



Code Visualization

Milestone 3

1. Data type detection

Curtice Gough

Overview

1. Encoded locals
 - a. Stack variables
 - b. Heap variables
2. Parsing heap references
 - a. JSON structure
 - b. Type recognition
 - c. Getting values
3. Other structures
 - a. Object instances
 - b. Compiling multiple files

```
44 trace.py
19 19 @@ -19,8 +19,10 @@ def main(argv):
20 20     trace_output_dict = json.loads(trace_proc.communicate(input=trace_input_json.encode())[0])
21 21
22 + # DEBUG
23 + #print(json.dumps(trace_output_dict))
24 + #quit()
25 +
26 26     step_bro(json.loads(trace_input_json), trace_output_dict)
27 27
28 28 #-----#
29 29 @@ -29,7 +33,7 @@ def jsonify_java(filename):
30 30     trace_input_dict = {
31 31         "usercode": "",
32 32         "options": {},
33 33         "args": [],
34 34         "args": [], # passed to main()
35 35         "stdin": ""
36 36     }
37 37
38 38 @@ -58,12 +51,48 @@ def step_bro(input_json, output_json):
39 39     for state in events:
40 40         print(f"state['event']: {state['stack_to_render'][0]['func_name']}")
41 41         print(f"code: {state['line']} | source_code[{state['line'] - 1}.strip()}'")
42 42         print(f"locals: {state['stack_to_render'][0]['encoded_locals']}")
43 43
44 44     print_locals(state)
45 45
46 46     print(f"globals: {state['globals']}")
47 47     print(f"stdout: {(v[state['stdout']] if v)}")
48 48     print("")
49 49     input("Press ENTER to continue...")
50 50
51 51 + def print_locals(state):
52 52 +     encoded_locals = state['stack_to_render'][0]['encoded_locals']
53 53 +
54 54 +     print("\nlocals:")
55 55 +     for var in encoded_locals:
56 56 +         if isinstance(encoded_locals[var], list):
57 57 +             if encoded_locals[var][0] == "HEAP":
58 58 +                 print(f"({var}):", get_heap_item(state, encoded_locals[var][1]))
59 59 +             else:
60 60 +                 print(f"({var}):", encoded_locals[var])
61 61 +
62 62 + def get_heap_item(state, heap_id):
63 63 +     heap_item_type = state['heap'][str(heap_id)][0]
64 64 +     if heap_item_type == "LIST":
65 65 +         return get_list(state, heap_id)
66 66 +     else:
67 67 +         return f"UNKNOWN DATA TYPE: {heap_item_type}"
68 68 +
69 69 + def get_list(state, heap_id):
70 70 +     old_list = state['heap'][str(heap_id)]
71 71 +     new_list = []
72 72 +     for item in old_list[1:]:
73 73 +         if isinstance(item, list):
74 74 +             if item[0] == "LLDIE":
75 75 +                 prev_item = new_list[-1]
76 76 +                 for i in range(len(item)):
77 77 +                     new_list.append(prev_item)
78 78 +                 else:
79 79 +                     print("ERROR: unknown:", item)
80 80 +                 else:
81 81 +                     new_list.append(item)
82 82 +
83 83 +     return new_list
84 84
85 85 #-----#
86 86 # This is where the fun begins :D #
87 87 #-----#
```

Encoded locals

- Stack locals show just the value
- Heap locals contain a “REF”

```
root@4fd976bd5c7: /code-v x + v
{
  "stdout": "\0\n1\n2\n3\n4\n5\n6\n7\n8\n",
  "event": "step_line",
  "line": 6,
  "stack_to_render": [
    {
      "func_name": "main:6",
      "encoded_locals": {
        "i": 9,
        "arr": [
          "REF",
          424
        ]
      },
      "ordered_varnames": [
        "arr",
        "i"
      ],
      "parent_frame_id_list": [],
      "is_highlighted": true,
      "is_zombie": false,
      "is_parent": false,
      "unique_hash": "142",
      "frame_id": 142
    }
  ],
  "globals": {},
  "ordered_globals": [],
  "func_name": "main",
  "heap": {
    "424": [
      "LIST",
      0,
      1,
      2,
      3,
      4,
      5,
      6,
      7,
      8,
      9
    ]
  }
},
```

Parsing heap references

- Matching IDs
- First item is data type
- Repeated list values denoted with “ELIDE”

```
root@4fd5976bd5c7: /code-v x + v
{
  "stdout": "0\n1\n2\n3\n4\n5\n6\n7\n8\n",
  "event": "step_line",
  "line": 6,
  "stack_to_render": [
    {
      "func_name": "main:6",
      "encoded_locals": {
        "i": 9,
        "arr": [
          "REF",
          424
        ]
      },
      "ordered_varnames": [
        "arr",
        "i"
      ],
      "parent_frame_id_list": [],
      "is_highlighted": true,
      "is_zombie": false,
      "is_parent": false,
      "unique_hash": "142",
      "frame_id": 142
    }
  ],
  "globals": {},
  "ordered_globals": [],
  "func_name": "main",
  "heap": {
    "424": [
      "LIST",
      0,
      1,
      2,
      3,
      4,
      5,
      6,
      7,
      8,
      9
    ]
  }
},
```

Upgraded array demo

Other structures

- Arbitrary classes are marked as “INSTANCE”
- Actual field values are not shown

```
root@4fd5c976bd5c7: /code-v x + v
{
  "stdout": "\n\n\n\n\n\n\n\n\n\n\n",
  "event": "step_line",
  "line": 6,
  "stack_to_render": [
    {
      "func_name": "main:6",
      "encoded_locals": {
        "i": 10,
        "linkedList": [
          "REF",
          428
        ]
      },
      "ordered_varnames": [
        "linkedList",
        "i"
      ],
      "parent_frame_id_list": [],
      "is_highlighted": true,
      "is_zombie": false,
      "is_parent": false,
      "unique_hash": "124",
      "frame_id": 124
    }
  ],
  "globals": {},
  "ordered_globals": [],
  "func_name": "main",
  "heap": {
    "428": [
      "INSTANCE",
      "java.util.LinkedList"
    ]
  }
},
{
  "stdout": "\n\n\n\n\n\n\n\n\n\n\n",
  "event": "step_line",
  "line": 10,
  "stack_to_render": [
    {
      "func_name": "main:10",
      "encoded_locals": {
        "linkedList": [

```

Custom LinkedList

- LinkedList compiled at the same time as main()
- All class info accessible by traceprinter

https://cscircles.cemc.uwaterloo.ca/java_visualize/example-code/LinkedList.java

```
root@4fd976bd5c7: /code-v x + v
// named after barrel of monkeys:
// each one hangs on to the next
public class LinkedList {

    // structure of items in list
    class Node {
        // each node knows "next" node
        Node next;
        // and stores a value
        String name;
        // constructor for nodes
        Node(String initialName) {
            name = initialName;
        }
    }

    // beginning of the list, initially empty
    private Node first = null;

    // a demo to create a length-3 list
    public void threeKongs() {
        first = new Node("DK Sr.");
        first.next = new Node("DK");
        first.next.next = new Node("DK Jr.");
    }

    // use a loop to print all
    public void printAll() {
        // a while loop also can work
        for (Node current = first;
            current != null;
            current = current.next) {
            System.out.println(current.name);
        }
    }

    public static void main(String[] args) {
        LinkedList mc = new LinkedList();
        mc.threeKongs();
        mc.printAll();
    }
}
~
~
~
"LinkedList.java" [New] 42L, 1011B written
41,1 All
```


Custom LinkedList

- LinkedList compiled at the same time as main()
- All class info accessible by traceprinter

https://cscircles.cemc.uwaterloo.ca/java_visualize/example-code/LinkedList.java

```
root@4fd976bd5c7: /code-v x + v
"ordered_globals": [],
"func_name": "<init>",
"heap": {
  "426": [
    "INSTANCE",
    "LinkedList",
    [
      "first",
      [
        "REF",
        471
      ]
    ]
  ],
  "471": [
    "INSTANCE",
    "Node",
    [
      "next",
      null
    ],
    [
      "name",
      "DK Sr."
    ]
  ],
  "473": [
    "INSTANCE",
    "Node",
    [
      "next",
      null
    ],
    [
      "name",
      null
    ]
  ]
}
}
"stdout": "",
"event": "step_line",
"line": 12,
"stack_to_render": [
```

Compiling multiple files

- Java source files can be compiled manually
- Classpath manipulation may also be possible

- Data structure definitions don't need to be in the same file as main()

https://github.com/daveagp/java_jail/blob/master/cp/traceprinter/InMemory.java

```
root@4fd976bd5c7: /code-v x + v
c2b.compilerOutput = new StringWriter();
c2b.options = Arrays.asList("-g", "-Xmaxerrs", "1");//, "--classpath", System.getProper
ty("java.class.path"));

DiagnosticCollector<JavaFileObject> errorCollector = new DiagnosticCollector<>();
c2b.diagnosticListener = errorCollector;

/*
For some reason the JVM at Princeton doesn't actually figure out
how to read these particular files off its classpath. So we'll
just throw them all in there manually.
TODO: Optimize and only use files actually referenced by student code.
*/
boolean isPrinceton = System.getProperty("java.class.path").contains("cos126");

if (isPrinceton) {
    String[][] fileinfo = new String[][] {
        {"Stack", getFileContents("cp/visualizer-stdlib/Stack.java")},
        {"Queue", getFileContents("cp/visualizer-stdlib/Queue.java")},
        {"ST", getFileContents("cp/visualizer-stdlib/ST.java")},
        {"StdIn", getFileContents("cp/visualizer-stdlib/StdIn.java")},
        {"StdOut", getFileContents("cp/visualizer-stdlib/StdOut.java")},
        {"Stopwatch", getFileContents("cp/visualizer-stdlib/Stopwatch.java")},
        {mainClass, usercode}
    };
    bytecode = c2b.compileFiles(fileinfo);
}
else {
    // do the normal thing
    bytecode = c2b.compileFile(mainClass, usercode);
}

if (bytecode == null) {
    for (Diagnostic<? extends JavaFileObject> err : errorCollector.getDiagnostics(
))
        if (err.getKind() == Diagnostic.Kind.ERROR) {
            compileError("Error: " + err.getMessage(null), Math.max(0, err.getLine
Number()),
                Math.max(0, err.getColumnNumber()));
            return;
        }
    compileError("Compiler did not work, but reported no ERROR?!?!", 0, 0);
    return;
}
}

163,1 56%
```

2. More GUI

Joshua Hartzfeld



QT Designer

- Group Decision:
 - QT Design studio will be used for the layout and main creation / theme / style of the GUI
 - Animations / visualization / interactions with traceprinter will be handled VIA code

3. Custom data structures

Catherine DiResta



Testing

- Began creating the custom classes
- Coded Multiple Class Java Programs and Multiple Data Structure Java Programs



Testing Todo

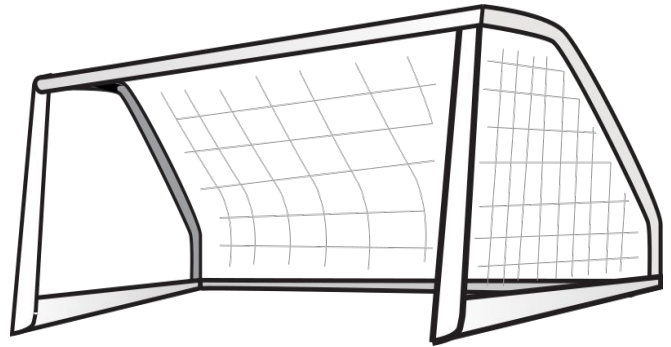
- Create custom classes for all supported data structures



Milestone 4

Goals

- Finish custom data structure implementations
 - LinkedList
 - Tree
 - Graph
- Write a presentable GUI
 - Main window
 - Example widget



Thank You

