LogicVis

Candice Miao
Leo Gao
Jed Chen
Glenn Zhang
Andrew Liu

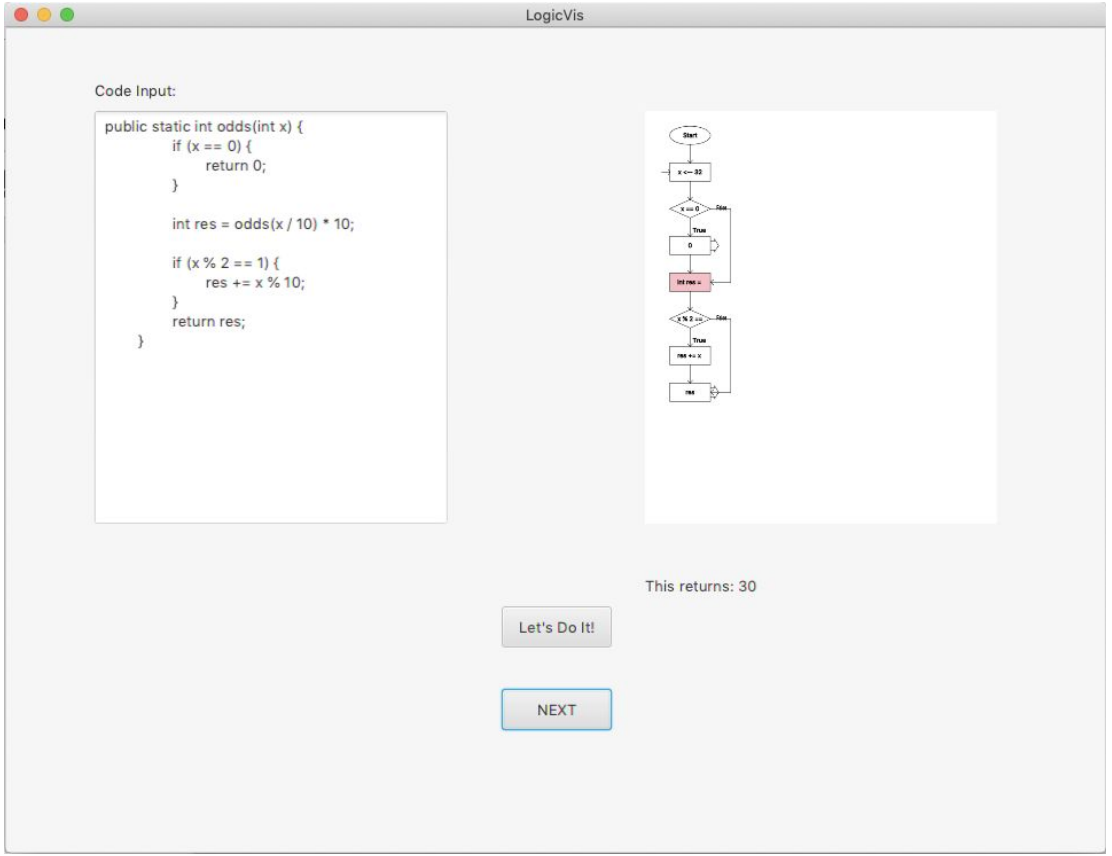
Recursion is a tough subject

Students are not using abstractions well enough

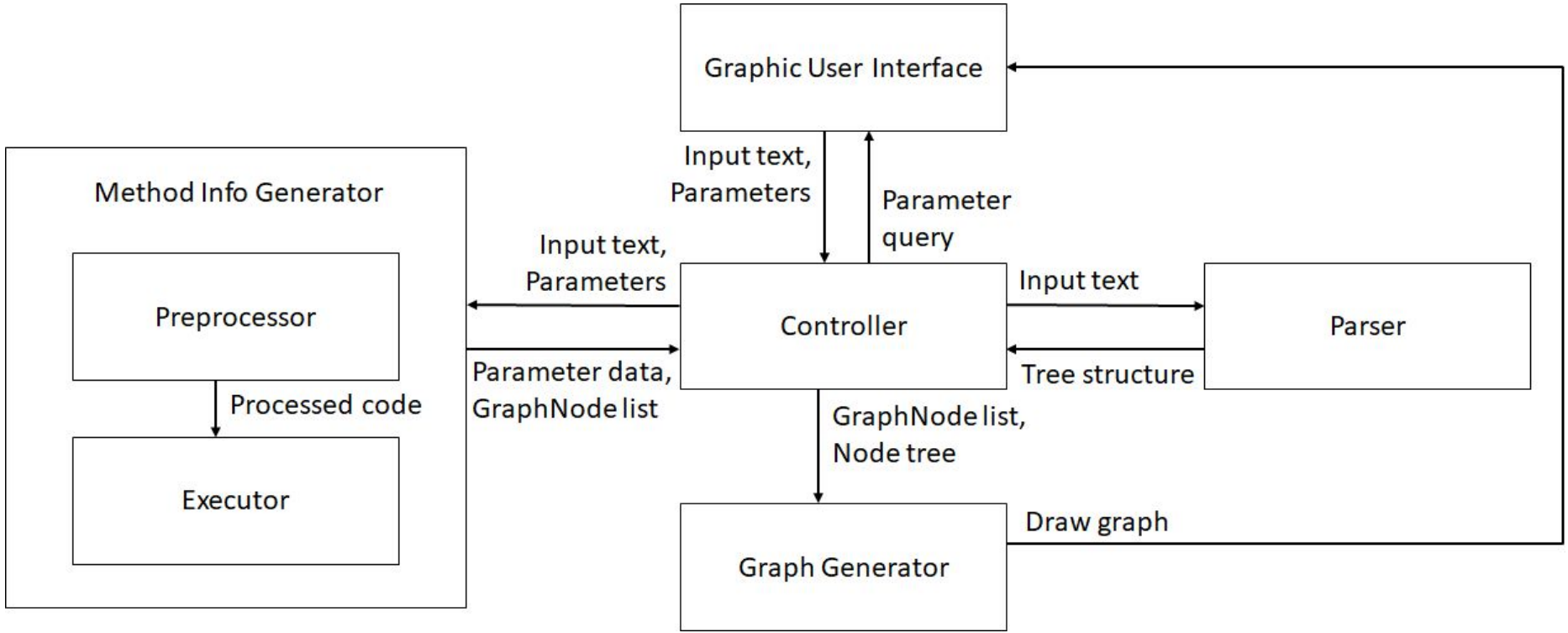
Lack of a proper methodology to represent a recursive solution

Approach

- Visualization provides an intermediate abstraction
- Our solution not only indicates the function call but also visualizes the specific call iteration
- Our solution provides the parameters of each function call



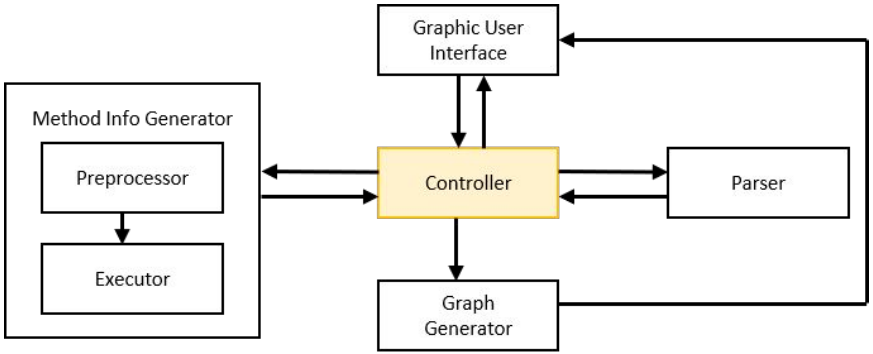
Architecture



Controller

Role: Sending information to different parts of the system

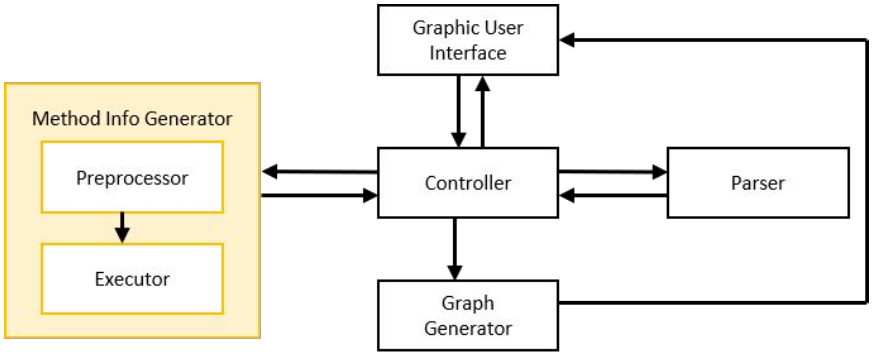
- Gets information from GUI
- Sends input to Parser to get the Node Tree
- Sends input to Info Generator to get the parameter information
- Sends parameter values to Info Generator to get Graph Nodes
- Sends the Node Tree and Graph Nodes to Graph Generator



Method Info Generator

Role: find all the method calls and their return values

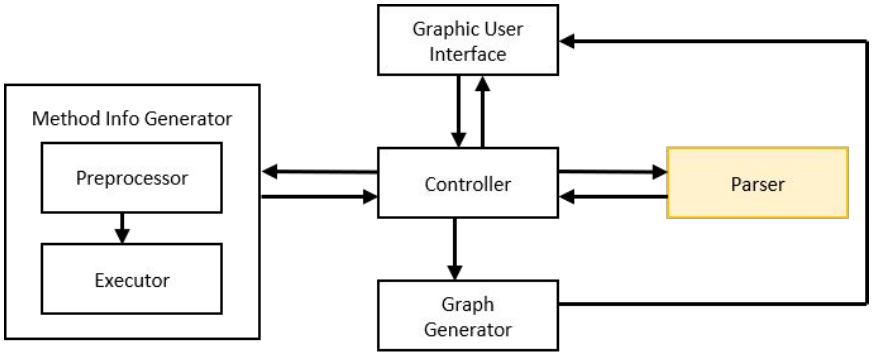
- Extract parameter, return, and depth information
  - GraphNode: contains this info
- Code Injection: BeanShell



Parser

Role: parse input code into a drawable graph

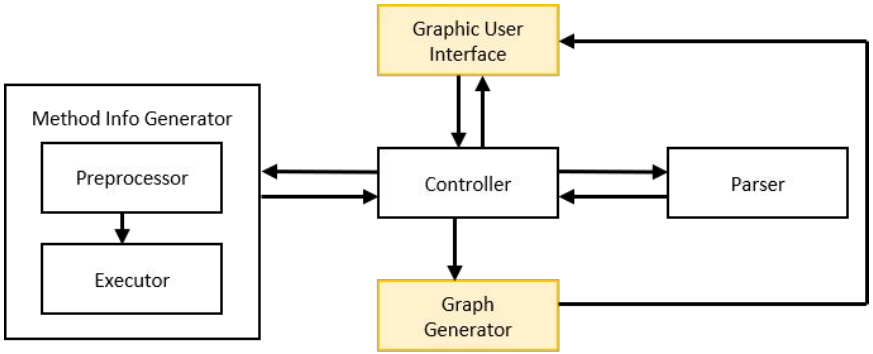
- Static Analysis: JavaParser and AST
- Constructed our own Node tree
  - content, type, children
  - Catered to our needs

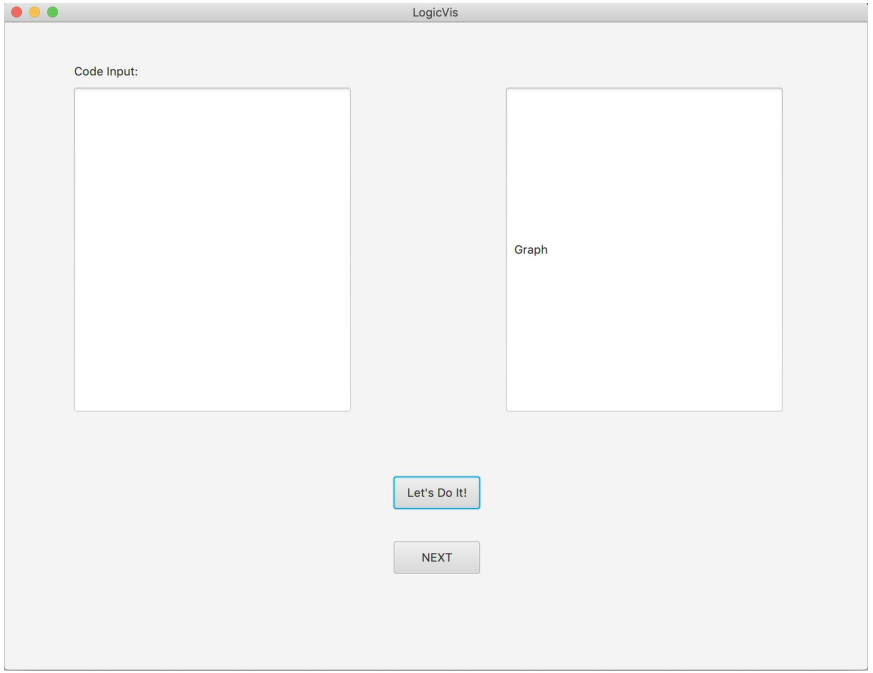


GUI & Graph Generator

Role: obtain and show information for the user

- JavaFX - drawing
  - Familiar, simple enough
  - JPEG Graph





Demo

User Studies

- Tested by letting users solve recursive programming questions with and without the tool
- Example: odds() and removeOnes() are the experiment questions
- The program only runs when user fills in compilable code

```java
/**
 * Get rid of all the digits that are even in the string.
 * Examples of this function are:
 * odds(323) = 33
 * odds(13524) = 135
 * odds(12345) = 135
 */
public static int odds(int x) {
    if (x == 0) return 0; // Base Case

    // Recursive Case
    int res = odds(x / 10) * 10;

    if (x % 2 == 1) {
        res += x % 10;
    }
    return res;
}
```

```java
/**
 * Replaces all ones in an integer with zeros.
 * Examples of this function are:
 * removeOnes(123) returns 23
 * removeOnes(212) returns 202
 * removeOnes(111111) returns 0
 */
private static int removeOnes(int n) {
    // Base Case
    if (______) {
        return n;
    }

    // Recursive Case
    if (n % 10 == 1) {  // if the next digit is 1
        return _______;
    }
    return ______;
}
```

Results

- First Experiment, the top 3, used a tool with worse graphics, and we have improved that in the second experiment

Subject Experience	Time Spent
Almost no Java experience, had Matlab experience (No recursion knowledge)	- 57 minutes to solve odds() without the tool - 32 minutes to solve printTwos() with the tool.
High School Computer Science (Has learned about recursion)	- 24 minutes to solve printTwos() without the tool - 6 minutes to solve odds() with the tool.
Basic Collegiate Programming and Data Structures (Has learned about recursion)	- 27 minutes to solve odds() with the tool
Basic Collegiate Programming (Has learned about recursion)	- 20 minutes to solve odds() without the tool - 31 minutes to solve removeOnes() with the tool

Results

- Users show an improvement in the time taken to understand the code
- The graph itself helps the user process information
- The user debugged using the graph

Subject Experience	Time Spent
Almost no Java experience, had Matlab experience (No recursion knowledge)	- 57 minutes to solve odds() without the tool - 32 minutes to solve printTwos() with the tool.
High School Computer Science (Has learned about recursion)	- 24 minutes to solve printTwos() without the tool - 6 minutes to solve odds() with the tool.
Basic Collegiate Programming and Data Structures (Has learned about recursion)	- 27 minutes to solve odds() with the tool
Basic Collegiate Programming (Has learned about recursion)	- 20 minutes to solve odds() without the tool - 31 minutes to solve removeOnes() with the tool

Limitations & Improvements

- Graphics can be better
- There are some bugs
- Fairly limited to single method recursions

Reflection

- Start earlier on implementation
- Design choices are difficult to change

Thank you!