Code Visualization

Milestone 1

Compare and Select Tools

Collaborative Tools

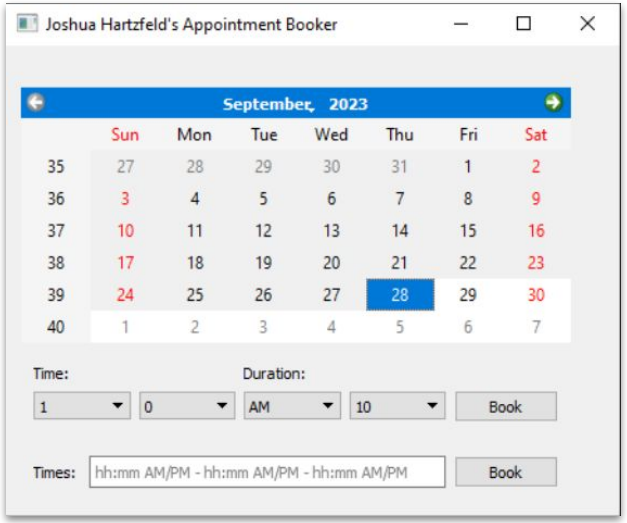
● Version control / task calendar	GitHub
● Documents / presentations	Google Docs / Overleaf
● Communication	Discord

Technical Tools

● Graphical User Interface	PyQt5
● Backend / code tracing	Traceprinter
● Target programming language	Java 8

“Hello World” Demos

Graphical User Interface PyQt5



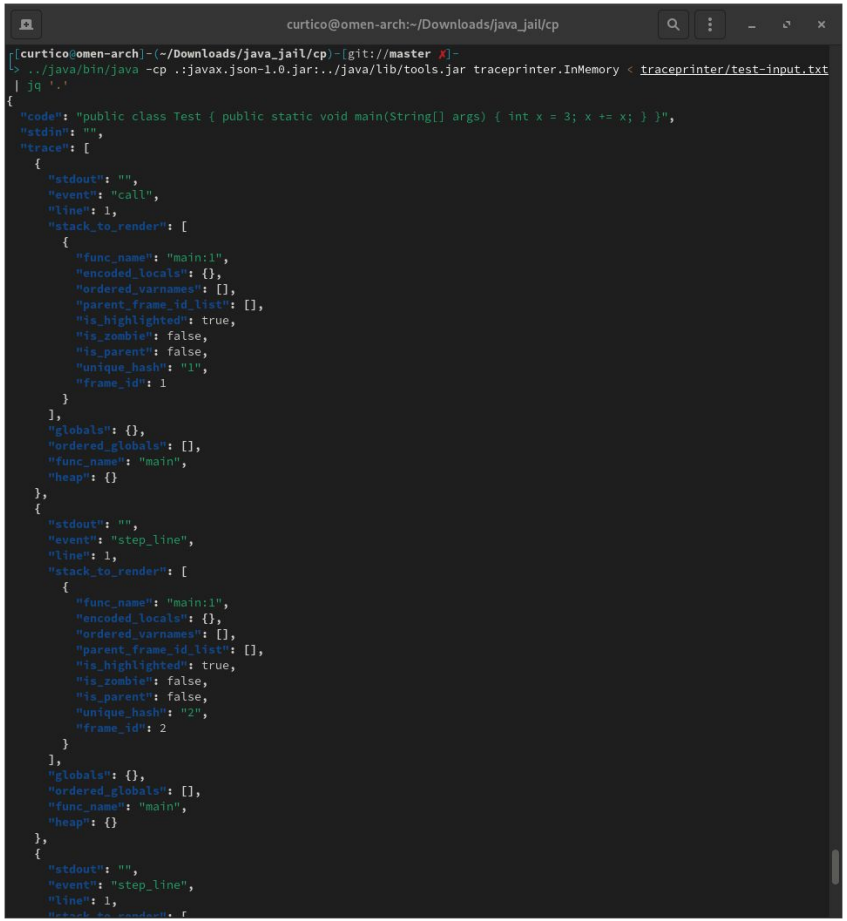

```python
class Window(QWidget):
    def __init__(self, parent=None):
        super(Window, self).__init__(parent)
        self.current_appointments = ["6:30am -7:00am", "7:00am -7:15am",
                                     "7:20am -7:30 am", "8:00am -8:25 am"]

        # Generic hooHaa

        durations = ['10', '15', '20', '25', '30']
        meridiems = ['AM', 'PM']
        hours = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12']
        minutes = [str(x) for x in range(60)]
        self.setWindowTitle("Joshua Hartzfeld's Appointment Booker")

        # Top Date Bar
        self.top_label = QLabel()

        # Appointment Bookings Frame
        self.current_Bookings = QFrame()
        self.booking_grid = QGridLayout()
        self.current_Bookings.setLayout(self.booking_grid)

        # calendar
        self.calendar = QCalendarWidget()
        self.calendar.setMinimumDate(datetime.datetime.now())

        # Single appointment frame
        self.booking_Frame = QFrame()
        frame_grid = QGridLayout()
        self.hour = QComboBox()
        self.hour.addItems(hours)
        self.minute = QComboBox()
        self.minute.addItems(minutes)
        self.am_pm = QComboBox()
        self.am_pm.addItems(meridiems)
        self.duration_box = QComboBox()
        self.duration_box.addItems(durations)
        self.booking_Frame.setLayout(frame_grid)
        self.frame_push_button = QPushButton()
```

Backend / Code Tracing - Traceprinter

```
┌[curtico@omen-arch]-(~/Downloads/java_jail/cp)-[git://master ✗]-
└> cat traceprinter/test-input.txt
{
"usercode":
 "public class Test { public static void main(String[] args) { int x = 3; x += x; } }",
"options": {},
"args": [],
"stdin": ""
}
```

Backend / Code Tracing

Traceprinter



Requirements / SRS

Interfaces

- Graphical User Interface
  - Data Structures View
  - Source View
  - Structures List
- Java Traceprinter Interface
  - Convert code to usable JSON input
  - Receive JSON output of traced code

Supported Data Structures

- Array
- java.util.ArrayList
- java.util.LinkedList
- java.util.Queue
- java.util.Stack
- codeviz.structures.Tree
- codeviz.structures.BinaryTree
- codeviz.structures.Graph

Functional Requirements

- The system shall allow users to paste/input Java code for analysis
- The system shall spawn an instance of Traceprinter as a child process
- The system shall correctly parse JSON output from Traceprinter
- The system shall visualize pre-defined detected data structures in the GUI
- The system shall visually animate the movement of data between data structures
- The system shall allow users to re-type and re-name data structure diagrams
- The system shall represent code execution line-by-line at the user's discretion

Design

GUI Layout

Main Features:

- **Data Structures View -** visualization aspect
- **Source View -** section of GUI for user to see code step by step
- **Structures List View** - Section of GUI to allow user to choose what data structure to visualize

Additional Key Features:

- **View Toggle Bar** - A bar with toggles to hide or display one of the views listed above
- Other Features will be added when deemed essential

Main Design 1

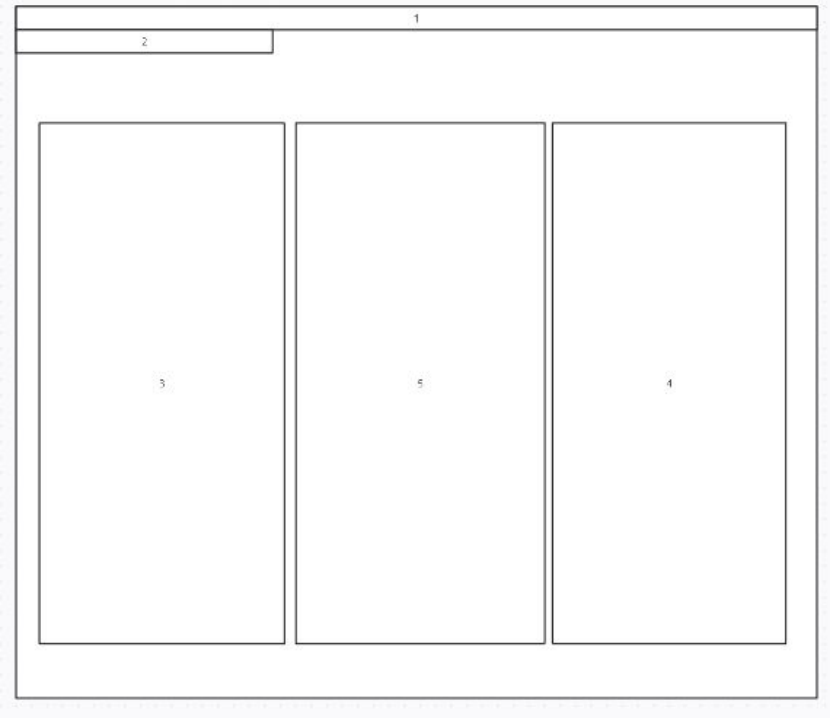
1 - top window bar

2 - Toggle View bar

3- Source Code input/display

4- Structures List

5- Visualization area



Main Design 2 (for lists)

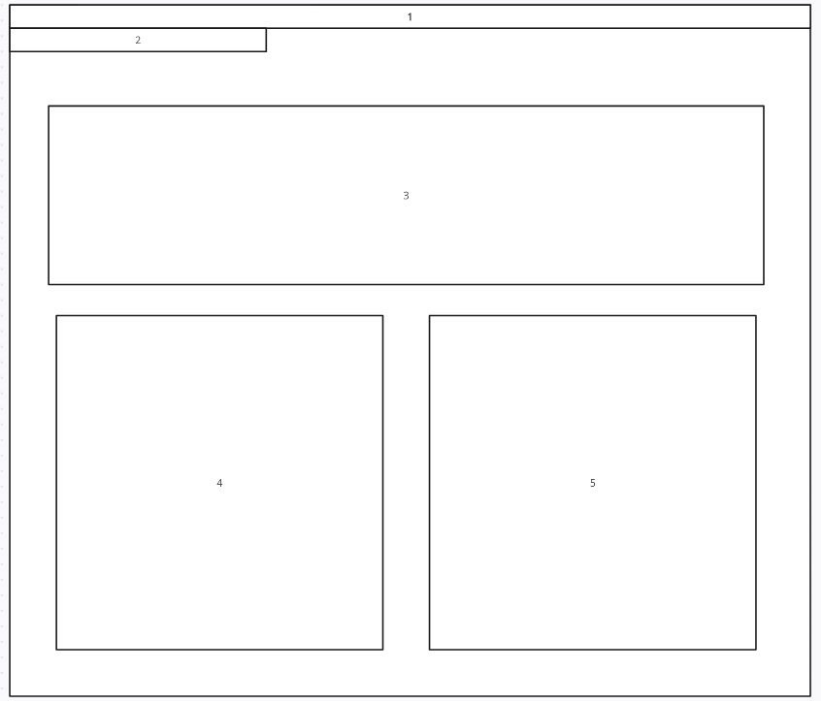
1 - top window bar

2 - Toggle View bar

3- Source Code input/display

4- Structures List

5- Visualization area



Test Plan

Interfaces

- Data Structure View
  - Verify that the data structure properly shows within the Visualization area.
- Source View
  - Test that the user is able to type their Java code within this area and they are able to scroll through the code.
- Structures List
  - Ensure it list all the data structures within source code and that the user is able to rename or redefine it.

Functional Requirements

- Source Code Input
  - Test that the user is able to both type or copy and paste Java code into the Source View
- Traceprinter Subprocess
  - Verify that software launches an instance of Traceprinter that has correct arguments, sends source code as input and receives JSON output.
- JSON Trace Parsing
  - Verify the parsing of the JSON data received from the Traceprinter Subprocess.
- Structure Visualization
  - Test that the data structures are correctly detected and visualized in the GUI

Functional Requirements

- Animations
  - Confirm the movement of data is accurately represented in the Data Structure View when the code trace happens
- Re-Naming
  - Test that the user is able to redefine or rename a structure
- Line-By-Line Execution
  - Verify the representation of each program state.

Milestone 2

GUI Groundwork

- Construct main GUI window
  - Prioritize Data Structures View
- Set up basic layout
- Begin implementing custom PyQt5 Widget
  - Each diagram will be an instance of this widget

Primary team member assigned: Joshua Hartzfeld

GUI Testing

- Test each element as it’s developed
- Follow Test Plan
- Multiple environments
    - Windows
    - MacOS
    - Linux

Primary team member assigned:

Equal parts each member

Example Java Programs

- Used for testing Traceprinter interface
- Used for testing data visualization / animation
- Shows users examples of valid programs

Primary team member assigned: Catherine DiResta

Traceprinter JSON Parsing

- Convert JSON output to Python dictionary
- Implement main state loop
  - CLI only for now
  - Print info on each structure at each point in execution

Thank You