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

Michelle Le Pham, Anna Nguyen, and Rebecca Schreib

RICE

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

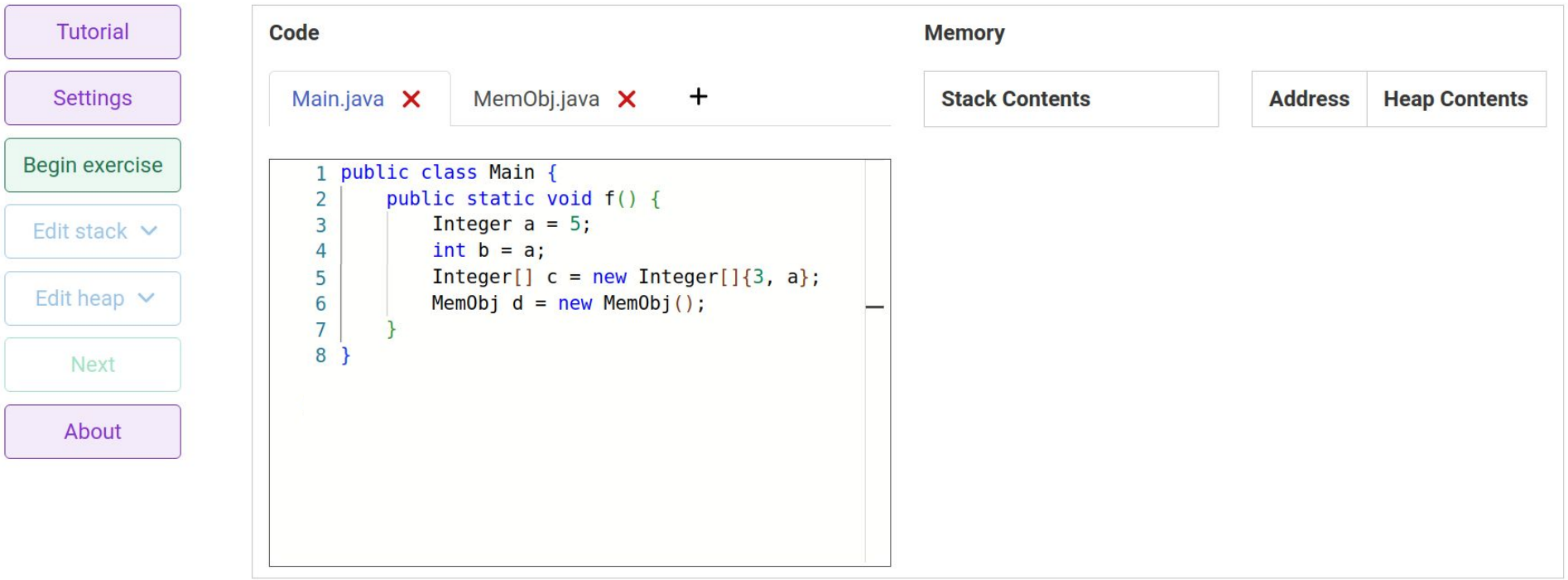
Goals

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

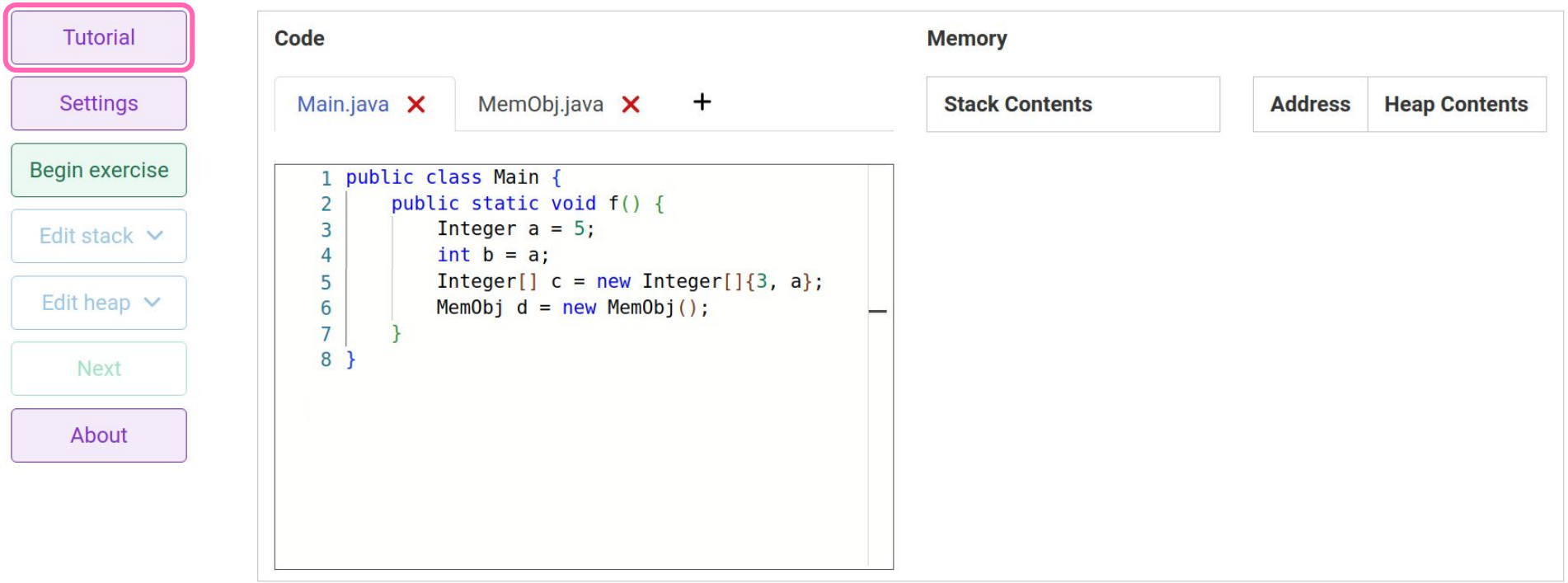
Desired features:

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

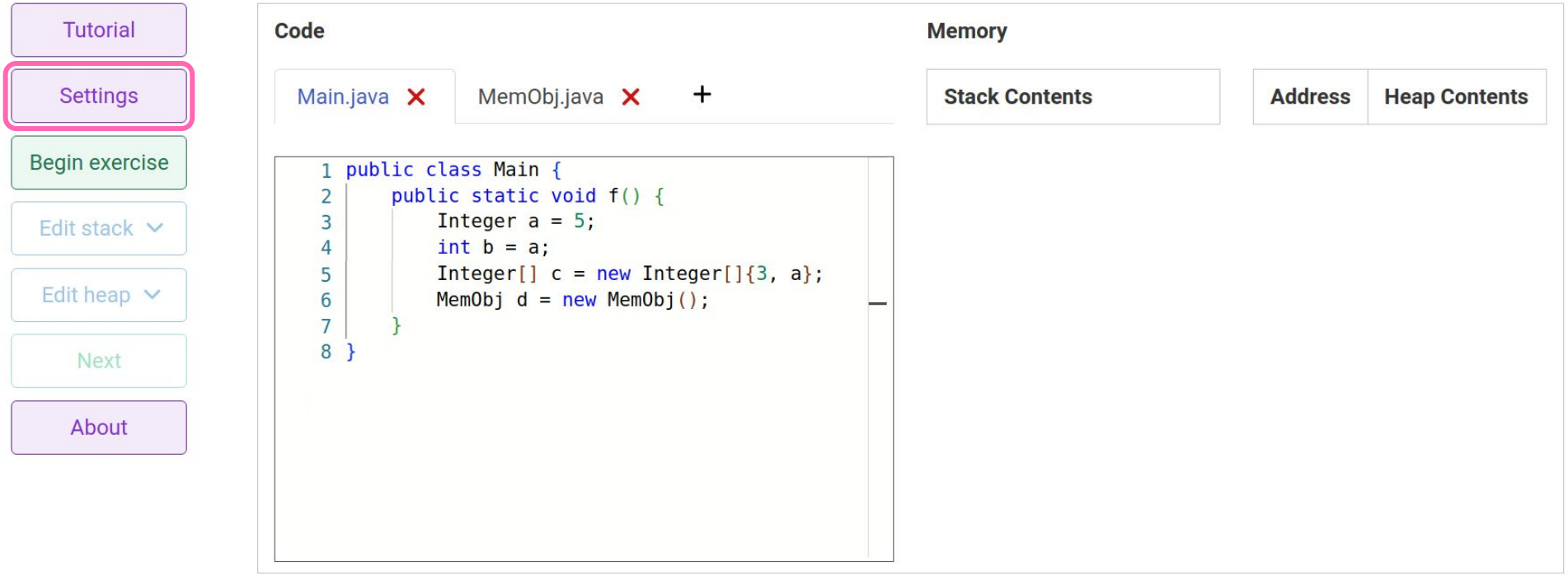
User Workflow

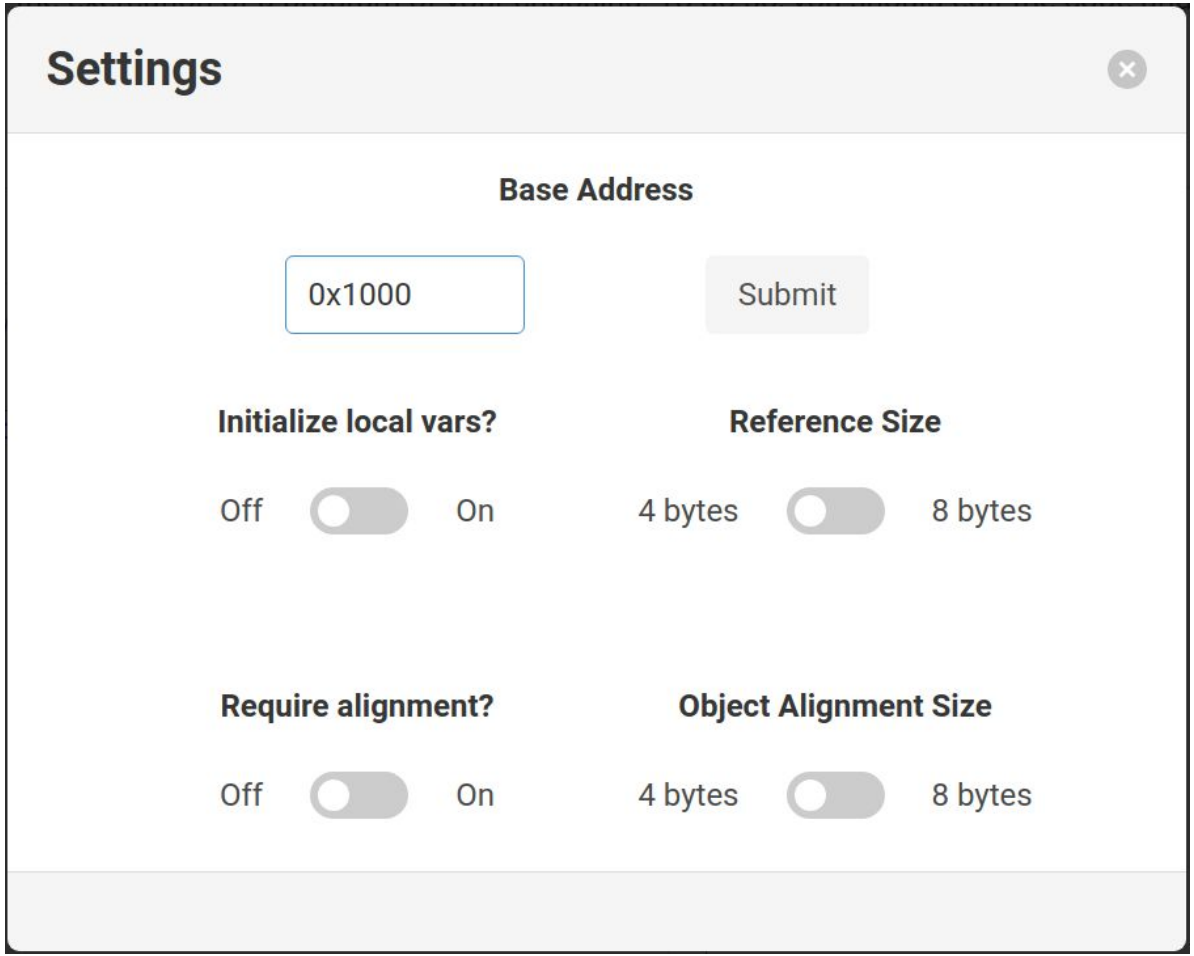


User Workflow

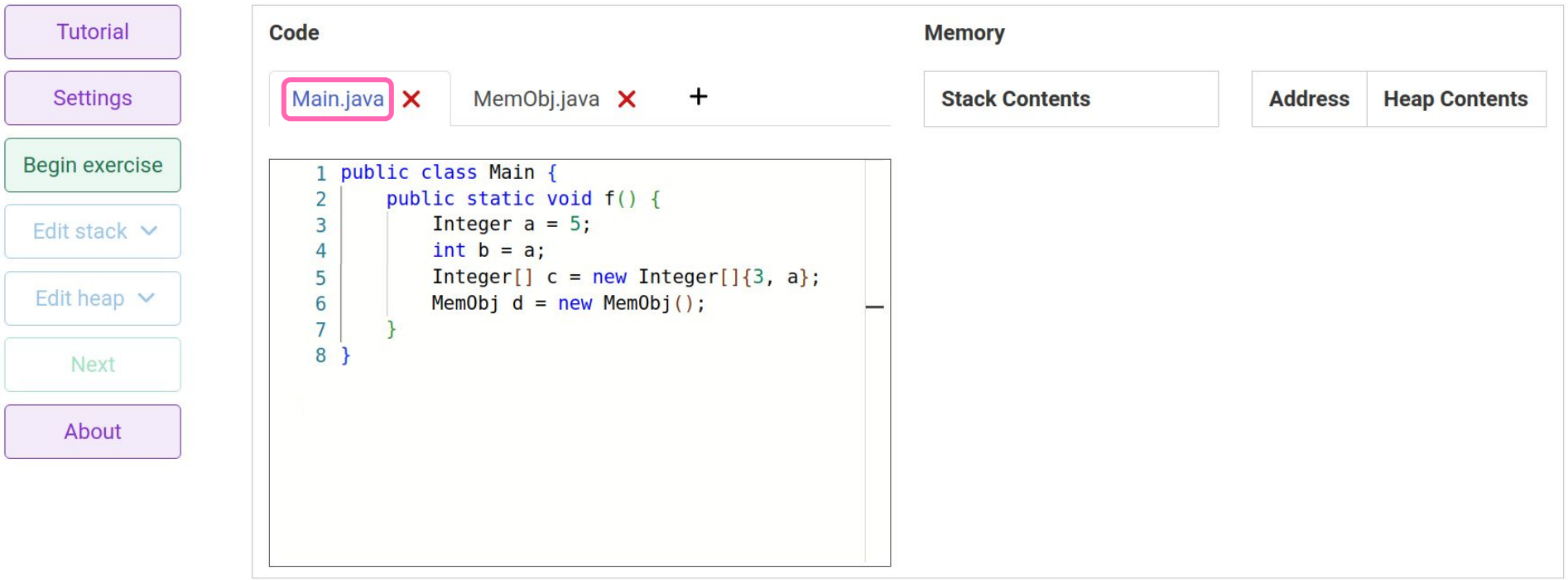


User Workflow

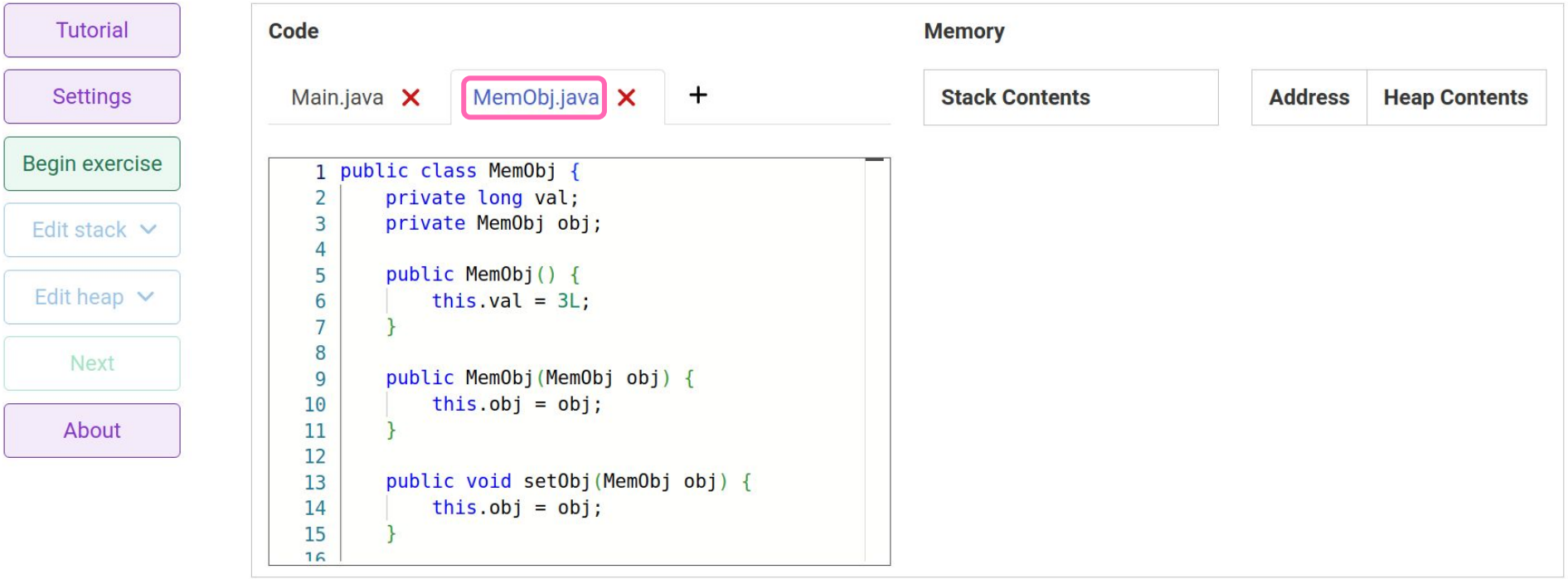



Settings
Base Address
0x1000
Submit
Initialize local vars?
Off
On
Reference Size
4 bytes
8 bytes
Require alignment?
Off
On
Object Alignment Size
4 bytes
8 bytes

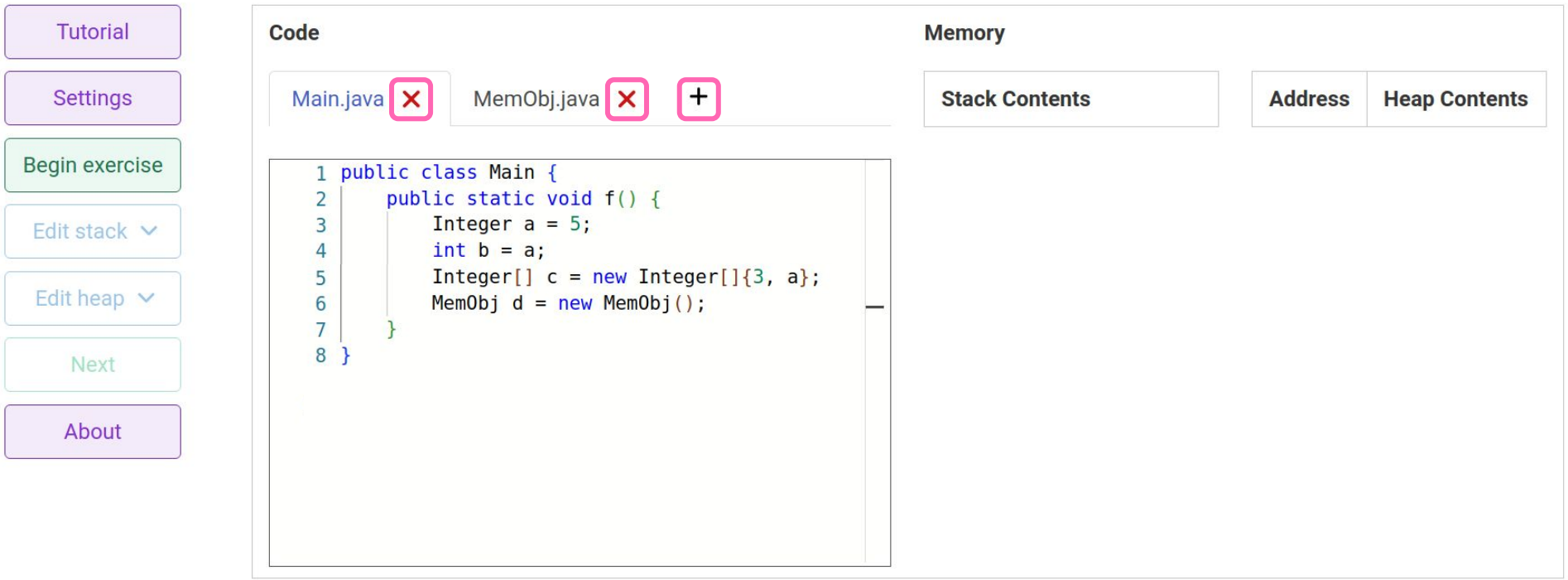
User Workflow



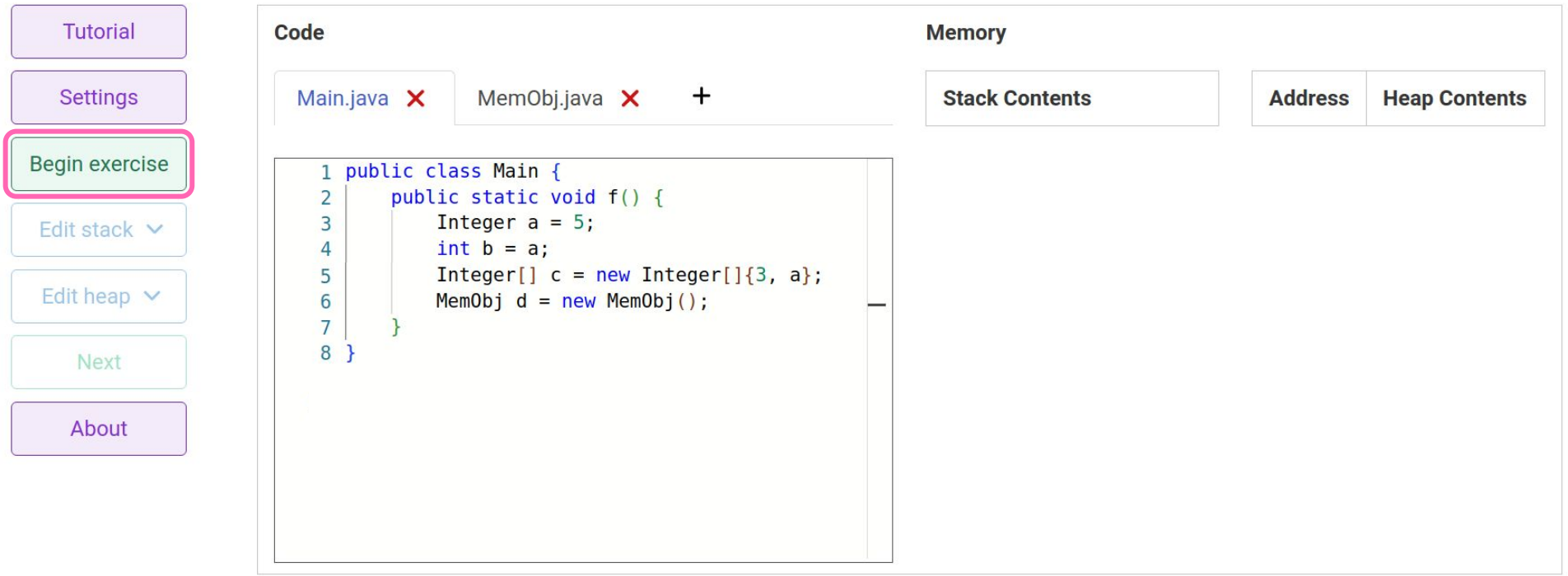
User Workflow



User Workflow



User Workflow



Settings

Begin exercise

Edit stack

Edit heap

Next

About

Main.java ✕ MemObj.java ✕ +

Stack Contents

Address **Heap Contents**

```
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 }
```

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

Settings

Begin exercise

Edit stack

Edit heap

Next

About

Main.java ✕ MemObj.java ✕ +

Stack Contents

Address **Heap Contents**

```
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 }
```

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

Settings

Begin exercise

Edit stack

Edit heap

Next

About

Main.java ✕ MemObj.java ✕ +

Stack Contents

Address **Heap Contents**

```
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 }
```

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

Settings

Stop exercise

Edit stack

Edit heap

Next

About

Main.java | MemObj.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 }
```

Stack Contents
Method name: Main.f
Local variables:

Address	Heap Contents
0x101c	
0x1018	
0x1014	
0x1010	
0x100c	
0x1008	
0x1004	
0x1000	

Settings

Stop exercise

Edit stack

Edit heap

Next

About

Main.java | MemObj.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 }
```

Stack Contents
Method name: Main.f
Local variables:

Address	Heap Contents
0x101c	
0x1018	
0x1014	
0x1010	
0x100c	
0x1008	
0x1004	
0x1000	

Settings

Stop exercise

Edit stack

Edit heap

Next

About

Main.java | MemObj.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 }
```

Stack Contents
Method name: Main.f
Local variables:

Address	Heap Contents
0x101c	
0x1018	
0x1014	
0x1010	
0x100c	
0x1008	
0x1004	
0x1000	

Settings

Stop exercise

Edit stack

Edit heap

Next

About

Main.java | MemObj.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 }
```

Stack Contents
Method name: Main.f
Local variables:

Address	Heap Contents
0x101c	
0x1018	
0x1014	
0x1010	
0x100c	
0x1008	
0x1004	
0x1000	

Settings

Stop exercise

Edit stack

Edit heap

- Add object to heap
- Add array to heap
- Update heap value

Main.java | MemObj.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 }
```

Stack Contents
Method name: Main.f
Local variables:

Address	Heap Contents
0x101c	
0x1018	
0x1014	
0x1010	
0x100c	
0x1008	
0x1004	
0x1000	

Settings

Stop exercise

Edit stack

- Add variable to stack
- Update stack value
- Add stack frame
- Deallocate stack frame

Main.java | MemObj.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 }
```

Stack Contents
Method name: Main.f
Local variables:

Address	Heap Contents
0x101c	
0x1018	
0x1014	
0x1010	
0x100c	
0x1008	
0x1004	
0x1000	

Settings

Stop exercise

Edit stack

- Add variable to stack
- Update stack value
- Add stack frame
- Deallocate stack frame

Main.java | MemObj.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 }
```

Stack Contents
Method name: Main.f
Local variables:

Address	Heap Contents
0x101c	
0x1018	
0x1014	
0x1010	
0x100c	
0x1008	
0x1004	
0x1000	

Settings

Stop exercise

Edit stack

Edit heap

Next

About

Main.java | MemObj.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 }
```

Stack Contents
Method name: Main.f
Local variables:

Address	Heap Contents
0x101c	
0x1018	
0x1014	
0x1010	
0x100c	
0x1008	
0x1004	
0x1000	

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

Variable's name:

Enter the variable's name

Variable's value:

Enter the variable's value

Submit

Settings

Stop exercise

Edit stack

Edit heap

Next

About

Main.java | MemObj.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 }
```

Stack Contents
Method name: Main.f
Local variables:

Address	Heap Contents
0x101c	
0x1018	
0x1014	
0x1010	
0x100c	
0x1008	
0x1004	
0x1000	

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:

5

Submit

Settings

Stop exercise

Edit stack

Edit heap

Next

About

Main.java | MemObj.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 }
```

Stack Contents
Method name: Main.f
Local variables:

Address	Heap Contents
0x101c	
0x1018	
0x1014	
0x1010	
0x100c	
0x1008	
0x1004	
0x1000	

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:

5

Submit

Settings

Stop exercise

Edit stack

Edit heap

Next

About

Main.java | MemObj.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 }
```

Stack Contents
Method name: Main.f
Local variables:

Address	Heap Contents
0x101c	
0x1018	
0x1014	
0x1010	
0x100c	
0x1008	
0x1004	
0x1000	

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).

Settings

Stop exercise

Edit stack

Edit heap

- Add object to heap
- Add array to heap
- Update heap value

Main.java | MemObj.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 }
```

Stack Contents
Method name: Main.f
Local variables:

Address	Heap Contents
0x101c	
0x1018	
0x1014	
0x1010	
0x100c	
0x1008	
0x1004	
0x1000	

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).

Settings

Stop exercise

Edit stack

Edit heap

Next

About

Main.java | MemObj.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 }
```

Stack Contents
Method name: Main.f
Local variables:

Address	Heap Contents
0x101c	
0x1018	
0x1014	
0x1010	
0x100c	
0x1008	
0x1004	
0x1000	

This operation will allocate a new object on the heap. Please specify the object to be allocated.

Starting address:

Enter the starting address at which to allocate the object

Number of fields:

Enter the number of fields that the object being allocated has

Submit

Settings

Stop exercise

Edit stack

Edit heap

Next

About

Main.java | MemObj.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 }
```

Stack Contents
Method name: Main.f
Local variables:

Address	Heap Contents
0x101c	
0x1018	
0x1014	
0x1010	
0x100c	
0x1008	
0x1004	
0x1000	

This operation will allocate a new object on the heap. Please specify the object to be allocated.

Starting address:

0x1000

Number of fields:

1

Submit

Settings

Stop exercise

Edit stack

Edit heap

Next

About

Main.java | MemObj.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 }
```

Stack Contents
Method name: Main.f
Local variables:

Address	Heap Contents
0x101c	
0x1018	
0x1014	
0x1010	
0x100c	
0x1008	
0x1004	
0x1000	

This operation will allocate a new object on the heap. Please specify the object to be allocated.

Starting address:

0x1000

Number of fields:

1

Submit

Edit stack

Edit heap

Next

About

```
3         Integer a = 5;
4         int b = a;
5         Integer[] c = new Integer[]{3, a};
6         MemObj d = new MemObj();
7     }
8 }
```

Local variables:

0x1014	
0x1010	
0x100c	
0x1008	
0x1004	
0x1000	

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

Please specify the first field.

Starting address of the field:

Enter the starting address of the field

Size of the field:

Enter the size of the field

Initial value of the field:

Enter the initial value of the field

Submit

Edit stack

Edit heap

Next

About

```
3         Integer a = 5;
4         int b = a;
5         Integer[] c = new Integer[]{3, a};
6         MemObj d = new MemObj();
7     }
8 }
```

Local variables:

0x1014	
0x1010	
0x100c	
0x1008	
0x1004	
0x1000	

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

Edit stack

Edit heap

Next

About

```
3         Integer a = 5;
4         int b = a;
5         Integer[] c = new Integer[]{3, a};
6         MemObj d = new MemObj();
7     }
8 }
```

Local variables:

0x1014	
0x1010	
0x100c	
0x1008	
0x1004	
0x1000	

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

Settings

Stop exercise

Edit stack

Edit heap

Next

About

Main.java | MemObj.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 }
```

Stack Contents
Method name: Main.f
Local variables:

Address	Heap Contents
0x101c	
0x1018	
0x1014	
0x1010	
0x100c	
0x1008	
0x1004	
0x1000	5

Settings

Stop exercise

Edit stack

- Add variable to stack
- Update stack value
- Add stack frame
- Deallocate stack frame

Main.java | MemObj.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 }
```

Stack Contents
Method name: Main.f
Local variables:

Address	Heap Contents
0x101c	
0x1018	
0x1014	
0x1010	
0x100c	
0x1008	
0x1004	
0x1000	5

Settings

Stop exercise

Edit stack

Edit heap

Next

About

Main.java | MemObj.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 }
```

Stack Contents
Method name: Main.f
Local variables:

Address	Heap Contents
0x101c	
0x1018	
0x1014	
0x1010	
0x100c	
0x1008	
0x1004	
0x1000	5

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

Variable's name:

Enter the variable's name

Variable's value:

Enter the variable's value

Submit

Settings

Stop exercise

Edit stack

Edit heap

Next

About

Main.java | MemObj.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 }
```

Stack Contents
Method name: Main.f
Local variables:

Address	Heap Contents
0x101c	
0x1018	
0x1014	
0x1010	
0x100c	
0x1008	
0x1004	
0x1000	5

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

Settings

Stop exercise

Edit stack

Edit heap

Next

About

Main.java | MemObj.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 }
```

Stack Contents
Method name: Main.f
Local variables:

Address	Heap Contents
0x101c	
0x1018	
0x1014	
0x1010	
0x100c	
0x1008	
0x1004	
0x1000	5

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

Settings

Stop exercise

Edit stack

Edit heap

Next

About

Main.java | MemObj.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 }
```

Stack Contents
Method name: Main.f Local variables: a: 0x1000

Address	Heap Contents
0x101c	
0x1018	
0x1014	
0x1010	
0x100c	
0x1008	
0x1004	
0x1000	5

Settings

Stop exercise

Edit stack

Edit heap

Next

About

Main.java | MemObj.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 }
```

Stack Contents
Method name: Main.f Local variables: a: 0x1000

Address	Heap Contents
0x101c	
0x1018	
0x1014	
0x1010	
0x100c	
0x1008	
0x1004	
0x1000	5

Settings

Stop exercise

Edit stack

Edit heap

Next

About

Main.java | MemObj.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 }
```

Stack Contents
Method name: Main.f Local variables: a: 0x1000

Address	Heap Contents
0x101c	
0x1018	
0x1014	
0x1010	
0x100c	
0x1008	
0x1004	
0x1000	5

Code

Main.java | MemObj.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 }
```

Memory

Stack Contents
Method name: Main.f Local variables: a: 0x1000 b: 5 c: 0x1008 d: 0x1010

Address	Heap Contents
0x101c	
0x1018	null
0x1014	*continuation of 0x1010*
0x1010	3L
0x100c	0x1000
0x1008	0x1004
0x1004	3
0x1000	5

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

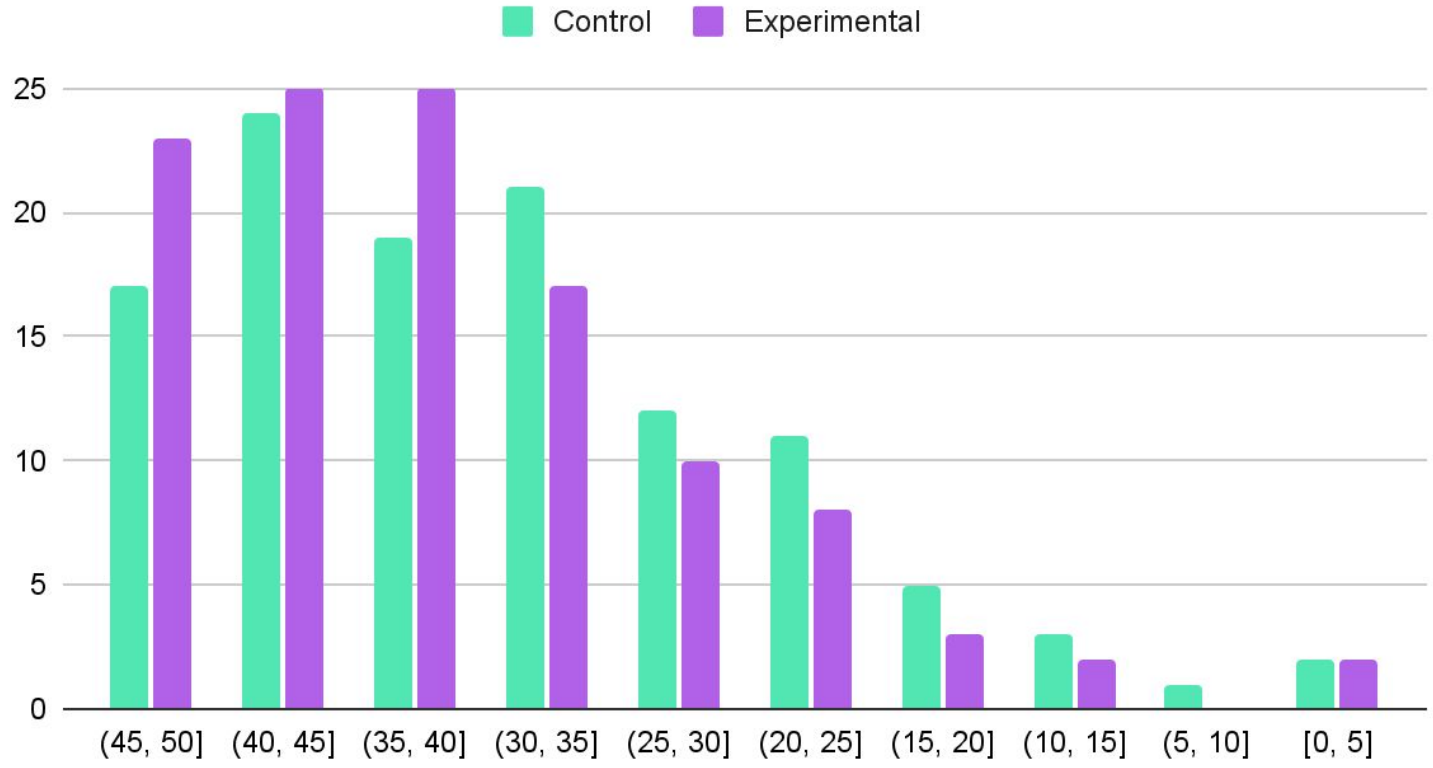
- **Post-study survey**
  - **All** students rated **agreement** with statements about the training
  - **Experimental** group provided **qualitative** feedback

- **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.

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

1 = Strongly Disagree ⟷ 5 = Strongly Agree

Qualitative Feedback

n = 112

What was your overall impression of MemStep?

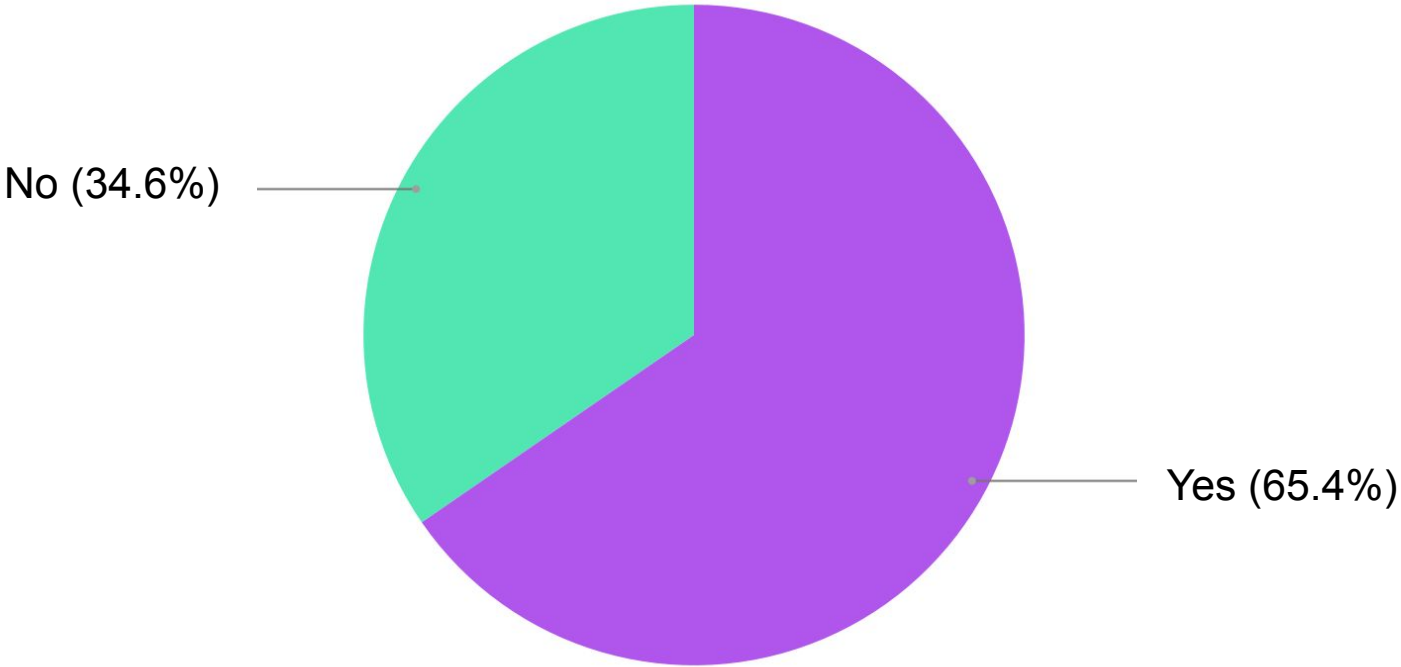
Overall Tone	Freq.
Positive	105
Neutral	5
Negative	2

Theme	Freq.
Helpful/effective for learning	58
Liked the feedback	24
Interactive/hands on	23
Fun/enjoyable	22
Easy to use	15
Well-designed	14
Engaging	14

End-of-Semester Survey Results

n = 237

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



End-of-Semester Survey Results

n = 155

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

	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

End-of-Semester Survey Results

n = 155

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

	Avg. Rating
Having the ability to walk through arbitrary Java programs using MemStep made me **curious** to try different programs in order **to see how different types of data get laid out** in memory	3.91
Having the ability to walk through arbitrary Java programs using MemStep made me **curious** to try different programs in order **to see what different different steps happen** under the hood during execution.	3.87

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