Code Visualization

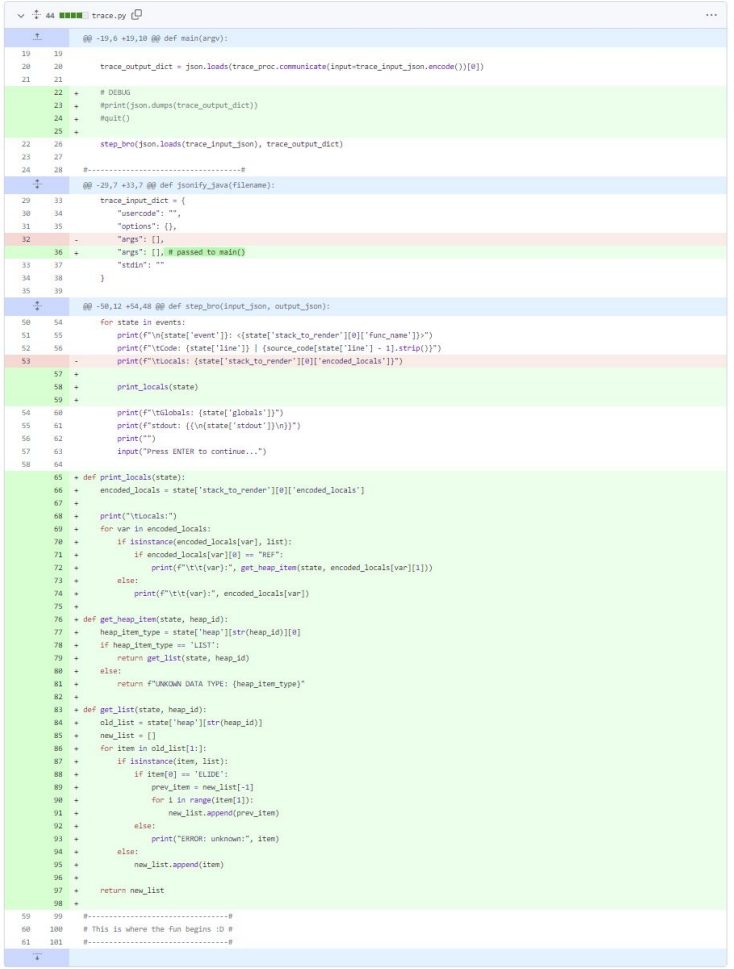
Milestone 3

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

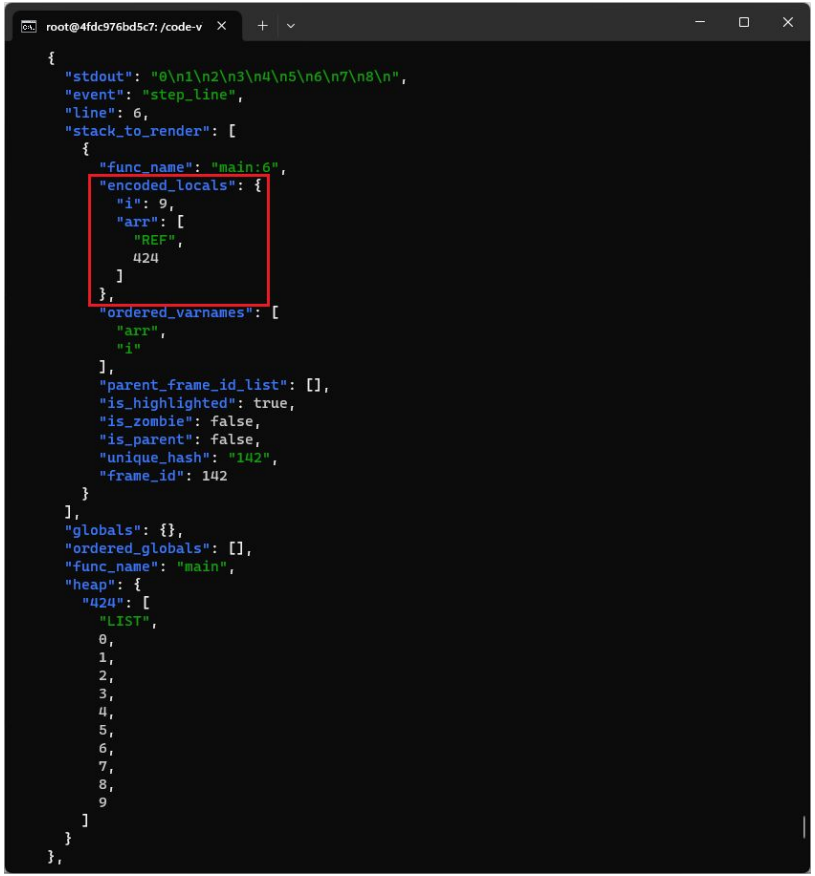


trace.py

```
@@ -19,6 +19,10 @@ def main(argv):

    trace_output_dict = json.loads(trace_proc.communicate(input=trace_input_json.encode())[0])

+   # DEBUG
+   #print(json.dumps(trace_output_dict))
+   #quit()
+
    step_bro(json.loads(trace_input_json), trace_output_dict)

#---------------------------------#
@@ -29,7 +33,7 @@ def jsonify_java(filename):
    trace_input_dict = {
        "usercode": "",
        "options": {},
-       "args": [],
+       "args": [], # passed to main()
        "stdin": ""
    }

@@ -50,12 +54,48 @@ def step_bro(input_json, output_json):
    for state in events:
        print(f"\n{state['event']}: <{state['stack_to_render'][0]['func_name']}>")
        print(f"\tCode: {state['line']} | {source_code[state['line'] - 1].strip()}")
-       print(f"\tLocals: {state['stack_to_render'][0]['encoded_locals']}")

+       print_locals(state)
+

        print(f"\tGlobals: {state['globals']}")
        print(f"stdout: {{\n{state['stdout']}\n}}")
        print("")
        input("Press ENTER to continue...")

+def print_locals(state):
+    encoded_locals = state['stack_to_render'][0]['encoded_locals']
+
+    print("\tLocals:")
+    for var in encoded_locals:
+        if isinstance(encoded_locals[var], list):
+            if encoded_locals[var][0] == "REF":
+                print(f"\t\t{var}:", get_heap_item(state, encoded_locals[var][1]))
+        else:
+            print(f"\t\t{var}:", encoded_locals[var])
+
+def get_heap_item(state, heap_id):
+    heap_item_type = state['heap'][str(heap_id)][0]
+    if heap_item_type == 'LIST':
+        return get_list(state, heap_id)
+    else:
+        return f"UNKOWN DATA TYPE: {heap_item_type}"
+
+def get_list(state, heap_id):
+    old_list = state['heap'][str(heap_id)]
+    new_list = []
+    for item in old_list[1:]:
+        if isinstance(item, list):
+            if item[0] == 'ELIDE':
+                prev_item = new_list[-1]
+                for i in range(item[1]):
+                    new_list.append(prev_item)
+            else:
+                print("ERROR: unknown:", item)
+        else:
+            new_list.append(item)
+
+    return new_list
+
#---------------------------------#
# This is where the fun begins :D #
#---------------------------------#
```

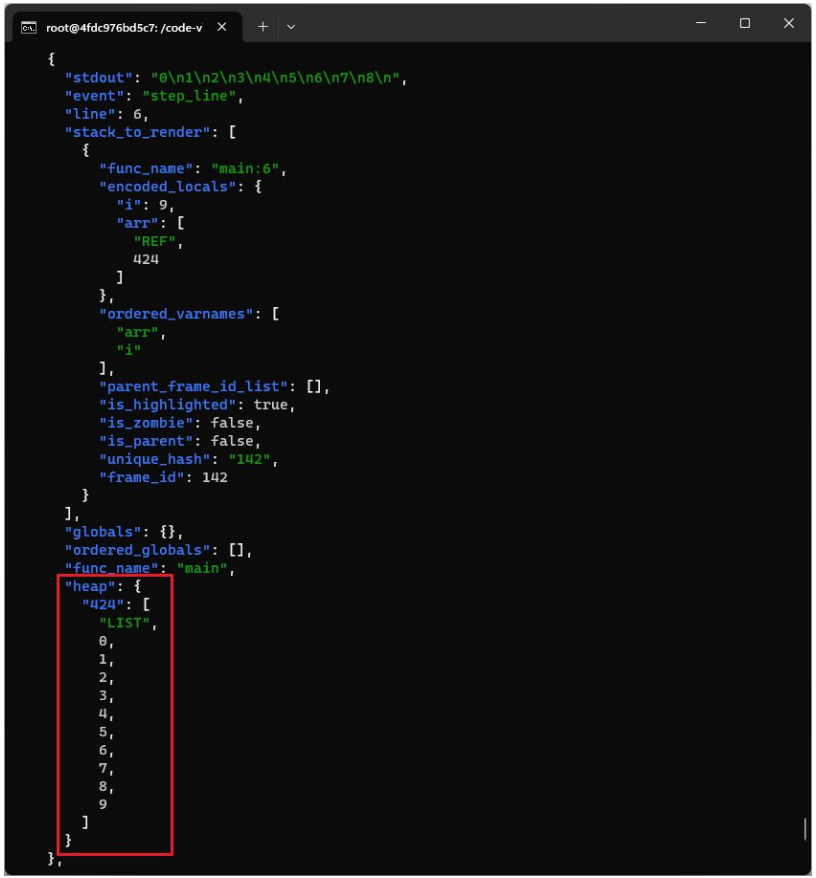
Encoded locals

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



Parsing heap references

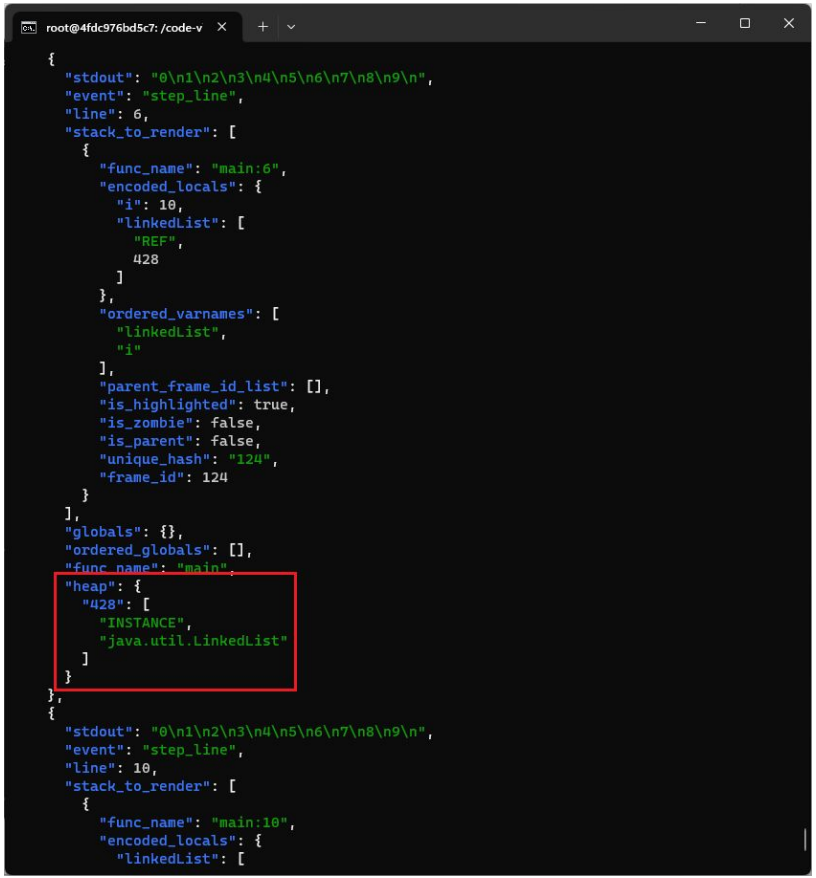
- Matching IDs
- First item is data type
- Repeated list values denoted with “ELIDE”



Upgraded array demo

Other structures

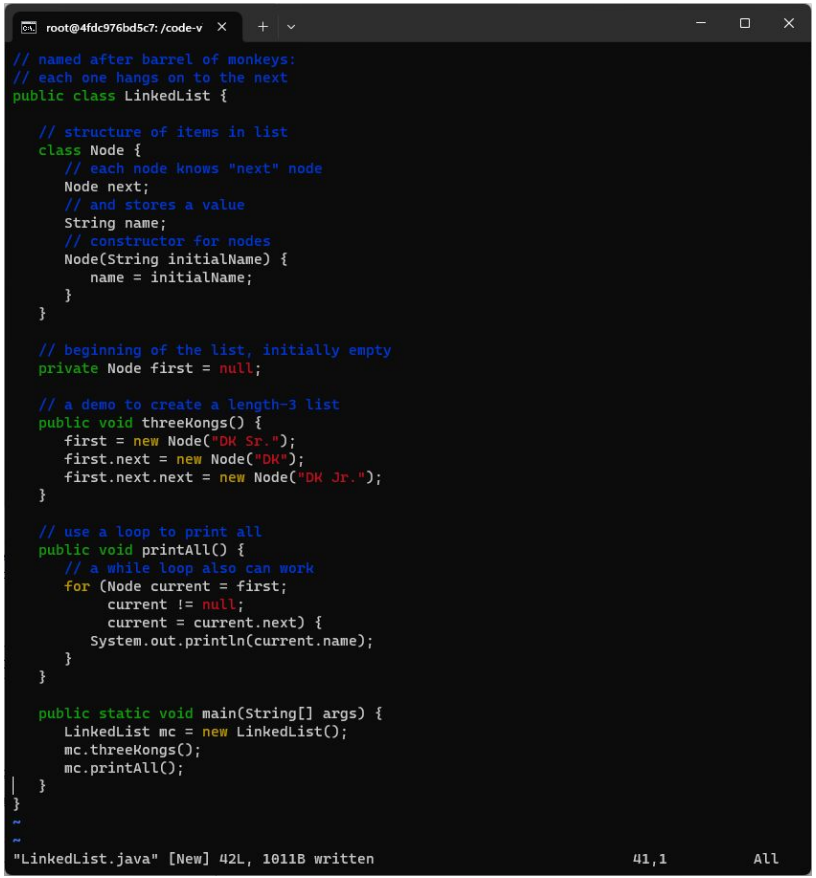
- Arbitrary classes are marked as “INSTANCE”
- Actual field values are not shown



Custom LinkedList

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

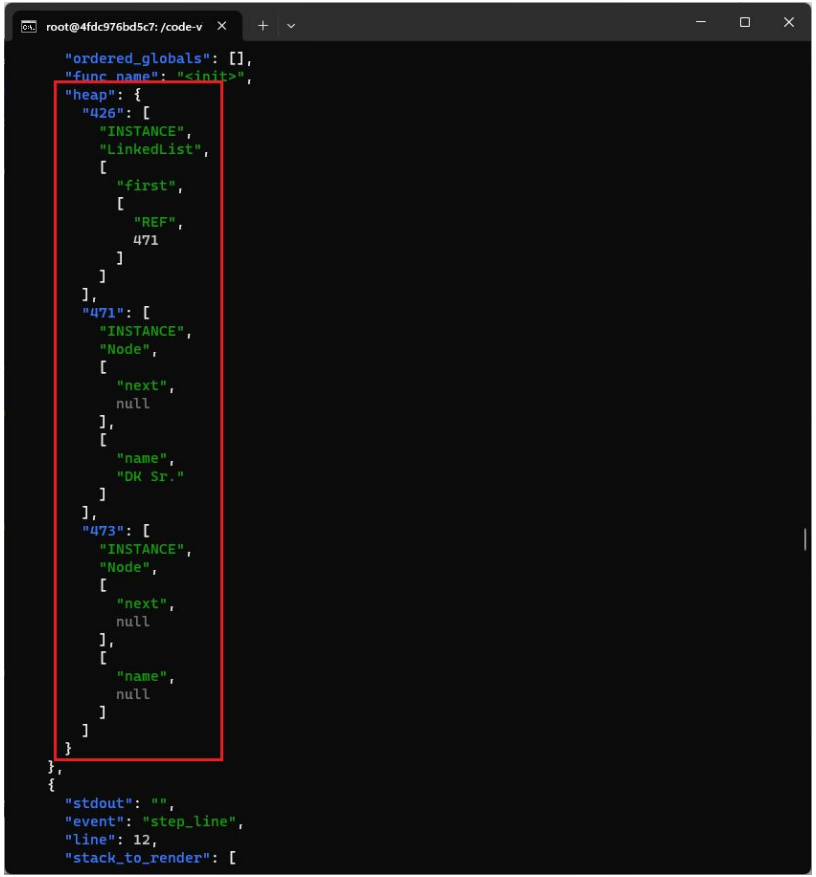
https://cscircles.cemc.uwaterloo.ca/java_visualize/example-code/LinkedList.java



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



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

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

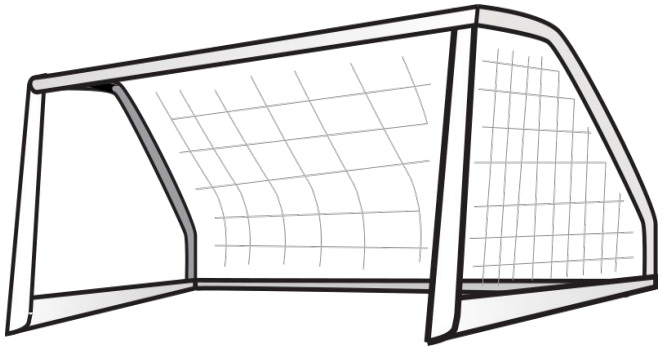
Testing Todo

- 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