

CS 600.226: Data Structures

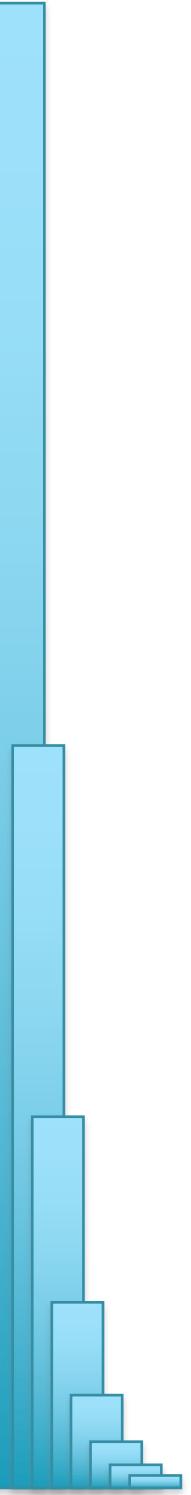
Michael Schatz

Sept 7 2018

Lecture 4: Linked Lists



Agenda

- 
- 1. Review HWI***
 - 2. Review Java Arrays***
 - 3. References and Linked Lists***

Assignment I: Due Friday Sept 14 @ 10pm

<https://github.com/schatzlab/datastructures2018/blob/master/assignments/assignment01/assignment01.md>

Assignment 1: Warming Up

- Out on: September 7, 2016
- Due by: September 14, 2016 before 10:00 pm
- Collaboration: None
- Grading:
 - Functionality 65%
 - ADT Solution 30%
 - Solution Design and README 5%
 - Style 0%

Overview

The first assignment is mostly a warmup exercise to refresh your knowledge of Java and an ADT problem to start you thinking more abstractly about your data.

Assignment I: Due Friday Sept 14 @ 10pm

<https://github.com/schatzlab/datastructures2018/blob/master/assignments/assignment01/assignment01.md>

Problem 1: Unique Numbers (35%)

Your first task is to write a simple Java program `Unique` that analyzes the command line it is given in a peculiar way. The program accepts any number of integers as command line arguments and prints each unique integer it was presented with as its output. For example, the invocation

```
java Unique 0 0 10 0 1 0 0 0 10 1
```

should generate the output

```
0  
10  
1
```

while the invocation

```
java Unique 1 9 2 3 1 4 9 5 3 6 0
```

should generate the output

```
1  
9  
2  
3  
4  
5  
6  
0
```

instead. Note that order doesn't matter as long as you print the correct set of numbers, one line per number, without any additional output!

As an added complication, you are not allowed to use any Java classes that serve as advanced data structures, specifically not Java collection classes like `ArrayList` or `HashMap`. You can use regular Java arrays, and in fact that's probably the best way to go; the only "problem" is that we don't specify an upper limit on the number of arguments, you'll have to figure out how to deal with that...

Hints

- If you feel like you need to sort something, think again! You don't have to sort anything to get this problem done.
- The "command line" is the array of strings passed to the main method of your program. If you're using a graphical development environment you may have to first figure out how you can start the program with a command line.

Assignment I: Due Friday Sept 14 @ 10pm

<https://github.com/schatzlab/datastructures2018/blob/master/assignments/assignment01/assignment01.md>

Problem 2: Counter Varieties (30%)

Your second task is to write a number of "counters" that can be used interchangably (at least as far as Java is concerned). You are given the following interface (put it into a file Counter.java please):

```
/** The essence of any counter. */
public interface Counter {
    /** Current value of this counter. */
    int value();
    /** Increment this counter. */
    void up();
    /** Decrement this counter. */
    void down();
}
```

Develop the following:

- An interface ResetableCounter that supports the method `void reset()` in addition to those of Counter ; this method should set the counter to its initial value
- An implementation of ResetableCounter called BasicCounter that starts at the value 0 and counts up and down by +1 and -1 respectively.
- An implementation of ResetableCounter called EvenCounter that starts at the value 0, counts up by adding 2, and counts down by subtracting 2
- An implementation of ResetableCounter called TenCounter that starts at the value 1, counts up by multiplying by 10, and counts down by dividing by 10. This should round up to the nearest integer if needed
- An implementation of ResetableCounter called FlexibleCounter that allows clients to specify a start value as well as an additive increment (used for counting up) when a counter is created. For example `new FlexibleCounter(-10, 42)` would yield a counter with the current value -10; after a call to `up()` its value would be 32.

All of your implementations should be resetable, and each should contain a main method that tests whether the implementation works as expected using assert as we did in lecture (this is a simple approach to unit testing, we'll cover a better approach later).

Finally, make sure that your four counters work with the `PolyCount.java` test program we provide; it's probably a good idea to read and understand it. :-)

Hints

- Pay attention to your use of `public` and `private` ! The essence of those counters is not just to hold a bunch of data, but to ensure that a certain approach to counting is followed; making everything public is a bad idea here.
- Remember that interfaces can extend one another in a way similar to classes (using the `extends` keyword). Classes implement interfaces however (using the `implements` keyword).

Assignment I: Due Friday Sept 14 @ 10pm

<https://github.com/schatzlab/datastructures2018/blob/master/assignments/assignment01/assignment01.md>

Problem 3: List ADT (30%)

In lecture we derived an algebraic specifications for the abstract data type `Array`. Following that example, develop a specification for the related ADT `List` supporting these operations:

1. Create a new empty list
2. Insert a new integer at a particular position in the list.
3. Return true if the list is empty, otherwise false.
4. Clear the contents of the list
5. Return the number of integers currently in the list.
6. Retrieve the integer at a particular position in the list.
7. Delete the integer at a particular position in the list.

From this description, the input and output of each operation should hopefully be clear. Later we will discuss how to implement this in an efficient way (Hint: Arrays will have bad performance if you expect to insert/delete frequently from the middle).

Do this in a text file named `ListADT.txt`

Notes

- You can assume the Boolean ADT is available that specifies 'true' and 'false'
- Make sure to list all axioms and preconditions -- what assertions and exceptions would you write if you were implementing this ADT?

Assignment I: Due Friday Sept 14 @ 10pm

<https://github.com/schatzlab/datastructures2018/blob/master/assignments/assignment01/assignment01.md>

Solution Design (5%)

- You will be graded on your general solution and design. Discuss in your README file anything related to how you solved the problems and justify why you believe your solution is a good one.
- This is relatively subjective but we are generally looking for good practices in Java (helper methods, inheritance, etc...)

Even More Hints

- Ensure that the version of your code you hand in does not produce any extraneous debugging output anymore!
- Pay attention to edge cases in the input your classes and programs are expected to handle! For example, make sure that you handle an empty command line in a reasonable way for Problem 1.
- You will not be deducted for style on this assignment, however you will still receive checkstyle feedback and general style feedback. In the future, style will be worth roughly 10% of the assignment and you will get checkstyle feedback in the autograder.
- Submitting compiling code is always better, no compilation means no functionality points. You will get freebie points just for having a submission that compiles.

Assignment I: Due Friday Sept 14 @ 10pm

<https://github.com/schatzlab/datastructures2018/blob/master/assignments/assignment01/assignment01.md>

Deliverables

Go to the assignment 1 page for Gradescope and click submit. Note that you can resubmit any time up until the deadline. You will be prompted to upload your files at which point you will upload all of the necessary source files. In the future we might not list them out, but for this assignment they are listed explicitly below:

```
Unique.java  
BasicCounter.java  
EvenCounter.java  
FlexibleCounter.java  
ResetableCounter.java  
TenCounter.java  
README  
ListADT.txt
```

Note (especially for Java files) your files must be named exactly as we are expecting them for them to work in the autograder.

Also note that for provided files (such as `Counter.java`), we will be dropping in the provided version with your solution. So if you change `Counter.java` and try to submit it, the original distributed `Counter.java` we have will overwrite it. It is important not to modify those given interface files.

After you submit, the autograder will run and you will get feedback on your functionality and how you performed on our test cases. For this assignment, we will display all of the test cases we run in the autograder to you so you will know exactly what test case failed. The test cases are what gets you the functionality points on the assignment. If for some reason your code did not compile, you should get that output from the autograder showing you the error messages it received. If you cannot figure out why your code is not working in the autograder, but works for you locally, post a private message on piazza.

Include a `README` file that briefly explains what your programs do and contains any other notes you want us to check out before grading. This is also a

Finally, make sure to include your name and email address in every file you turn in (well, in every file for which it makes sense to do so anyway)!

Assignment I: Due Friday Sept 14 @ 10pm

<https://github.com/schatzlab/datastructures2018/blob/master/assignments/assignment01/assignment01.md>

Grading

For reference, here is a short explanation of the grading criteria; some of the criteria don't apply to all problems, and not all of the criteria are used on all assignments.

Packaging refers to the proper organization of the stuff you hand in, following both the guidelines for Deliverables above as well as the general submission instructions for assignments.

Style refers to Java programming style, including things like consistent indentation, appropriate identifier names, useful comments, suitable javadoc documentation, etc. Many aspects of this are enforced automatically by [Checkstyle](#) when run with the configuration file available on [github](#). Style also includes proper modularization of your code (into interfaces, classes, methods, using `public`, `protected`, and `private` appropriately, etc.). Simple, clean, readable code is what you should be aiming for.

Testing refers to proper unit tests for all of the data structure classes you developed for this assignment, using the [JUnit 4](#) framework as introduced in lecture. Make sure you test all (implied) axioms that you can think of and all exception conditions that are relevant.

Performance refers to how fast/with how little memory your program can produce the required results compared to other submissions.

Functionality refers to your programs being able to do what they should according to the specification given above; if the specification is ambiguous and you had to make a certain choice, defend that choice in your `README` file.

If your programs cannot be built you will get no points whatsoever. If your programs cannot be built without warnings using `javac -Xlint:all` we will take off 10% (except if you document a very good reason; no, you cannot use the `@SuppressWarnings` annotation either). If your programs fail miserably even once, i.e. terminate with an exception of any kind, we will take off 10% (however we'll also take those 10% off if you're trying to be "excessively smart" by wrapping your whole program into a universal try-catch).

GradeScope.com

Entry Code: MDJYER

The screenshot shows a web browser window for GradeScope. The URL in the address bar is gradescope.com. The main sidebar on the left is titled "Gradescope 202" and includes links for "Advanced Gradescope Features", "Dashboard", and "Regrade Requests". Below that is an "INSTRUCTOR" section with a profile for "Michael Schatz". The main content area is titled "Autograder Details" and "STUDENT". A modal dialog box is open, titled "Submit Programming Assignment". It instructs the user to "Upload all files for your submission". Under "SUBMISSION METHOD", the "Upload" option is selected. Below it, there is a placeholder text: "Add files via Drag & Drop or Browse Files." A table lists seven Java files with their sizes:

NAME	SIZE	PROGRESS
BasicCounter.java	0.4 KB	[Progress Bar]
EvenCounter.java	0.4 KB	[Progress Bar]
FlexibleCounter.java	1.4 KB	[Progress Bar]
PolyCount.java	2.3 KB	[Progress Bar]
ResetableCounter.java	0.3 KB	[Progress Bar]
TenCounter.java	0.6 KB	[Progress Bar]
Unique.java	2.1 KB	[Progress Bar]

At the bottom of the modal are two buttons: "Upload" (green) and "Cancel" (red). The background of the page shows some text from a previous submission attempt.

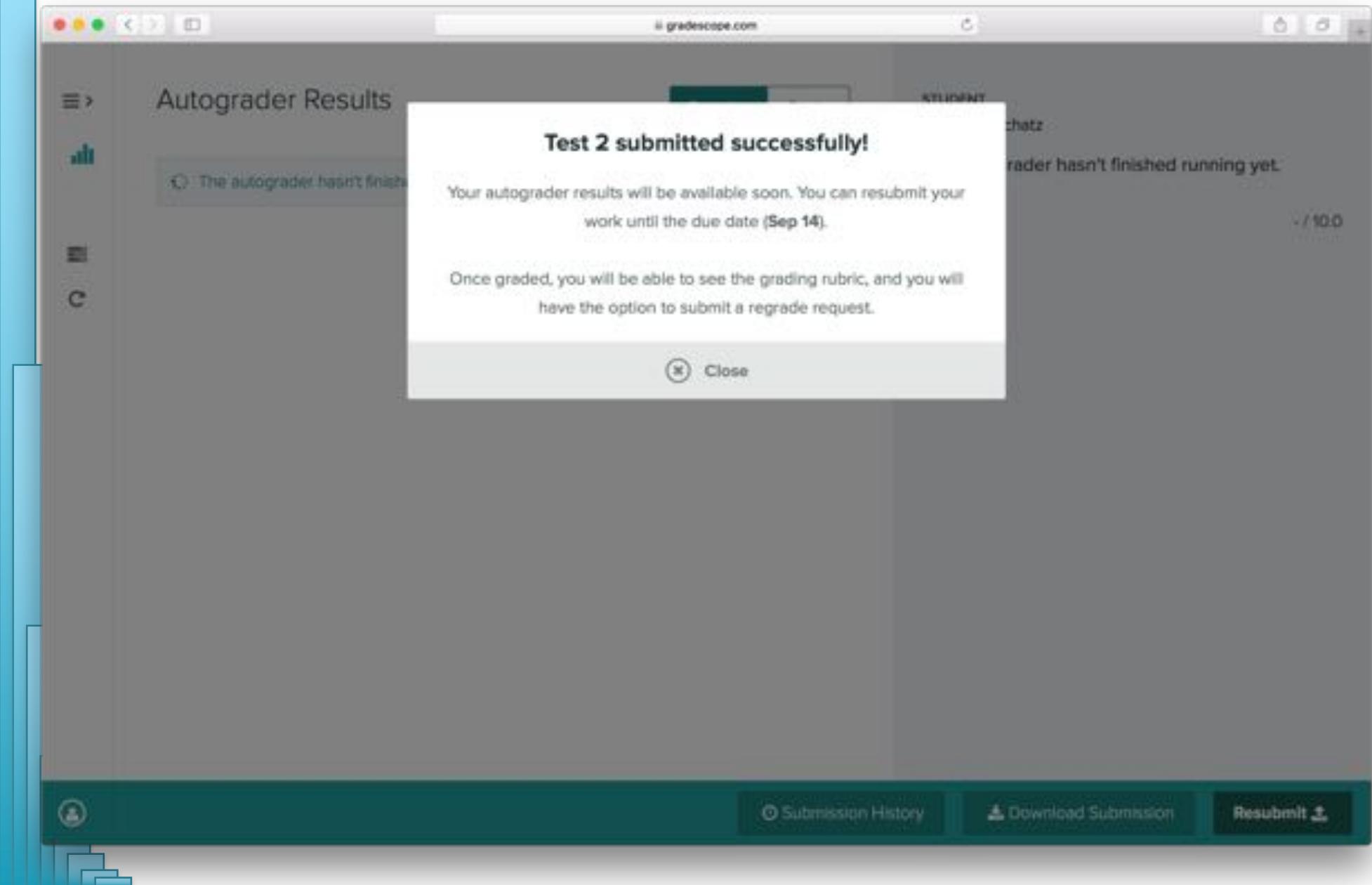
New even counter has default value 0 (1.0/1.0)

its by 1 (1.0/1.0)
counter has default value 0 (1.0/1.0)
ments by 1 (1.0/1.0)
to initial value. (1.0/1.0)
n() combination. (1.0/1.0)
s the value (1.0/1.0)
ases the value (1.0/1.0)
unter has default value 0 (1.0/1.0)
ments by 2 (1.0/1.0)
ts by 2 (1.0/1.0)
to initial value. (1.0/1.0)
n() combination. (1.0/1.0)
s the value (1.0/1.0)
ases the value (1.0/1.0)
ts by increment value. (1.0/1.0)
ement value triggers exception
ments by decrement value. (1.0/1.0)
counter has correct default value.
to initial value. (1.0/1.0)
n() combination. (1.0/1.0)
s the value (1.0/1.0)
down() decreases the value (1.0/1.0)
up() multiplies by 10. (1.0/1.0)

Account Submission History Download Submission Resubmit

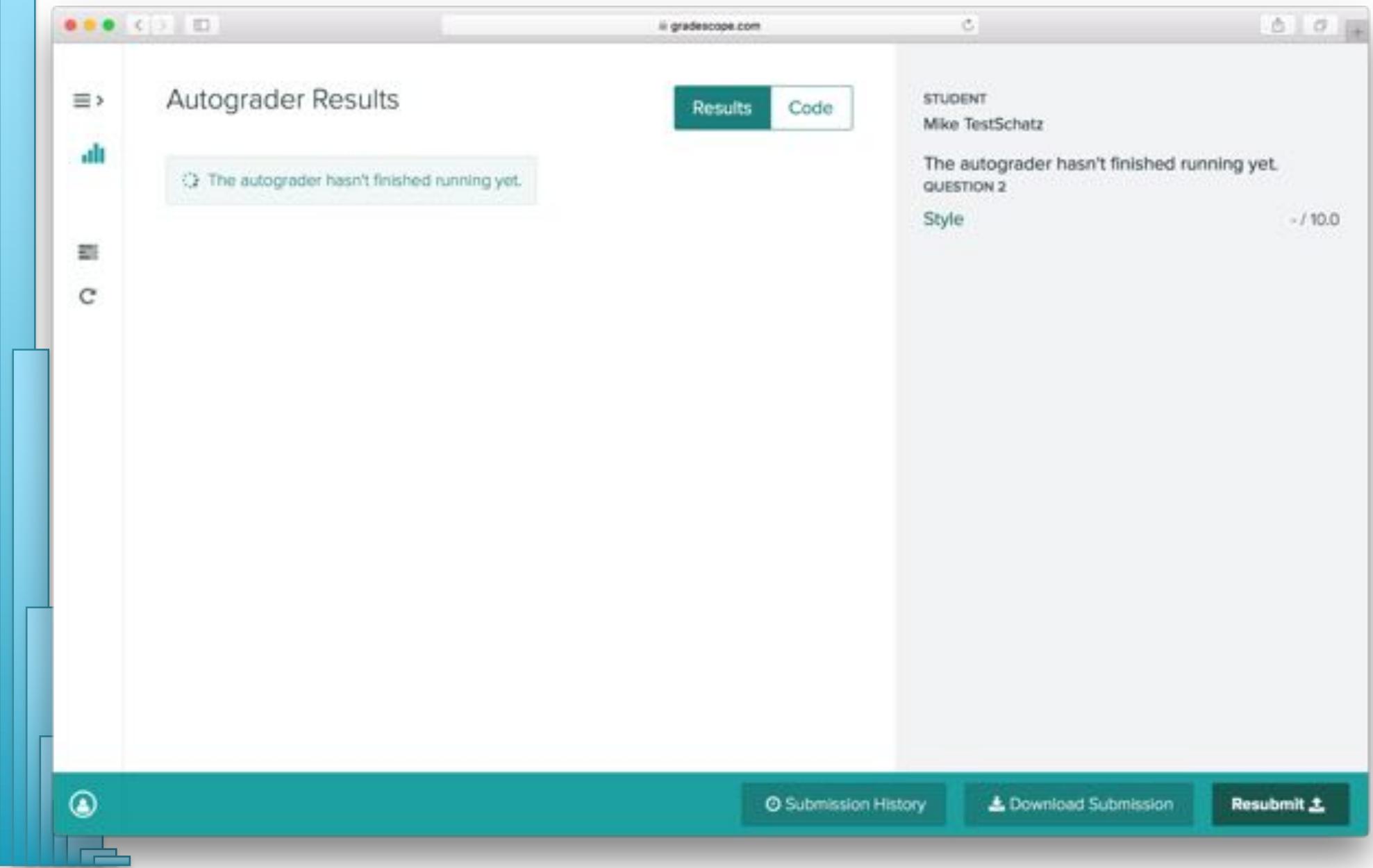
GradeScope.com

Entry Code: MDJYER



GradeScope.com

Entry Code: MDJYER



A screenshot of a web browser displaying the GradeScope.com Autograder Results page. The URL in the address bar is "gradescope.com". The main content area shows "Autograder Results" and a message: "The autograder hasn't finished running yet." Below this, under "QUESTION 2", it says "Style - / 10.0". On the left, there's a sidebar with icons for "Autograder Results", "Submission History", "Download Submission", and "Resubmit". At the bottom, there are buttons for "Submission History", "Download Submission", and "Resubmit".

Autograder Results

The autograder hasn't finished running yet.

STUDENT
Mike TestSchatz

The autograder hasn't finished running yet.
QUESTION 2

Style - / 10.0

Submission History Download Submission Resubmit

GradeScope.com

Entry Code: MDJYER

The screenshot shows the GradeScope Autograder Results page for a submission. The left sidebar includes links for Gradescope 202, Advanced Gradescope Features, Dashboard, and Regrade Requests. The top navigation bar shows the URL gradescope.com. The main content area displays the Autograder Results with two tabs: 'Results' (selected) and 'Code'. On the right, student information is shown: STUDENT Mike TestSchatz and AUTOGRADER SCORE 45.0 / 90.0. Below this, PASSED TESTS are listed, each with a green checkmark icon and a test name followed by its outcome (e.g., up() increments by 1 (1.0/1.0)). A large orange callout box highlights the text "All green tests 😊". At the bottom, there are links for Account, Submission History, Download Submission, and Resubmit.

Autograder Results

Results Code

STUDENT
Mike TestSchatz

AUTOGRADER SCORE
45.0 / 90.0

PASSED TESTS

- up() increments by 1 (1.0/1.0)
- New basic counter has default value 0 (1.0/1.0)
- down() decrements by 1 (1.0/1.0)
- reset() resets to initial value. (1.0/1.0)
- up() and down() combination. (1.0/1.0)
- up() increases the value (1.0/1.0)
- down() decreases the value (1.0/1.0)
- New even counter has default value 0 (1.0/1.0)
- down() decrements by 2 (1.0/1.0)
- up() increments by 2 (1.0/1.0)
- reset() resets to initial value. (1.0/1.0)
- up() and down() combination. (1.0/1.0)
- up() increases the value (1.0/1.0)
- down() decreases the value (1.0/1.0)
- up() increments by increment value. (1.0/1.0)
- Negative increment value triggers exception (1.0/1.0)
- down() decrements by decrement value. (1.0/1.0)
- New flexible counter has correct default value. (1.0/1.0)
- reset() resets to initial value. (1.0/1.0)
- up() and down() combination. (1.0/1.0)
- up() increases the value (1.0/1.0)
- down() decreases the value (1.0/1.0)
- up() multiplies by 10. (1.0/1.0)

All green tests 😊

Account Submission History Download Submission Resubmit

GradeScope.com

Entry Code: MDJYER

The screenshot shows the GradeScope Autograder Results page for a submission. The page has a sidebar with navigation icons (Home, Autograder, Submission History, Download Submission, Resubmit) and a main content area.

Autograder Results

STUDENT: Mike TestSchatz

AUTOGRADE SCORE: 43.0 / 90.0

FAILED TESTS:

- up() increments by 1 (1.0/1.0)
- New basic counter has default value 0 (1.0/1.0)
- down() decrements by 1 (1.0/1.0)
- reset() resets to initial value. (1.0/1.0)
- up() and down() combination. (1.0/1.0)
- up() increases the value (1.0/1.0)
- down() decreases the value (1.0/1.0)
- New even counter has default value 0 (1.0/1.0)
- down() decrements by 2 (1.0/1.0)
- up() increments by 2 (1.0/1.0)
- reset() resets to initial value. (1.0/1.0)
- up() and down() combination. (1.0/1.0)
- up() increases the value (1.0/1.0)
- down() decreases the value (1.0/1.0)
- ... (increment value triggers exception) (1.0/1.0)
- down() decrements by decrement value. (1.0/1.0)
- New flexible counter has correct default value. (1.0/1.0)
- reset() resets to initial value. (1.0/1.0)
- up() and down() combination. (1.0/1.0)

PASSED TESTS:

- up() increments by 1 (1.0/1.0)
- New basic counter has default value 0 (1.0/1.0)
- down() decrements by 1 (1.0/1.0)
- reset() resets to initial value. (1.0/1.0)
- up() and down() combination. (1.0/1.0)
- up() increases the value (1.0/1.0)
- down() decreases the value (1.0/1.0)
- New even counter has default value 0 (1.0/1.0)
- down() decrements by 2 (1.0/1.0)
- up() increments by 2 (1.0/1.0)
- reset() resets to initial value. (1.0/1.0)
- up() and down() combination. (1.0/1.0)
- up() increases the value (1.0/1.0)
- down() decreases the value (1.0/1.0)
- ... (increment value triggers exception) (1.0/1.0)
- down() decrements by decrement value. (1.0/1.0)
- New flexible counter has correct default value. (1.0/1.0)
- reset() resets to initial value. (1.0/1.0)
- up() and down() combination. (1.0/1.0)

Double-check your logic, resubmit

Submission History | **Download Submission** | **Resubmit**

GradeScope.com

Entry Code: MDJYER

The screenshot shows a web browser window for GradeScope.com. The URL in the address bar is gradescope.com. The main content area displays "Autograder Results". On the left sidebar, under "Gradescope 202", are links for "Advanced Gradescope Features", "Dashboard", and "Regrade Requests". Under "INSTRUCTOR", it shows "Michael Schatz". The central "Autograder Results" section has tabs for "Results" (which is selected) and "Code". The "Results" tab shows a message: "The autograder failed to execute correctly. Please ensure that your submission is valid. Contact your course staff for help in debugging this issue. Make sure to include a link to this page so that they can help you most effectively." To the right, under "STUDENT", it says "Mike TestSchatz". Below that, "AUTOGRADER SCORE" is listed as "0.0 / 90.0". Under "QUESTION 2", it says "Style" with a value of "- / 10.0". At the bottom of the page, there is a yellow callout box with the text: "Did you miss a file? Are you sure it compiles correctly? Fix and resubmit. Check Piazza. Message the CAs". The bottom navigation bar includes links for "Account", "Submission History", "Download Submission", and "Resubmit".

Did you miss a file? Are you sure it compiles correctly?
Fix and resubmit. Check Piazza. Message the CAs

gradescope

Gradescope 202

Advanced Gradescope Features

Dashboard

Regrade Requests

INSTRUCTOR

Michael Schatz

Autograder Results

Results Code

The autograder failed to execute correctly. Please ensure that your submission is valid. Contact your course staff for help in debugging this issue. Make sure to include a link to this page so that they can help you most effectively.

STUDENT

Mike TestSchatz

AUTOGRADER SCORE

0.0 / 90.0

QUESTION 2

Style - / 10.0

Account

Submission History

Download Submission

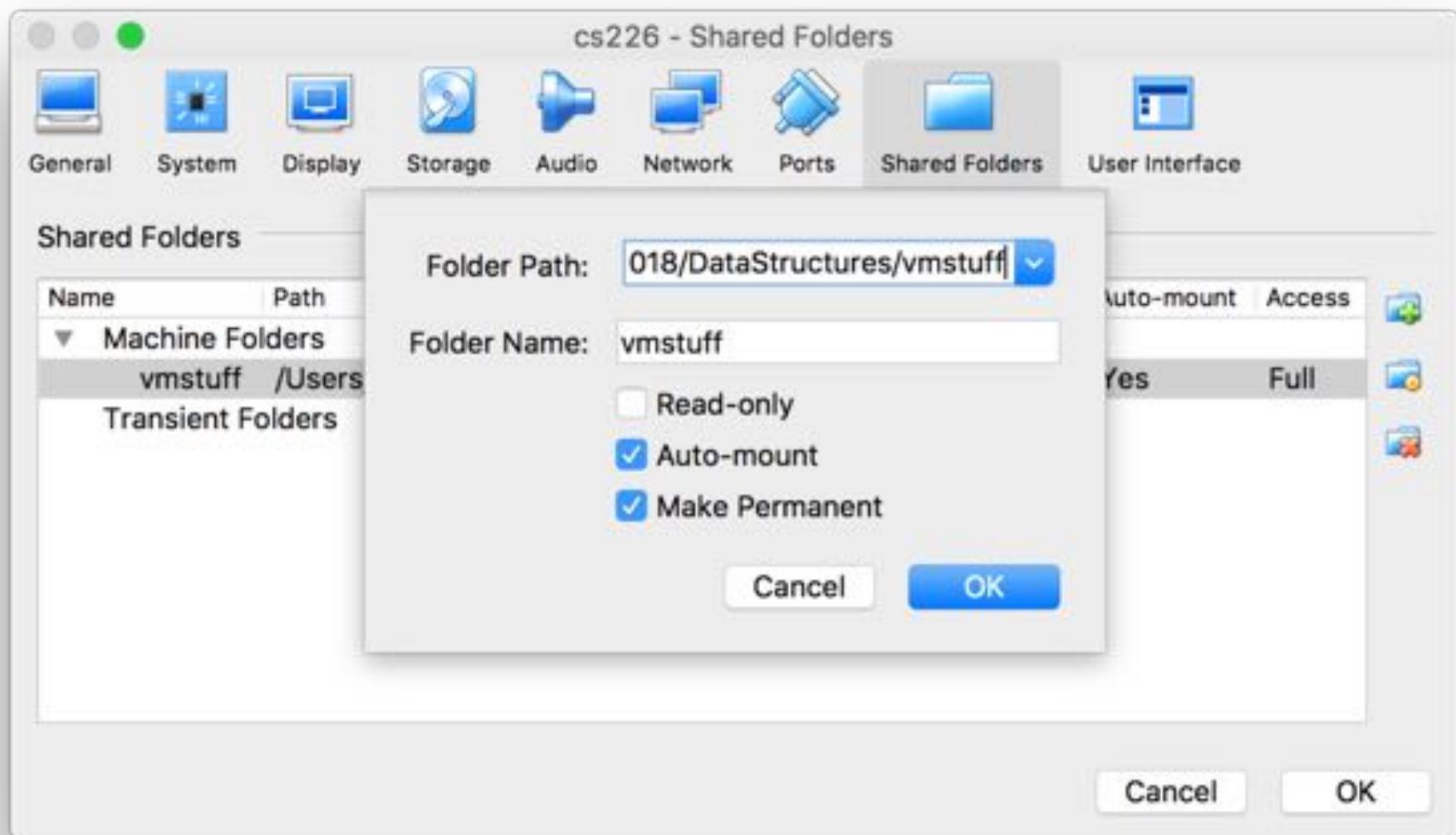
Resubmit

VirtualBox

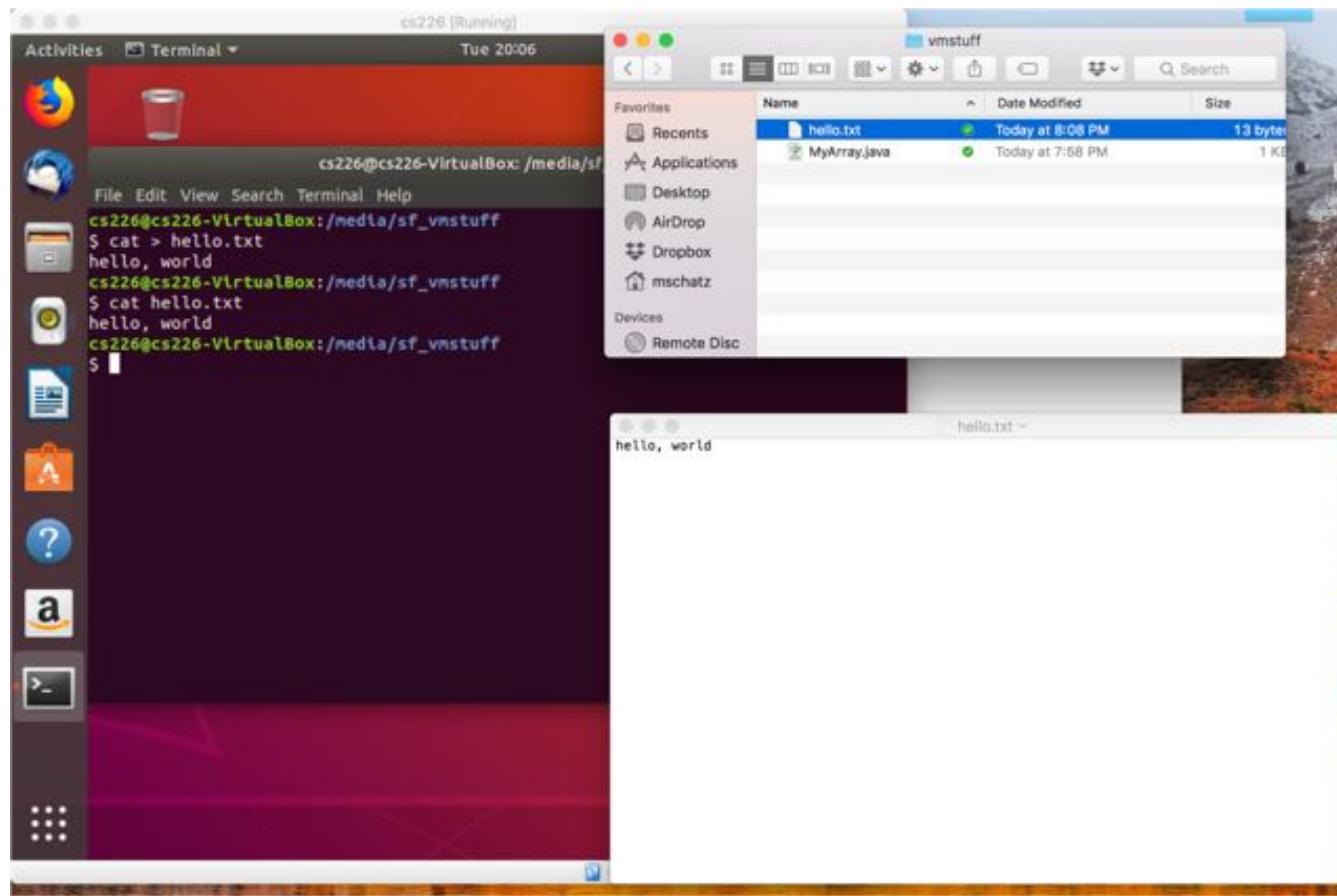


- Client application available for Mac, Windows, Linux
- Available to run our reference virtual machine running linux
 - Guaranteed that your development environment matches testing environment
 - Make sure to install the Extension Pack and Guest Additions too

VirtualBox Shared Folders



VirtualBox Shared Folders



```
# Install Guest Additions  
$ sudo apt-get update  
$ sudo /media/cs226/VBox_GAs_5.2.18/VBoxLinuxAdditions.run
```

```
# Fix the permissions  
$ sudo usermod -aG vboxsf cs226  
$ /sbin/shutdown -r now
```

Make sure to shutdown cleanly!

Java Environments

Command Line Everything

```
/*
public class SimpleCounter implements Counter {
    // Current value of the counter.
    private int value;

    /**
     * Simple assert-based unit tests for this counter.
     *
     * Make sure you run SimpleCounter with -enableassertions!
     * We'll learn a much better approach to unit testing later.
     */
    @param args Command line arguments.
 */
public static void main(String[] args) {
    Counter c = new SimpleCounter();
    assertEquals(c.value(), 0);
    System.out.println("Counter is now: " + c.value());
    c.up();
    assertEquals(c.value(), 1);
    System.out.println("Counter is now: " + c.value());
    c.down();
    assertEquals(c.value(), 0);
    System.out.println("Counter is now: " + c.value());
    c.down();
    c.up();
    c.up();
    c.up();
    System.out.println("Counter is now: " + c.value());
    assertEquals(c.value(), 2);
}
```

```
$ vim HelloWorld.java
```

```
$ javac HelloWorld.java  
$ java HelloWorld
```

*Universal, fast, flexible
Steep learning curve*

GUI Editor ***+ Command Line***

```
Demonstration

base64.cc

31 void base64_encode(const uint8_t * data, size_t length, char
32 {
33     size_t src_idx = 0;
34     size_t dst_idx = 0;
35     for (; (src_idx + 3) < length; src_idx += 3, dst_idx += 6)
36     {
37         uint8_t s0 = data[src_idx];
38         uint8_t s1 = data[src_idx + 1];
39         uint8_t s2 = data[src_idx + 2];
40
41         dst[dst_idx + 0] = charset[(s0 & 0xfc) >> 2];
42         dst[dst_idx + 1] = charset[((s0 & 0x03) << 4) | ((s1
43             & 0xf0) << 2)];
44         dst[dst_idx + 2] = charset[((s1 & 0x0f) << 2) | (s2
45             & 0xc0)];
46
47     if (src_idx < length)
48     {
49         uint8_t s0 = data[src_idx];
50         uint8_t s1 = (src_idx + 1 < length) ? data[src_idx
51
52             + 1] : 0;
53         dst[dst_idx++ >>= 2] = charset[((s0 & 0xfc) >> 2)];
54         dst[dst_idx++ >>= 2] = charset[((s0 & 0x03) << 4) | ((s1
55             & 0xf0) << 2)];
56     }
57 }

Line 31, Column 55
```

Sublime Text

```
$ javac HelloWorld.java  
$ java HelloWorld
```

Nearly universal, flexible Moderate learning curve

Integrated Development Environment (IDE)

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows a Java project named "HelloWorld" with packages "HelloWorld" and "HelloWorld.ws". The file "HelloM10.java" is selected.
- Code Editor:** Displays the Java code for "HelloM10.java":

```
1 package HelloWorld;
2
3 import javax.jws.WebService;
4
5 @WebService
6 public class HelloM10 {
7
8     /**
9      * Say hello to everyone.
10     */
11    public String helloAll() {
12        return "Hello All";
13    }
14
15    /**
16     * Protected constructor.
17     */
18    protected HelloM10() {
19    }
20
21    /**
22     * Simple said.
23     */
24    public String sayHello(String name) {
25        return "Hello, " + name;
26    }
27
28    /**
29     * Main method.
30     */
31    public static void main(String[] args) {
32        for (int i = 0; i < args.length; i++) {
33            System.out.println("Hello, " + args[i] + "!");
34        }
35    }
36}
```

- Terminal:** Shows the command-line output of the application running:

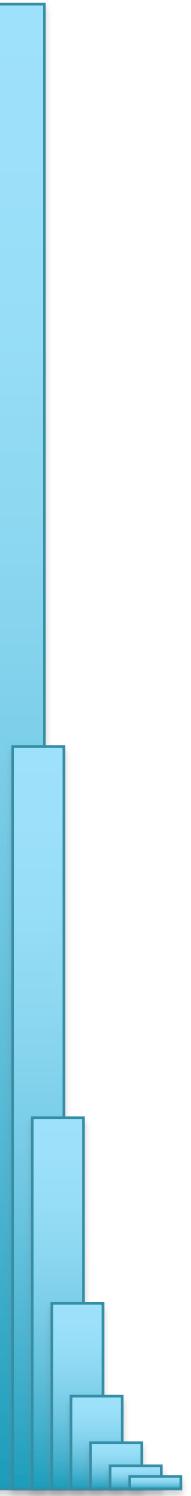
```
Hello, Mike!
Hello, Peter!
Hello, Kevin!
```

Eclipse / IntelliJ

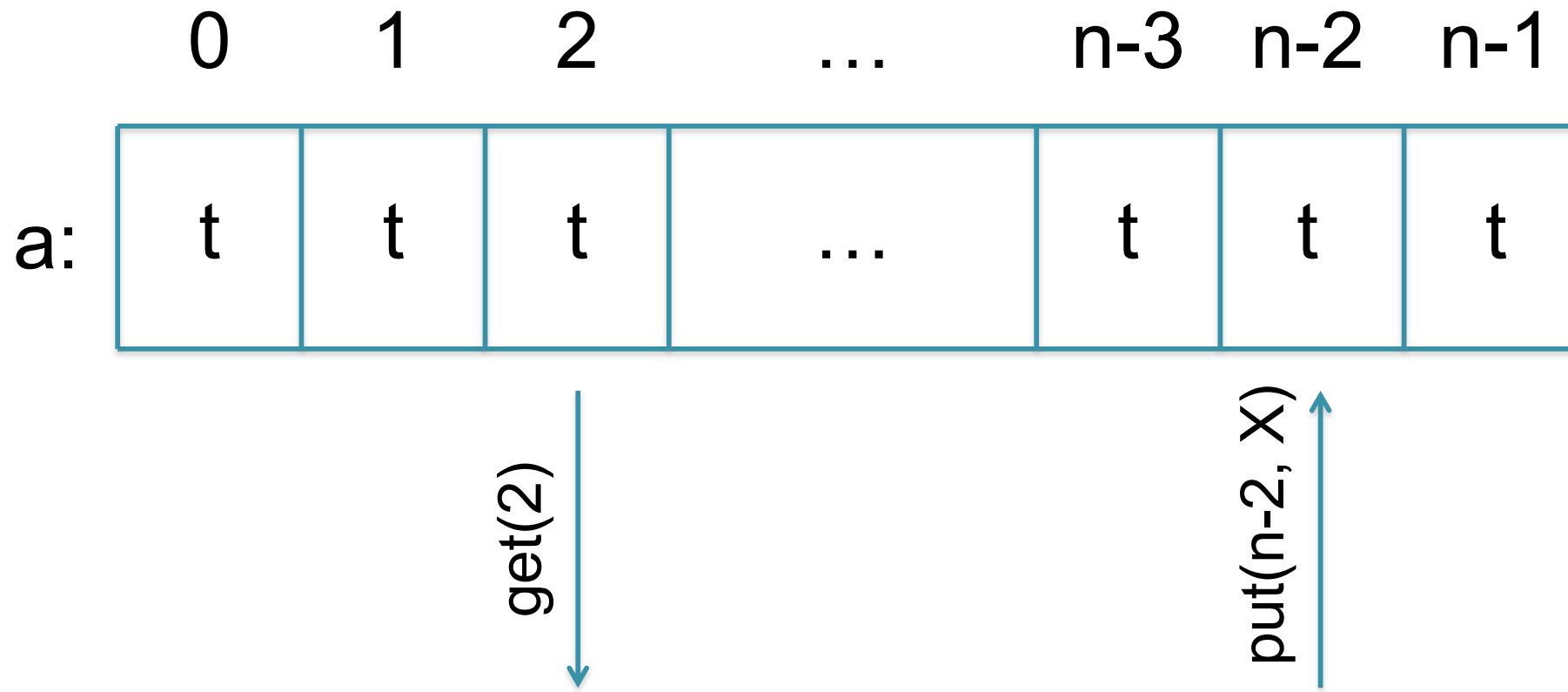
Most Support Most “magical”

*Code may not work
during grading* ☹

Agenda

- 
- 1. Review HWI***
 - 2. Review Java Arrays***
 - 3. References and Linked Lists***

ADT:Arrays



- Fixed length data structure
- Constant time $get()$ and $put()$ methods
- Definitely needs to be generic ☺

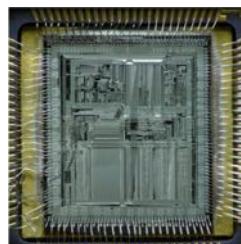
What is a computer?

[hardware]



Hard Drive

Permanent Storage – 1TB
(big, slow, cheap)



Processor
Arithmetic, logic
cores, clock speed



Display
Human Interface



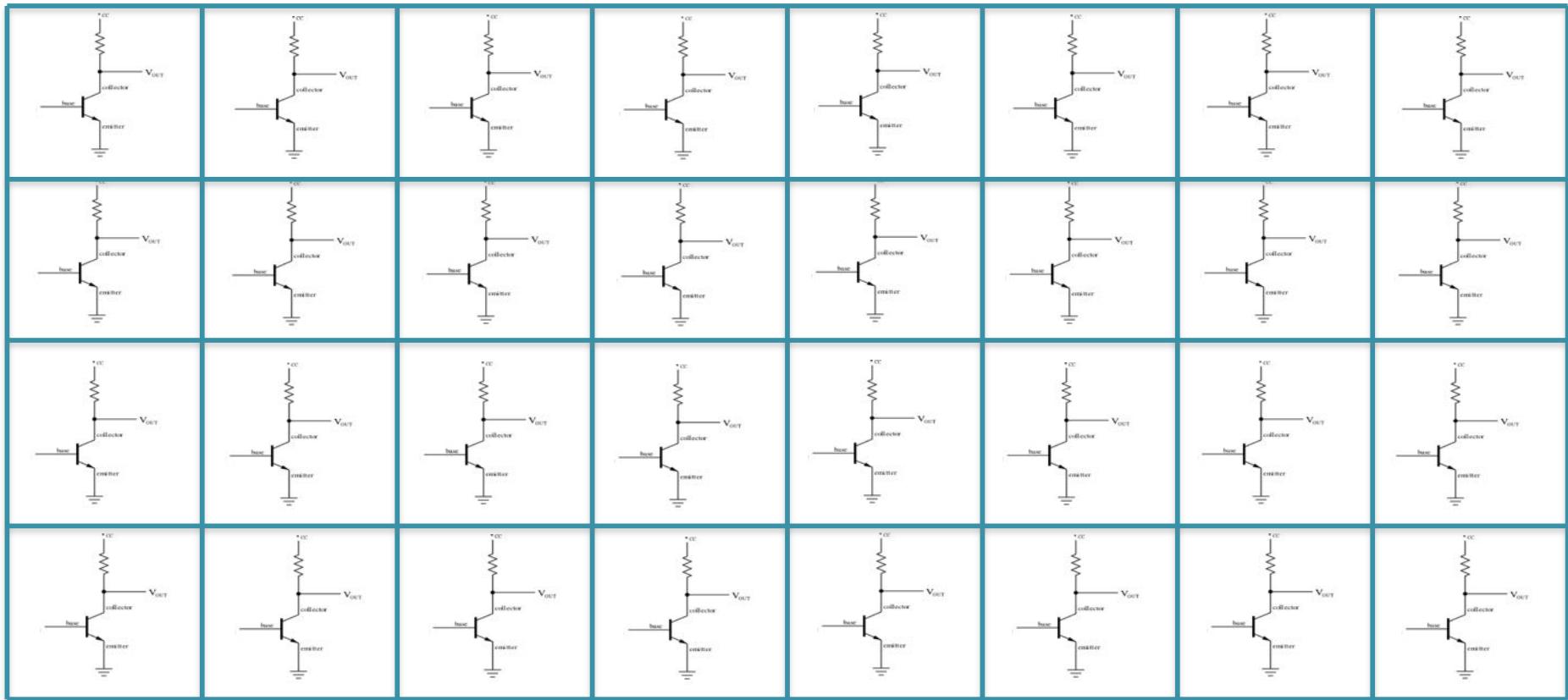
RAM

Working Storage – 8 GB
(small, fast, expensive)



Network
Computer Interface
Home: 10Mb/s, JHU: 1Gb/s

Accessing RAM

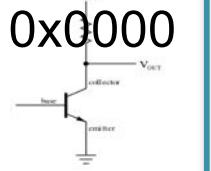
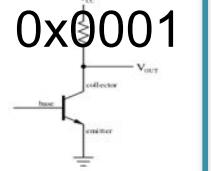
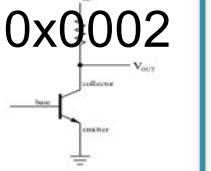
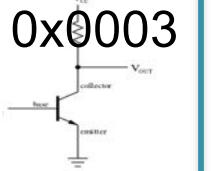
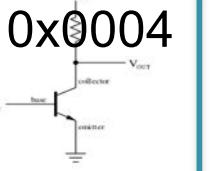
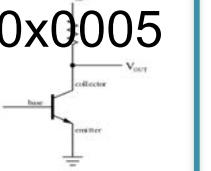
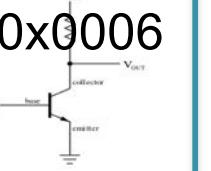
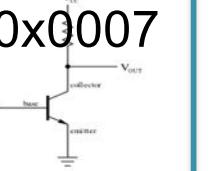
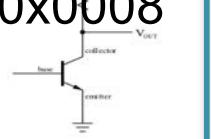
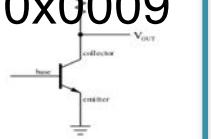
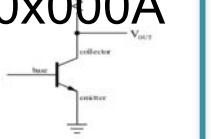
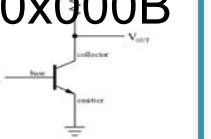
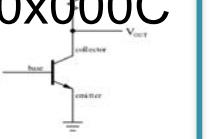
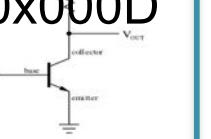
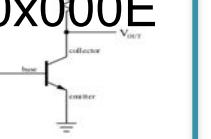
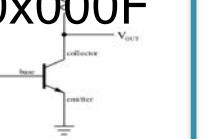
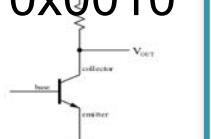
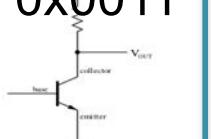
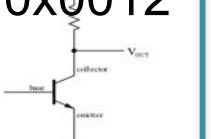
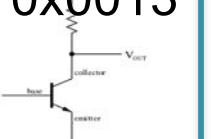
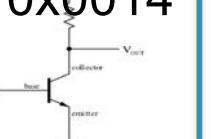
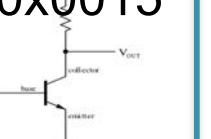
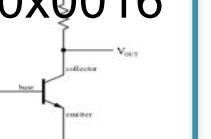
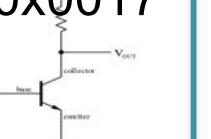
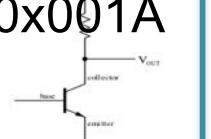
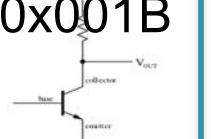
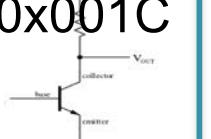
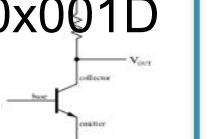
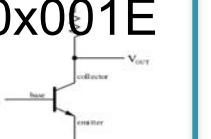
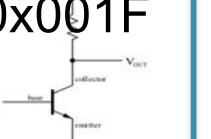


`movl var, %eax`

Move the contents of memory location var into number register %eax.

<https://docs.oracle.com/cd/E19120-01/open.solaris/817-5477/6mkuavhrv/index.html>

Accessing RAM

0x0000	0x0001	0x0002	0x0003	0x0004	0x0005	0x0006	0x0007
							
0x0008	0x0009	0x000A	0x000B	0x000C	0x000D	0x000E	0x000F
							
0x0010	0x0011	0x0012	0x0013	0x0014	0x0015	0x0016	0x0017
							
0x0018	0x0019	0x001A	0x001B	0x001C	0x001D	0x001E	0x001F
							

movl var, %eax

Move the contents of memory location var into number register %eax.

<https://docs.oracle.com/cd/E19120-01/open.solaris/817-5477/6mkuavhrv/index.html>

Accessing RAM

0x0000	0x0001	0x0002	0x0003	0x0004	0x0005	0x0006	0x0007
0x0008	0x0009	0x000A	0x000B	0x000C	0x000D	0x000E	0x000F
0x0010	0x0011	0x0012	0x0013	0x0014	0x0015	0x0016	0x0017
0x0018	0x0019	0x001A	0x001B	0x001C	0x001D	0x001E	0x001F

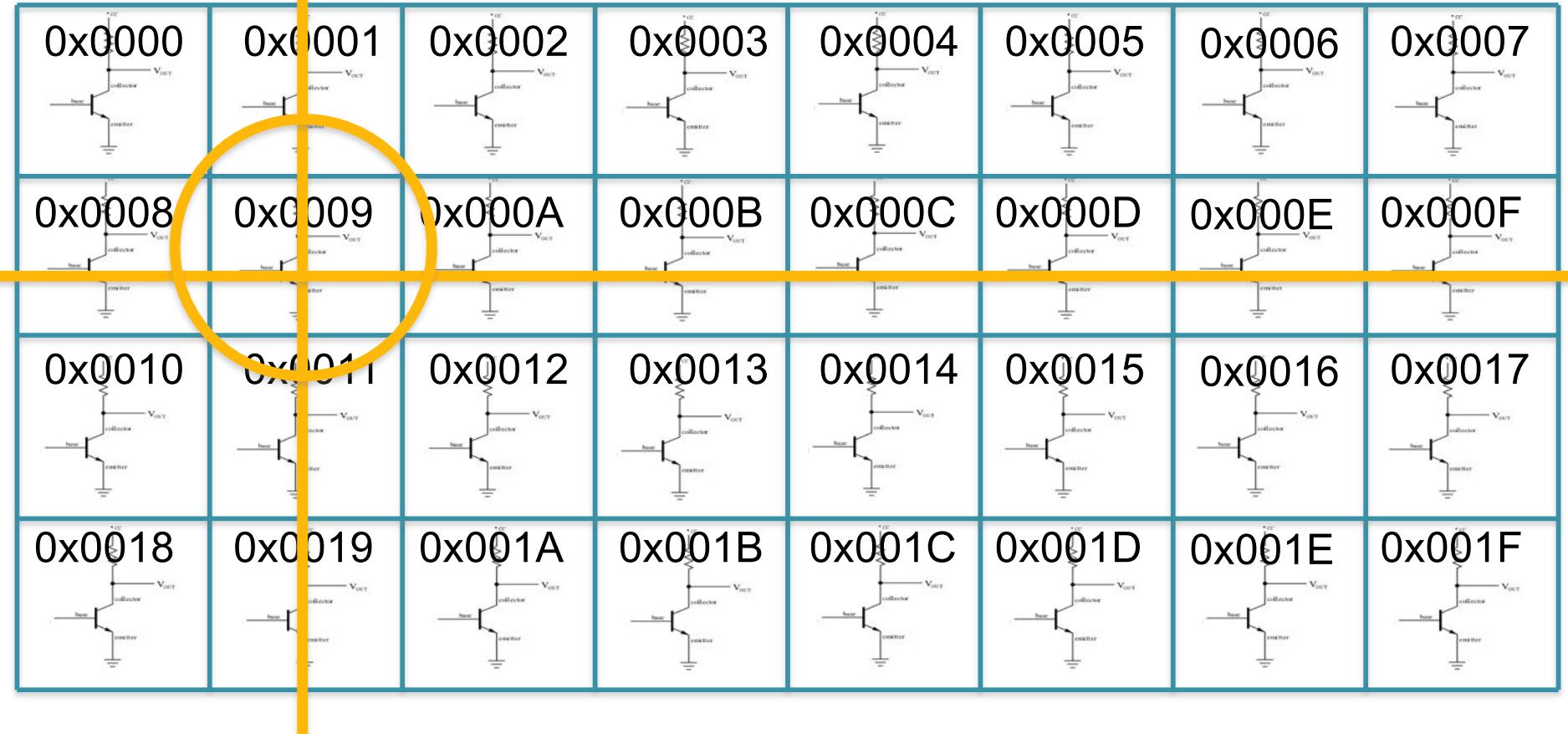
The diagram illustrates memory access using a grid of 16 transistors, each representing a memory location. The columns are labeled with addresses from 0x0000 to 0x001F. The rows are also labeled with addresses from 0x0000 to 0x001F. A yellow cross highlights the memory location at address 0x0014, which is circled in yellow. This visual representation shows how a specific memory location can be selected and accessed.

```
movl 0x0014, %eax
```

Move the contents of memory location var into number register %eax.

<https://docs.oracle.com/cd/E19120-01/open.solaris/817-5477/6mkuavhrv/index.html>

Accessing RAM



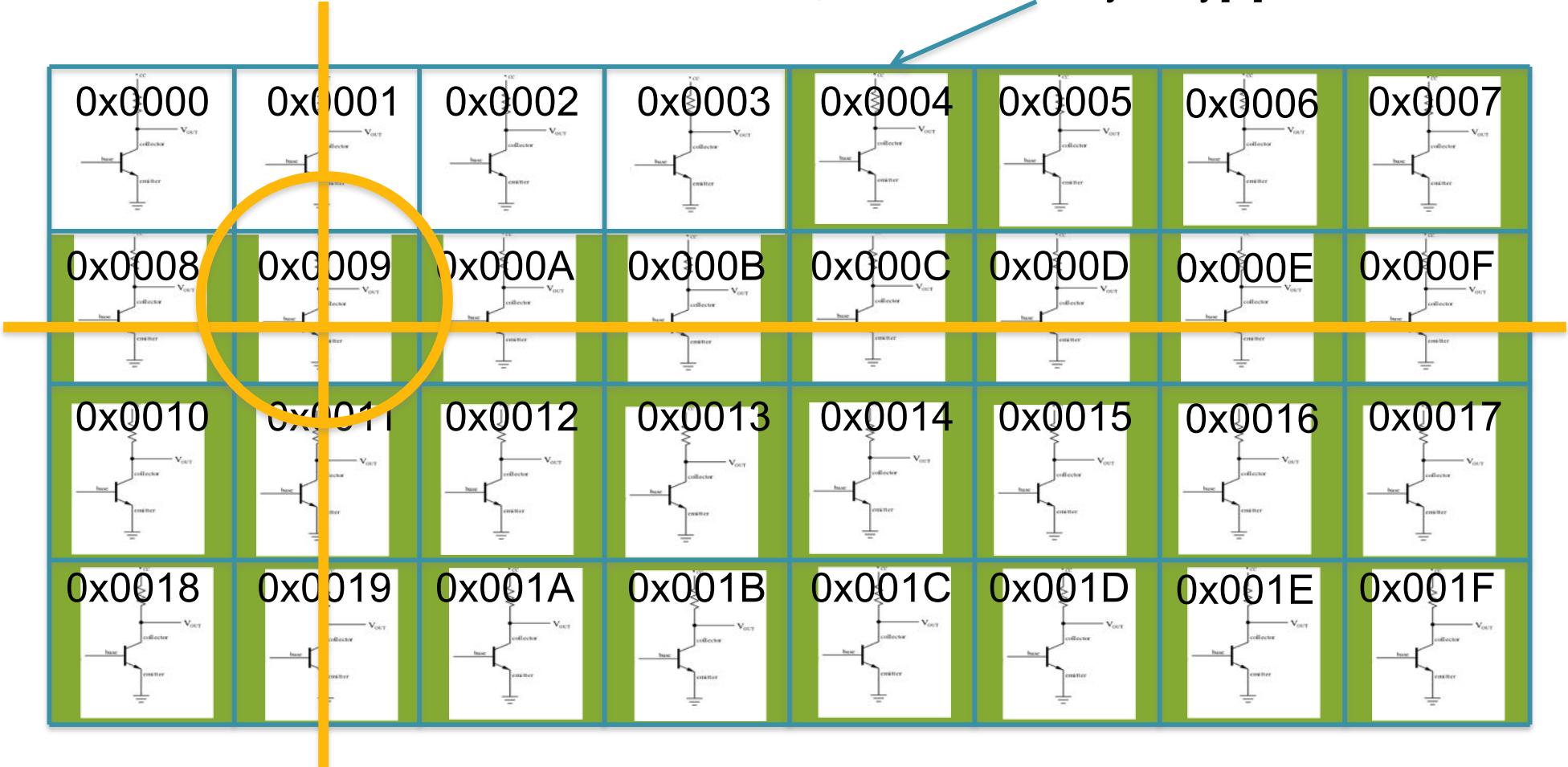
```
movl 0x0009, %eax
```

Move the contents of memory location var into number register %eax.

<https://docs.oracle.com/cd/E19120-01/open.solaris/817-5477/6mkuavhrv/index.html>

Accessing RAM

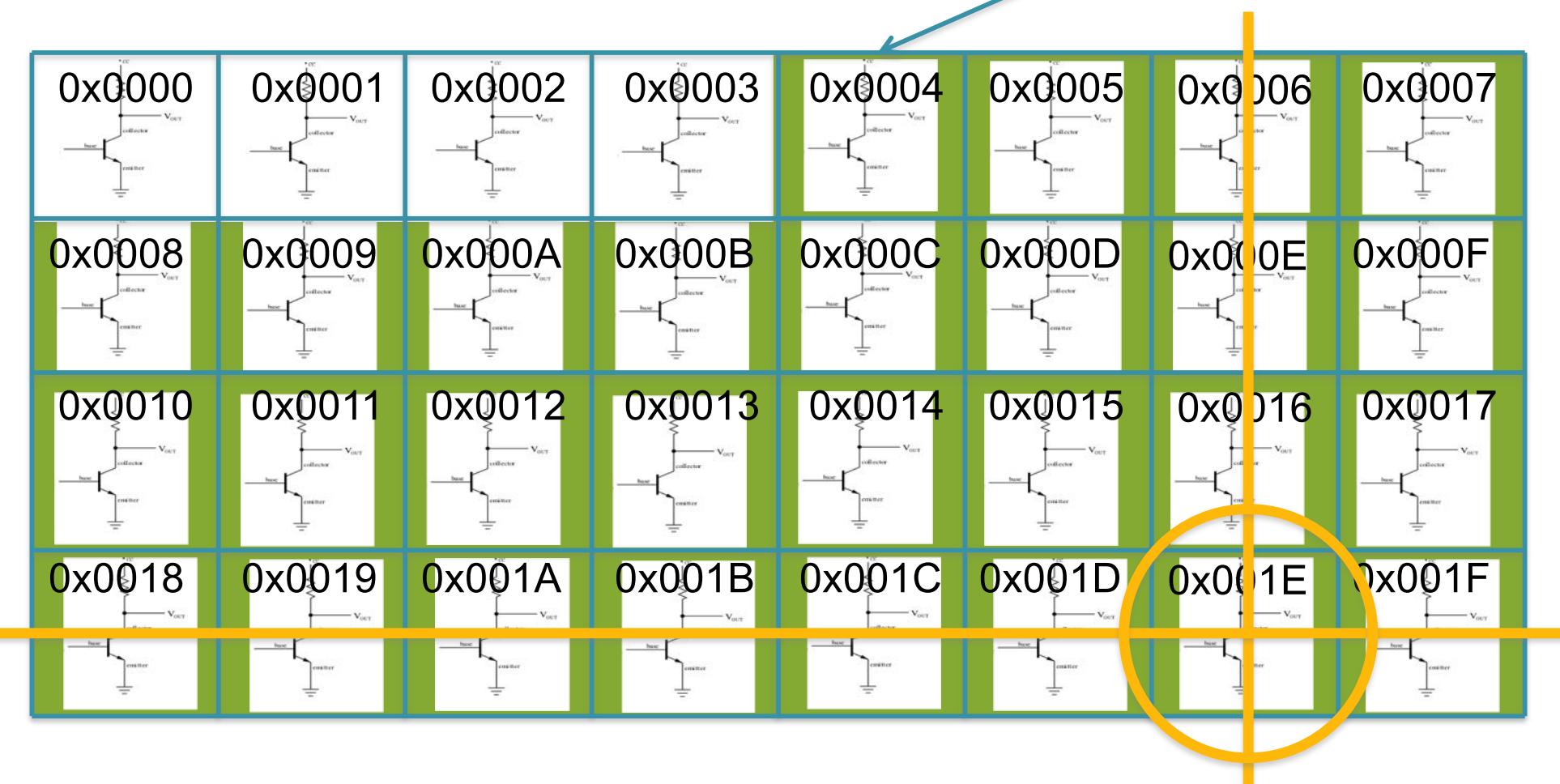
myarray[0]



```
Byte [] myarray = new myarray[5000]; // myarray now starts at 0x0004
```

```
myarray[5] => offset for myarray + 5 * (sizeof type) => 0x04 + 0x05 * 1 => movl 0x09, %eax
```

Accessing RAM



```
Byte [] myarray = new myarray[5000]; // myarray now starts at 0x0004
```

```
myarray[5] => offset for myarray + 5 * (sizeof type) => 0x04 + 0x05 * 1 => movl 0x09, %eax
```

```
myarray[26] => offset for myarray + 26 * (sizeof type) => 0x04 + 0x01A * 1 => movl 0x1E, %eax
```

Simple Array Implementation

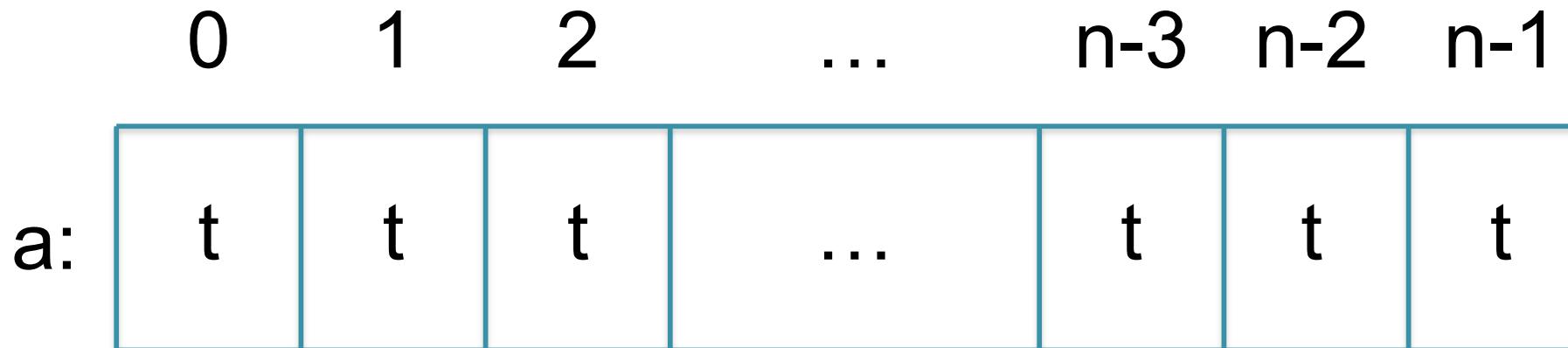
```
1 public class SimpleArray<T> implements Array<T> {
2     private T[] data;
3
4     public SimpleArray(int n, T t) throws LengthException {
5         if (n <= 0) {
6             throw new LengthException();
7         }
8
9         this.data = (T[]) new Object[n];
10
11        for (int i = 0; i < n; i++) {
12            this.data[i] = t;
13        }
14    }
15
16    public T get(int i) throws IndexException {
17        try {
18            return this.data[i];
19        } catch (ArrayIndexOutOfBoundsException e) {
20            throw new IndexException();
21        }
22    }
23
24    public void put(int i, T t) throws IndexException {
25        try {
26            this.data[i] = t;
27        } catch (ArrayIndexOutOfBoundsException e) {
28            throw new IndexException();
29        }
30    }
31
32    public int length() {
33        return this.data.length;
34    }
35}
36}
```

Generic class implementation for a generic interface

Uses a Object array as the backing storage, but we could replace it with something else if needed

Use our own exception types to encapsulate (hide) the underlying data types

ADT:Arrays



- Fixed length data structure
- Constant time get() and put() methods
- Definitely needs to be generic ☺

What are the limitations of arrays?

- Fixed length data structure
 - wastes space and/or cant grow
- Adding/removing from the middle is slow
- Searching can be slow (if not sorted)

Agenda

- 1. Review HWI***
- 2. Review Java Arrays***
- 3. Java References and Linked Lists***

Java References

Student.java

```
public class Student {  
    private String name;  
    private int grade;  
  
    public Student(String n, int g) {  
        this.name = n;  
        this.grade = g;  
    }  
  
    public void setName(String s) {  
        this.name = s;  
    }  
  
    public void setGrade(int g) {  
        this.grade = g;  
    }  
}  
  
Student s = new Student("Mike", 100);
```

s



Student

String name: "Mike"
int grade: 100

Java References

Student.java

```
public class Student {  
    private String name;  
    private int grade;  
  
    public Student(String n, int g) {  
        this.name = n;  
        this.grade = g;  
    }  
  
    public void setName(String s) {  
        this.name = s;  
    }  
  
    public void setGrade(int g) {  
        this.grade = g;  
    }  
  
    Student s = new Student("Mike", 100);  
    s.setName("Peter");  
    s.setGrade(95);
```

s



Student

String name: "Peter"
int grade: 95

Java References

Student.java

```
public class Student {  
    private String name;  
    private int grade;  
    private Student partner;  
  
    ...  
  
    public void setPartner(Student p) {  
        this.partner = p;  
    }  
}  
  
Student s1 = new Student("Mike", 100);  
Student s2 = new Student("Peter", 95);  
  
s1.setPartner(s2);  
s2.setPartner(s1);
```

```
System.out.println(s1.getName() + " " + s1.getGrade()  
                  + " " + s1.getPartner().getName());  
System.out.println(s2.getName() + " " + s2.getGrade()  
                  + " " + s2.getPartner().getName());
```

Student

String name: "Mike"
int grade: 100
Student partner: <s2>

s1

Student

String name: "Peter"
int grade: 95
Student partner: <s1>

s2

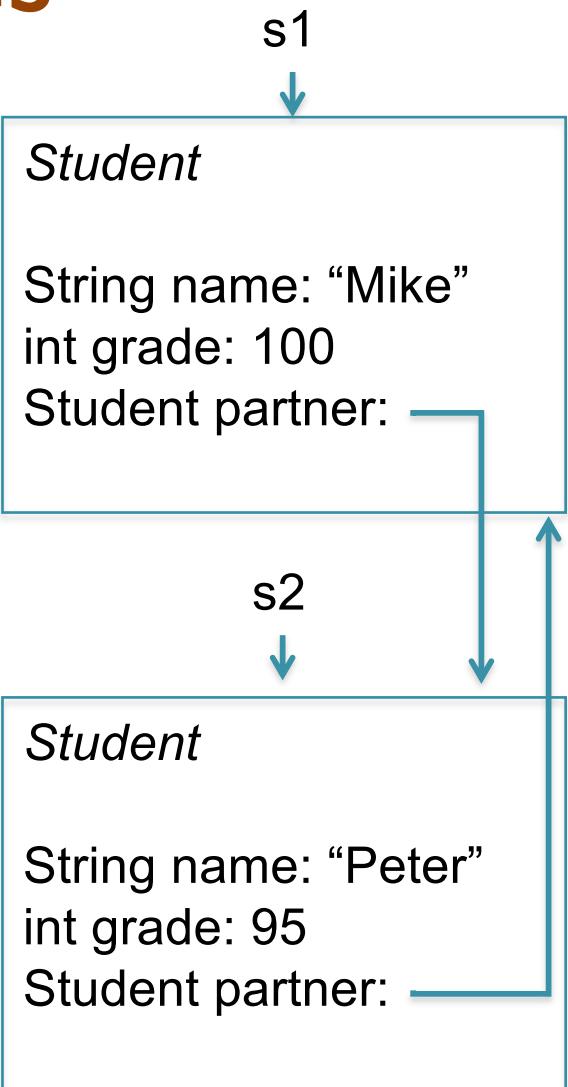
Mike 100 Peter
Peter 95 Mike

Java References

Student.java

```
public class Student {  
    private String name;  
    private int grade;  
    private Student partner;  
  
    ...  
  
    public void setPartner(Student p) {  
        this.partner = p;  
    }  
}  
  
Student s1 = new Student("Mike", 100);  
Student s2 = new Student("Peter", 95);  
  
s1.setPartner(s2);  
s2.setPartner(s1);
```

s1 and s2 are “just” references to the “Mike” and “Peter” objects



```
System.out.println(s1.getName() + " " + s1.getGrade()  
                  + " " + s1.getPartner().getName());  
System.out.println(s2.getName() + " " + s2.getGrade()  
                  + " " + s2.getPartner().getName());
```

Mike 100 Peter
Peter 95 Mike

Java References

```
Student s1 = new Student("Mike", 100);
Student s2 = new Student("Peter", 95);

s1.setPartner(s2);
s2.setPartner(s1);

s2 = new Student("Kelly", 99);

System.out.println(s1.getName() + " " + s1.getGrade()
                  + " " + s1.getPartner().getName());
System.out.println(s2.getName() + " " + s2.getGrade()
                  + " " + s2.getPartner().getName());
```

Any Guesses?

Java References

```
Student s1 = new Student("Mike", 100);
Student s2 = new Student("Peter", 95);

s1.setPartner(s2);
s2.setPartner(s1);

s2 = new Student("Kelly", 99);

System.out.println(s1.getName() + " " + s1.getGrade()
                  + " " + s1.getPartner().getName());
System.out.println(s2.getName() + " " + s2.getGrade()
                  + " " + s2.getPartner().getName());
```

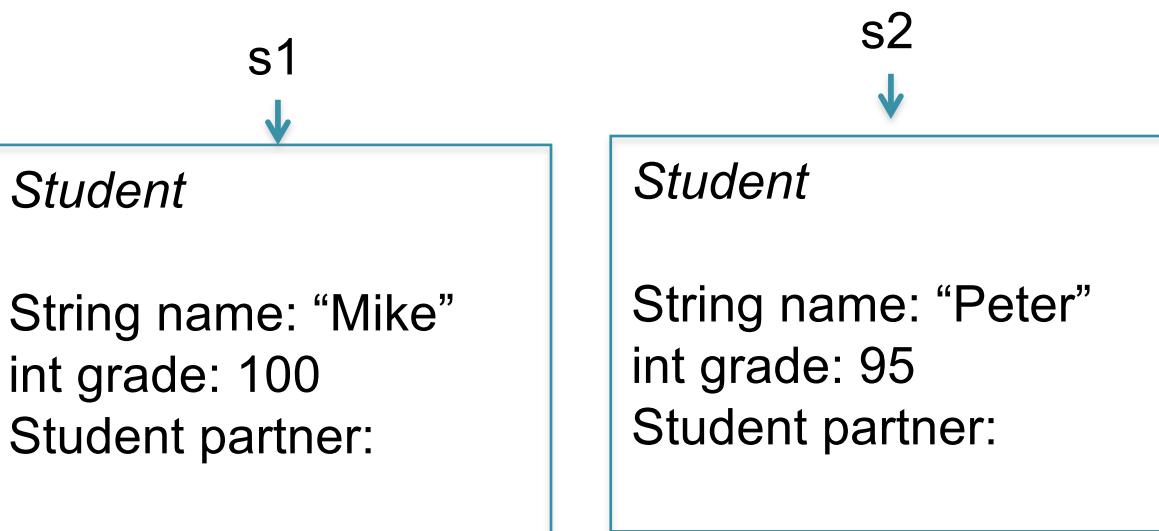
Java References

```
Student s1 = new Student("Mike", 100);
Student s2 = new Student("Peter", 95);

s1.setPartner(s2);
s2.setPartner(s1);

s2 = new Student("Kelly", 99);

System.out.println(s1.getName() + " " + s1.getGrade()
                  + " " + s1.getPartner().getName());
System.out.println(s2.getName() + " " + s2.getGrade()
                  + " " + s2.getPartner().getName());
```



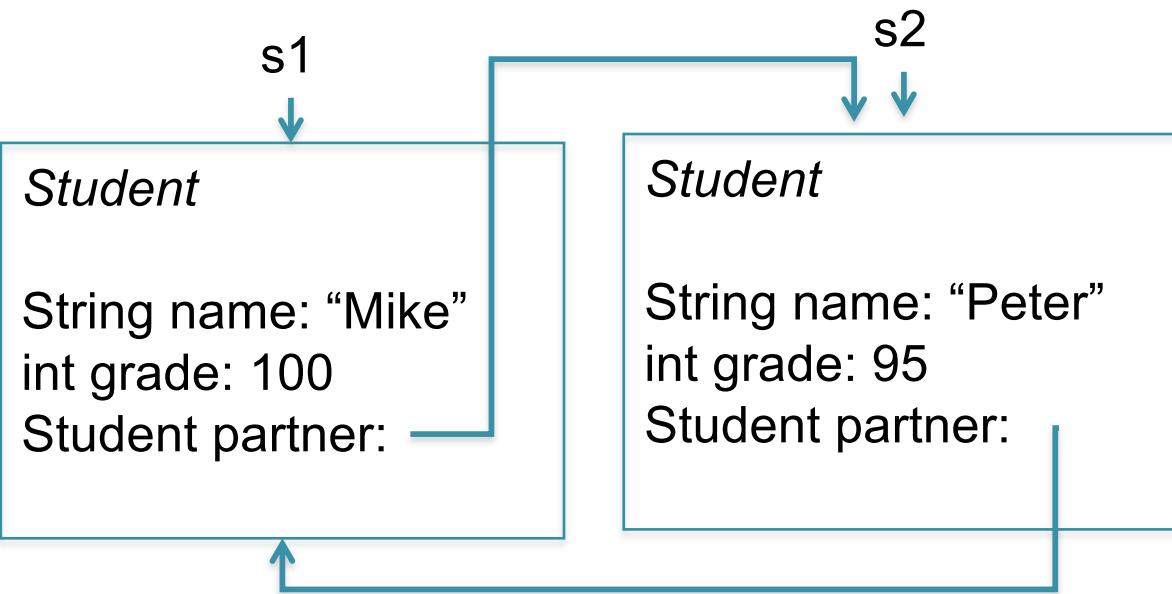
Java References

```
Student s1 = new Student("Mike", 100);
Student s2 = new Student("Peter", 95);

s1.setPartner(s2);
s2.setPartner(s1);

s2 = new Student("Kelly", 99);

System.out.println(s1.getName() + " " + s1.getGrade()
                  + " " + s1.getPartner().getName());
System.out.println(s2.getName() + " " + s2.getGrade()
                  + " " + s2.getPartner().getName());
```



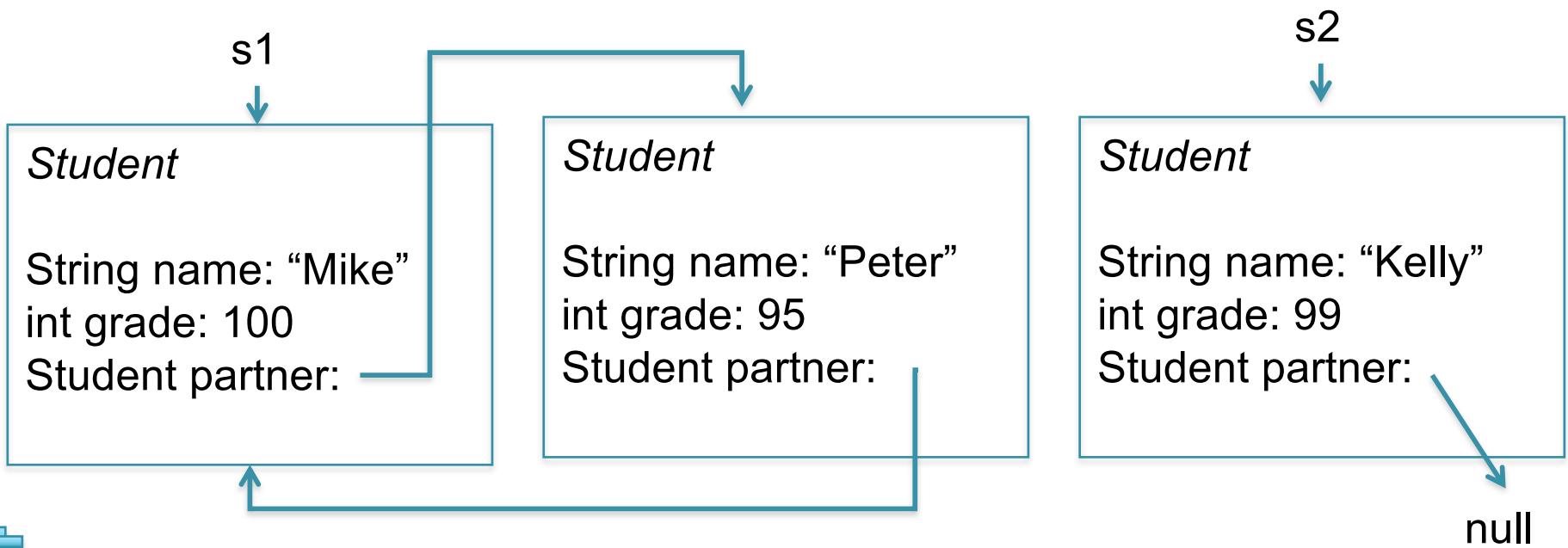
Java References

```
Student s1 = new Student("Mike", 100);
Student s2 = new Student("Peter", 95);

s1.setPartner(s2);
s2.setPartner(s1);

s2 = new Student("Kelly", 99);

System.out.println(s1.getName() + " " + s1.getGrade()
                  + " " + s1.getPartner().getName());
System.out.println(s2.getName() + " " + s2.getGrade()
                  + " " + s2.getPartner().getName());
```



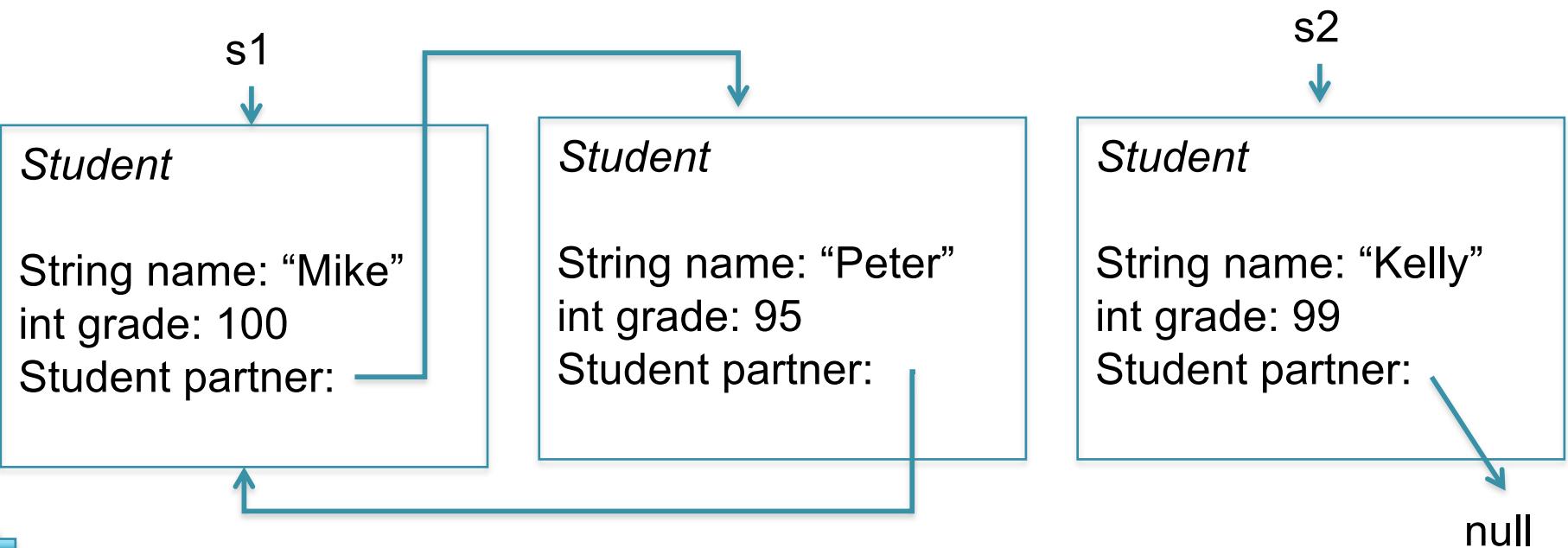
Java References

```
Student s1 = new Student("Mike", 100);
Student s2 = new Student("Peter", 95);
```

```
Mike 100 Peter (s2);
s2.setPartner(s1);
```

```
s2 = new Student("Kelly", 99);
```

```
System.out.println(s1.getName() + " " + s1.getGrade()
+ " " + s1.getPartner().getName());
System.out.println(s2.getName() + " " + s2.getGrade()
+ " " + s2.getPartner().getName());
```

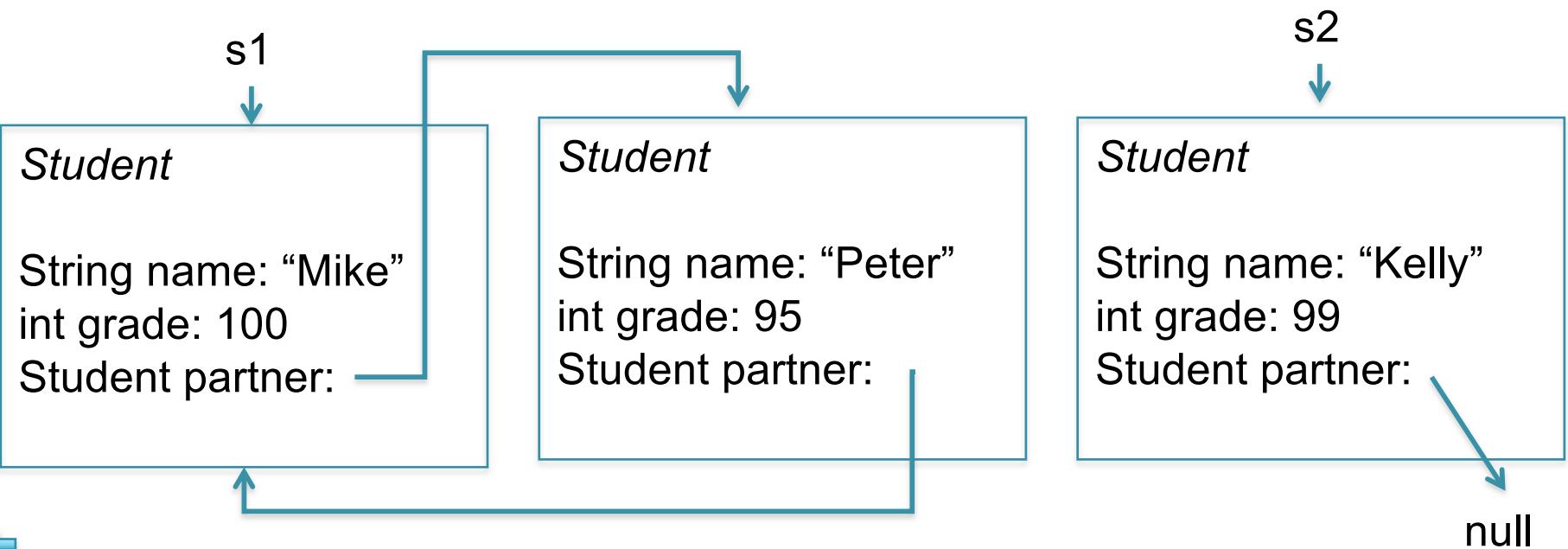


Java References

```
Student s1 = new Student("Mike", 100);
Student s2 = new Student("Peter", 95);
```

```
Mike 100 Peter
Exception in thread "main" java.lang.NullPointerException
at Student.main(Student.java:48)
s2 = new Student("Kelly", 99);
```

```
System.out.println(s1.getName() + " " + s1.getGrade()
+ " " + s1.getPartner().getName());
System.out.println(s2.getName() + " " + s2.getGrade()
+ " " + s2.getPartner().getName());
```

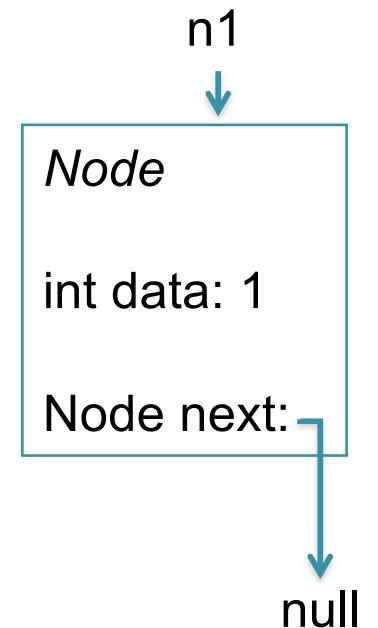


Java Nodes

```
public class Node {  
    private int data;  
    private Node next;  
  
    public Node(int d) {  
        this.data = d;  
        this.next = null; // will do this by default  
    }  
  
    public void setNext(Node n) {  
        this.next = n;  
    }  
  
    public Node getNext() {  
        return this.next;  
    }  
  
    public int getData() {  
        return this.data;  
    }  
}
```

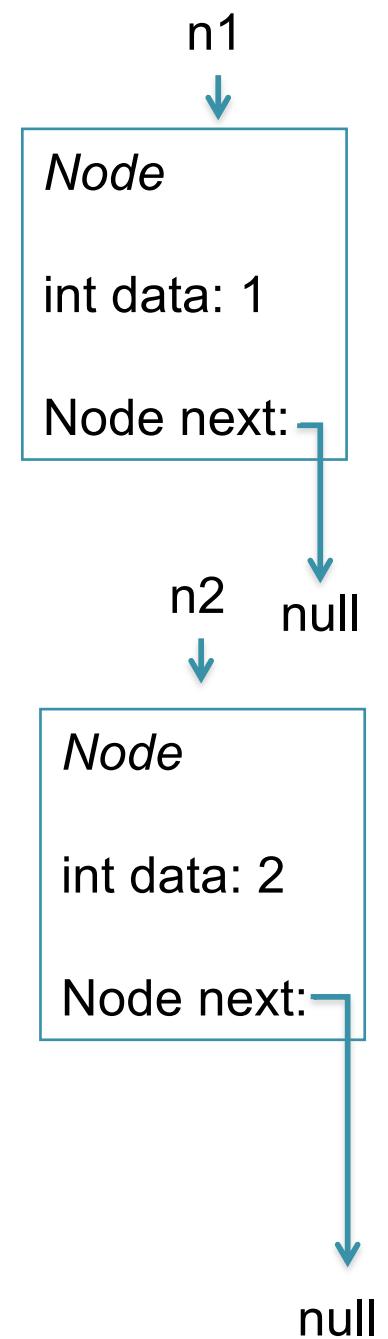
Java Nodes

```
public class Node {  
    private int data;  
    private Node next;  
  
    public Node(int d) {  
        this.data = d;  
        this.next = null; // will do this by default  
    }  
  
    public void setNext(Node n) {  
        this.next = n;  
    }  
  
    public Node getNext() {  
        return this.next;  
    }  
  
    public int getData() {  
        return this.data;  
    }  
  
Node n1 = new Node(1);
```



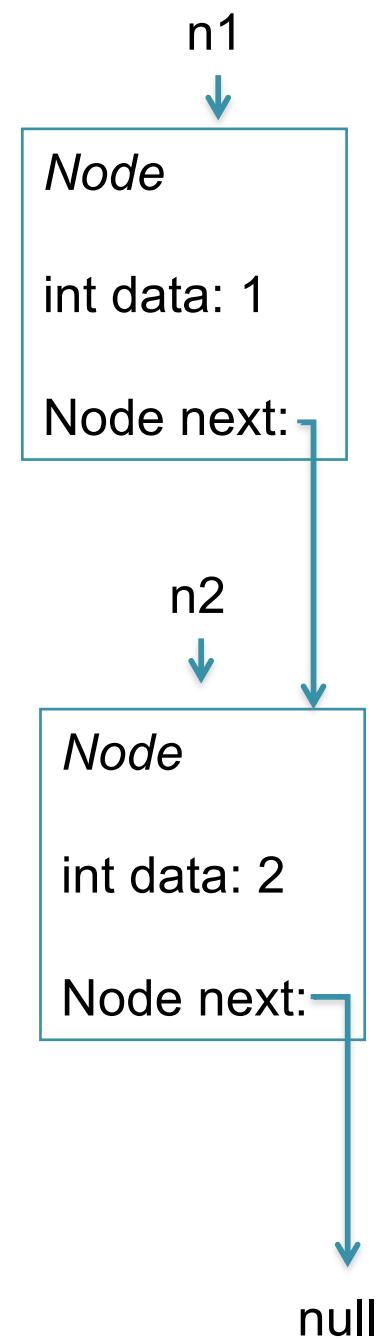
Java Nodes

```
public class Node {  
    private int data;  
    private Node next;  
  
    public Node(int d) {  
        this.data = d;  
        this.next = null; // will do this by default  
    }  
  
    public void setNext(Node n) {  
        this.next = n;  
    }  
  
    public Node getNext() {  
        return this.next;  
    }  
  
    public int getData() {  
        return this.data;  
    }  
  
Node n1 = new Node(1);  
Node n2 = new Node(2);
```



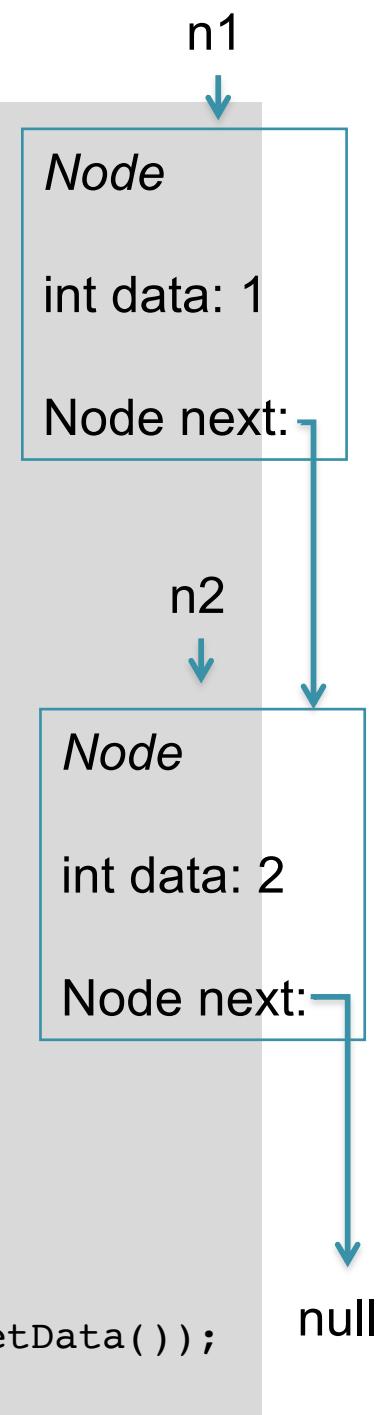
Java Nodes

```
public class Node {  
    private int data;  
    private Node next;  
  
    public Node(int d) {  
        this.data = d;  
        this.next = null; // will do this by default  
    }  
  
    public void setNext(Node n) {  
        this.next = n;  
    }  
  
    public Node getNext() {  
        return this.next;  
    }  
  
    public int getData() {  
        return this.data;  
    }  
  
Node n1 = new Node(1);  
Node n2 = new Node(2);  
n1.setNext(n2);
```



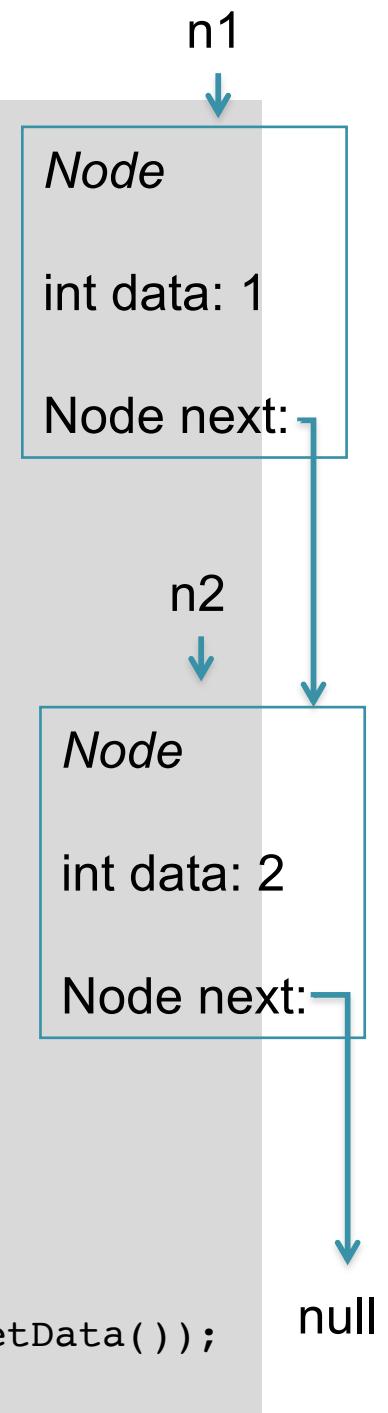
Java Nodes

```
public class Node {  
    private int data;  
    private Node next;  
  
    public Node(int d) {  
        this.data = d;  
        this.next = null; // will do this by default  
    }  
  
    public void setNext(Node n) {  
        this.next = n;  
    }  
  
    public Node getNext() {  
        return this.next;  
    }  
  
    public int getData() {  
        return this.data;  
    }  
  
Node n1 = new Node(1);  
Node n2 = new Node(2);  
n1.setNext(n2);  
  
System.out.println(n1.getData() + " -> " + n1.getNext().getData());
```



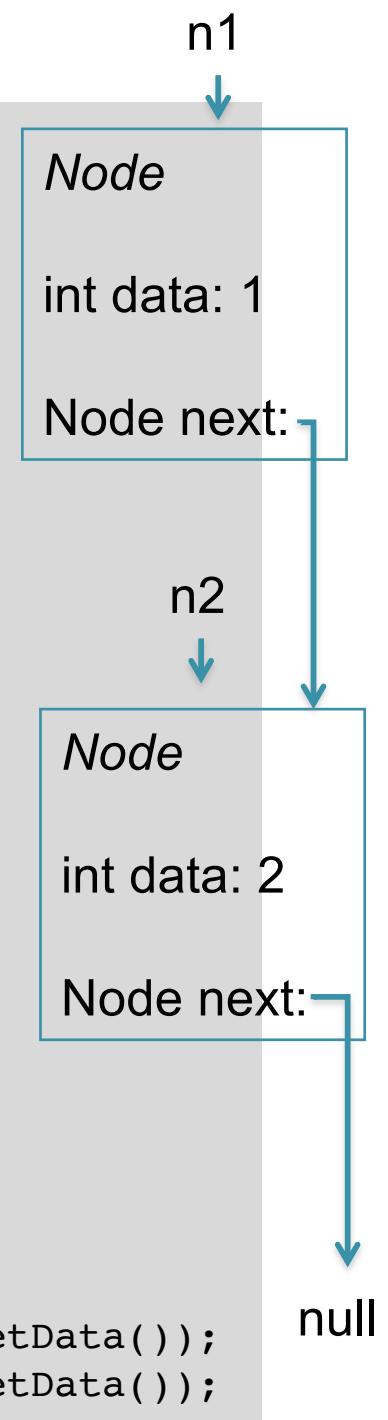
Java Nodes

```
public class Node {  
    private int data;  
    private Node next;  
  
    public Node(int d) {  
        this.data = d;  
        this.next = null; // will do this by default  
    }  
  
    public void setNext(Node n) {  
        this.next = n;  
    }  
  
    public Node getNext() {  
        return this.next;  
    }  
    1 -> 2    public int getData() {  
        return this.data;  
    }  
  
Node n1 = new Node(1);  
Node n2 = new Node(2);  
n1.setNext(n2);  
  
System.out.println(n1.getData() + " -> " + n1.getNext().getData());
```



Java Nodes

```
public class Node {  
    private int data;  
    private Node next;  
  
    public Node(int d) {  
        this.data = d;  
        this.next = null; // will do this by default  
    }  
  
    public void setNext(Node n) {  
        this.next = n;  
    }  
  
    public Node getNext() {  
        return this.next;  
    }  
    1 -> 2    public int getData() {  
        return this.data;  
    }  
  
Node n1 = new Node(1);  
Node n2 = new Node(2);  
n1.setNext(n2);  
  
System.out.println(n1.getData() + " -> " + n1.getNext().getData());  
System.out.println(n2.getData() + " -> " + n2.getNext().getData());
```



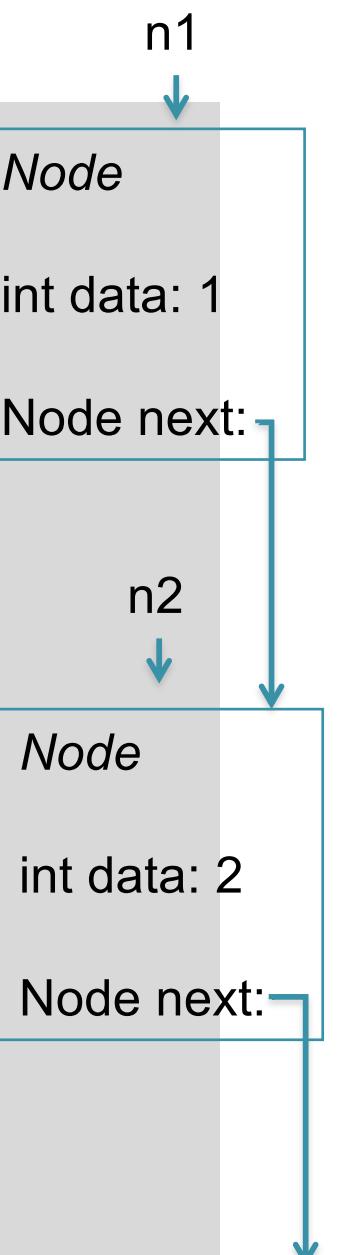
Java Nodes

```
public class Node {  
    private int data;  
    private Node next;  
  
    public Node(int d) {  
        this.data = d;  
        this.next = null; // will do this by default  
    }  
  
    public void setNext(Node n) {  
        this.next = n;  
    }  
  
    public Node getNext() {  
        return this.next;  
    }  
}
```

```
1 -> 2  
Exception in thread "main" java.lang.NullPointerException  
at Node.main(Node.java:29)
```

```
Node n1 = new Node(1);  
Node n2 = new Node(2);  
n1.setNext(n2);
```

```
System.out.println(n1.getData() + " -> " + n1.getNext().getData());  
System.out.println(n2.getData() + " -> " + n2.getNext().getData());
```



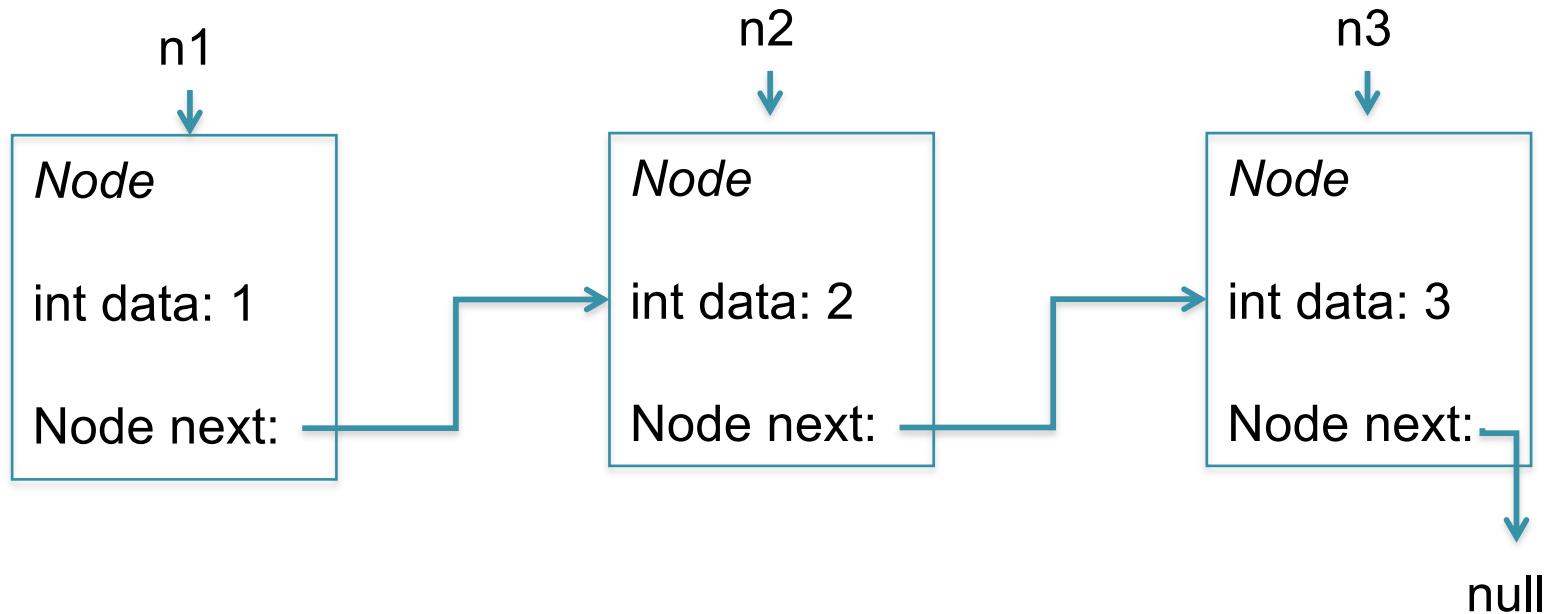
null

Java Nodes

```
Node n1 = new Node(1);
Node n2 = new Node(2);
Node n3 = new Node(3);
n1.setNext(n2);
n2.setNext(n3);

System.out.println(n1.getData() + " -> " + n1.getNext().getData());
System.out.println(n2.getData() + " -> " + n2.getNext().getData());
System.out.println(n3.getData() + " -> " + n3.getNext().getData());
```

1 -> 2

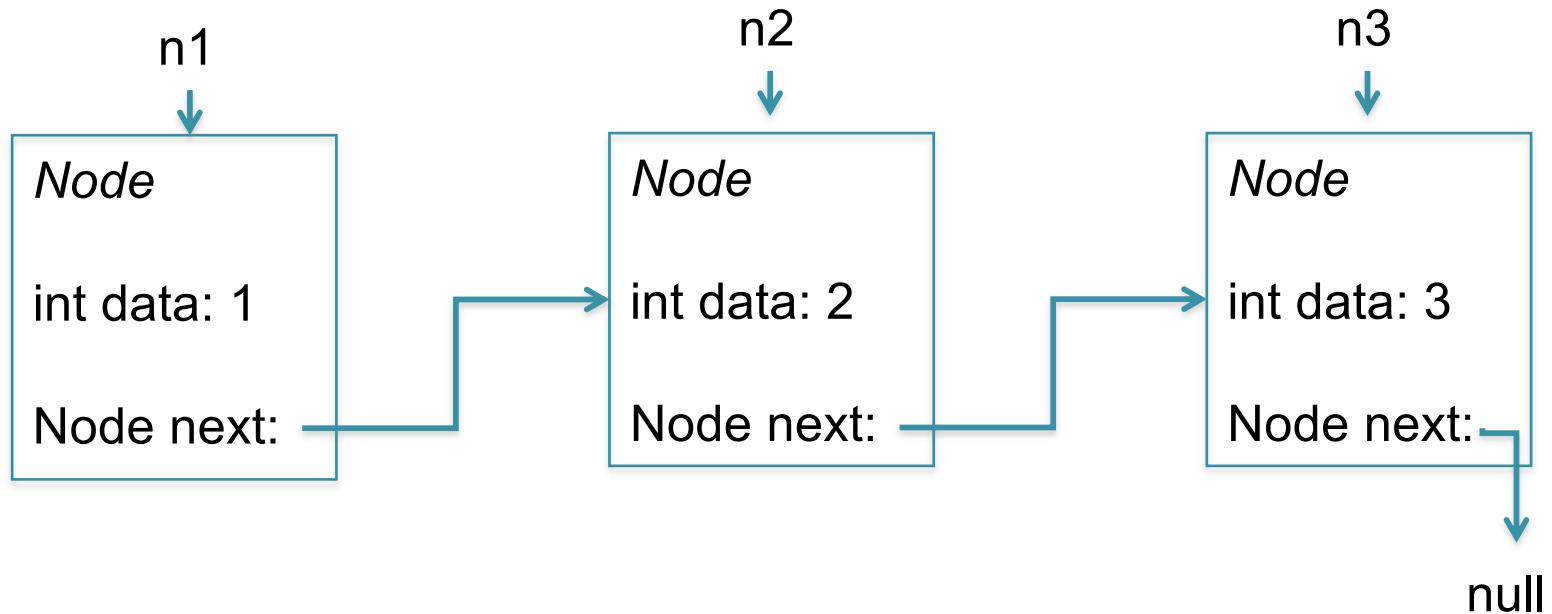


Java Nodes

```
Node n1 = new Node(1);
Node n2 = new Node(2);
Node n3 = new Node(3);
n1.setNext(n2);
n2.setNext(n3);

System.out.println(n1.getData() + " -> " + n1.getNext().getData());
System.out.println(n2.getData() + " -> " + n2.getNext().getData());
System.out.println(n3.getData() + " -> " + n3.getNext().getData());
```

1 -> 2
2 -> 3

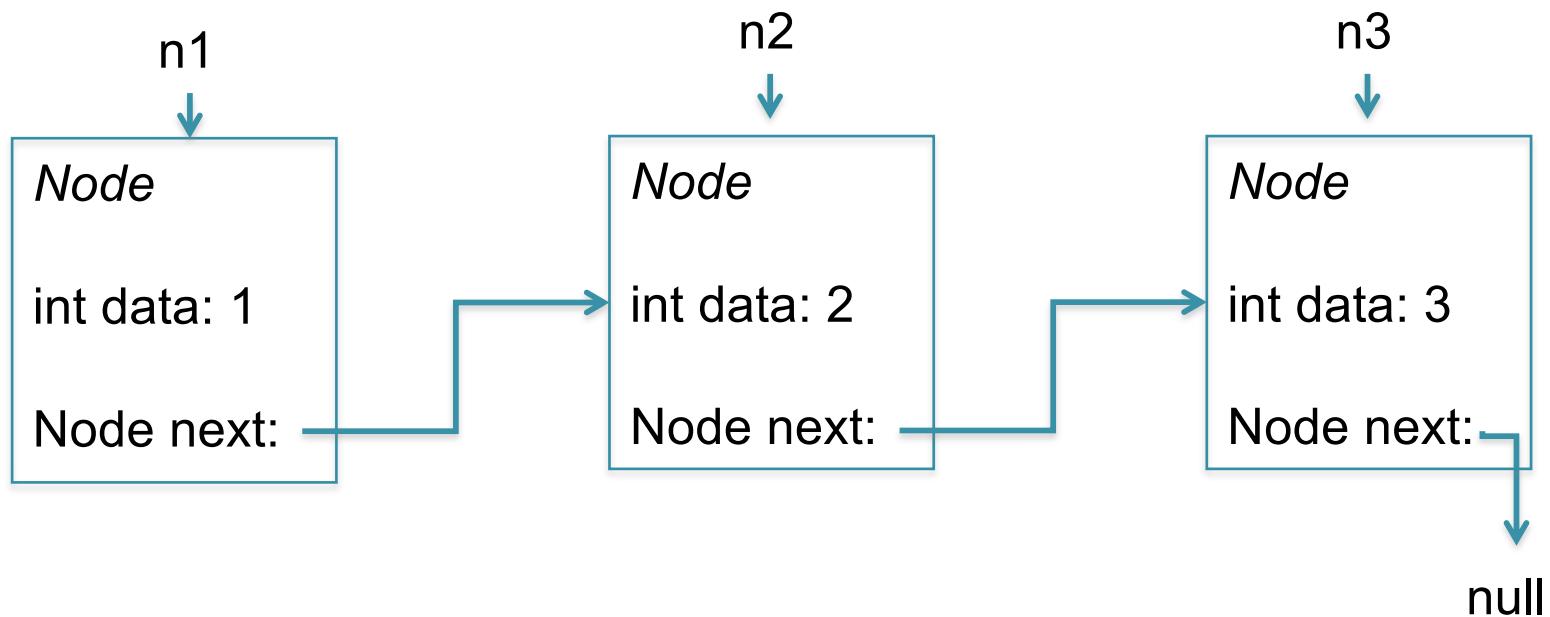


Java Nodes

```
Node n1 = new Node(1);
Node n2 = new Node(2);
Node n3 = new Node(3);
n1.setNext(n2);
n2.setNext(n3);

System.out.println(n1.getData() + " -> " + n1.getNext().getData());
System.out.println(n2.getData() + " -> " + n2.getNext().getData());
System.out.println(n3.getData() + " -> " + n3.getNext().getData());
```

```
1 -> 2
2 -> 3
Exception in thread "main" java.lang.NullPointerException
at Node.main(Node.java:32)
```

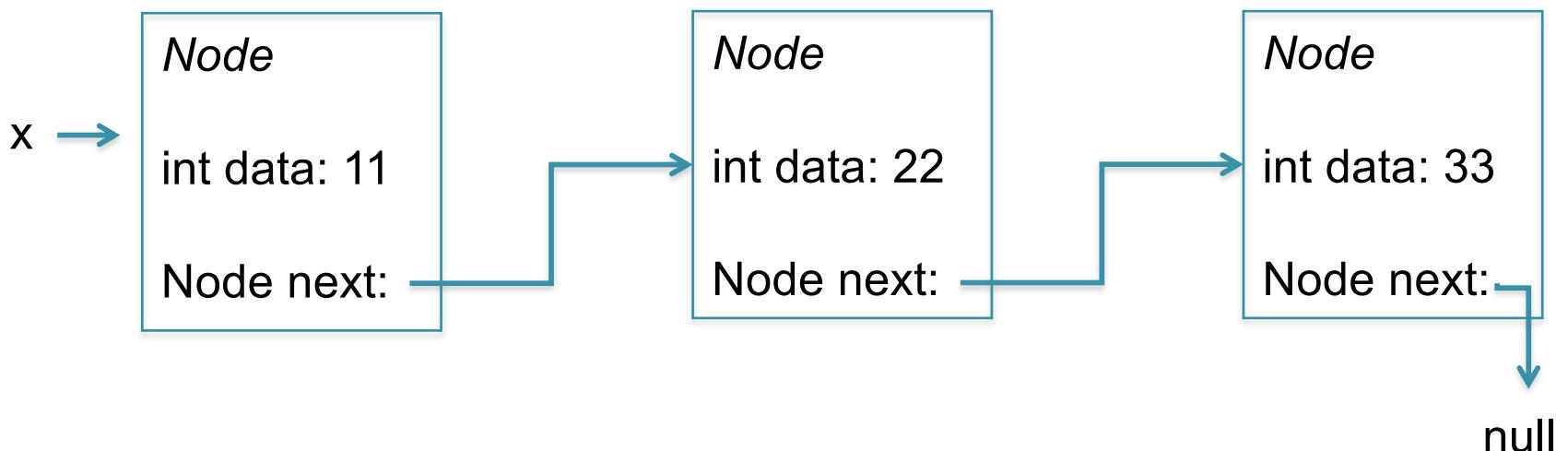


Java Nodes

```
Node x = new Node(11);
x.setNext(new Node(22));
x.getNext().setNext(new Node(33));

System.out.println(x.getData() + " -> " +
                   x.getNext().getData());
System.out.println(x.getNext().getData() + " -> " +
                   x.getNext().getNext().getData());
System.out.println(x.getNext().getNext().getData() + " -> " +
                   x.getNext().getNext().getNext().getData());
```

11 -> 22

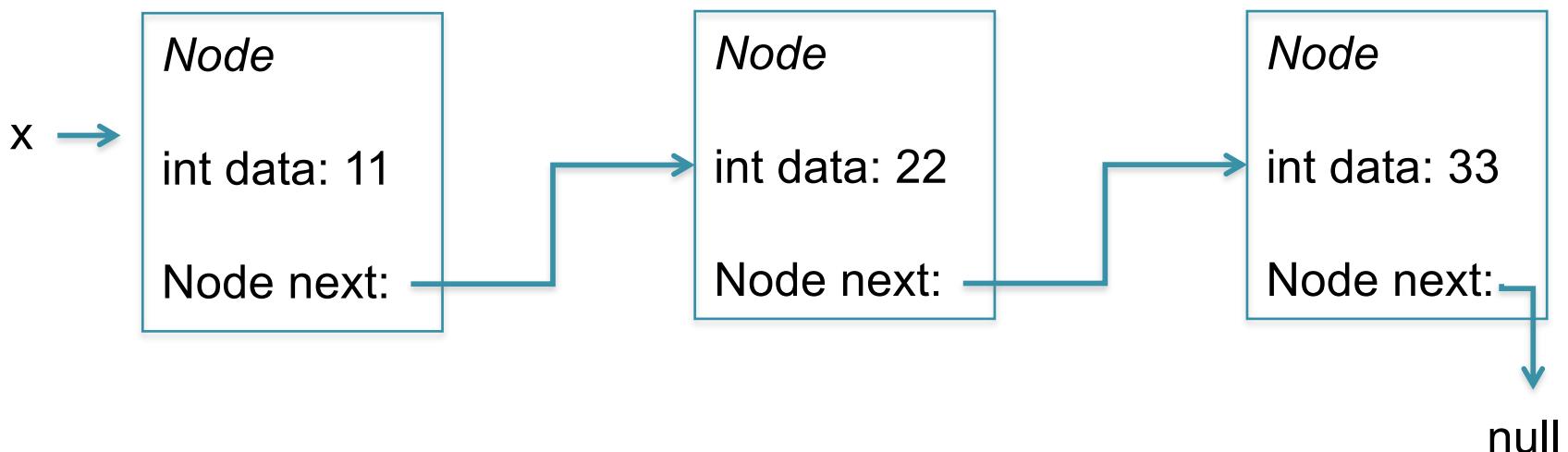


Java Nodes

```
Node x = new Node(11);
x.setNext(new Node(22));
x.getNext().setNext(new Node(33));

System.out.println(x.getData() + " -> " +
                   x.getNext().getData());
System.out.println(x.getNext().getData() + " -> " +
                   x.getNext().getNext().getData());
System.out.println(x.getNext().getNext().getData() + " -> " +
                   x.getNext().getNext().getNext().getData());
```

11 -> 22
22 -> 33

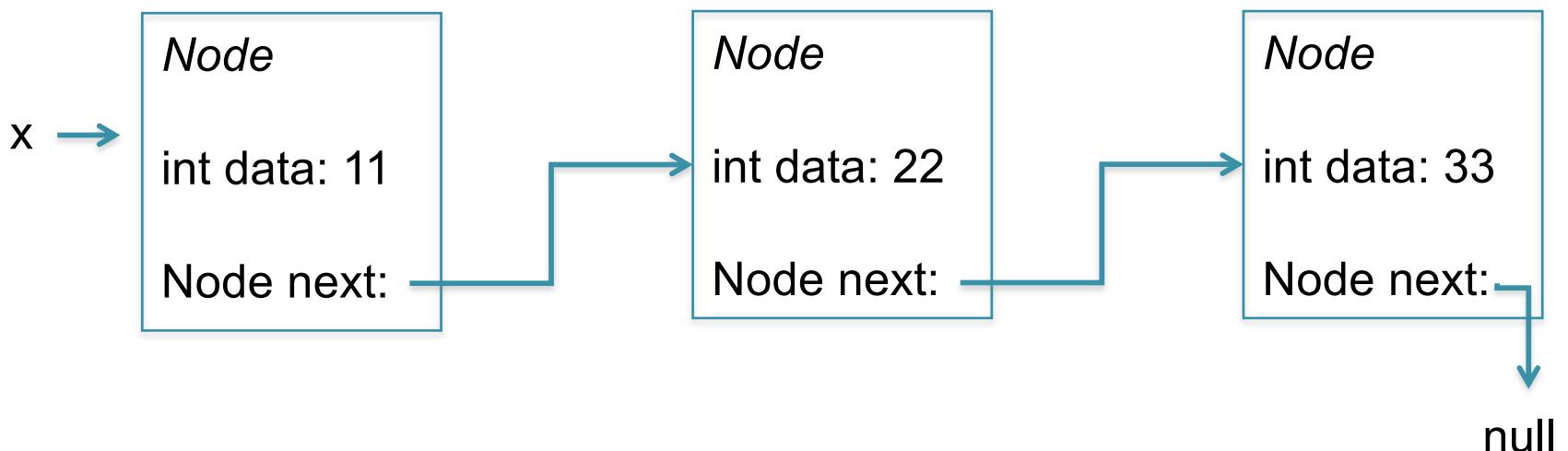


Java Nodes

```
Node x = new Node(11);
x.setNext(new Node(22));
x.getNext().setNext(new Node(33));

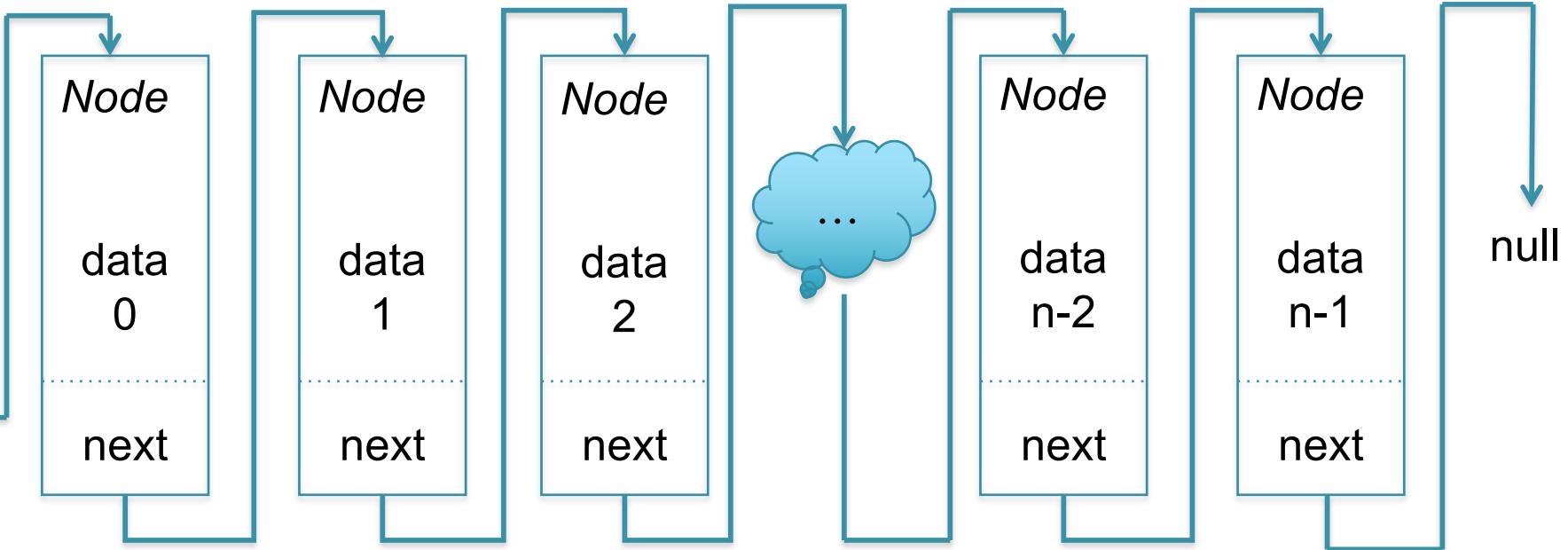
System.out.println(x.getData() + " -> " +
                   x.getNext().getData());
System.out.println(x.getNext().getData() + " -> " +
                   x.getNext().getNext().getData());
System.out.println(x.getNext().getNext().getData() + " -> " +
                   x.getNext().getNext().getNext().getData());
```

```
11 -> 22
22 -> 33
Exception in thread "main" java.lang.NullPointerException
at Node.main(Node.java:32)
```



ADT: Linked List

(Singly Linked List)



- Variable length data structure
- Constant time to head of list
- Linear time seek, easy to add/remove to middle once found

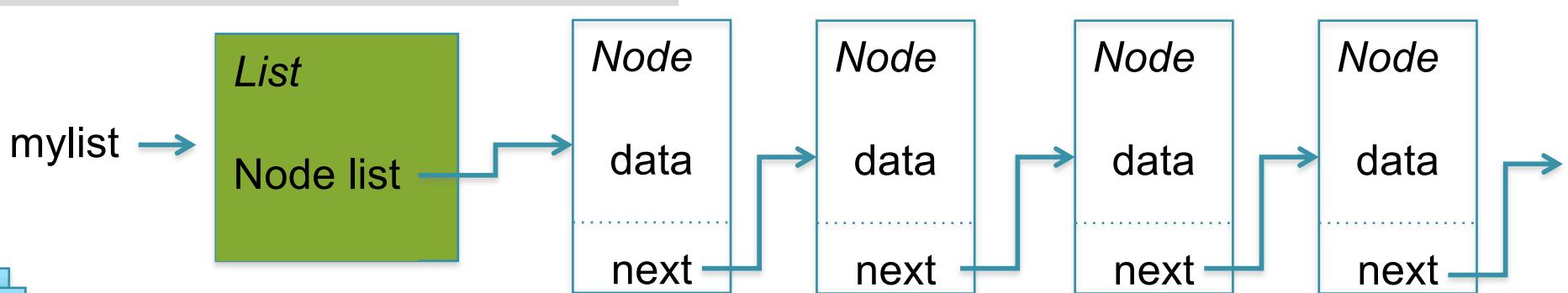
List Class

StringNode.java

```
public class StringNode {  
    private String data;  
    private StringNode next;  
  
    public StringNode(String d) {  
        this.data = d;  
        this.next = null; // not necessary  
    }  
  
    public void setNext(StringNode n) {  
        this.next = n;  
    }  
  
    public StringNode getNext() {  
        return this.next;  
    }  
  
    public String getData() {  
        return this.data;  
    }  
};
```

List.java

```
public class List {  
    private StringNode list;  
  
    public List() {  
        this.list = null; // not necessary  
    }  
  
    public static void main(String[] args) {  
        List mylist = new List();  
    }  
};
```



Linked List Construction

list → null

```
mylist.add("Mike");
```

Linked List Construction

list → null

mylist.add("Mike");

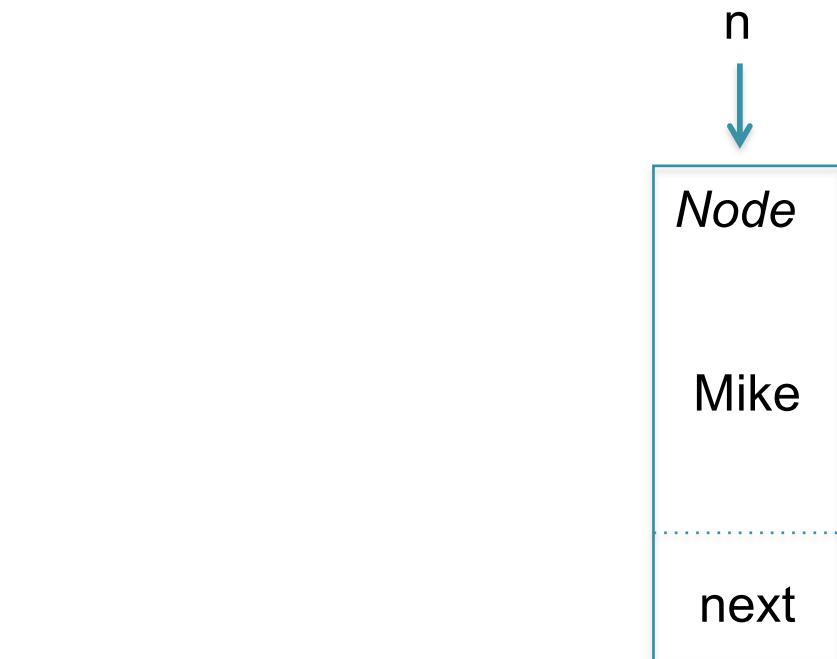
List.java

```
public void add(String value){  
    StringNode n = new StringNode(value);  
    n.setNext(this.list);  
    this.list = n;  
}
```

Linked List Construction

list → null

mylist.add("Mike");



List.java

```
public void add(String value){  
    StringNode n = new StringNode(value);  
    n.setNext(this.list);  
    this.list = n;  
}
```

Linked List Construction



```
mylist.add("Mike");
```

List.java

```
public void add(String value){  
    StringNode n = new StringNode(value);  
    n.setNext(this.list);  
    this.list = n;  
}
```

Linked List Construction

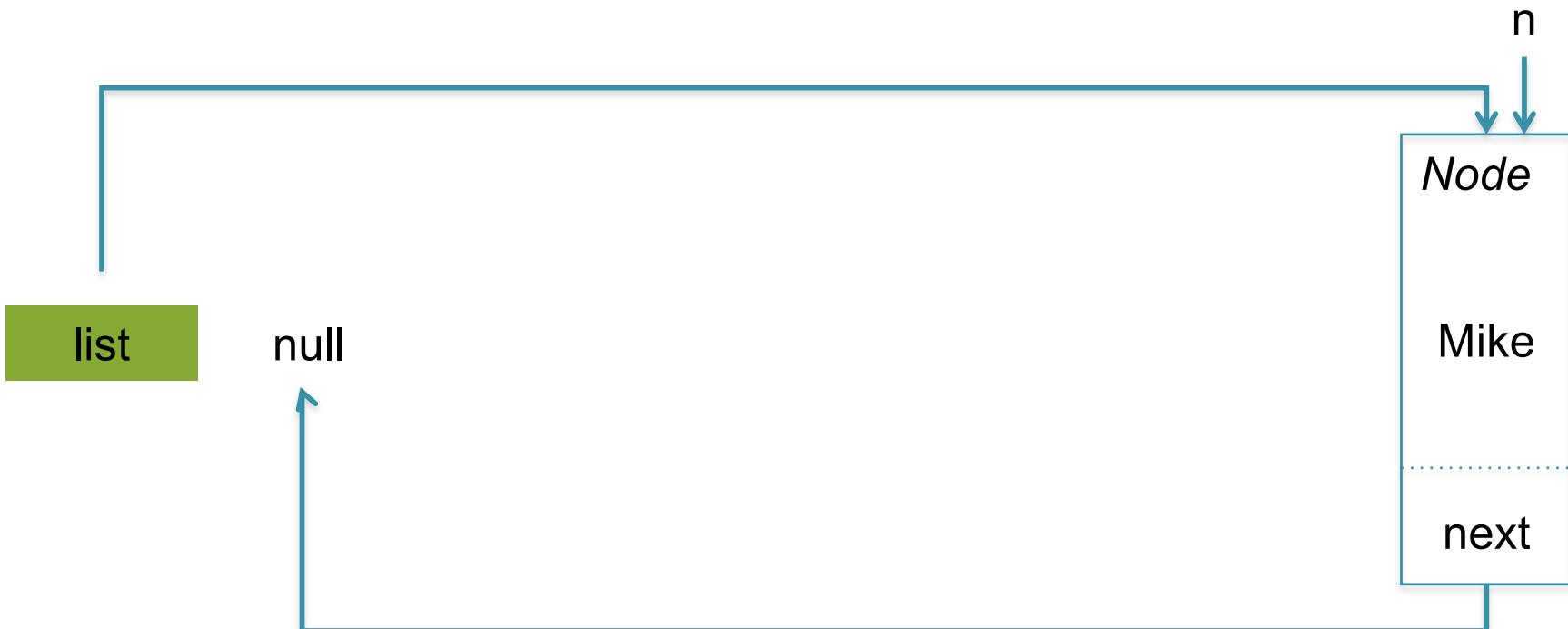


```
mylist.add("Mike");
```

List.java

```
public void add(String value){  
    StringNode n = new StringNode(value);  
    n.setNext(this.list);  
    this.list = n;  
}
```

Linked List Construction

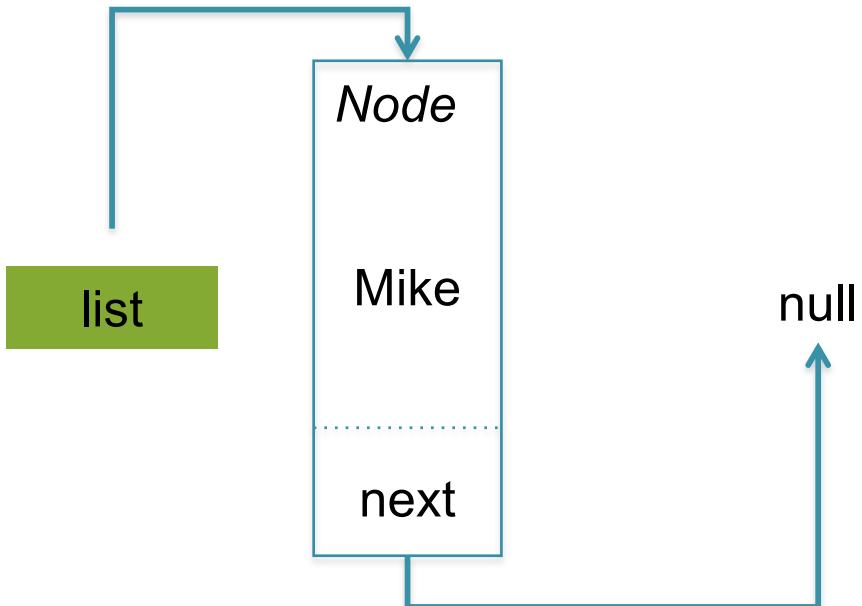


```
mylist.add("Mike");
```

List.java

```
public void add(String value){  
    StringNode n = new StringNode(value);  
    n.setNext(this.list);  
    this.list = n;  
}
```

Linked List Construction

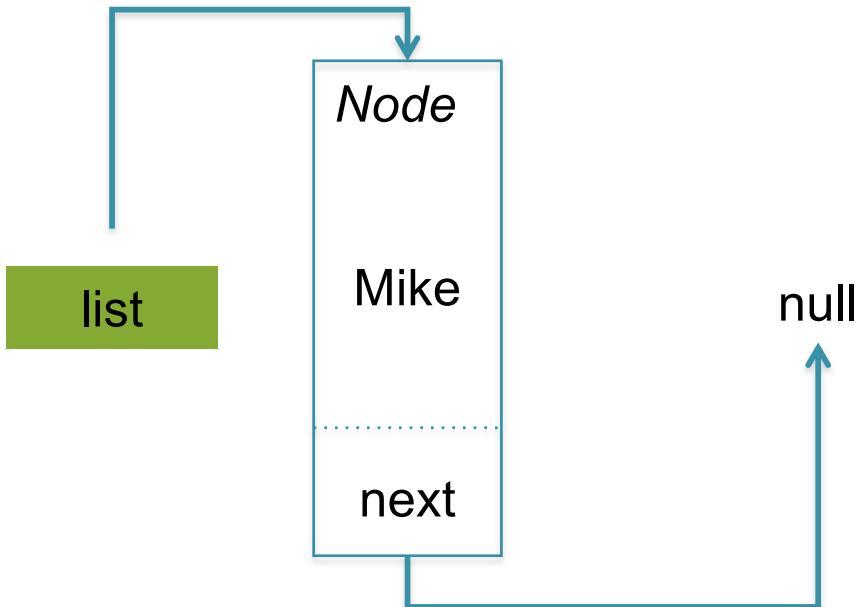


```
mylist.add("Mike");
```

List.java

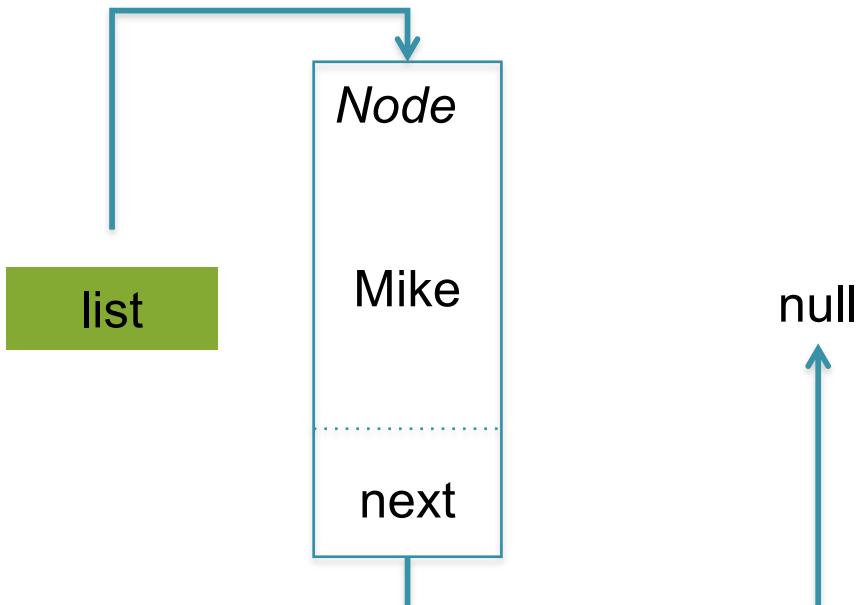
```
public void add(String value){  
    StringNode n = new StringNode(value);  
    n.setNext(this.list);  
    this.list = n;  
}
```

Linked List Construction



First node successfully added (prepend)
Questions?

Linked List Construction

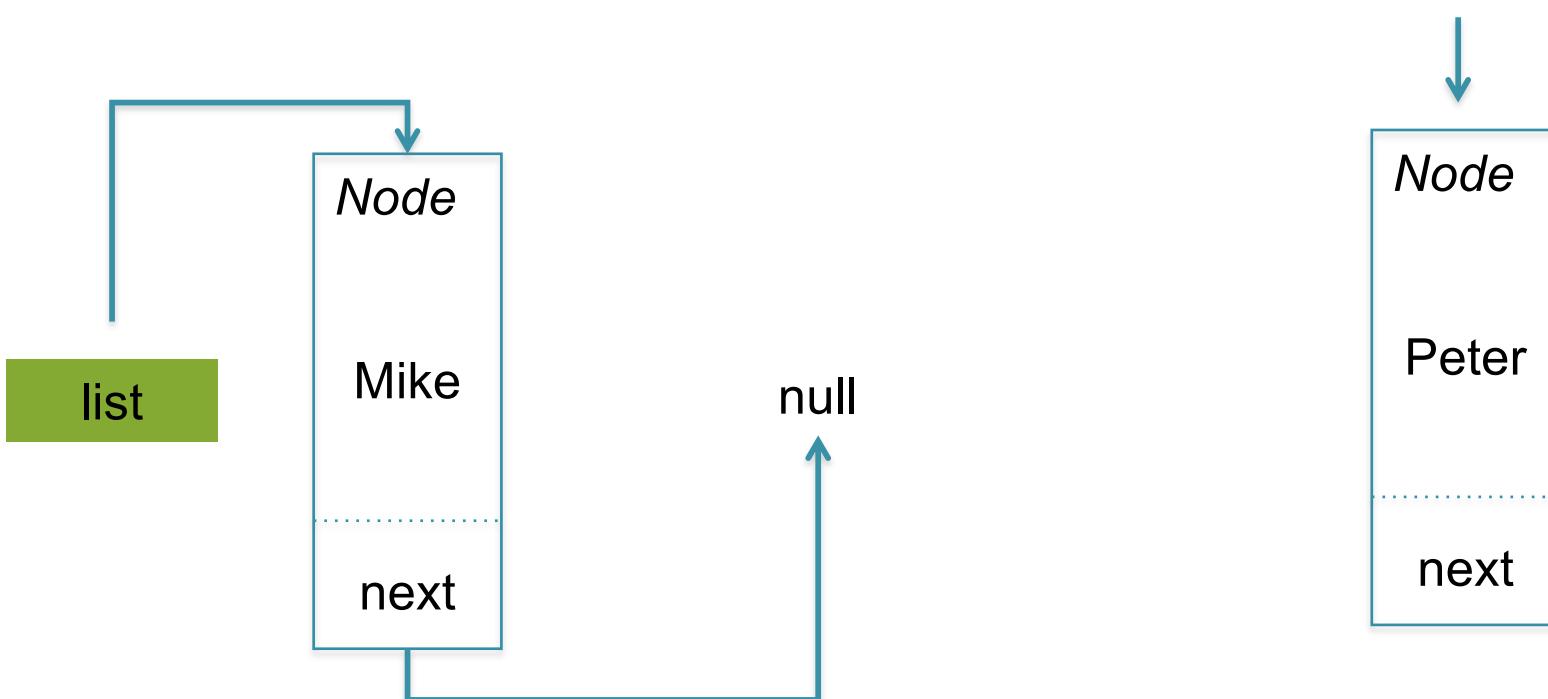


```
mylist.add("Peter");
```

List.java

```
public void add(String value){  
    StringNode n = new StringNode(value);  
    n.setNext(this.list);  
    this.list = n;  
}
```

Linked List Construction

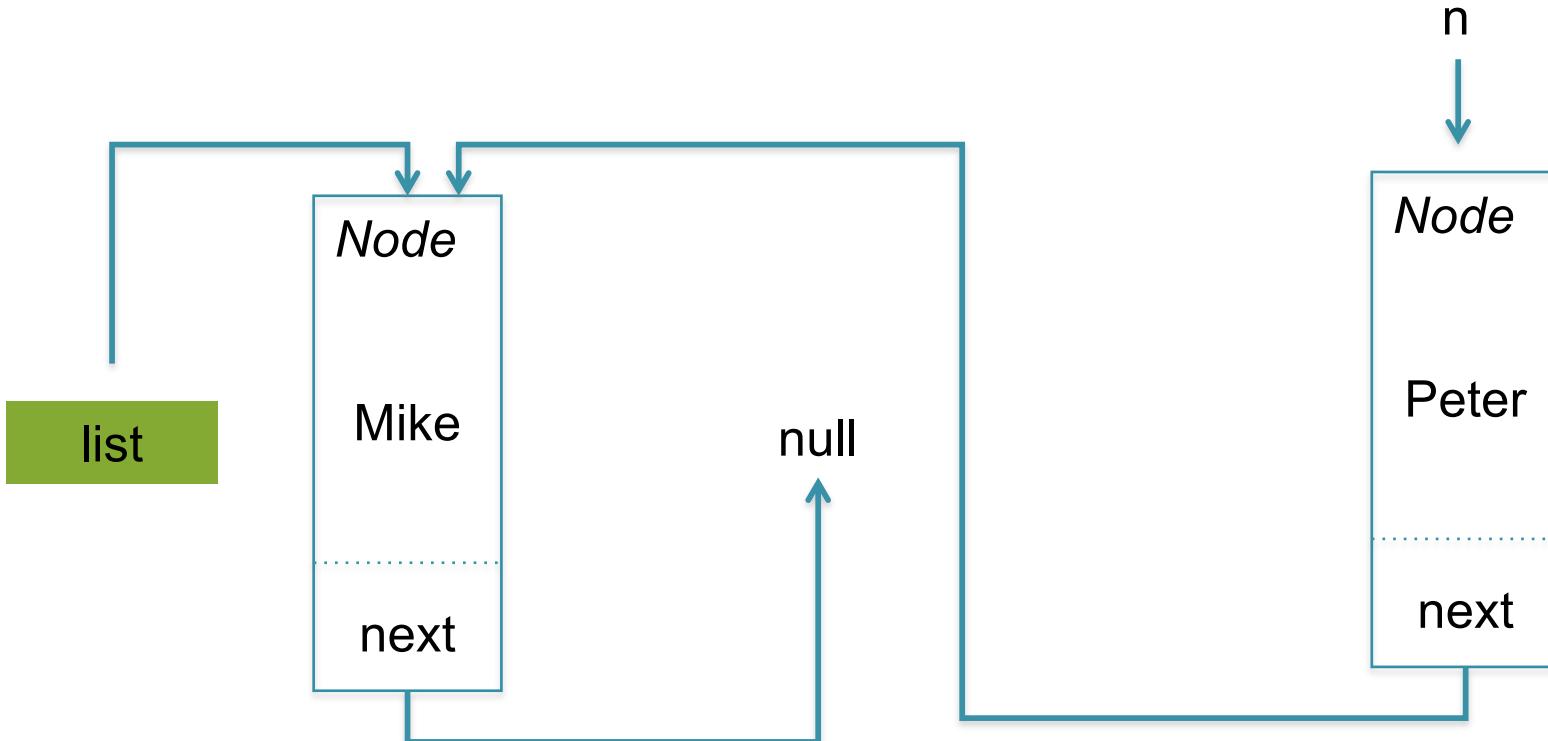


```
mylist.add("Peter");
```

List.java

```
public void add(String value){  
    StringNode n = new StringNode(value);  
    n.setNext(this.list);  
    this.list = n;  
}
```

Linked List Construction

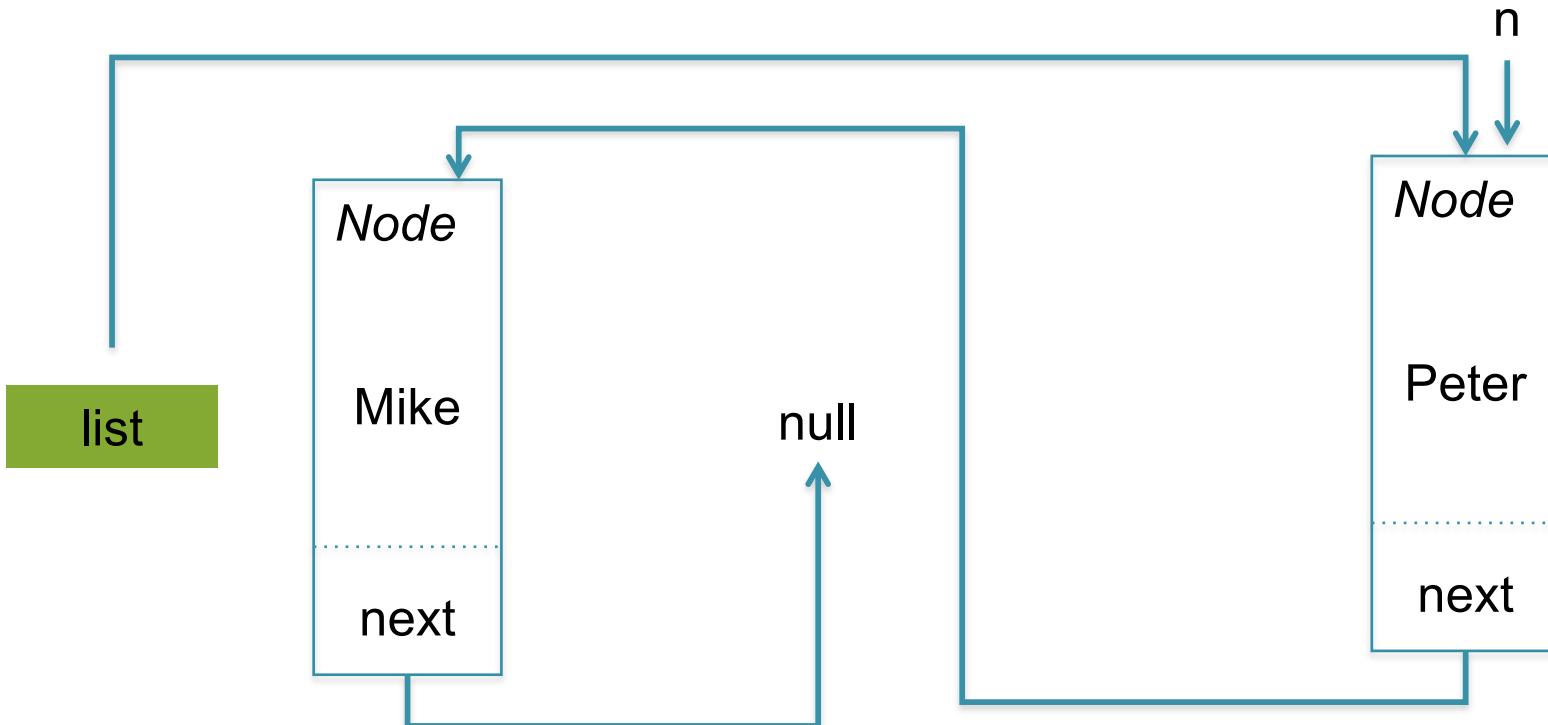


```
mylist.add("Peter");
```

List.java

```
public void add(String value){  
    StringNode n = new StringNode(value);  
    n.setNext(this.list);  
    this.list = n;  
}
```

Linked List Construction

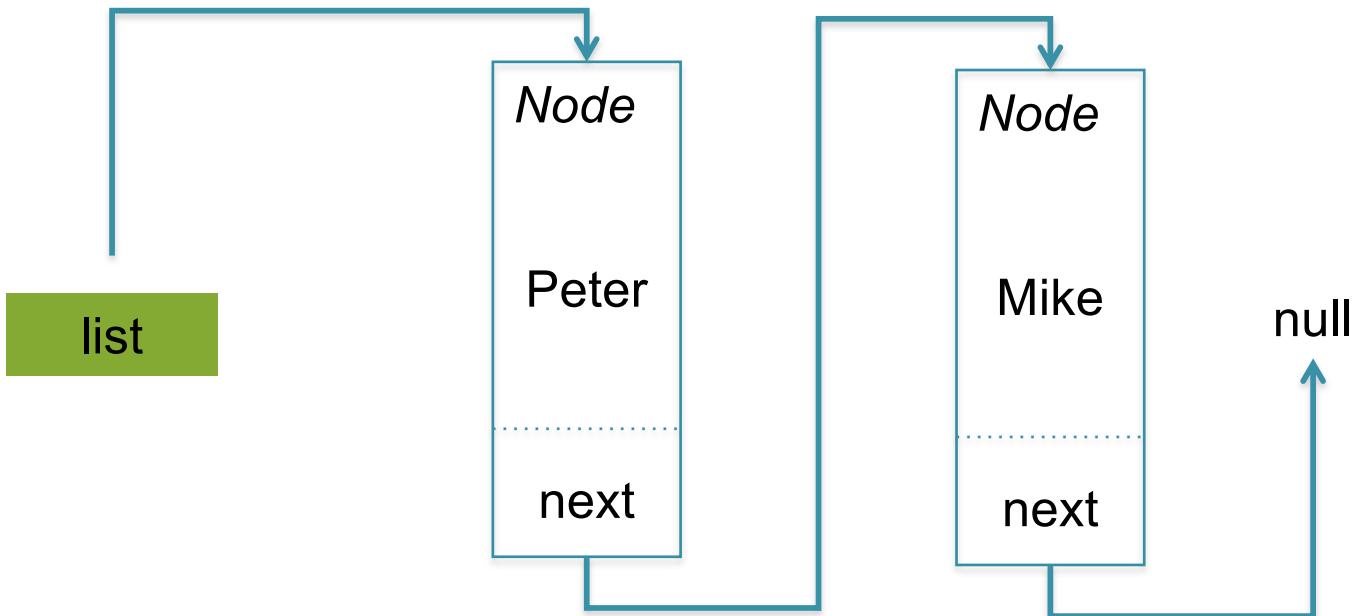


```
mylist.add("Peter");
```

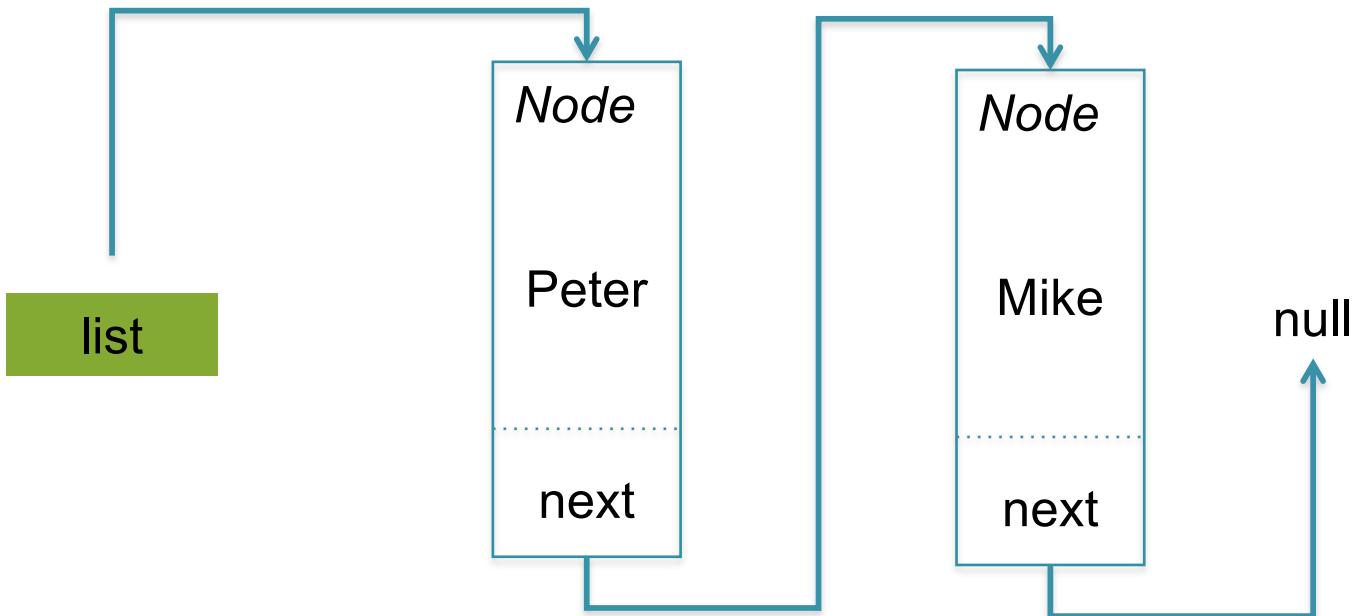
List.java

```
public void add(String value){  
    StringNode n = new StringNode(value);  
    n.setNext(this.list);  
    this.list = n;  
}
```

Linked List Construction



Linked List Construction

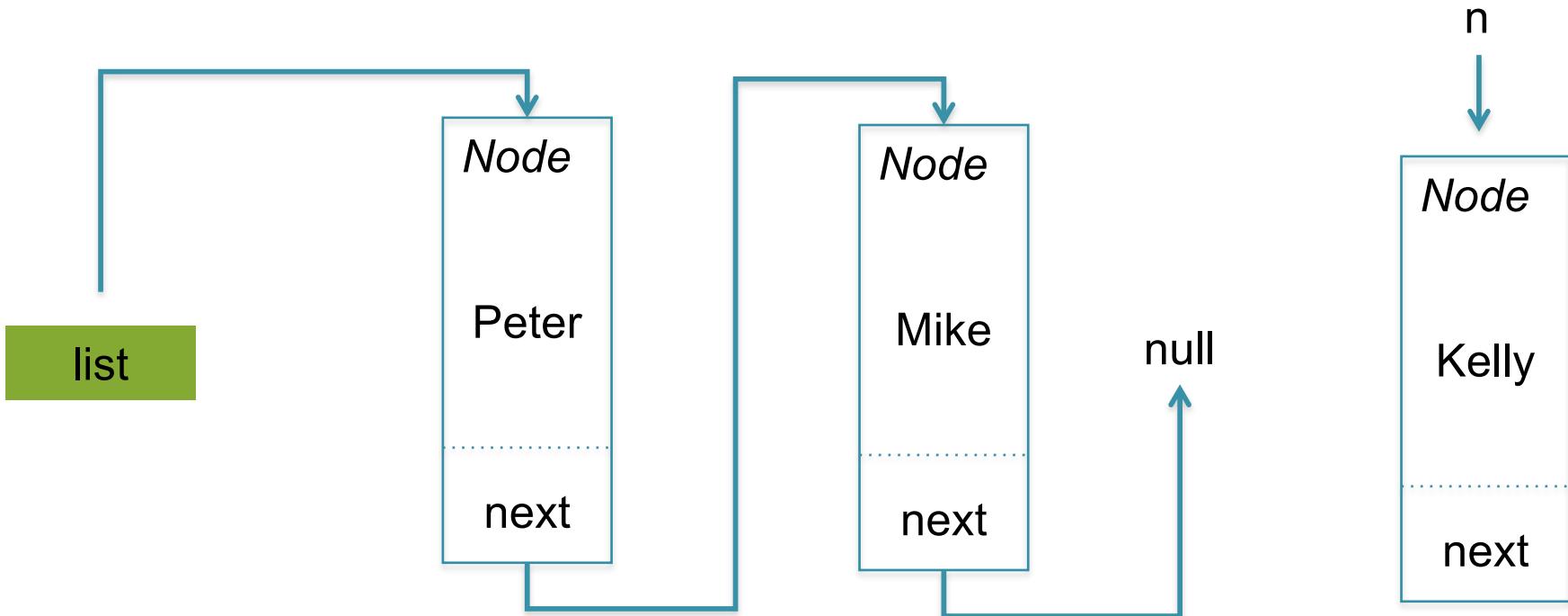


```
mylist.add("Kelly");
```

List.java

```
public void add(String value){  
    StringNode n = new StringNode(value);  
    n.setNext(this.list);  
    this.list = n;  
}
```

Linked List Construction

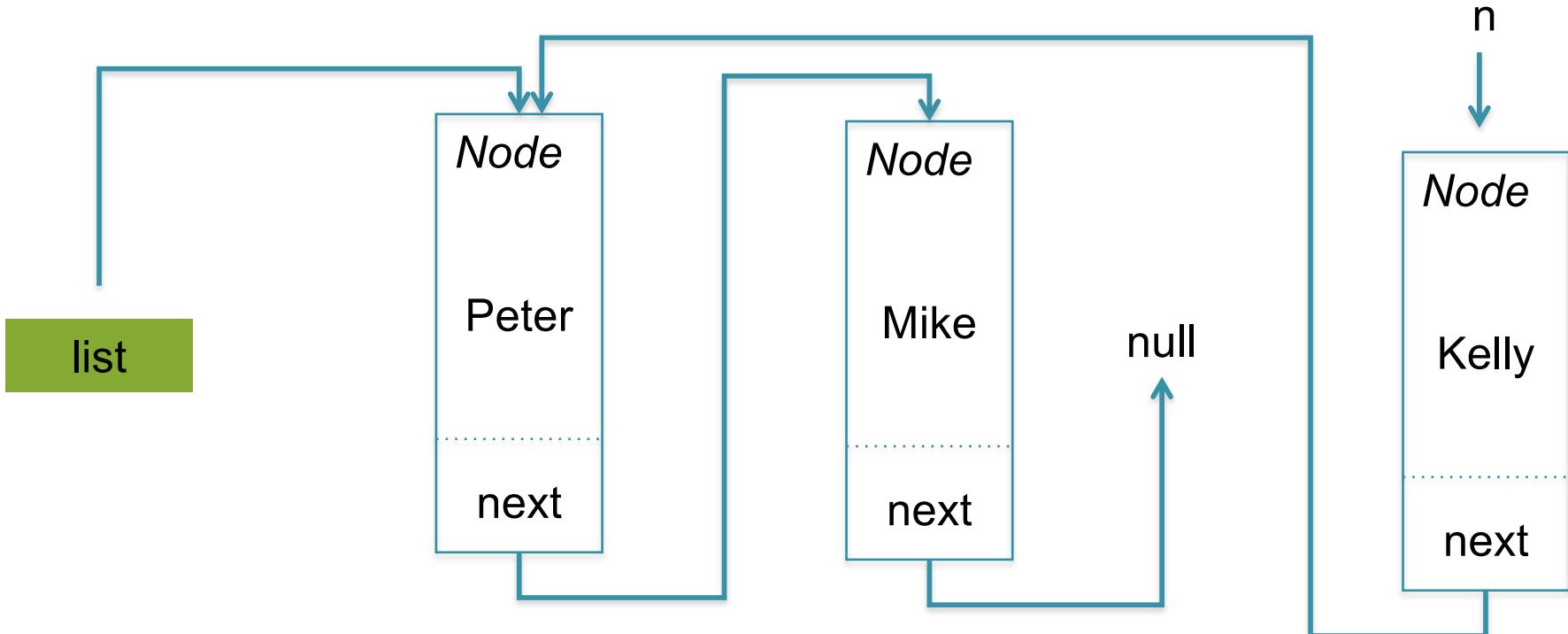


```
mylist.add("Kelly");
```

List.java

```
public void add(String value){  
    StringNode n = new StringNode(value);  
    n.setNext(this.list);  
    this.list = n;  
}
```

Linked List Construction

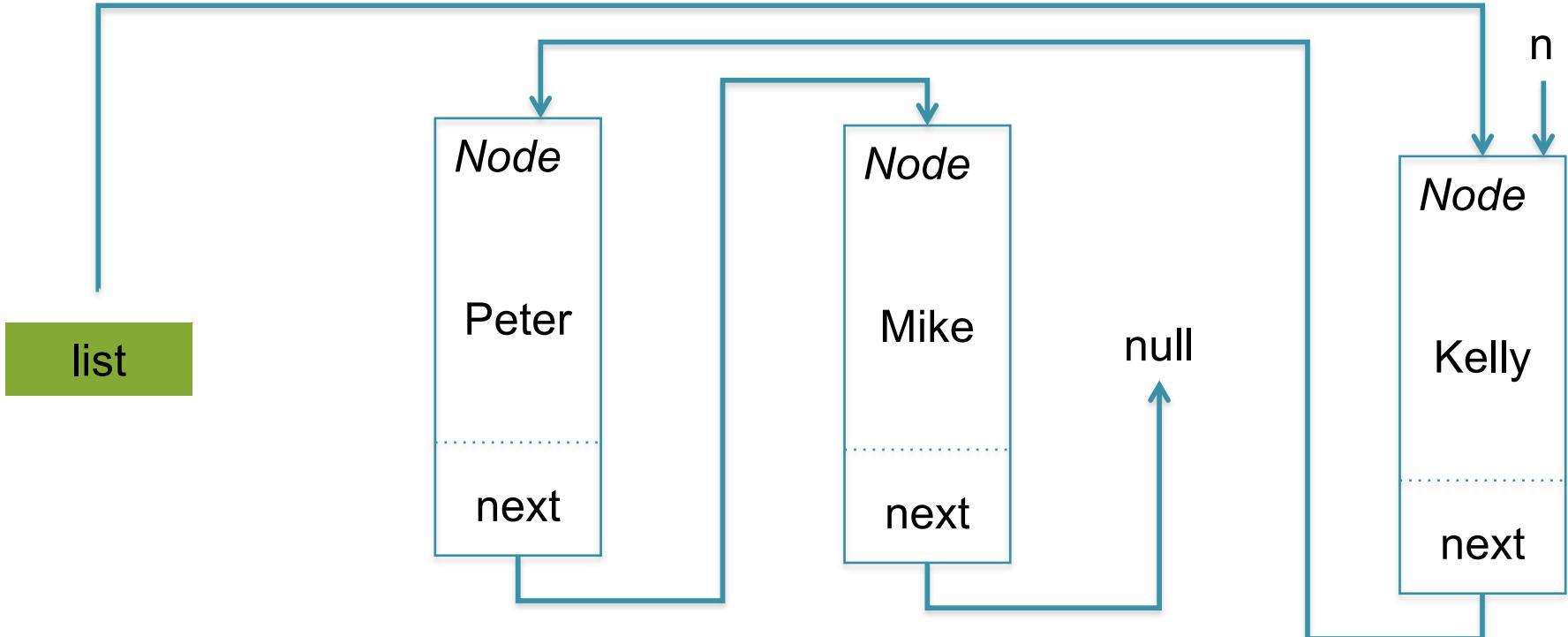


```
mylist.add("Kelly");
```

List.java

```
public void add(String value){  
    StringNode n = new StringNode(value);  
    n.setNext(this.list);  
    this.list = n;  
}
```

Linked List Construction

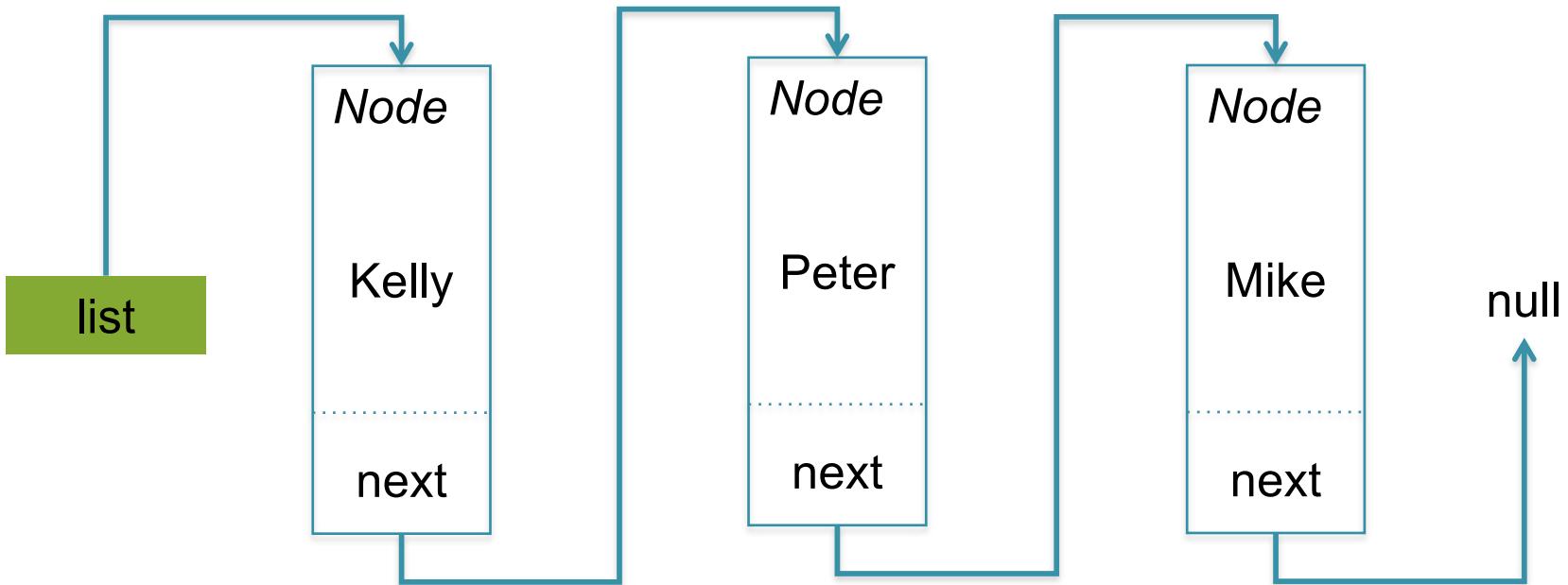


```
mylist.add("Kelly");
```

List.java

```
public void add(String value){  
    StringNode n = new StringNode(value);  
    n.setNext(this.list);  
    this.list = n;  
}
```

Linked List Construction



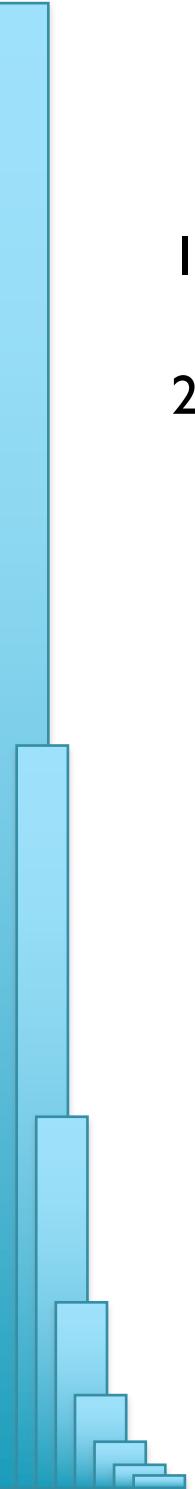
*Why do we insert at the beginning of the list (prepend)?
How long would it take to insert at the tail?
How could you make that faster?*

Linked List Construction



What

end)?



Next Steps

- I. Work on HWI
2. Check on Piazza for tips & corrections!