2.8     1.0       11.2        19     ./Person.java
```

Cycle 2



```
5 file analyzed.
==============================================================
NLOC    Avg.NLOC  AvgCCN  Avg.token  function_cnt    file
--------------------------------------------------------------
    22       4.0     1.0       15.0         3     ./RelationshipEdge.java
    75      12.6     3.4      101.0         5     ./ParseFile.java
   177      56.3     7.7      447.3         3     ./FamilyGraph.java
    45       3.0     1.0       11.5        12     ./Relationship.java
    66       2.8     1.0       11.2        19     ./Person.java
```

Cycle 1

- More complex means harder to maintain
- General guide for how many tests to write
- Keeping CC < 15 is good practice

Tool used: Lizard
(https://github.com/terryyin/lizard)

Teamwork & What We’ve Learned

- We learned early on that since everyone is a at a different level of computer science education we needed to plan appropriately
  - People were assigned tasks based on ability and equitable effort
- A successful deliverable requires close communication and teamwork in order to ensure that all the requirements are met to an adequate standard
  - Cycle 2 led to a little slacking on this point, something we hopefully learned for Cycle 3
- Holding each other accountable for work by having clear assignments and due dates allowed us to ensure there wasn’t any significant blocking going on
- Before progress can begin on a project, it needs to be ensured everyone has the appropriate environment setup (git repository, IDE, etc)

What We Could’ve Done Differently

- Better shared understanding of the project
- More teamwork
- Much better refactoring
    - Splitting up functions into different classes
    - Breaking down very large functions into smaller functions
        - This is especially useful for writing tests
- Evened out the amount of hours spent working each week during each cycle
    - No more “Oh sh*t, Cycle X is due this Friday” on the Monday

Fin