Progress Evaluation: Milestone 2

Code Visualization

Team Members

- Curtice Gough cgough2019@my.fit.edu
- Joshua Hartzfeld jhartzfeld2020@my.fit.edu
- Catherine DiResta cdiresta2019@my.fit.edu

Client/Advisor

- Dr. Ryan Stansifer ryan@fit.edu

Progress Matrix

Task	Completion %	Curtice	Joshua	Catherine
1. GUI groundwork	50%	0%	50%	0%
2. Traceprinter JSON parsing	75%	75%	0%	0%
3. Example Java programs	50%	0%	0%	50%

Task Summary / Team Member Contribution

1. GUI groundwork

JOSHUA:

> I've adopted the QT Designer tool to help me construct a user friendly and efficient template for our GUI. Progress is slow but steady since I need to learn what QT Designer can offer and how I can make the template useable with our backend aka trace printer. QT Designer also allows easy animations and interactable text and graphical windows, so it satisfies what we need it for. Designing the window should no longer be a hurdle, however due to my unfamiliarity with the template system and PyQT5 engine's ability to load templates we may encounter an issue with communication between front and backend. Luckily for us we are using python so I assume there will be numerous solutions that we will discover while in development.

2. Traceprinter JSON parsing

CURTICE:

So far, I've written about 50 lines of code. The focus thus far has been on implementing the functions "main", "jsonify_java", and "step_bro". The function "jsonify_java" is fully operational. It takes a filename as its parameter, opens the corresponding file, reads in Java source code, and formats it as valid JSON input for Traceprinter. In "main", "jsonify_java" is called, and the resulting JSON is sent as input to a subprocess which runs Traceprinter. The output of this subprocess is captured and passed as a parameter to "step_bro", which is still a work in progress. This function will contain the necessary implementation for stepping through each traced program state and representing variables accordingly.

3. Example Java programs

CATHERINE:

The Java programs that will be used for testing throughout the project were started. The first batch of programs were designed to be easy to understand and simple to trace to allow for clear results if anything was wrong. To begin, one program was created for each of the supported data structures. The custom data structures were started as well. In order to test that the system wouldn't mistake any of the supported programs that use Hashmap and Heap were created for the non-supported data structure tests.

Milestone 3 Task Matrix

Task	Curtice	Josh	Catherine
1. Data type detection	100%	0%	0%
2. More GUI	0%	100%	0%
3. Custom data structure implementations	0%	0%	100%

Task Summary

1. Data type detection

   In the function "step_bro", begin adding functionality to detect various data types captured in "trace_output_json". Accurately represent the values of each variable at each stage of the trace.

2. More GUI

   Build enough of the main window that the Data Structure View is usable, and start writing the custom PyQt widget used as the superclass for data structure diagrams.

3. Custom data structure implementations

   Write a custom implementation of the "BinaryTree" data structure and add it to the "codeviz" package.

Client Meeting Dates

- 27 October 2023

Faculty Advisor Signature:_____ Date: _____