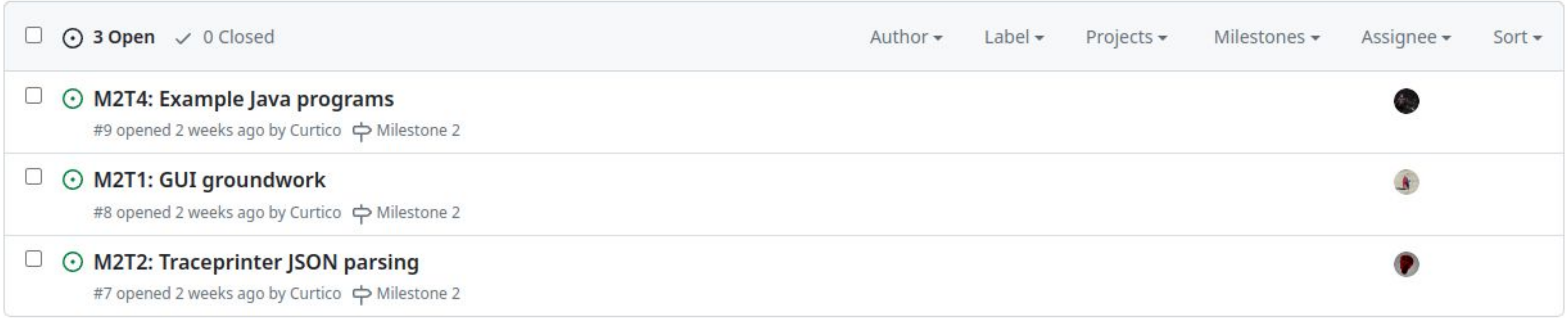
Code Visualization

Milestone 2

Task Calendar



3 Open ✓ 0 Closed

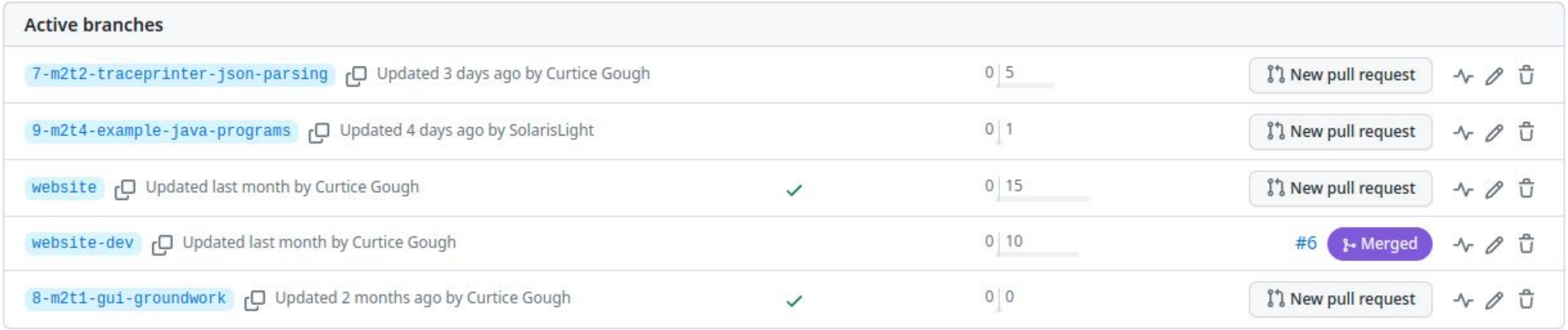
Author ▾ Label ▾ Projects ▾ Milestones ▾ Assignee ▾ Sort ▾

M2T4: Example Java programs
#9 opened 2 weeks ago by Curtico Milestone 2

M2T1: GUI groundwork
#8 opened 2 weeks ago by Curtico Milestone 2

M2T2: Traceprinter JSON parsing
#7 opened 2 weeks ago by Curtico Milestone 2

Git Branches

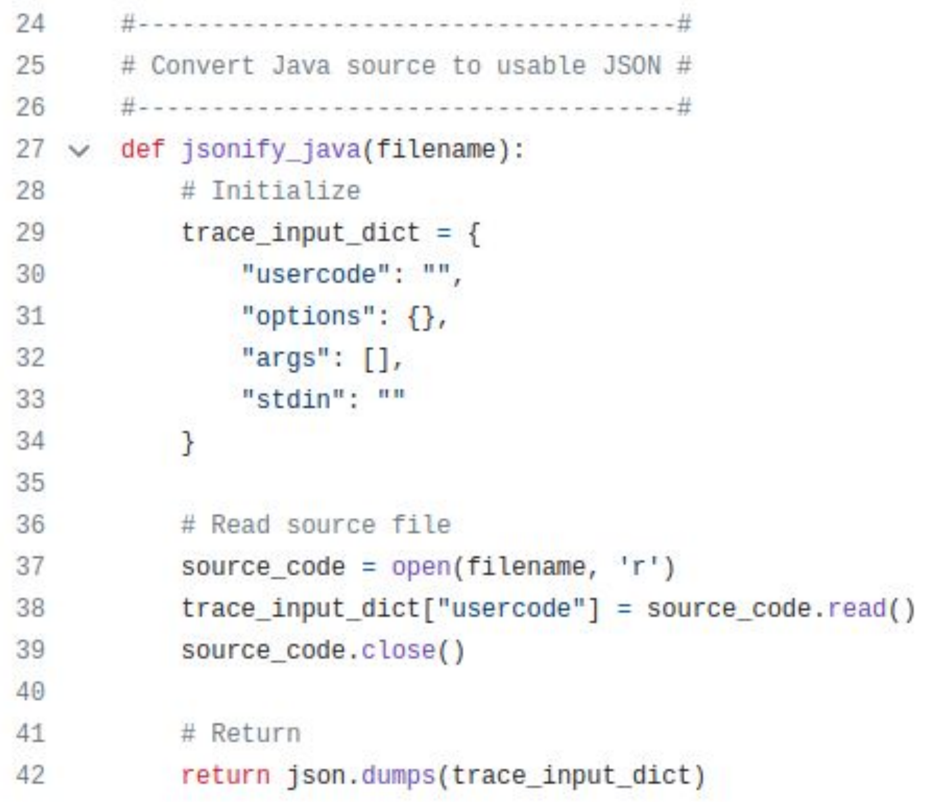


Active branches

7-m2t2-traceprinter-json-parsing	Updated 3 days ago by Curtice Gough		0 | 5	New pull request
9-m2t4-example-java-programs	Updated 4 days ago by SolarisLight		0 | 1	New pull request
website	Updated last month by Curtice Gough	✓	0 | 15	New pull request
website-dev	Updated last month by Curtice Gough		0 | 10	#6 Merged
8-m2t1-gui-groundwork	Updated 2 months ago by Curtice Gough	✓	0 | 0	New pull request

trace.py - `jsonify_json`

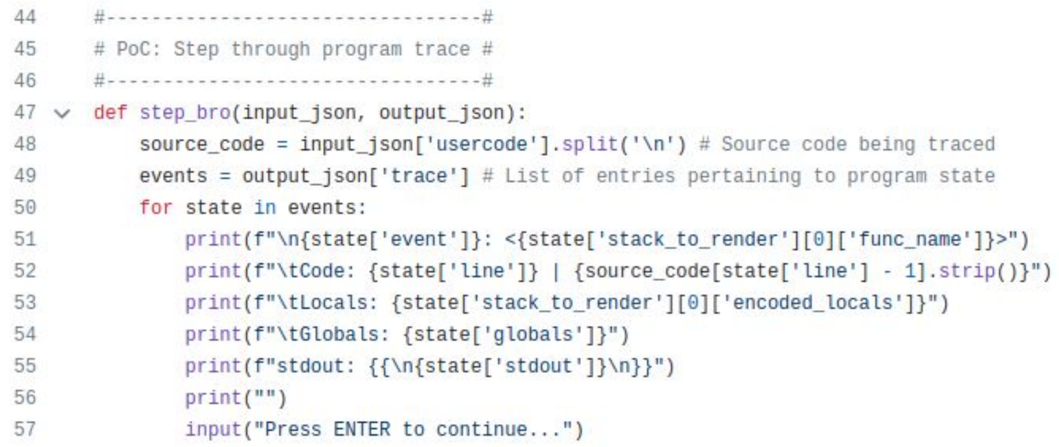
1. Open file
2. Set options
3. Save code



```python
#----------------------------------#
# Convert Java source to usable JSON #
#----------------------------------#
def jsonify_java(filename):
    # Initialize
    trace_input_dict = {
        "usercode": "",
        "options": {},
        "args": [],
        "stdin": ""
    }

    # Read source file
    source_code = open(filename, 'r')
    trace_input_dict["usercode"] = source_code.read()
    source_code.close()

    # Return
    return json.dumps(trace_input_dict)
```

trace.py - `step_bro`

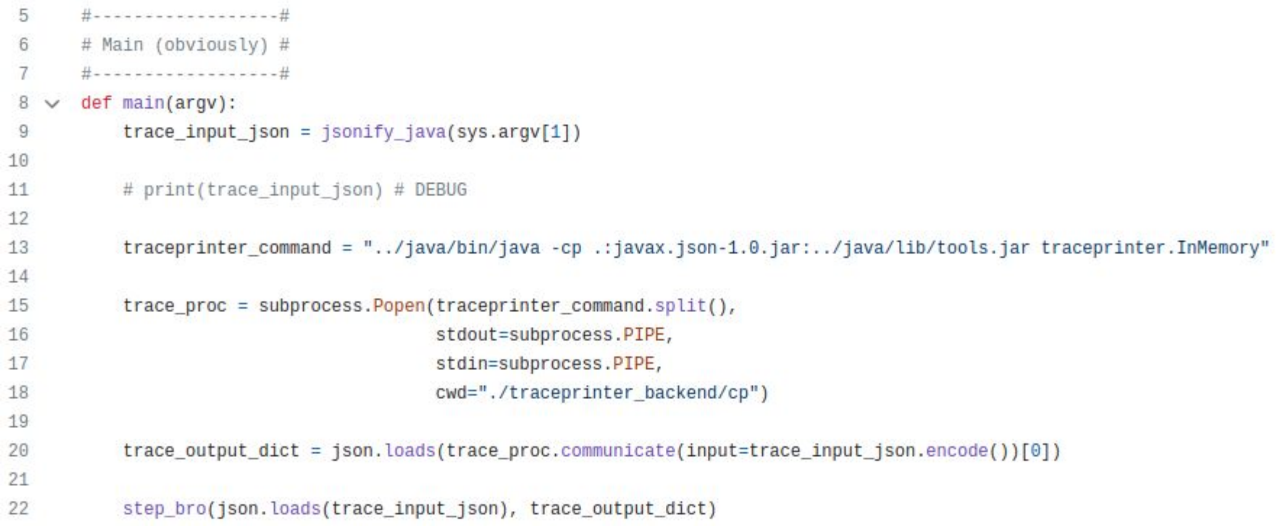
1. Read Traceprinter output
2. Read source code
3. Iterate through program states



```python
#--------------------------------#
# PoC: Step through program trace #
#--------------------------------#
def step_bro(input_json, output_json):
    source_code = input_json['usercode'].split('\n') # Source code being traced
    events = output_json['trace'] # List of entries pertaining to program state
    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(f"\tGlobals: {state['globals']}")
        print(f"stdout: {{\n{state['stdout']}\n}}")
        print("")
        input("Press ENTER to continue...")
```

trace.py - `main`

1. jsonify_java
2. Start subprocess
3. Send input
4. Capture output
5. step_bro

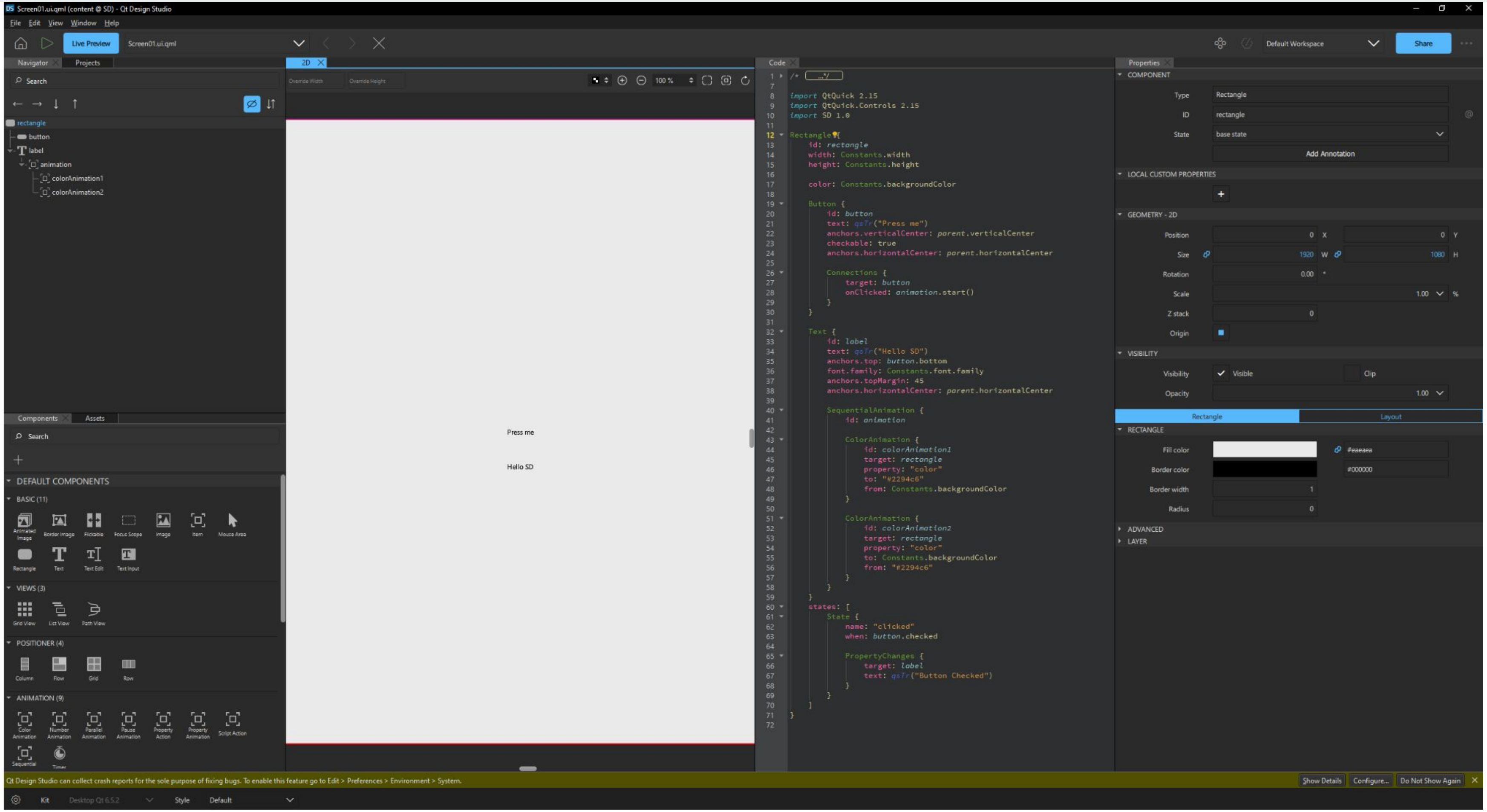


```python
#------------------#
# Main (obviously) #
#------------------#
def main(argv):
    trace_input_json = jsonify_java(sys.argv[1])

    # print(trace_input_json) # DEBUG

    traceprinter_command = "../java/bin/java -cp .:javax.json-1.0.jar:../java/lib/tools.jar traceprinter.InMemory"

    trace_proc = subprocess.Popen(traceprinter_command.split(),
                                  stdout=subprocess.PIPE,
                                  stdin=subprocess.PIPE,
                                  cwd="./traceprinter_backend/cp")

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

    step_bro(json.loads(trace_input_json), trace_output_dict)
```

GUI groundwork

- Change of tools regarding PYQT - now looking into the template engine based QT Design Studio same functionalities different development application.
    - All current progress on the GUI was lost in the transition

- Benefits of new tool
    - Native animation support, template based, support for custom importation of widgets/assets



```
/* ...*/

import QtQuick 2.15
import QtQuick.Controls 2.15
import SD 1.0

Rectangle {
    id: rectangle
    width: Constants.width
    height: Constants.height

    color: Constants.backgroundColor

    Button {
        id: button
        text: qsTr("Press me")
        anchors.verticalCenter: parent.verticalCenter
        checkable: true
        anchors.horizontalCenter: parent.horizontalCenter

        Connections {
            target: button
            onClicked: animation.start()
        }
    }

    Text {
        id: label
        text: qsTr("Hello SD")
        anchors.top: button.bottom
        font.family: Constants.font.family
        anchors.topMargin: 45
        anchors.horizontalCenter: parent.horizontalCenter

        SequentialAnimation {
            id: animation

            ColorAnimation {
                id: colorAnimation1
                target: rectangle
                property: "color"
                to: "#2294c6"
                from: Constants.backgroundColor
            }

            ColorAnimation {
                id: colorAnimation2
                target: rectangle
                property: "color"
                to: Constants.backgroundColor
                from: "#2294c6"
            }
        }
    }
    states: [
        State {
            name: "clicked"
            when: button.checked

            PropertyChanges {
                target: label
                text: qsTr("Button Checked")
            }
        }
    ]
}
```

Type	Rectangle
ID	rectangle
State	base state
Position	0 X 0 Y
Size	1920 W 1080 H
Rotation	0.00 °
Scale	1.00 %
Z stack	0
Visibility	✓ Visible Clip
Opacity	1.00
Fill color	#eaeaea
Border color	#000000
Border width	1
Radius	0

Qt Design Studio can collect crash reports for the sole purpose of fixing bugs. To enable this feature go to Edit > Preferences > Environment > System.

Kit Desktop Qt 6.5.2 Style Default

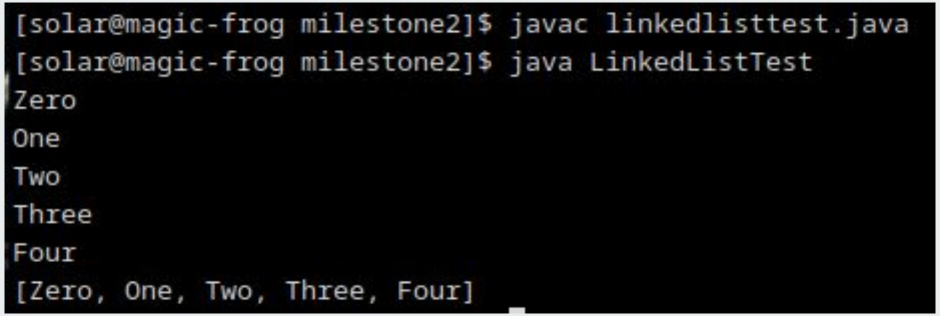
GUI Todo

- Research into custom widgets using QT Designer
- Research into Animation handling
- Research into PYQT5 engine to load templates

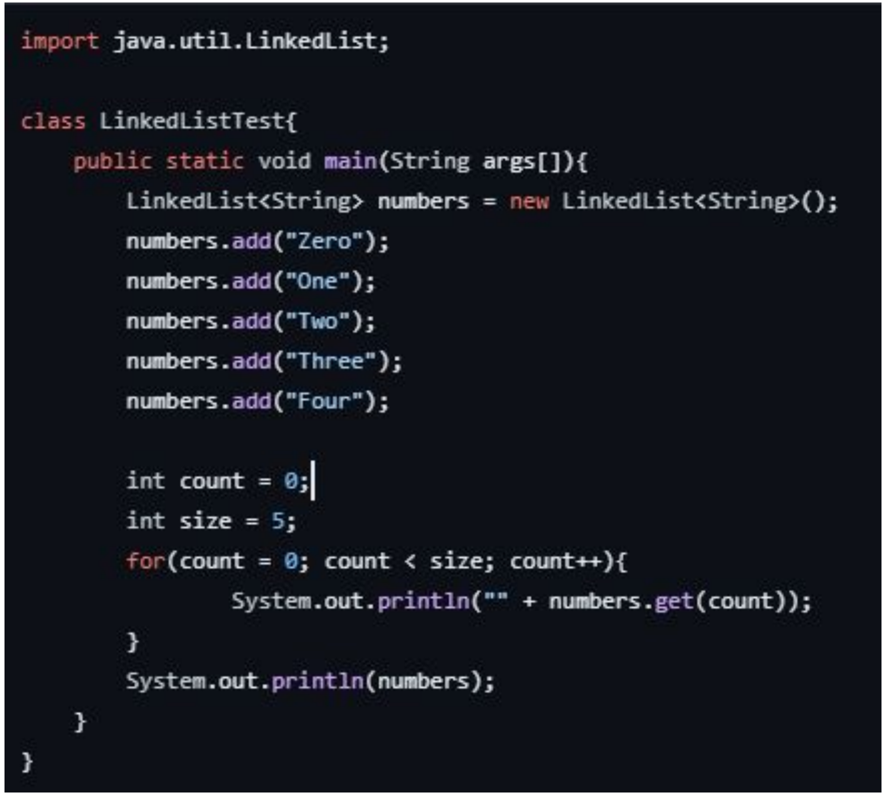
Testing

- Coded programs for Array, ArrayList, LinkedList, Queue and Stack.
- Code was created to test Traceprinter and potential animations in future milestones.
- All programs compiled and ran as expected.

Test Code



```
[solar@magic-frog milestone2]$ javac linkedlisttest.java
[solar@magic-frog milestone2]$ java LinkedListTest
Zero
One
Two
Three
Four
[Zero, One, Two, Three, Four]
```

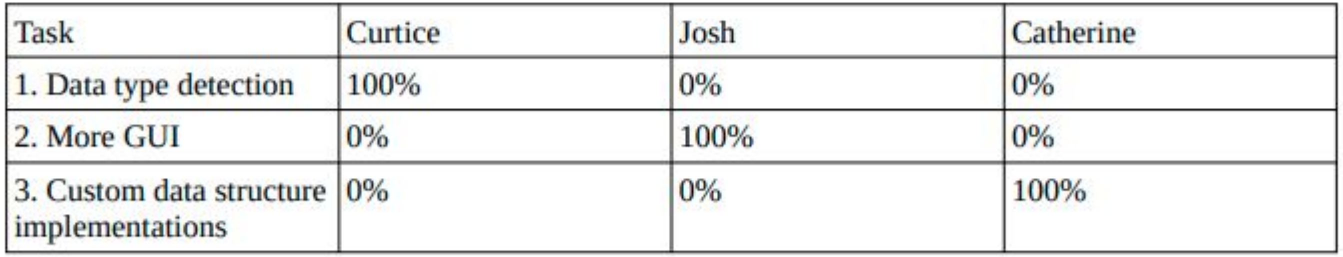


```
import java.util.LinkedList;

class LinkedListTest{
    public static void main(String args[]){
        LinkedList<String> numbers = new LinkedList<String>();
        numbers.add("Zero");
        numbers.add("One");
        numbers.add("Two");
        numbers.add("Three");
        numbers.add("Four");

        int count = 0;
        int size = 5;
        for(count = 0; count < size; count++){
                System.out.println("" + numbers.get(count));
        }
        System.out.println(numbers);
    }
}
```

Testing Todo

- Creating Custom Classes
- Coding Multiple Class Java Programs
- Implement Multiple Data Structure Java Programs

Milestone 3



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%

Thank You