MemStep: An Interactive Tool for Constructing and Visualizing the Run-Time Memory Layout of Java Programs

Michelle Le Pham, Anna Nguyen, and Rebecca Schreib
Motivation

Novices often hold misconceptions about the run-time behavior of programs
Motivation

Novices often hold misconceptions about the **run-time behavior** of programs.

Simply **writing** programs isn’t sufficient to help novices overcome these misconceptions.
Motivation

Novices often hold misconceptions about the **run-time behavior** of programs.

Simply **writing** programs isn’t sufficient to help novices overcome these misconceptions.

We must deliberately teach **conceptual models** of **program execution** and **memory layout**!
Goals

Design a tool that will help students to develop accurate conceptual models of program execution and the run-time layout of memory.
Goals

Design a tool that will help students to develop accurate conceptual models of program execution and the run-time layout of memory.

Desired features:

1. **Visualization:** Students can visualize the program’s run-time memory layout.
Goals

Design a tool that will help students to develop accurate **conceptual models** of **program execution** and the **run-time layout of memory**

**Desired features:**

1. **Visualization**: Students can visualize the program’s run-time memory layout
2. **Custom Code**: Students can trace the execution of arbitrary (Java) programs
Goals

Design a tool that will help students to develop accurate conceptual models of program execution and the run-time layout of memory

Desired features:

1. **Visualization**: Students can visualize the program’s run-time memory layout
2. **Custom Code**: Students can trace the execution of arbitrary (Java) programs
3. **Interactivity**: Students actively construct the run-time memory layout
Goals

Design a tool that will help students to develop accurate conceptual models of program execution and the run-time layout of memory

Desired features:

1. **Visualization**: Students can visualize the program’s run-time memory layout
2. **Custom Code**: Students can trace the execution of arbitrary (Java) programs
3. **Interactivity**: Students actively construct the run-time memory layout
4. **Feedback**: Students are given targeted formative feedback when they err
Goals

Design a tool that will help students to develop accurate conceptual models of program execution and the run-time layout of memory

Desired features:

1. **Visualization**: Students can visualize the program’s run-time memory layout
2. **Custom Code**: Students can trace the execution of arbitrary (Java) programs
3. **Interactivity**: Students actively construct the run-time memory layout
4. **Feedback**: Students are given targeted formative feedback when they err
5. **Realistic model**: Students interact with a notional machine that uses a relatively low level of abstraction
User Workflow

```java
1 public class Main {
2     public static void f() {
3         Integer a = 5;
4         int b = a;
5         Integer[] c = new Integer[]{3, a};
6         MemObj d = new MemObj();
7     }
8 }
```
User Workflow

```java
public class Main {
    public static void f() {
        Integer a = 5;
        int b = a;
        Integer[] c = new Integer[] {3, a};
        MemObj d = new MemObj();
    }
```
User Workflow

```java
public class Main {
    public static void f() {
        Integer a = 5;
        int b = a;
        Integer[] c = new Integer[]{3, a};
        MemObj d = new MemObj();
    }
}
```
Settings

Base Address

0x1000
Submit

Initialize local vars?
Off [ ] On

Reference Size
4 bytes [ ] 8 bytes

Require alignment?
Off [ ] On

Object Alignment Size
4 bytes [ ] 8 bytes
User Workflow

```java
public class Main {
    public static void f() {
        Integer a = 5;
        int b = a;
        Integer[] c = new Integer[]{3, a};
        MemObj d = new MemObj();
    }
}
```
User Workflow

```java
public class MemObj {
    private long val;
    private MemObj obj;

    public MemObj() {
        this.val = 3L;
    }

    public MemObj(MemObj obj) {
        this.obj = obj;
    }

    public void setObject(MemObj obj) {
        this.obj = obj;
    }
}
```
User Workflow

```java
public class Main {
    public static void f() {
        Integer a = 5;
        int b = a;
        Integer[] c = new Integer[]{3, a};
        MemObj d = new MemObj();
    }
}
```
public class Main {
    public static void f() {
        Integer a = 5;
        int b = a;
        Integer[] c = new Integer[]{3, a};
        MemObj d = new MemObj();
    }
}

Please specify the entry point for execution. The entry point should be a static method that takes zero arguments.

Class name:
Enter the name of the class that will serve as the entry point

Method name:
Enter the name of the method that will serve as the entry point

Submit
```java
public class Main {
    public static void f() {
        Integer a = 5;
        int b = a;
        Integer[] c = new Integer[]{3, a};
        MemObj d = new MemObj();
    }
}
```
public class Main {
    public static void f() {
        Integer a = 5;
        int b = a;
        Integer[] c = new Integer[]{3, a};
        MemObj d = new MemObj();
    }
}

Please specify the entry point for execution. The entry point should be a static method that takes zero arguments.

Class name:
Main

Method name:
f

Submit
```java
public class Main {

    public static void f() {
        Integer a = 5;
        int b = a;
        Integer[] c = new Integer[]{3, a};
        MemObj d = new MemObj();
    }
}
```
public class Main {
    public static void f() {
        Integer a = 5;
        int b = a;
        Integer[] c = new Integer[]{3, a};
        MemObj d = new MemObj();
    }
}
public class Main {
    public static void f() {
        Integer a = 5;
        int b = a;
        Integer[] c = new Integer[]{3, a};
        MemObj d = new MemObj();
    }
}
public class Main {
    public static void f() {
        Integer a = 5;
        int b = a;
        Integer[] c = new Integer[] {3, a};
        MemObj d = new MemObj();
    }
}
```java
public class Main {
    public static void f() {
        Integer a = 5;
        int b = a;
        Integer[] c = new Integer[]{3, a};
        MemObj d = new MemObj();
    }
}
```
```java
public class Main {
    public static void f() {
        Integer a = 5;
        int b = a;
        Integer[] c = new Integer[]{3, a};
        MemObj d = new MemObj();
    }
```
```java
public class Main {
    public static void f() {
        Integer a = 5;
        int b = a;
        Integer[] c = new Integer[]{3, a};
        MemObj d = new MemObj();
    }
```
public class Main {
    public static void f() {
        Integer a = 5;
        int b = a;
        Integer[] c = new Integer[]{3, a};
        MemObj d = new MemObj();
    }
}
public class Main {
    public static void f() {
        Integer a = 5;
        int b = a;
        Integer[] c = new Integer[]{3, a};
        MemObj d = new MemObj();
    }
}
```java
public class Main {
    public static void f() {
        Integer a = 5;
        int b = a;
        Integer[] c = new Integer[]{3, a};
        MemObj d = new MemObj();
    }
}
```
This line of code does add a variable to the stack, but there is at least one other operation that should be performed first. In particular, consider whether there is an object that should be allocated on the heap first. For example, if this statement uses a primitive value in a location where a wrapper type is declared, the primitive value needs to be autoboxed (converted to an instance of the corresponding wrapper class).
This line of code does add a variable to the stack, but there is at least one other operation that should be performed first. In particular, consider whether there is an object that should be allocated on the heap first. For example, if this statement uses a primitive value in a location where a wrapper type is declared, the primitive value needs to be autoboxed (converted to an instance of the corresponding wrapper class).
public class Main {
    public static void f() {
        Integer a = 5;
        int b = a;
        Integer[] c = new Integer[]{3, a};
        MemObj d = new MemObj();
    }
}
public class Main {
    public static void f() {
        Integer a = 5;
        int b = a;
        Integer[] c = new Integer[]{3, a};
        MemObj d = new MemObj();
    }
}
public class Main {
    public static void f() {
        Integer a = 5;
        int b = a;
        Integer[] c = new Integer[] {3, a};
        MemObj d = new MemObj();
    }
}
Integer a = 5;
int b = a;
Integer[] c = new Integer[3, a];
MemObj d = new MemObj();
Correct! Next, you need to provide preliminary information about each field.

Please specify the first field.

Starting address of the field:

0x1000

Size of the field:

4

Initial value of the field:

5

Submit
Correct! Next, you need to provide preliminary information about each field.

Please specify the first field.

Starting address of the field:
0x1000

Size of the field:
4

Initial value of the field:
5

Submit
```java
public class Main {
    public static void f() {
        Integer a = 5;
        int b = a;
        Integer[] c = new Integer[]{3, a};
        MemObj d = new MemObj();
    }
}
```
```java
public class Main {
    public static void f() {
        Integer a = 5;
        int b = a;
        Integer[] c = new Integer[3, a];
        MemObj d = new MemObj();
    }
}
```
public class Main {
    public static void f() {
        Integer a = 5;
        int b = a;
        Integer[] c = new Integer[] {3, a};
        MemObj d = new MemObj();
    }
}
```java
public class Main {
    public static void f() {
        Integer a = 5;
        int b = a;
        Integer[] c = new Integer[]{3, a};
        MemObj d = new MemObj();
    }
}
```
```java
public class Main {
    public static void f() {
        Integer a = 5;
        int b = a;
        Integer[] c = new Integer[]{3, a};
        MemObj d = new MemObj();
    }
}
```

This operation will add a new local variable to the current (top) stack frame. Please specify the variable to be added.

Variable's name:
```
a
```

Variable's value:
```
0x1000
```

Submit
public class Main {
    public static void f() {
        Integer a = 5;
        int b = a;
        Integer[] c = new Integer[] { 3, a };
        MemObj d = new MemObj();
    }
}
```java
public class Main {
    public static void f() {
        Integer a = 5;
        int b = a;
        Integer[] c = new Integer[]{3, a};
        MemObj d = new MemObj();
    }
}
```
public class Main {
    public static void f() {
        Integer a = 5;
        int b = a;
        Integer[] c = new Integer[]{3, a};
        MemObj d = new MemObj();
    }
}
public class Main {
    public static void f() {
        int b = a;
        Integer[] c = new Integer[]{3, a};
        MemObj d = new MemObj();
    }
}

Congratulations! You've correctly constructed the state of the stack and the heap for this program. If you'd like to try a different program, you can press the "Stop exercise" button and you will be given the opportunity to edit the code.
Research Questions

1. Is using MemStep more or less effective at improving student learning when compared to traditional learning activities?
Research Questions

1. Is using MemStep more or less effective at improving student learning when compared to traditional learning activities?

2. Is using MemStep more or less engaging and satisfying for students when compared to traditional learning activities?
Research Questions

1. Is using MemStep more or less effective at improving student learning when compared to traditional learning activities?

2. Is using MemStep more or less engaging and satisfying for students when compared to traditional learning activities?

3. How effective is MemStep at sparking technical curiosity?

From the CS2023 curriculum:

“Technical Curiosity: Students must develop interest in understanding how programs are executed, how programs and data are stored in memory, etc.”
Experimental Methodology

- Population: 248 students in a second-year program design course
- Random assignment of students to control & experimental groups
Experimental Methodology

- **Population:** 248 students in a second-year program design course
- **Random assignment** of students to control & experimental groups

- **Training exercises** (graded, outside of class)
  - All students watched the same video lecture (~45 min)
  - Experimental group completed four exercises using MemStep
  - Control group completed *same* four exercises in a worksheet
Experimental Methodology

- **Population:** 248 students in a second-year program design course
- **Random assignment** of students to control & experimental groups

- **Training exercises** (graded, outside of class)
  - All students watched the same video lecture (~45 min)
  - Experimental group completed four exercises using MemStep
  - Control group completed same four exercises in a worksheet

- **Test exercise** (ungraded, during class)
  - All students completed four exercises in a worksheet
Experimental Methodology

- Post-study survey
  - All students rated agreement with statements about the training
  - Experimental group provided qualitative feedback
Experimental Methodology

● Post-study survey
  ○ All students rated agreement with statements about the training
  ○ Experimental group provided qualitative feedback

● End-of-semester survey
  ○ All students who used MemStep while preparing for test 4 rated agreement with statements about its helpfulness
Experimental Methodology

● Post-study survey
  ○ All students rated agreement with statements about the training
  ○ Experimental group provided qualitative feedback

● End-of-semester survey
  ○ All students who used MemStep while preparing for test 4 rated agreement with statements about its helpfulness

● When applicable, we compared the experimental and control groups using a two-tailed unpaired t-test
Post-Test Results

n = 115 per group

Control:
Max: 49.5
Mean: 34.4
Min: 0

Experimental:
Max: 50
Mean: 36.7
Min: 1.5

p-value: 0.091
Cohen's d: 0.22
**Post-Survey Results**

n = 115 per group

Please rate the extent to which you agree with the following statements regarding the version of the Memory assignment that you completed in advance of today's class.

<table>
<thead>
<tr>
<th>Statement</th>
<th>Ctrl.</th>
<th>Exp.</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>The assignment was <strong>engaging</strong>; it captured my attention and I felt motivated to progress through the exercises.</td>
<td>3.22</td>
<td>4.24</td>
<td>&lt; 0.00001</td>
</tr>
<tr>
<td>The assignment was <strong>interactive</strong>; I felt like I was actively involved in an exchange of information that gradually deepened my understanding of the material.</td>
<td>3.11</td>
<td>4.62</td>
<td>&lt; 0.00001</td>
</tr>
<tr>
<td>The assignment was <strong>satisfying</strong> to complete; I felt a sense of accomplishment when I completed it.</td>
<td>3.44</td>
<td>4.30</td>
<td>&lt; 0.00001</td>
</tr>
</tbody>
</table>

1 = Strongly Disagree   ←   5 = Strongly Agree
Qualitative Feedback

What was your overall impression of MemStep?

<table>
<thead>
<tr>
<th>Overall Tone</th>
<th>Freq.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>105</td>
</tr>
<tr>
<td>Neutral</td>
<td>5</td>
</tr>
<tr>
<td>Negative</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Theme</th>
<th>Freq.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Helpful/effective for learning</td>
<td>58</td>
</tr>
<tr>
<td>Liked the feedback</td>
<td>24</td>
</tr>
<tr>
<td>Interactive/hands on</td>
<td>23</td>
</tr>
<tr>
<td>Fun/enjoyable</td>
<td>22</td>
</tr>
<tr>
<td>Easy to use</td>
<td>15</td>
</tr>
<tr>
<td>Well-designed</td>
<td>14</td>
</tr>
<tr>
<td>Engaging</td>
<td>14</td>
</tr>
</tbody>
</table>
End-of-Semester Survey Results

Did you use MemStep at all when preparing for test 4?

- Yes (65.4%)
- No (34.6%)

n = 237
# End-of-Semester Survey Results

Please rate the extent to which you agree with each of the following statements regarding your use of MemStep to prepare for test 4.

| Statement                                                                .MULTICELL| Avg. Rating |
|---------------------------------------------------------------------------|------------|-------------|
| Using MemStep helped me to **deepen my understanding** of how data gets laid out in memory during the execution of a Java program. |             | 4.61        |
| Using MemStep made it **easier to create** and walk through additional **practice problems** in preparation for the test. |             | 4.42        |
| Using MemStep helped to **prepare** me for similar problems on the test. |             | 4.50        |

1 = Strongly Disagree  ↔  5 = Strongly Agree

n = 155
### End-of-Semester Survey Results

Please rate the extent to which you agree with each of the following statements regarding your use of MemStep to prepare for test 4.

<table>
<thead>
<tr>
<th>Having the ability to walk through arbitrary Java programs using MemStep made me <strong>curious</strong> to try different programs in order <strong>to see how different types of data get laid out</strong> in memory</th>
<th>Avg. Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Having the ability to walk through arbitrary Java programs using MemStep made me <strong>curious</strong> to try different programs in order <strong>to see what different different steps happen</strong> under the hood during execution.</td>
<td>3.87</td>
</tr>
</tbody>
</table>

1 = Strongly Disagree   ➡️   5 = Strongly Agree
Conclusion

- MemStep actively engages students in the construction and visualization of the run-time layout of memory
Conclusion

- MemStep **actively** engages students in the construction and visualization of the run-time layout of **memory**

- Students who used MemStep to learn…
  - Performed **at least as well** as those who used a worksheet
  - Perceived MemStep as more **engaging** and **satisfying** to use
  - Typically agreed that MemStep sparked **technical curiosity**
Conclusion

- MemStep **actively** engages students in the construction and visualization of the run-time layout of **memory**

- Students who used MemStep to learn…
  - Performed **at least as well** as those who used a worksheet
  - Perceived MemStep as more **engaging** and **satisfying** to use
  - Typically agreed that MemStep sparked **technical curiosity**

- MemStep makes it easy for students & **instructors** to create infinite new practice problems!
Questions?

Try out MemStep: cs.rice.edu/~rjs7/courseware
Contact if you have questions: rjs@rice.edu