CS 600.226: Data Structures

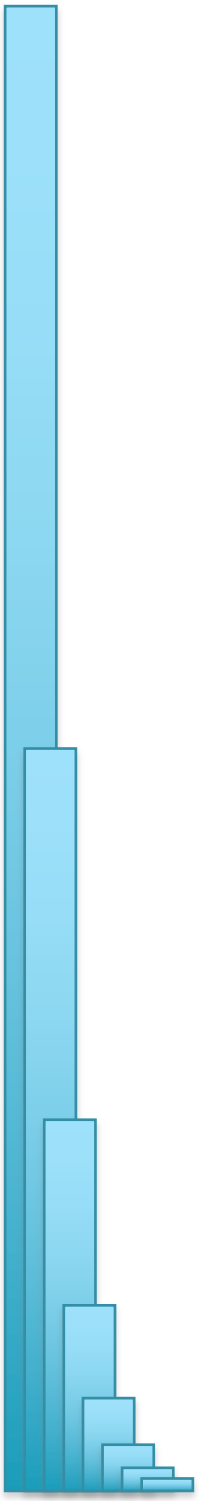
Michael Schatz

Sept 7 2018

Lecture 4: Linked Lists

Agenda

1. *Review HW1*
2. *Review Java Arrays*
3. *References and Linked Lists*



Assignment 1: 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 READMDE 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 1: 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 1: 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 substacting 2
- An implementation of `ResetableCounter` called `TenCounter` that starts at the value 1, counts up by multipling 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 1: 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 1: 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 1: 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 (esepcially 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 1: 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.

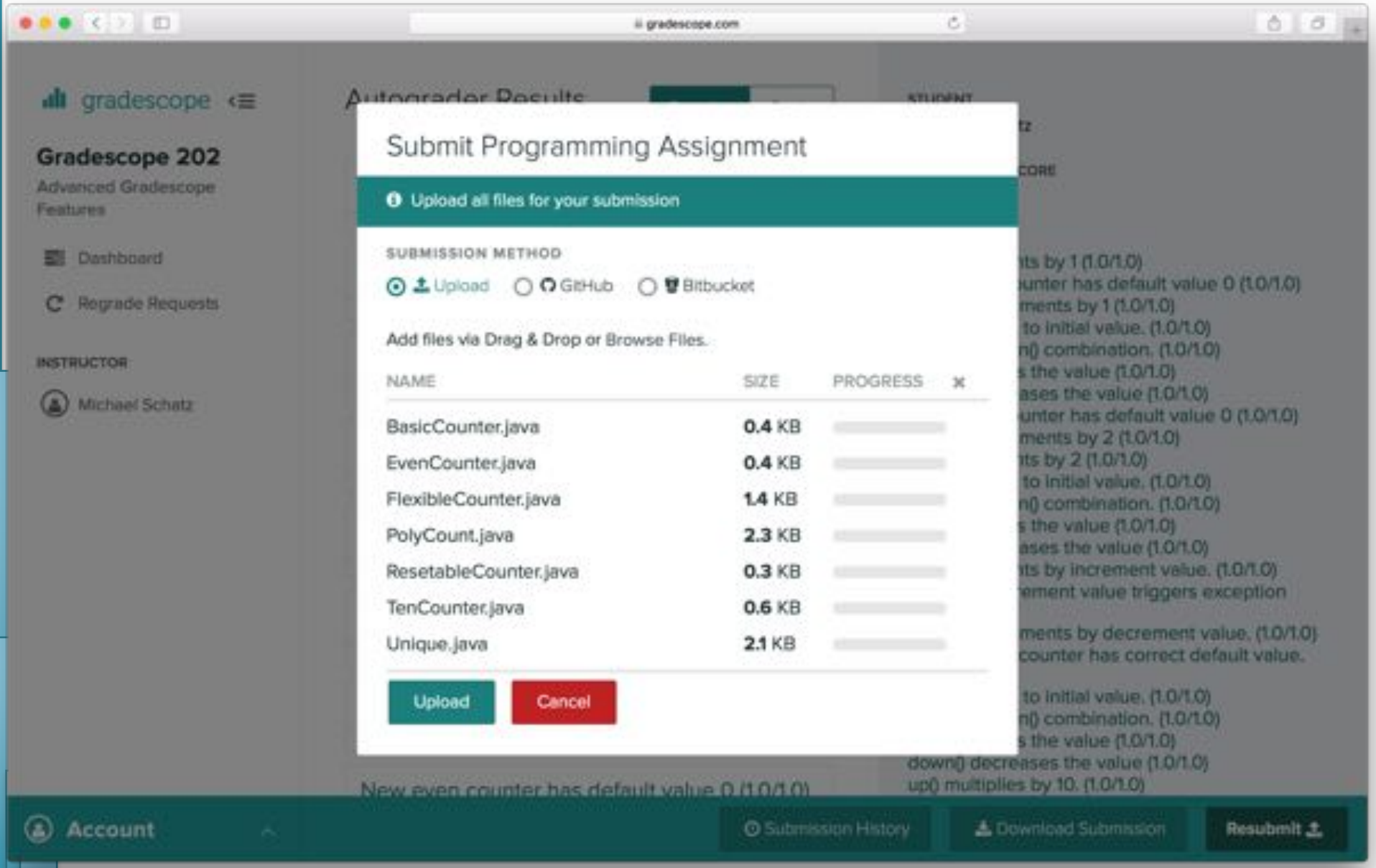
Testing refers to proper unit tests for all of the data structure classes you developed for this assignment, using the JUnit 4 framework as introduced in lecture. Make sure you test **all** (implied) axioms that you can think of and **all** exception conditions that are relevant.

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

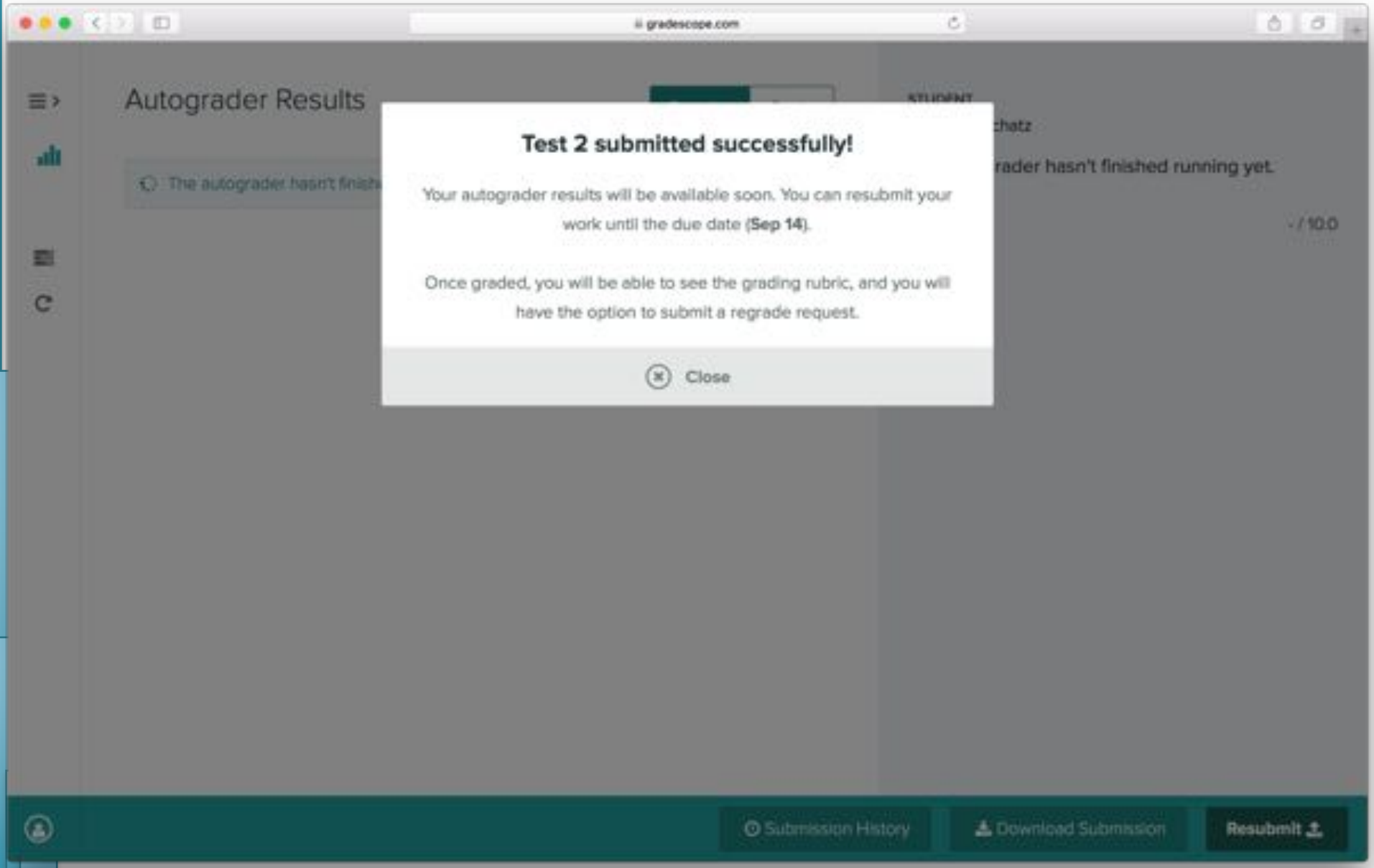
Functionality refers to your programs being able to do what they should according to the specification given above; if the specification is ambiguous and you had to make a certain choice, defend that choice in your `README` file.

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).

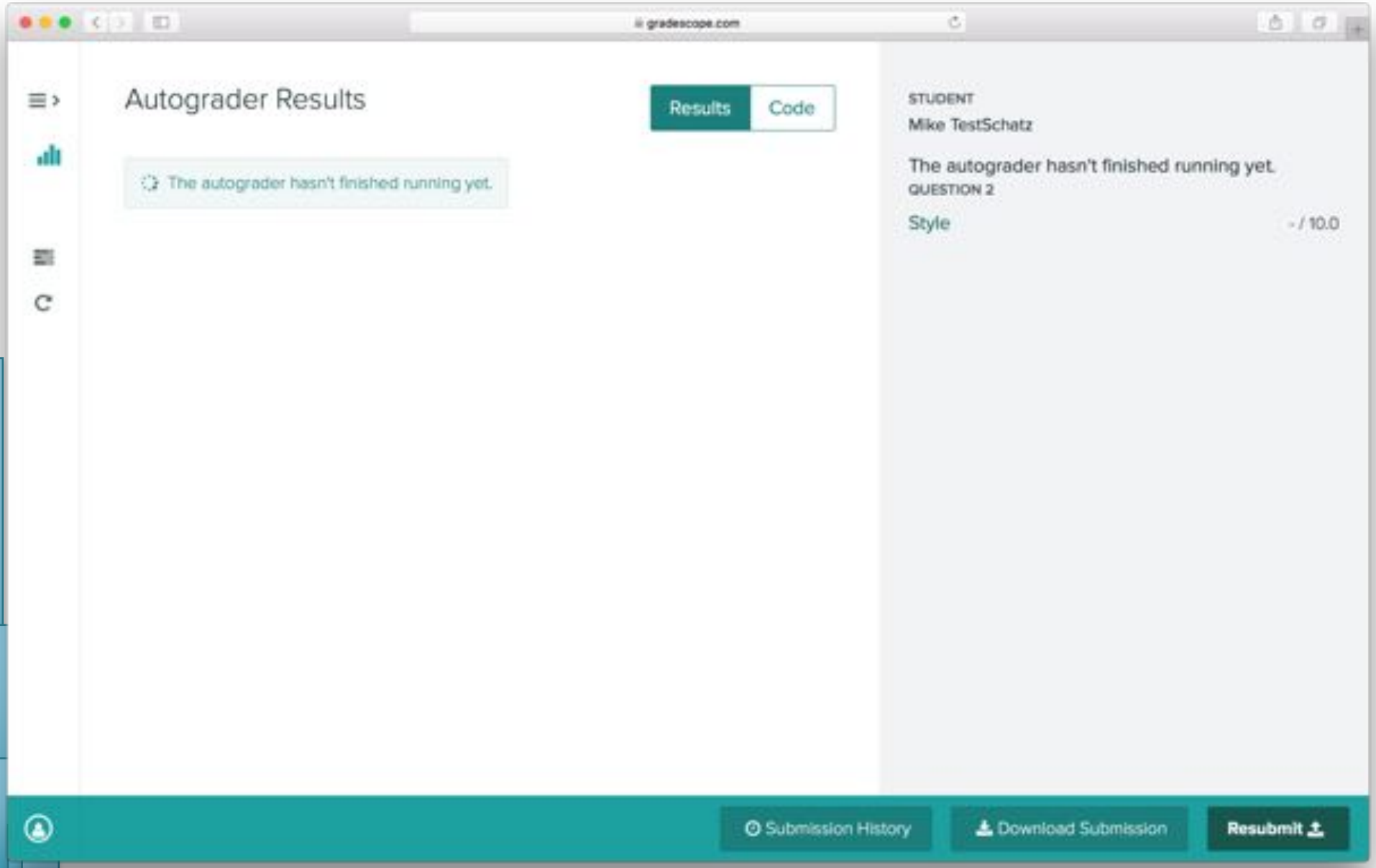
GradeScope.com
Entry Code: MDJYER



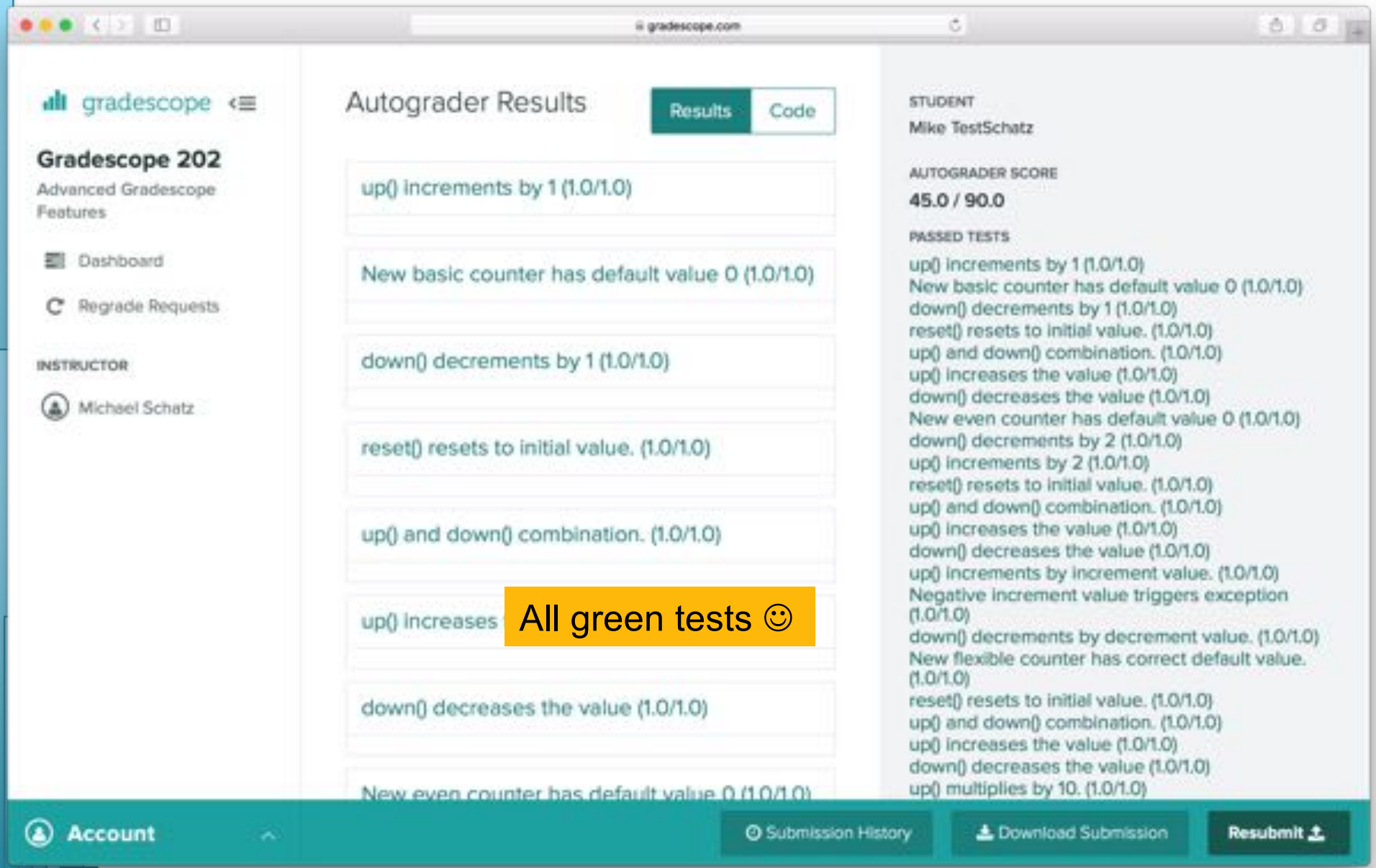
GradeScope.com
Entry Code: MDJYER



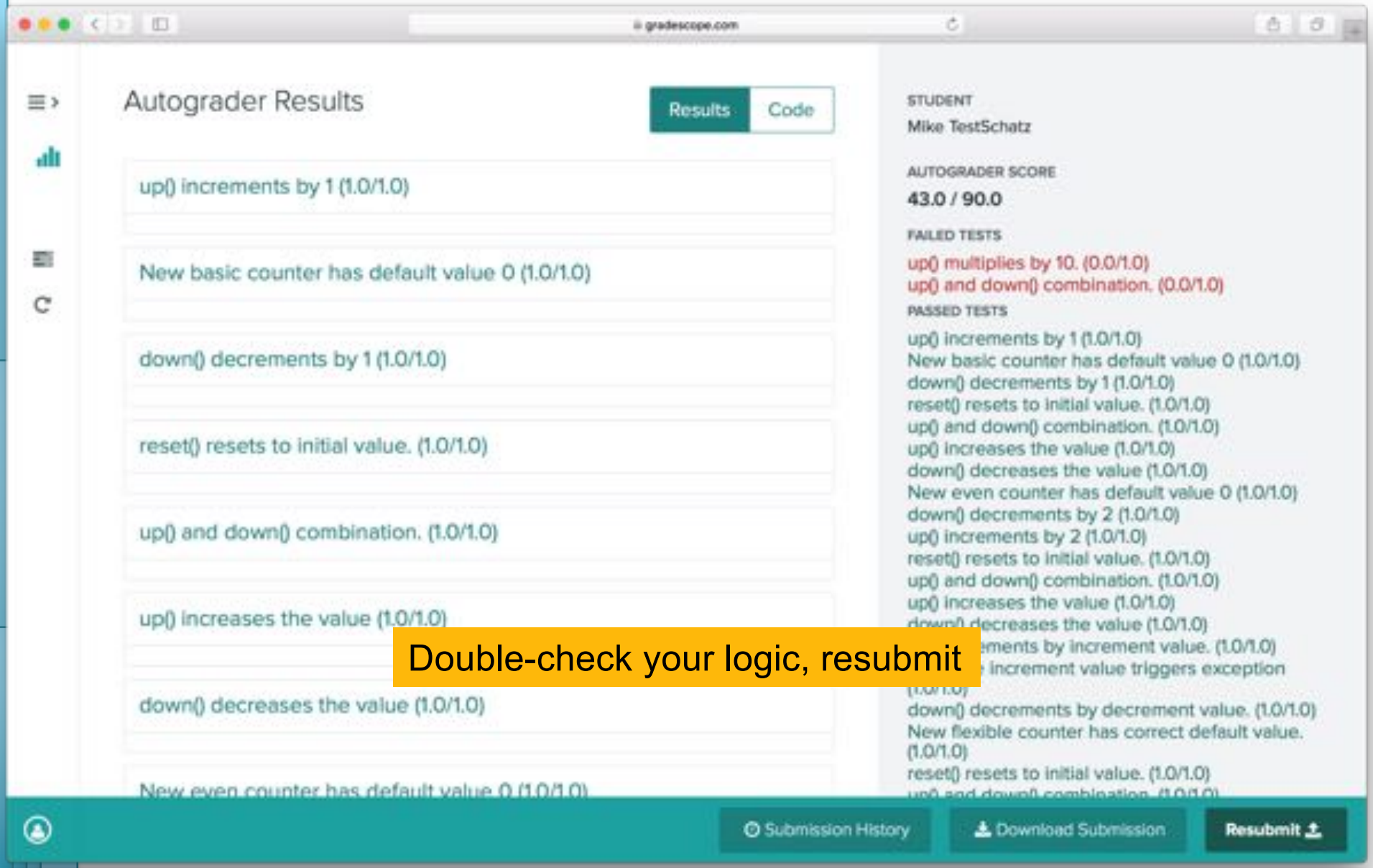
GradeScope.com
Entry Code: MDJYER



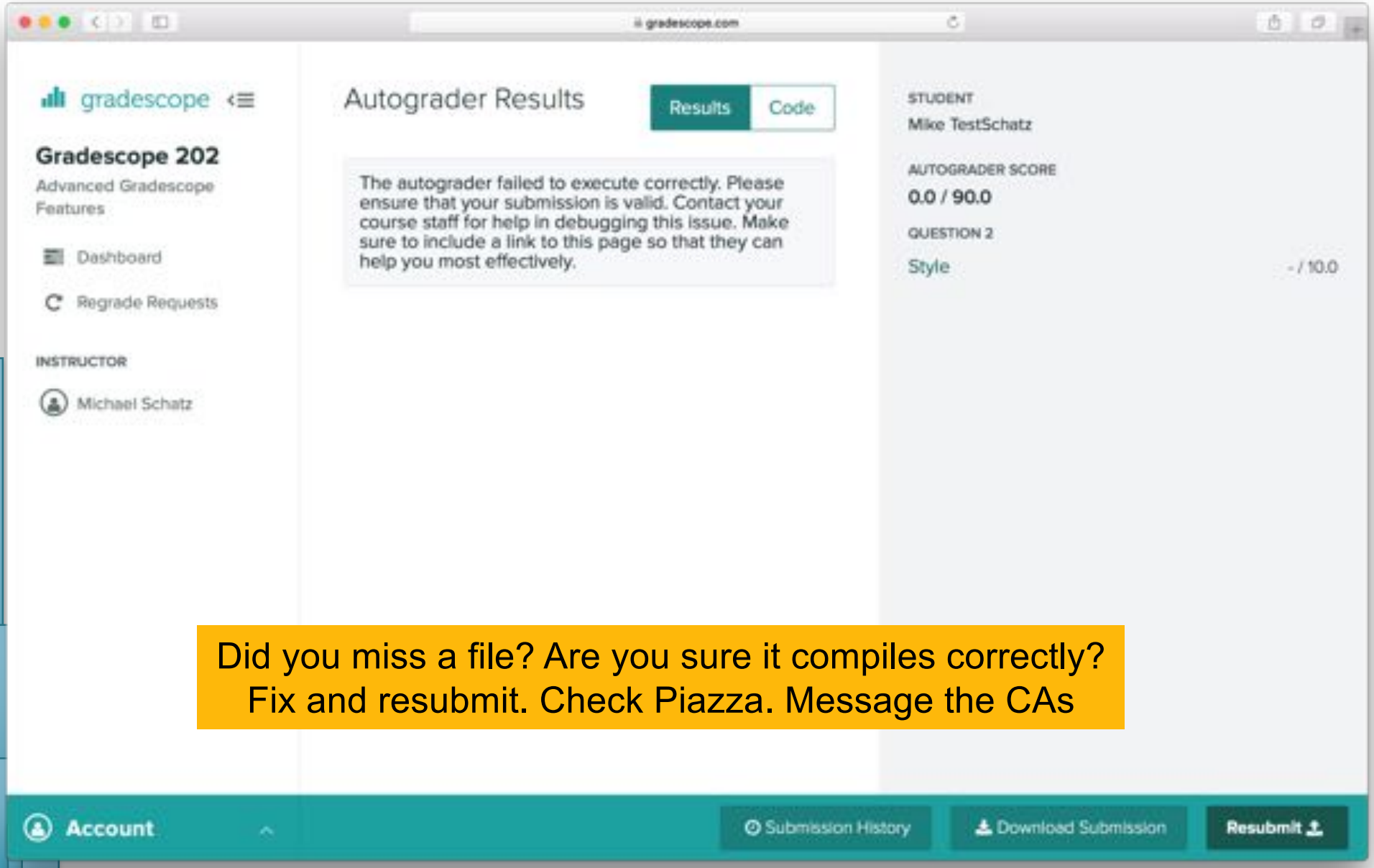
GradeScope.com
Entry Code: MDJYER



GradeScope.com
Entry Code: MDJYER



GradeScope.com
Entry Code: MDJYER

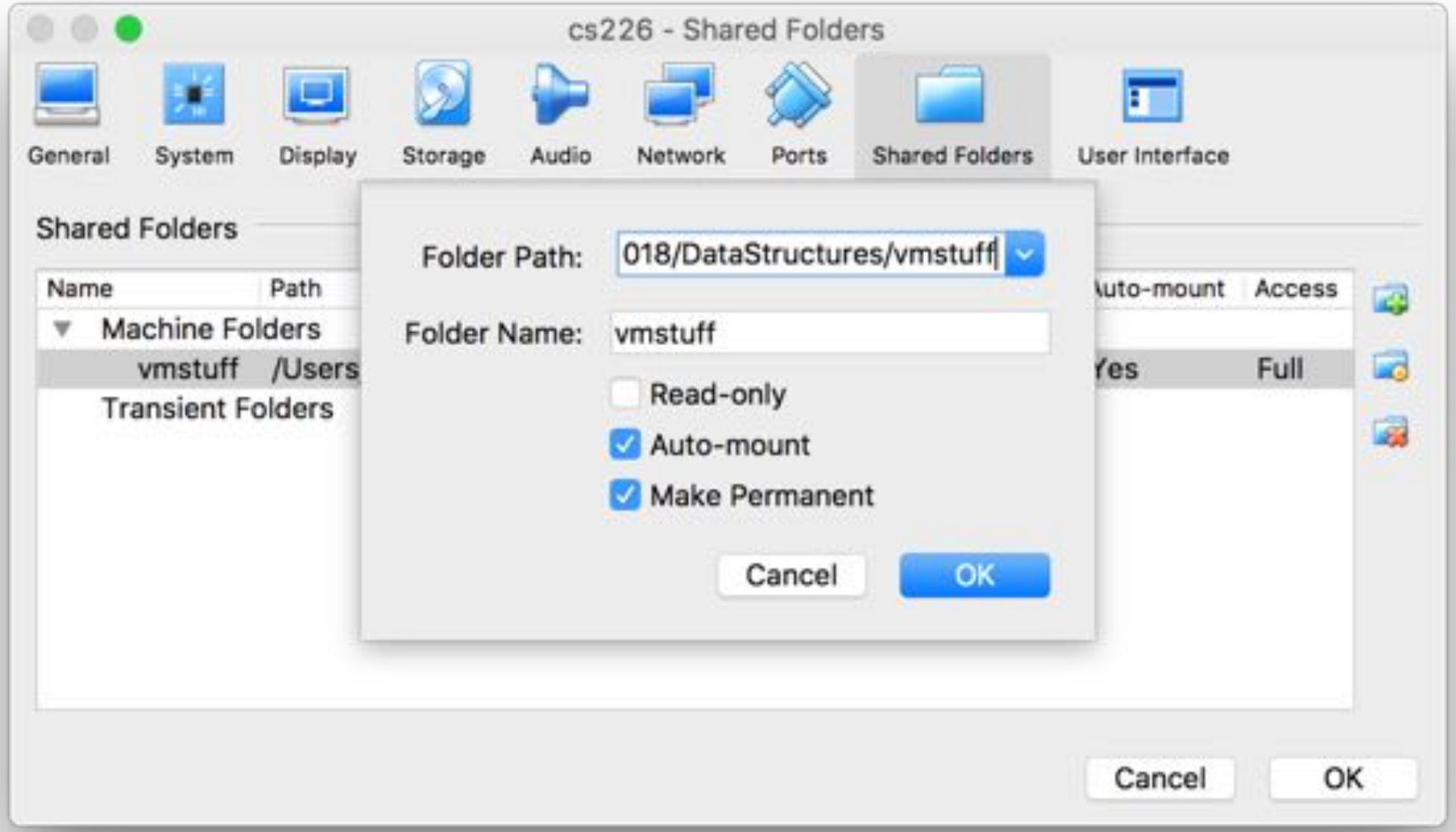


Did you miss a file? Are you sure it compiles correctly?
Fix and resubmit. Check Piazza. Message the CAs

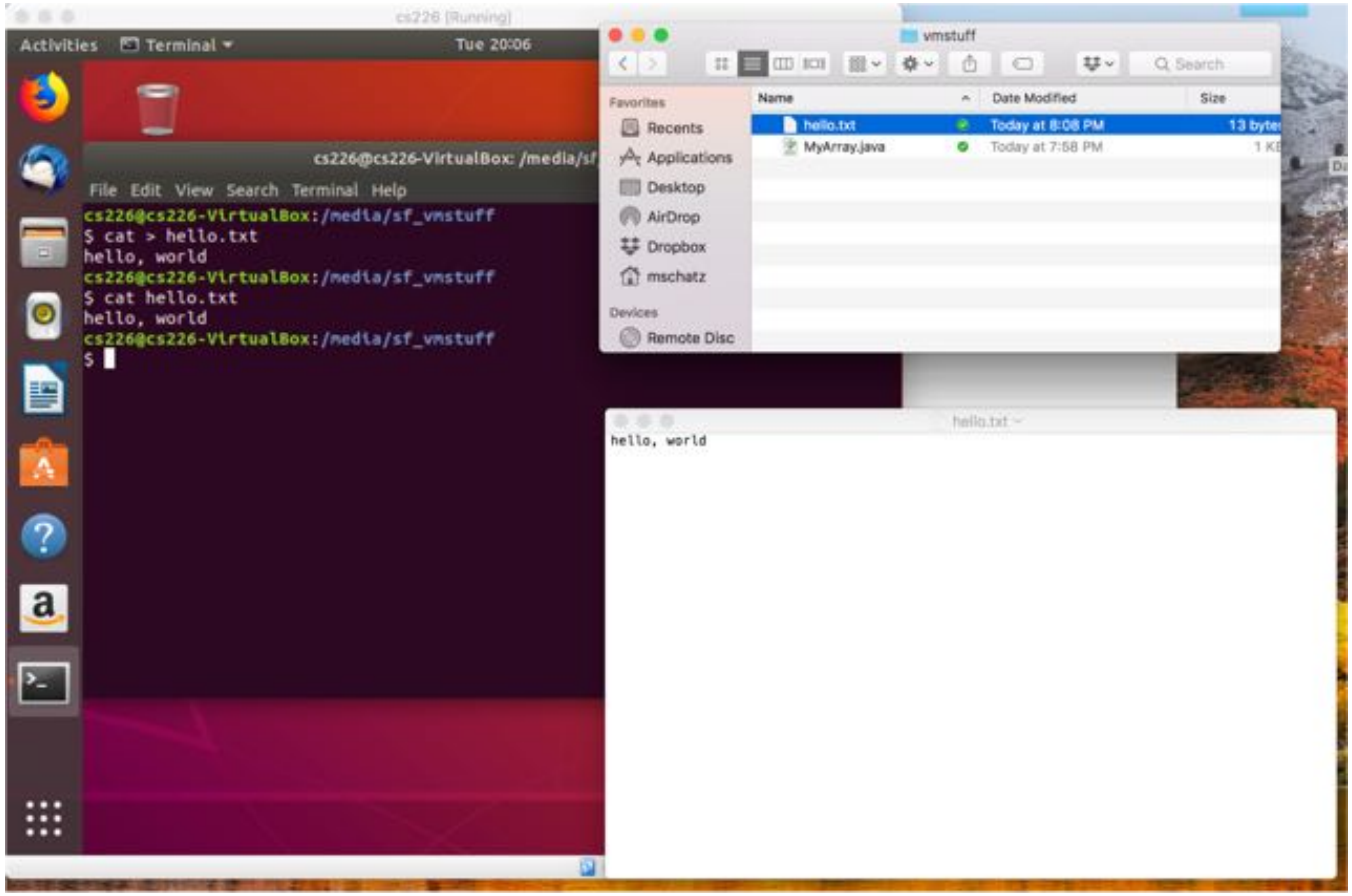
VirtualBox

- 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 permisions
$ sudo usermod -aG vboxsf cs226
$ /sbin/shutdown -r now
```

Make sure to shutdown cleanly!

Java Environments

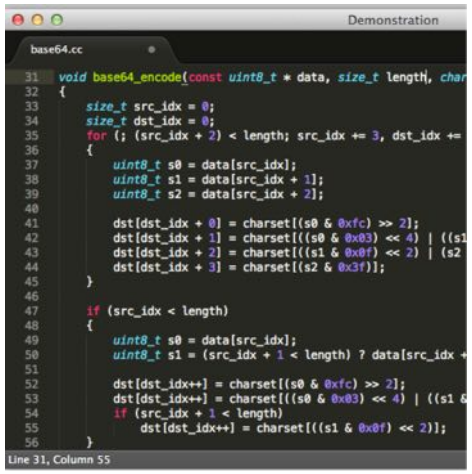
Command Line Everything

$ vim HelloWorld.java

$ javac HelloWorld.java
$ java HelloWorld

Universal, fast, flexible
Steep learning curve

GUI Editor + Command Line

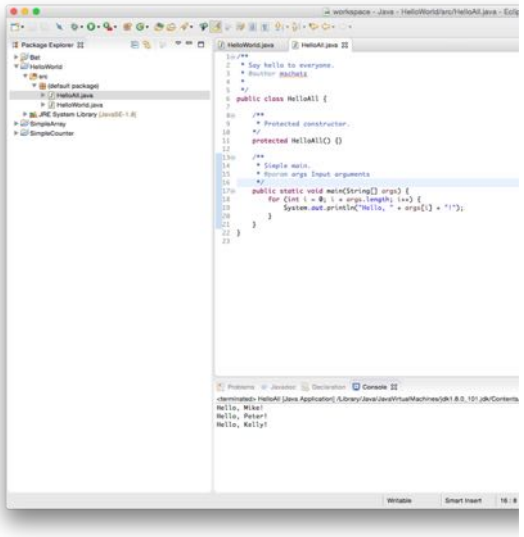


Sublime Text

$ javac HelloWorld.java
$ java HelloWorld

Nearly universal, flexible
Moderate learning curve

Integrated Development Environment (IDE)



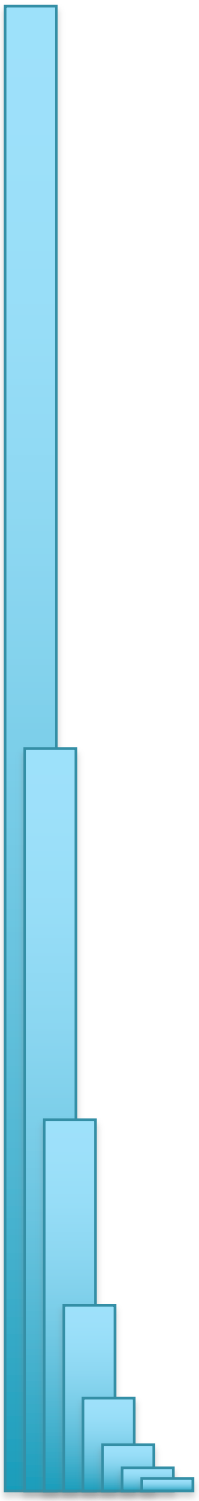
Eclipse / IntelliJ

Most Support
Most "magical"

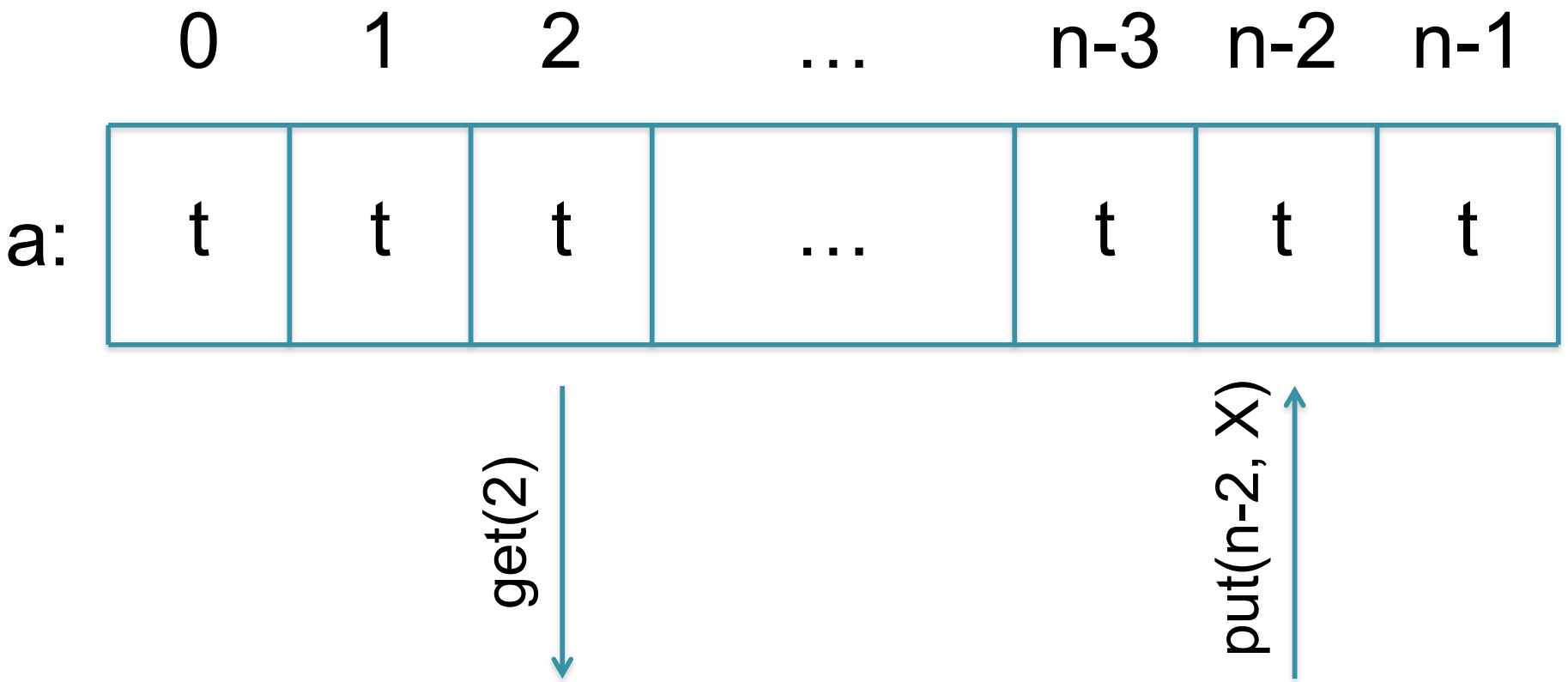
Code may not work during grading ☹

Agenda

1. *Review HW1*
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

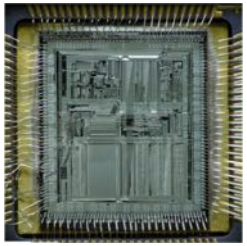
Permanent Storage – 1TB

(big, slow, cheap)

RAM

Working Storage – 8 GB

(small, fast, expensive)



Processor

Arithmetic, logic

cores, clock speed

Display

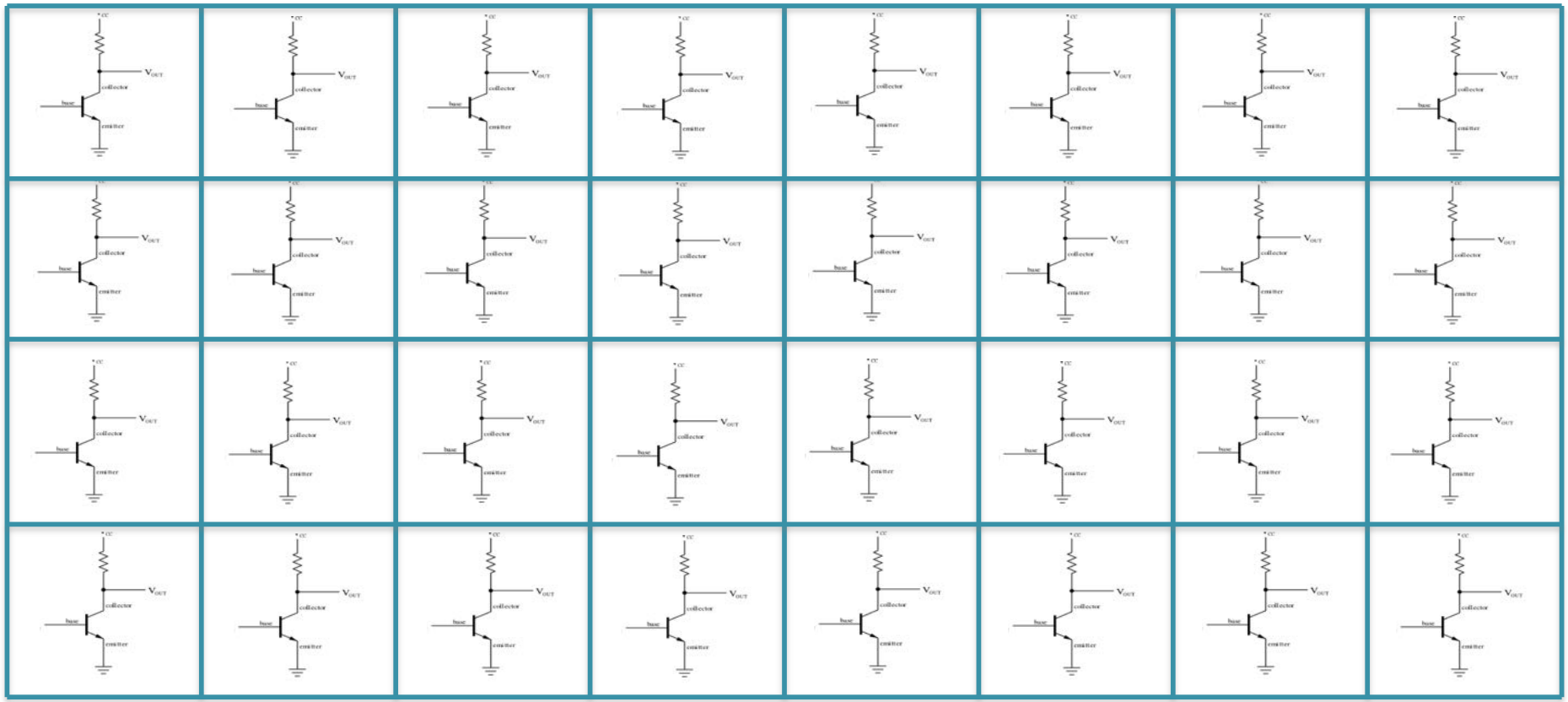
Human Interface

Network

Computer Interface

Home: 10Mb/s, JHU: 1Gb/s

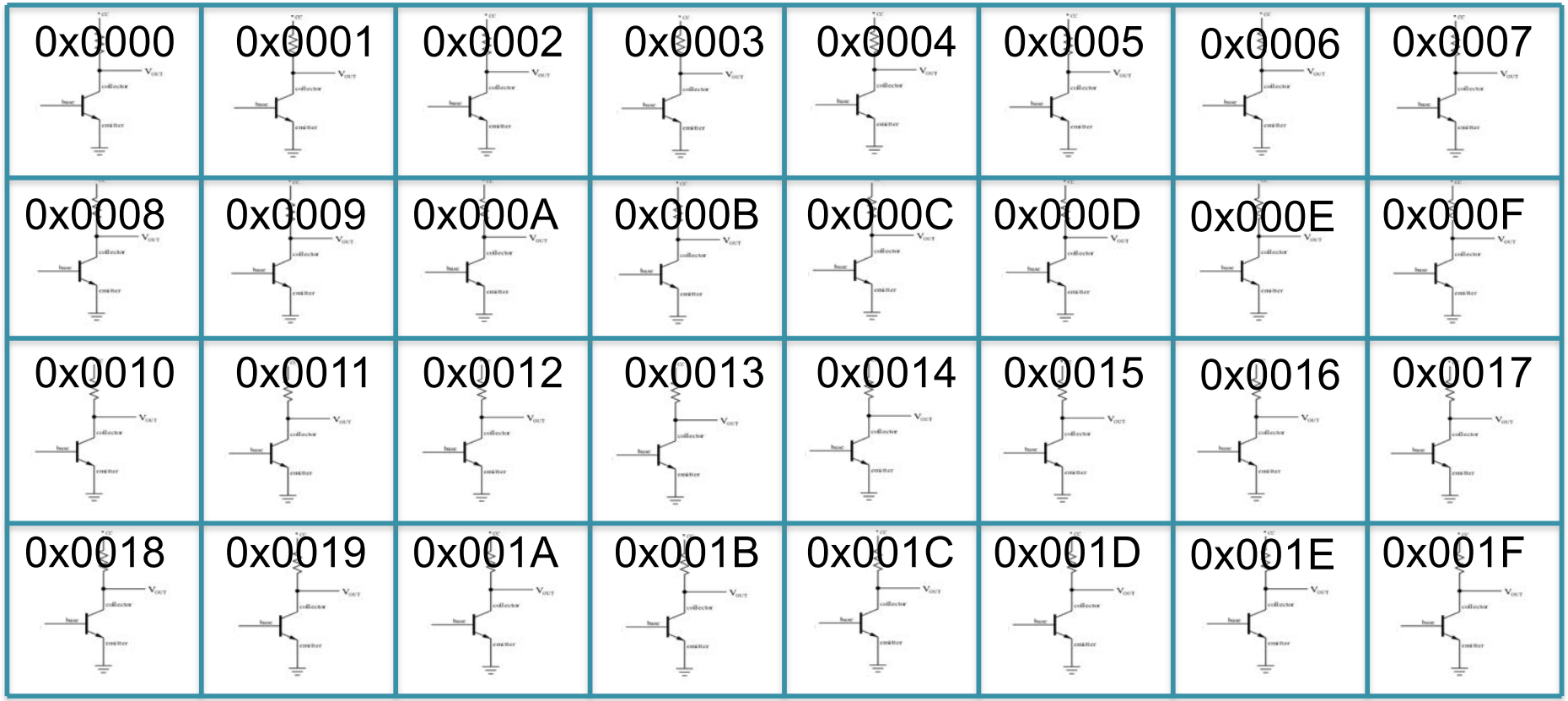
Accessing RAM



```
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

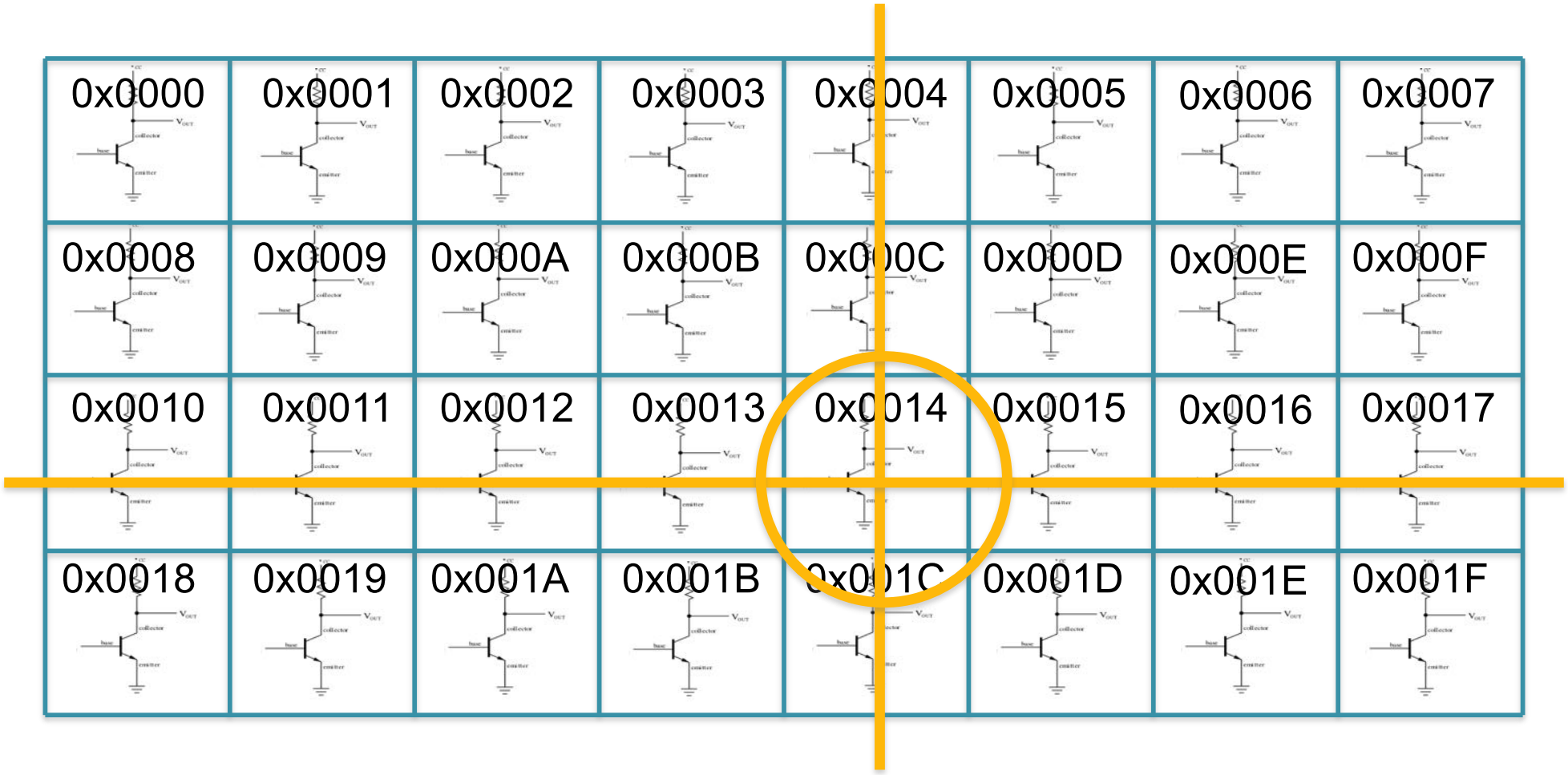
Accessing RAM




```
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

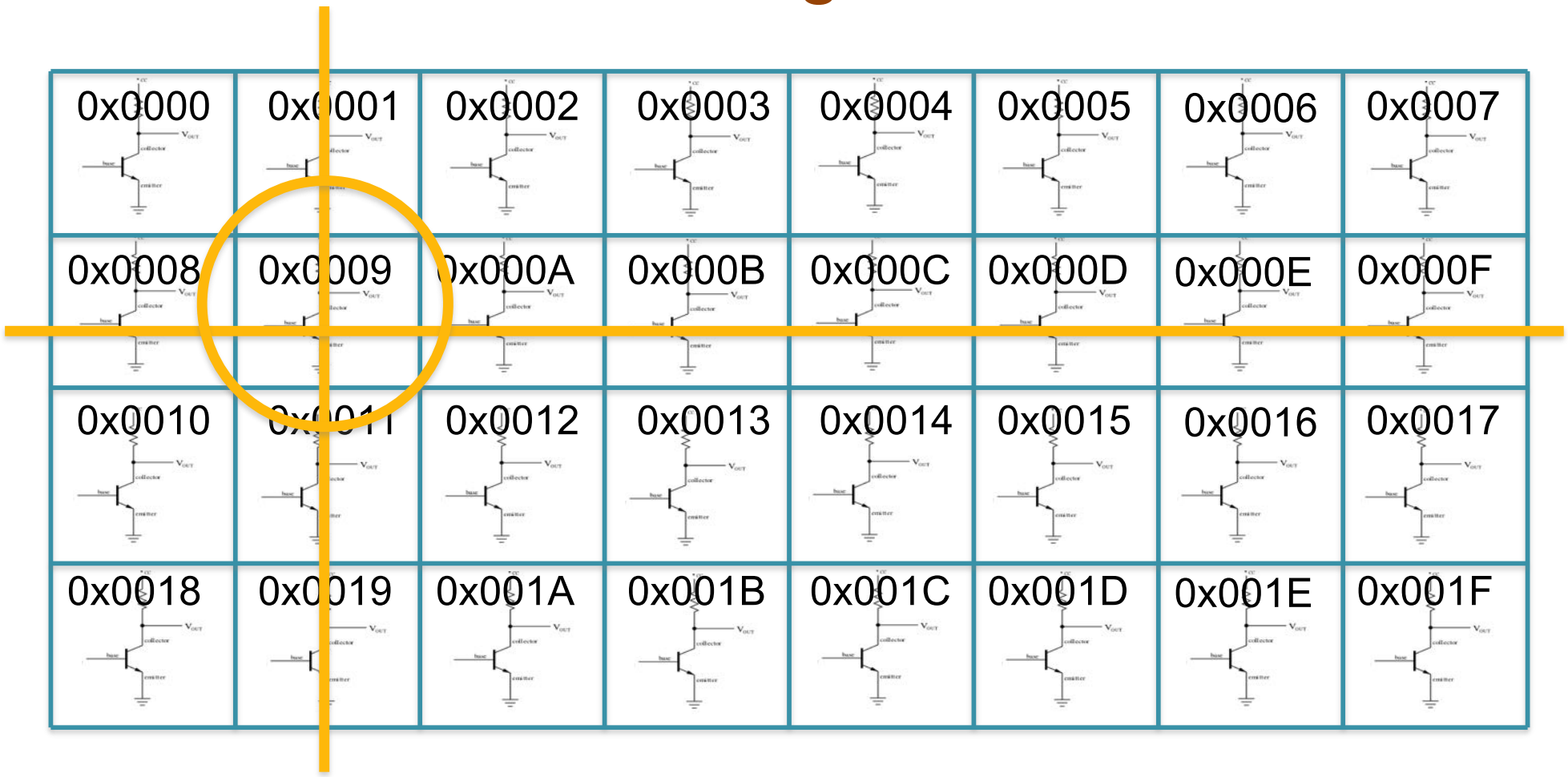
Accessing RAM




```
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

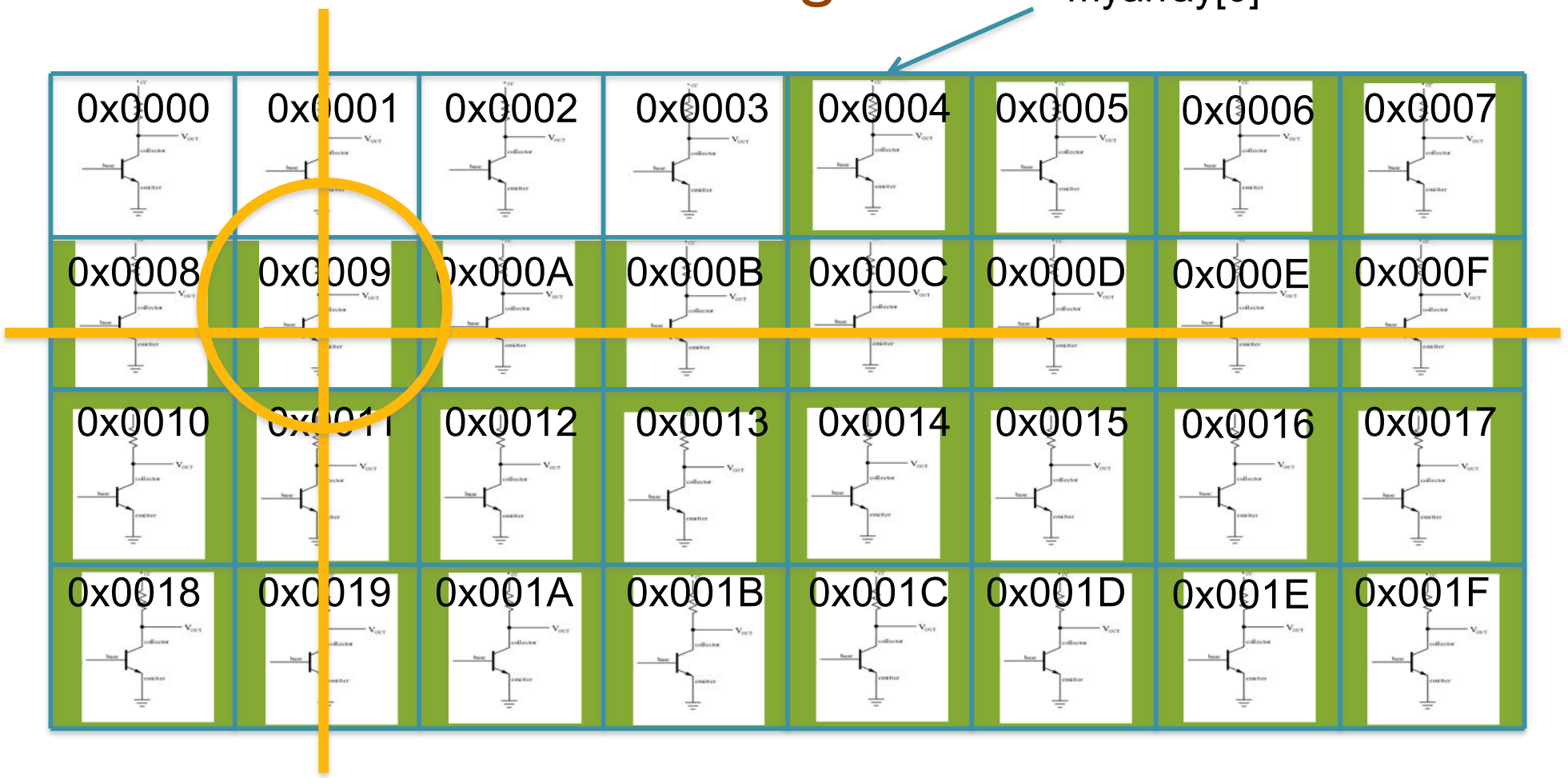
Accessing RAM




```
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

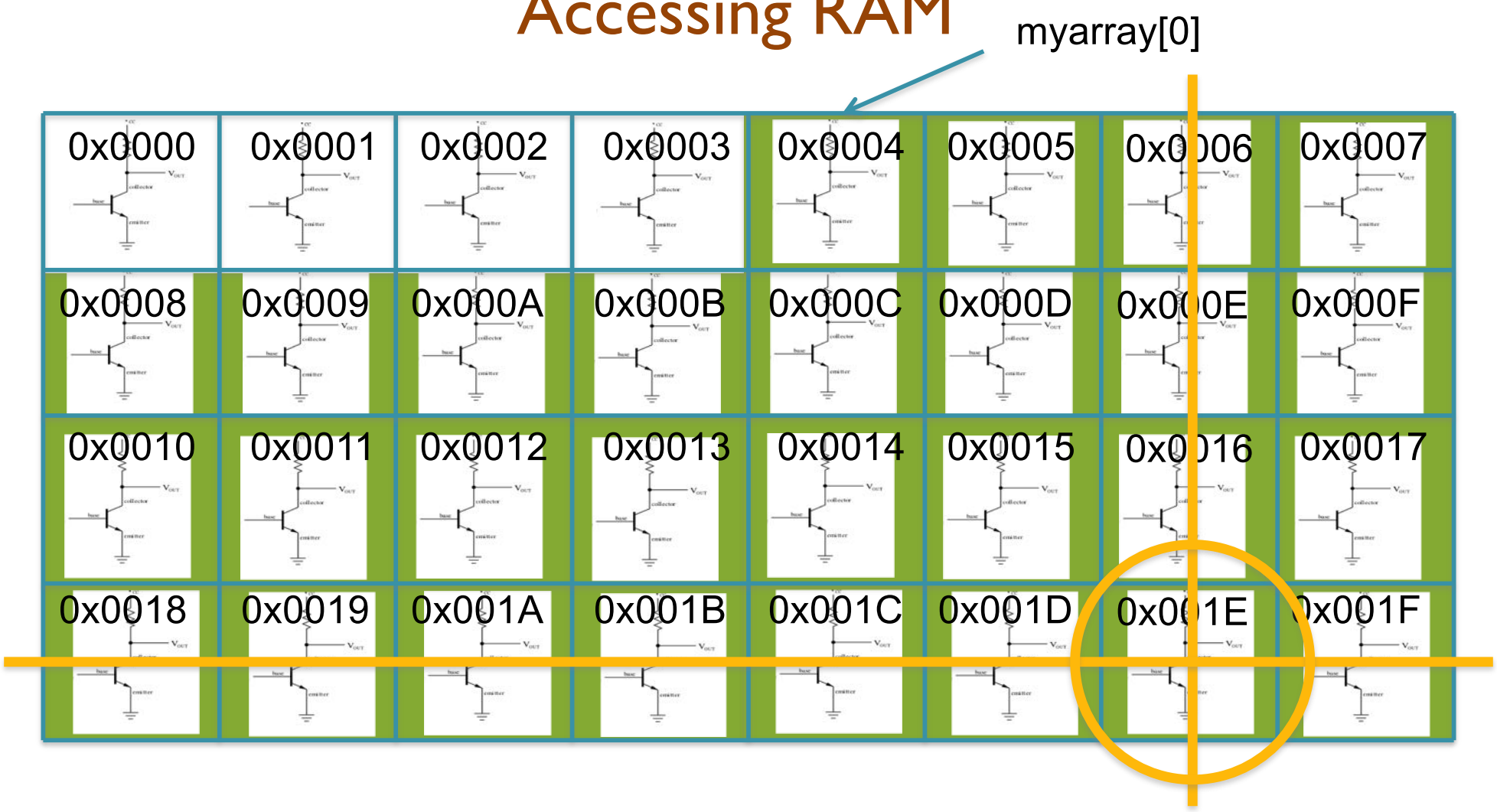
Accessing RAM



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

SimpleArray-short.java

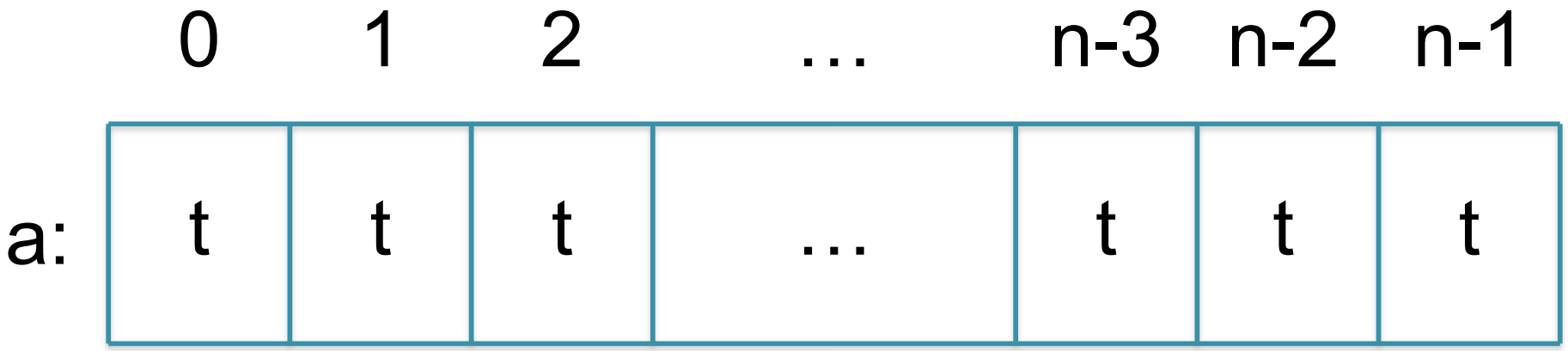
```java
public class SimpleArray<T> implements Array<T> {
    private T[] data;

    public SimpleArray(int n, T t) throws LengthException {
        if (n <= 0) {
            throw new LengthException();
        }

        this.data = (T[]) new Object[n];

        for (int i = 0; i < n; i++) {
            this.data[i] = t;
        }
    }

    public T get(int i) throws IndexException {
        try {
            return this.data[i];
        } catch (ArrayIndexOutOfBoundsException e) {
            throw new IndexException();
        }
    }

    public void put(int i, T t) throws IndexException {
        try {
            this.data[i] = t;
        } catch (ArrayIndexOutOfBoundsException e) {
            throw new IndexException();
        }
    }

    public int length() {
        return this.data.length;
    }
}
```

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



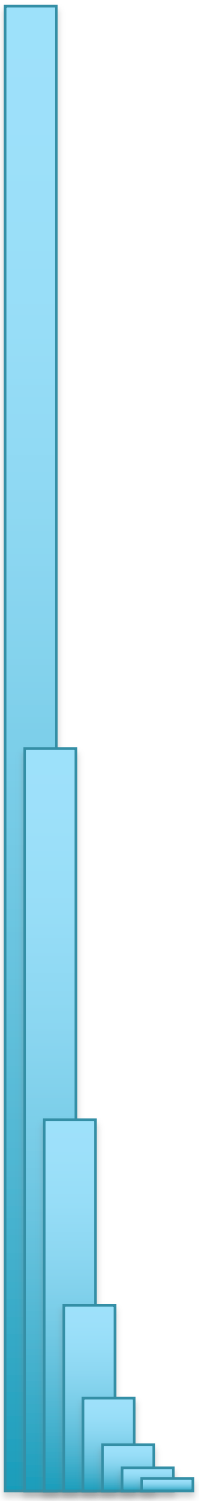
- Fixed length data structure
- Constant time get() and put() methods
- Definitely needs to be generic ☺

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

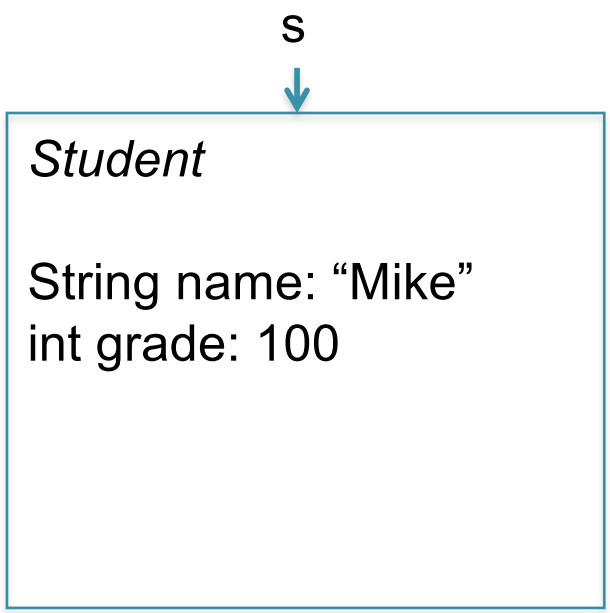
1. *Review HW1*
2. *Review Java Arrays*
3. *Java References and Linked Lists*



Java References

Student.java

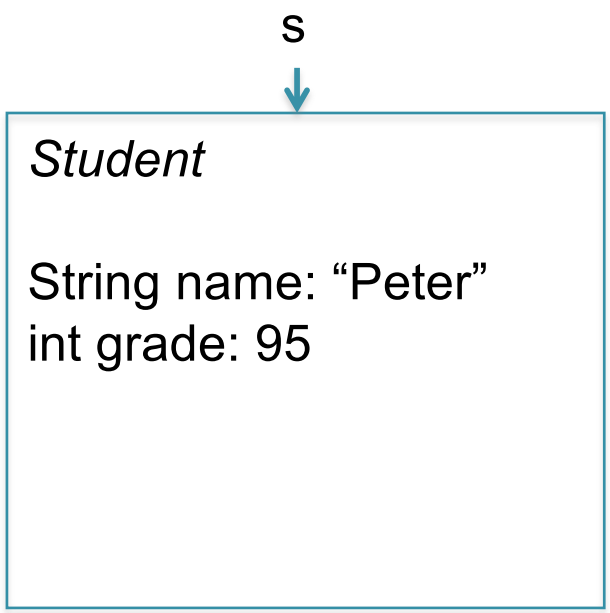
```
public class Student {
  private String name;
  private int grade;

  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);
```



Java References

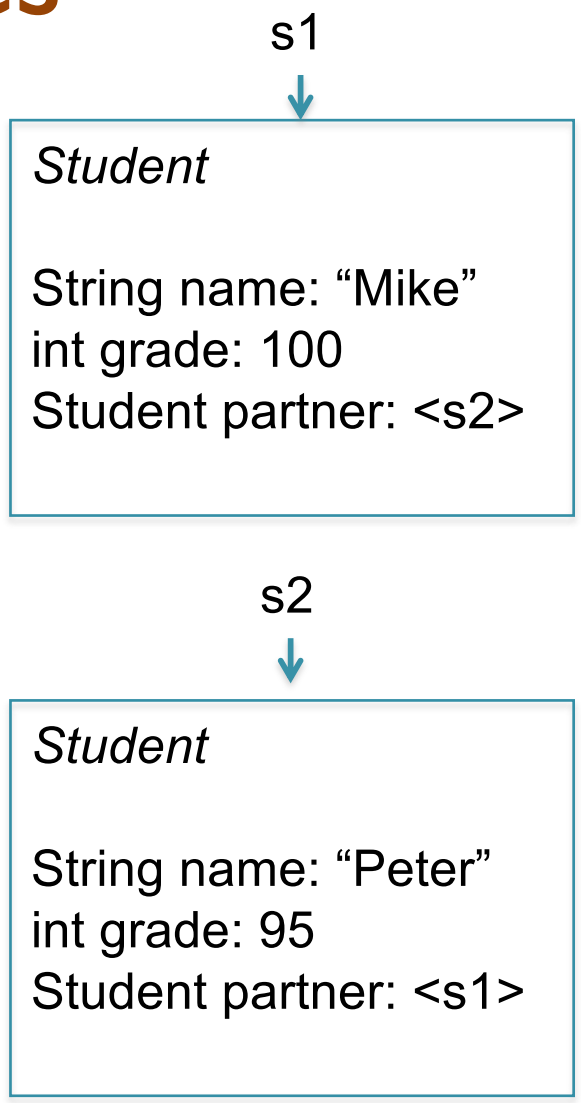
Student.java

```
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);
```



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);
```



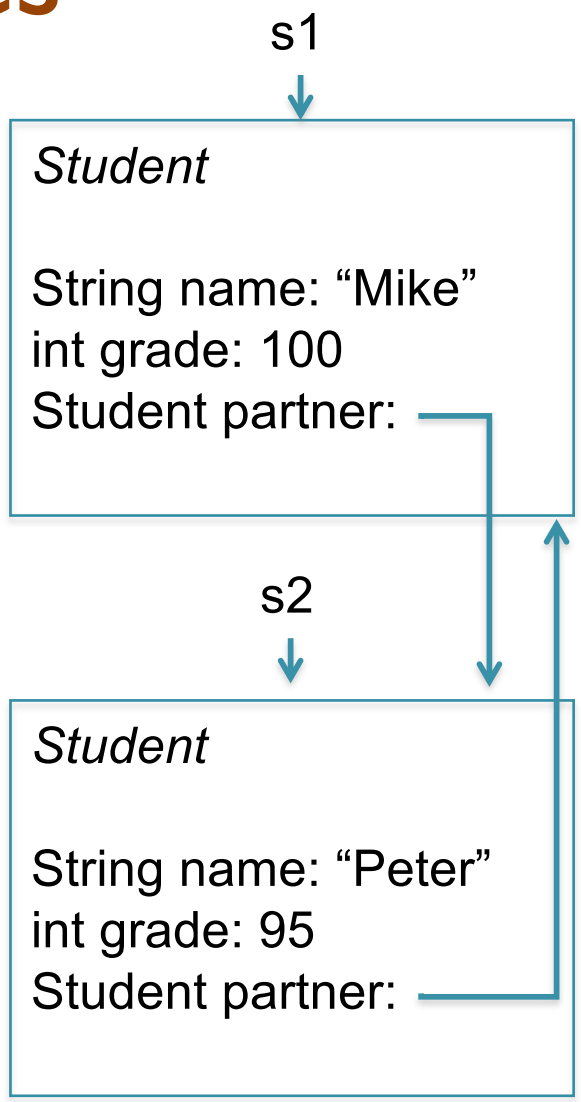

```
System.out.println(s1.getName() + " " + s1.getGrade()
                   + " " + s1.getPartner().getName());
System.out.println(s2.getName() + " " + s2.getGrade()
                   + " " + s2.getPartner().getName());
```

```
Mike 100 Peter
Peter 95 Mike
```

Java References

s1 and s2 are "just" references to the "Mike" and "Peter" objects

```
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());
```

```
Mike 100 Peter
Peter 95 Mike
```

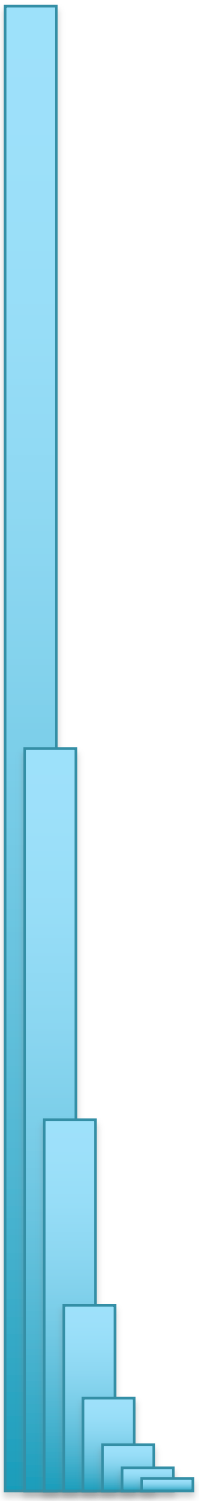
Java References

```
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?

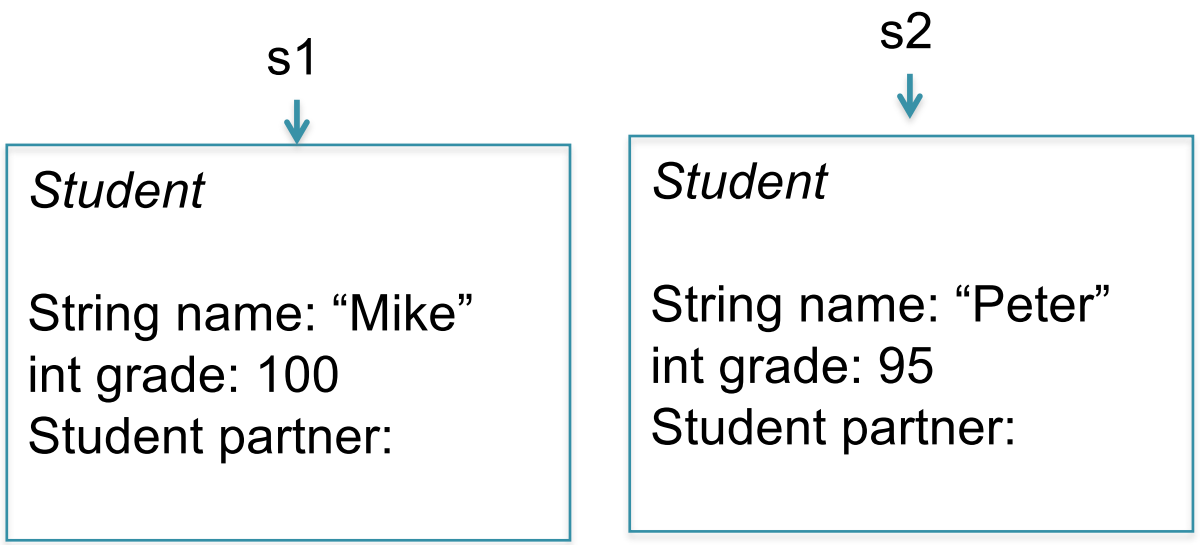
Java References

```
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());
```



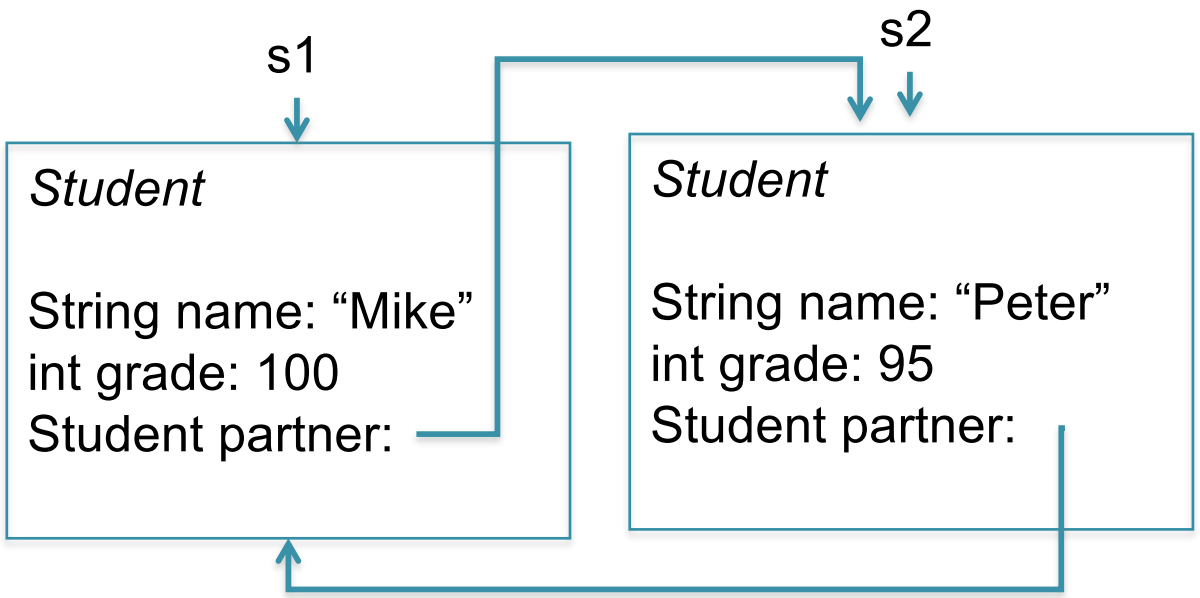
Java References

```
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());
```



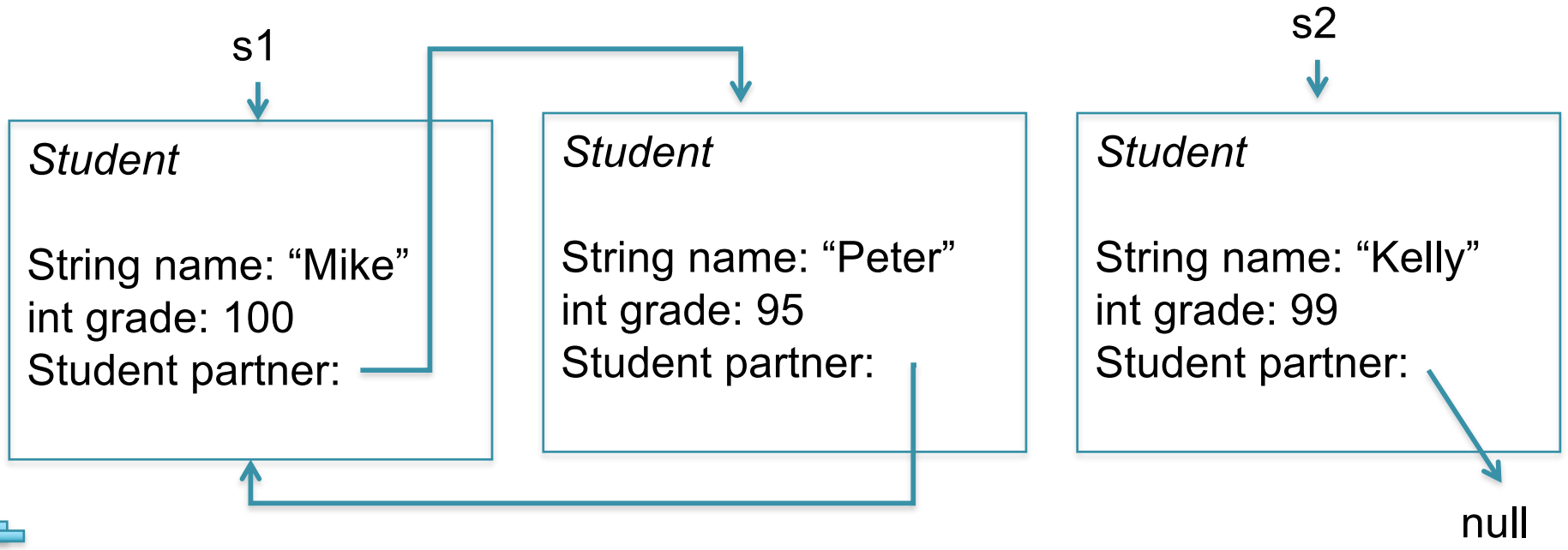
Java References

```
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());
```



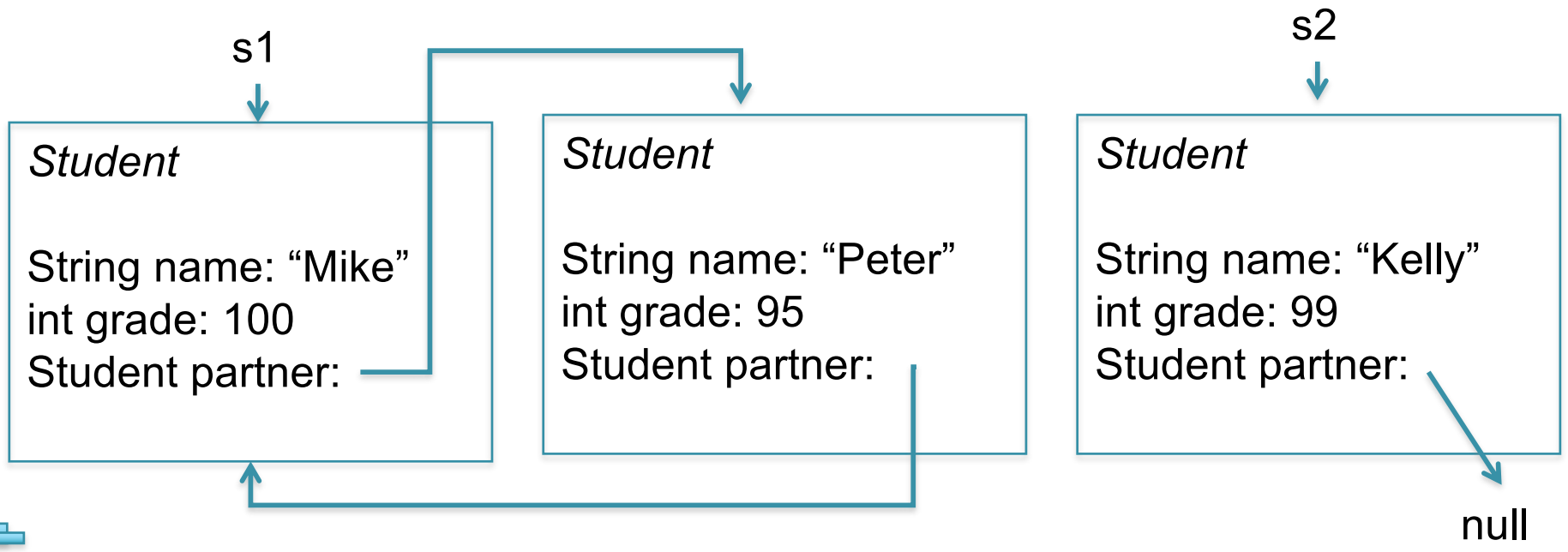
Java References

```
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());
```

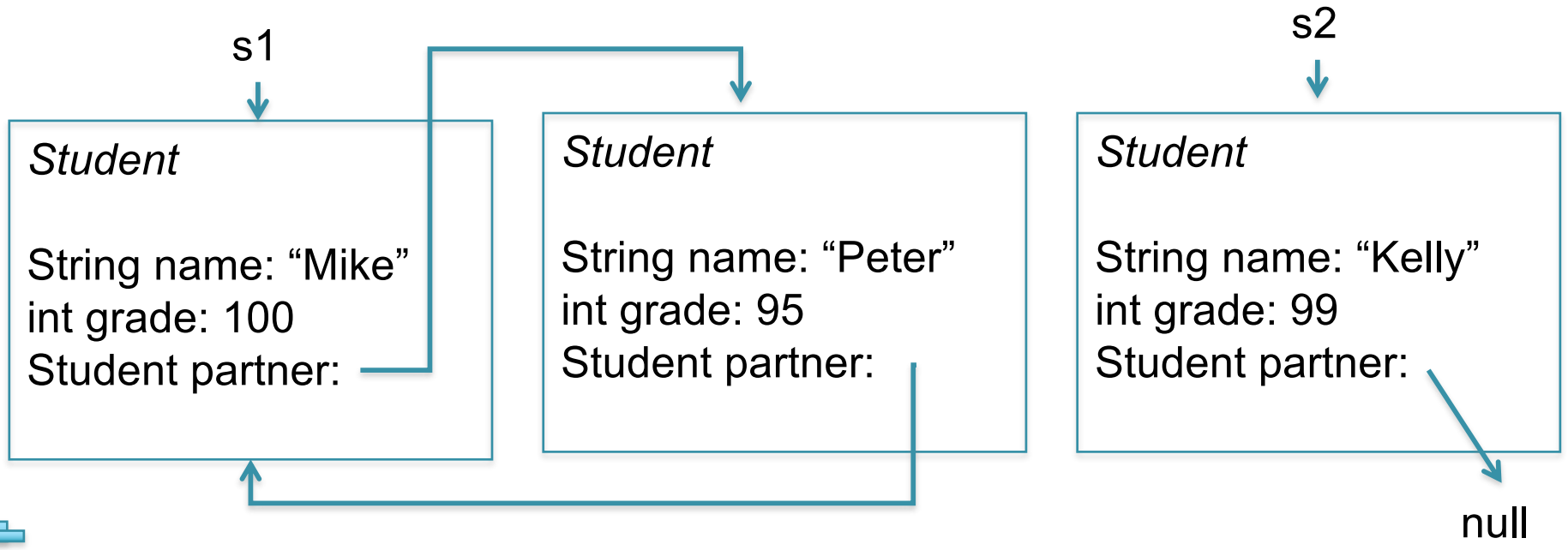


Java References

```
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());
```

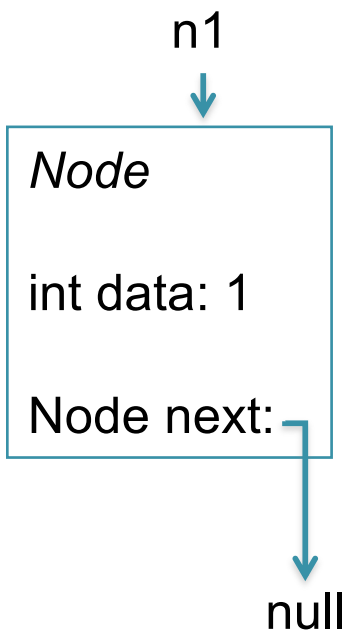


Java Nodes

```
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;
  }
```

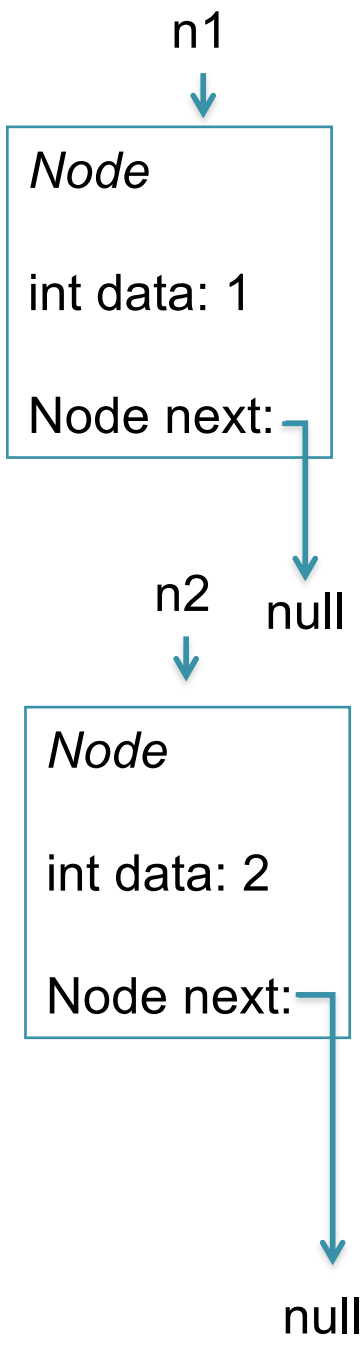
Java Nodes

```
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);
```



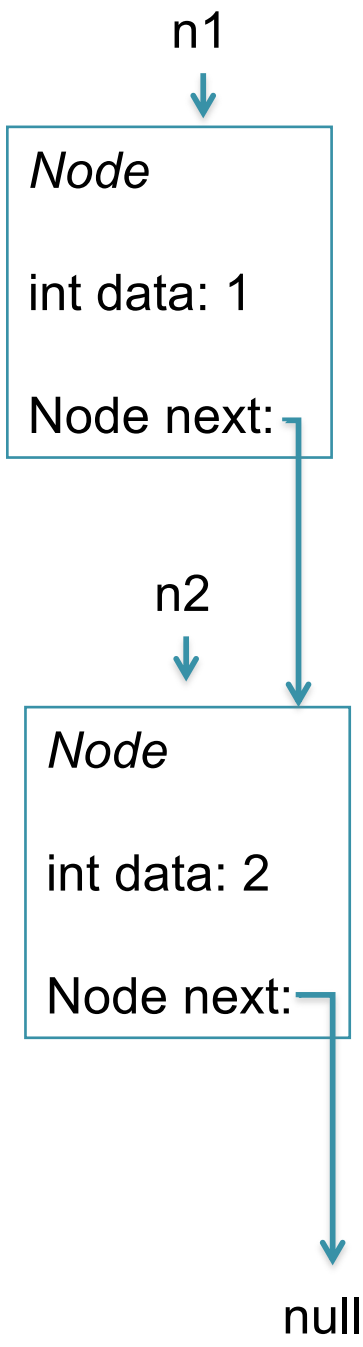
Java Nodes

```
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);
```



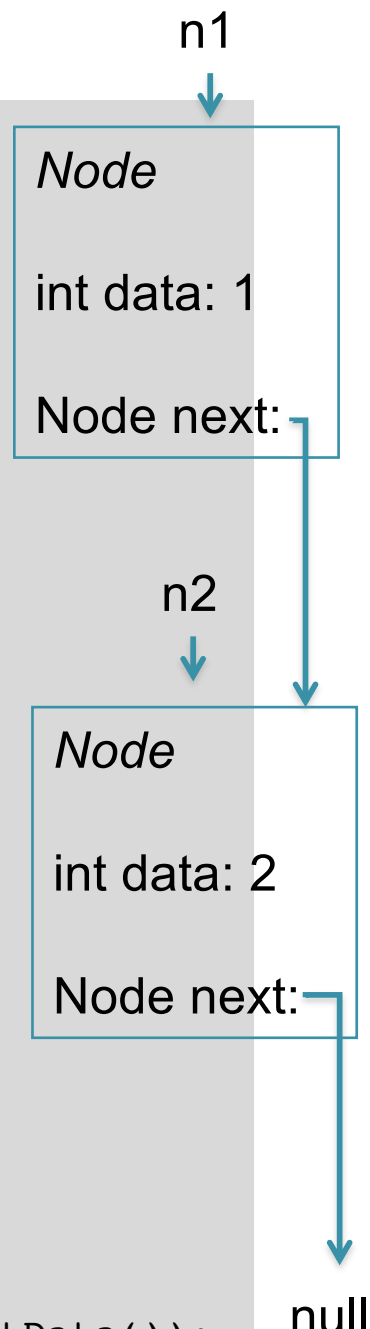
Java Nodes

```
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

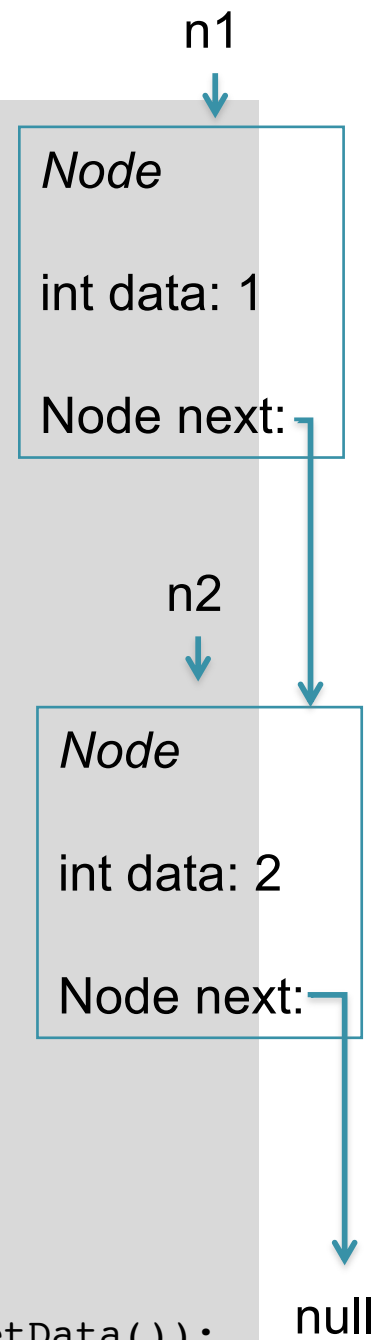
```
public class Node {
  private int data;
  private Node next;

  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;
  }

  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());
```

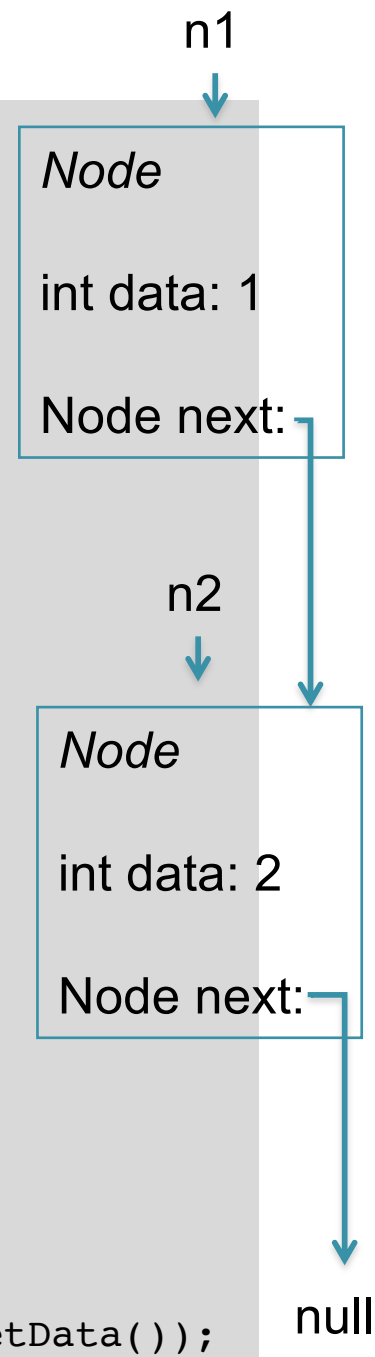
1 -> 2



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);

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

1 -> 2

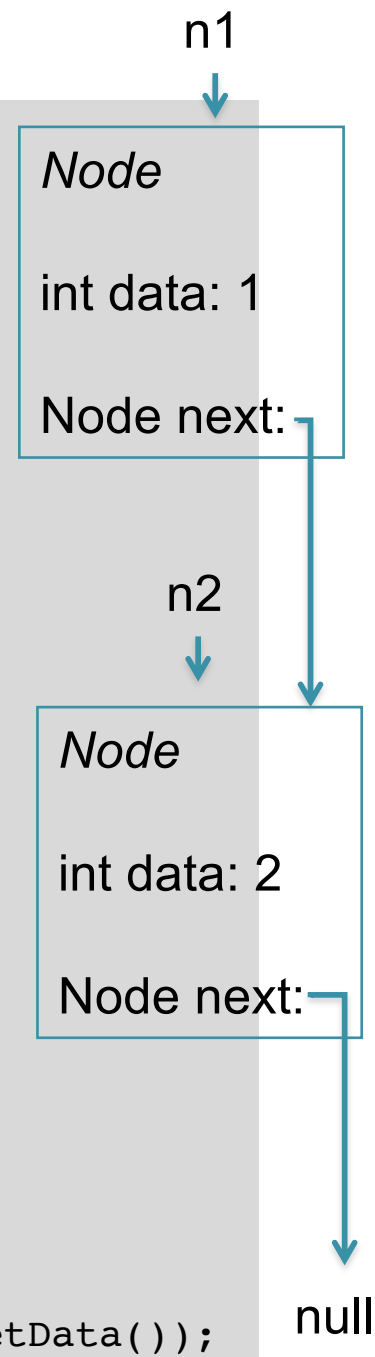


Java Nodes

```
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;
  }
```

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

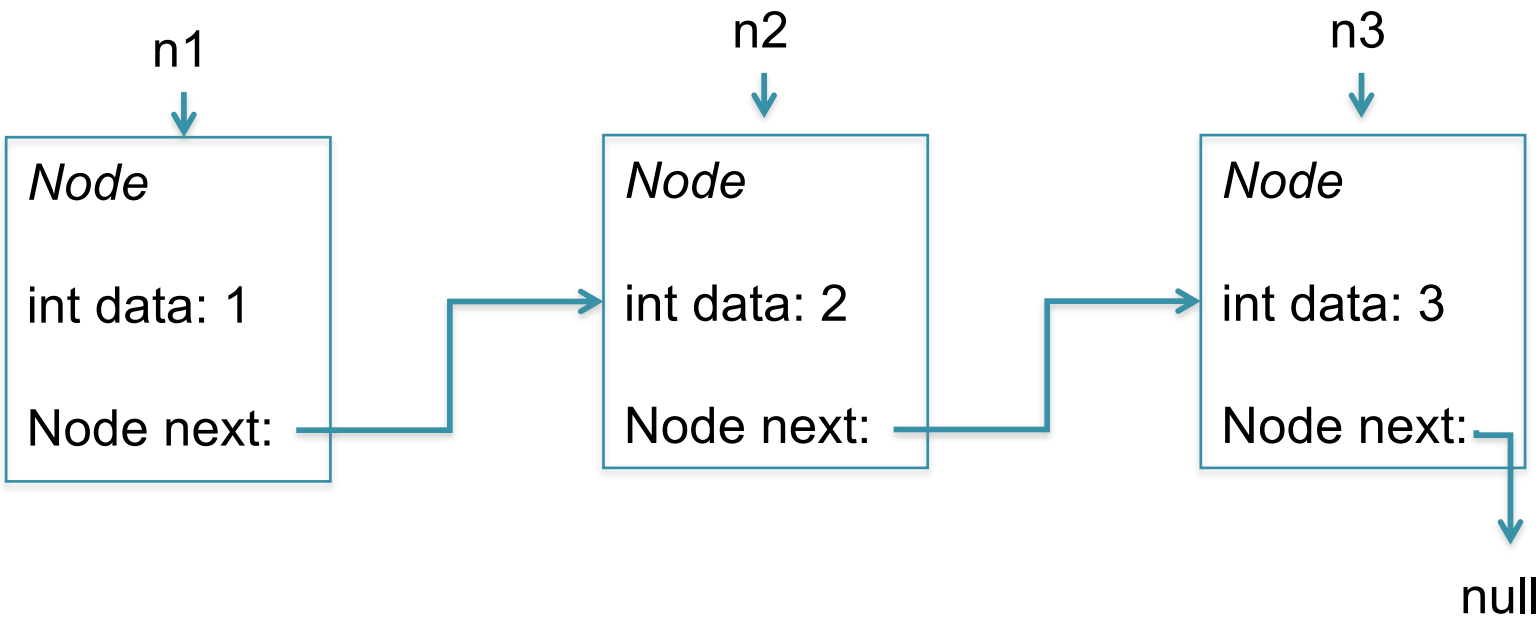
```
Node n1 = new Node(1);
Node n2 = new Node(2);
n1.setNext(n2);

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



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());
```

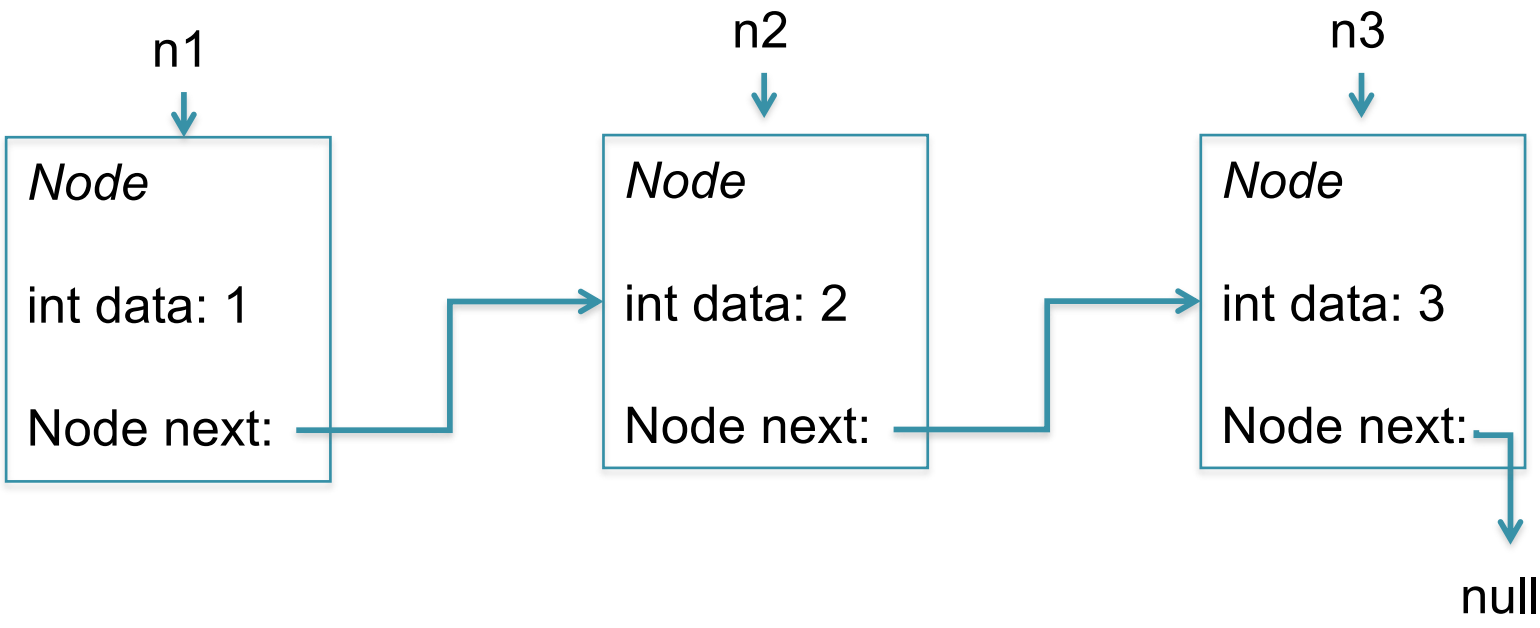
```
1 -> 2
```



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());
```

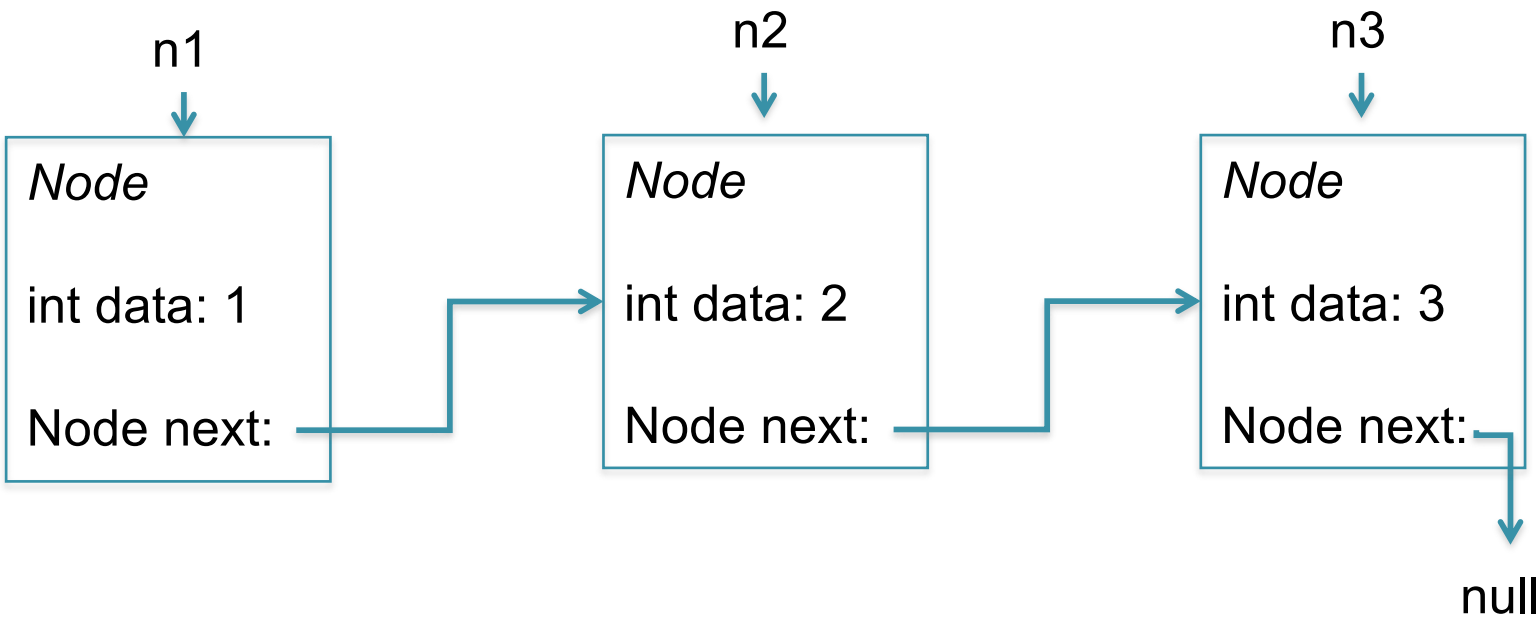
```
1 -> 2
2 -> 3
```



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());
```

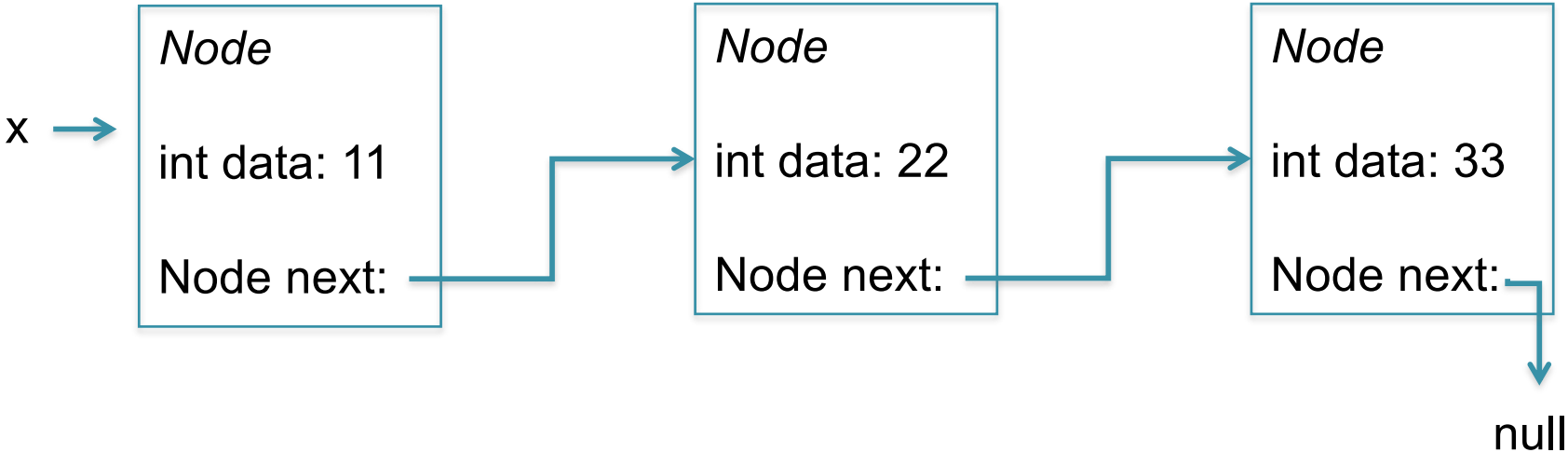
```
1 -> 2
2 -> 3
Exception in thread "main" java.lang.NullPointerException
        at Node.main(Node.java:32)
```



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());
```

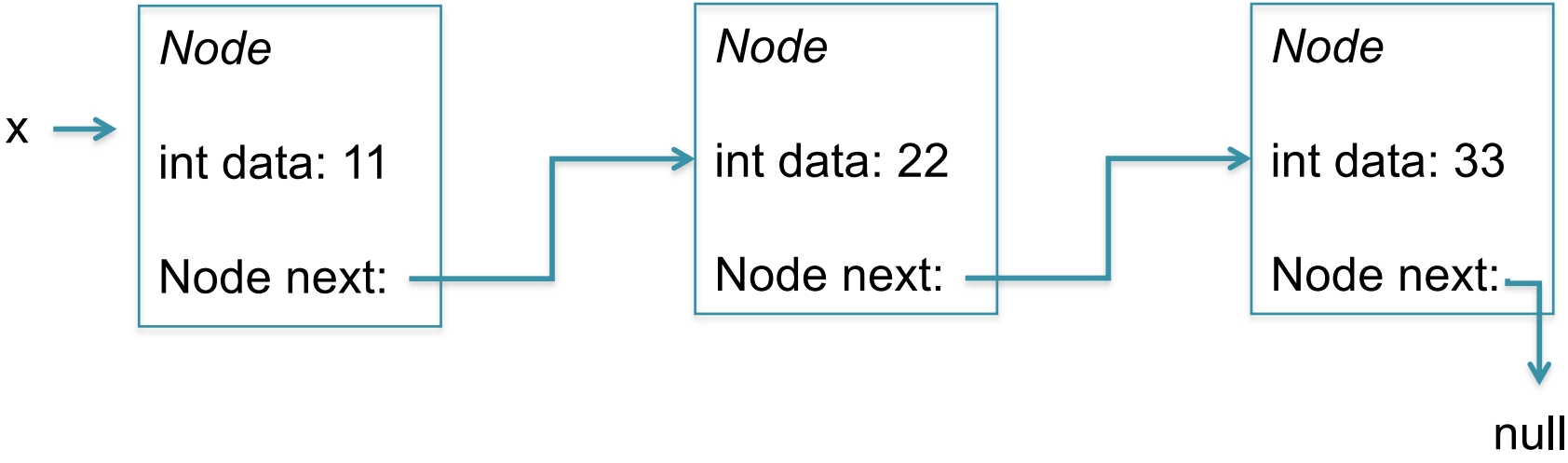
```
11 -> 22
```



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());
```

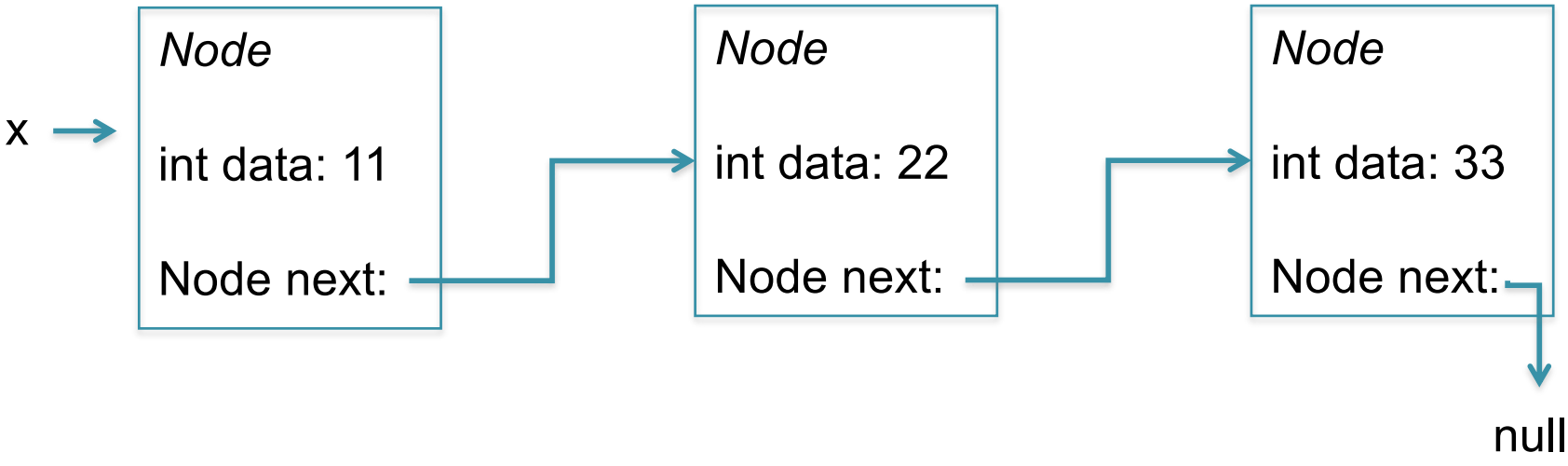
```
11 -> 22
22 -> 33
```



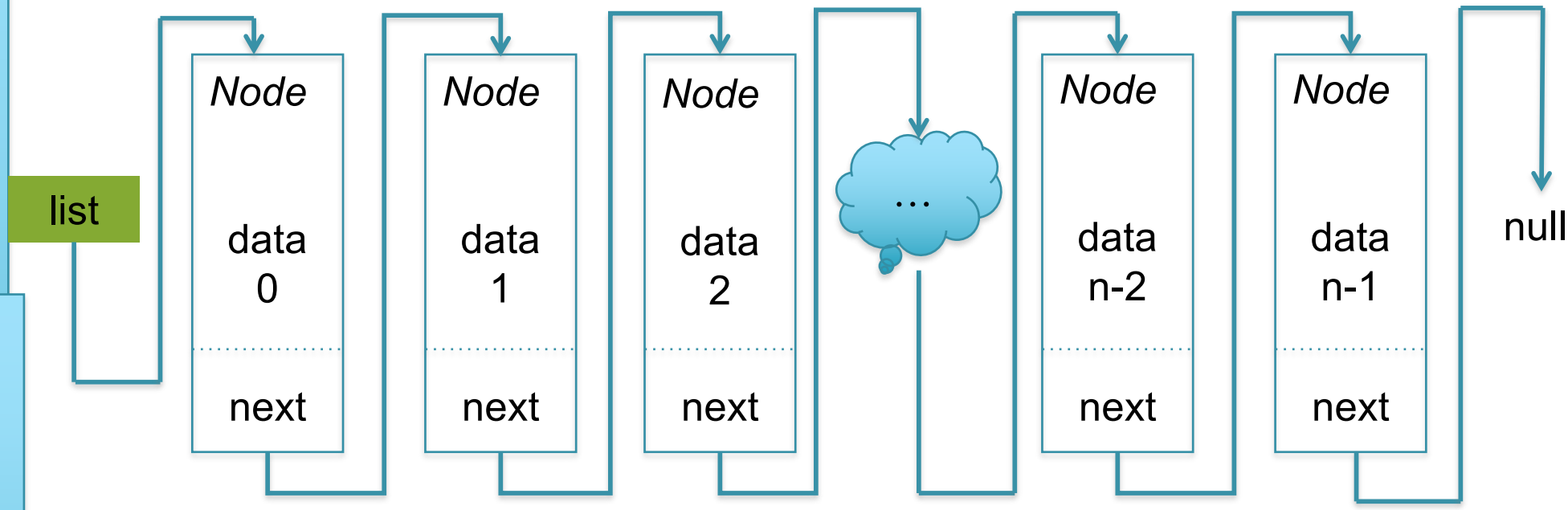
Java Nodes

```
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

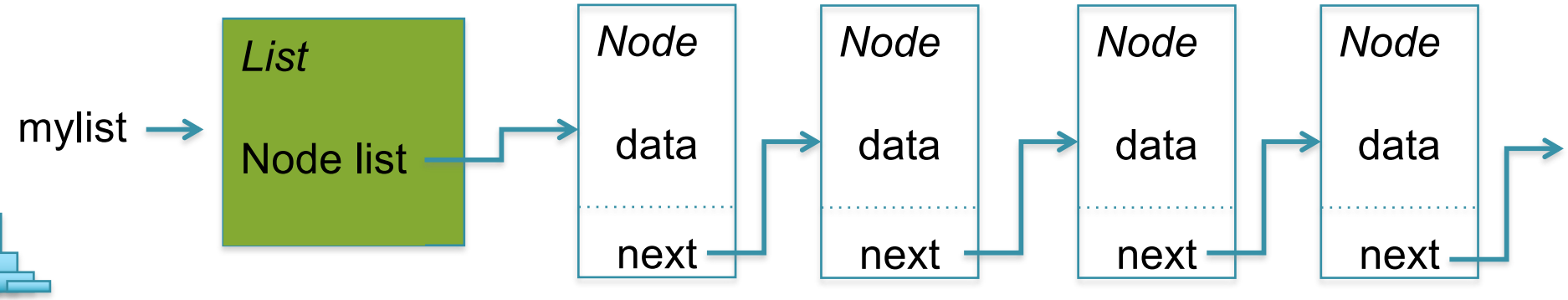
List Class

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


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

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;
 }
```

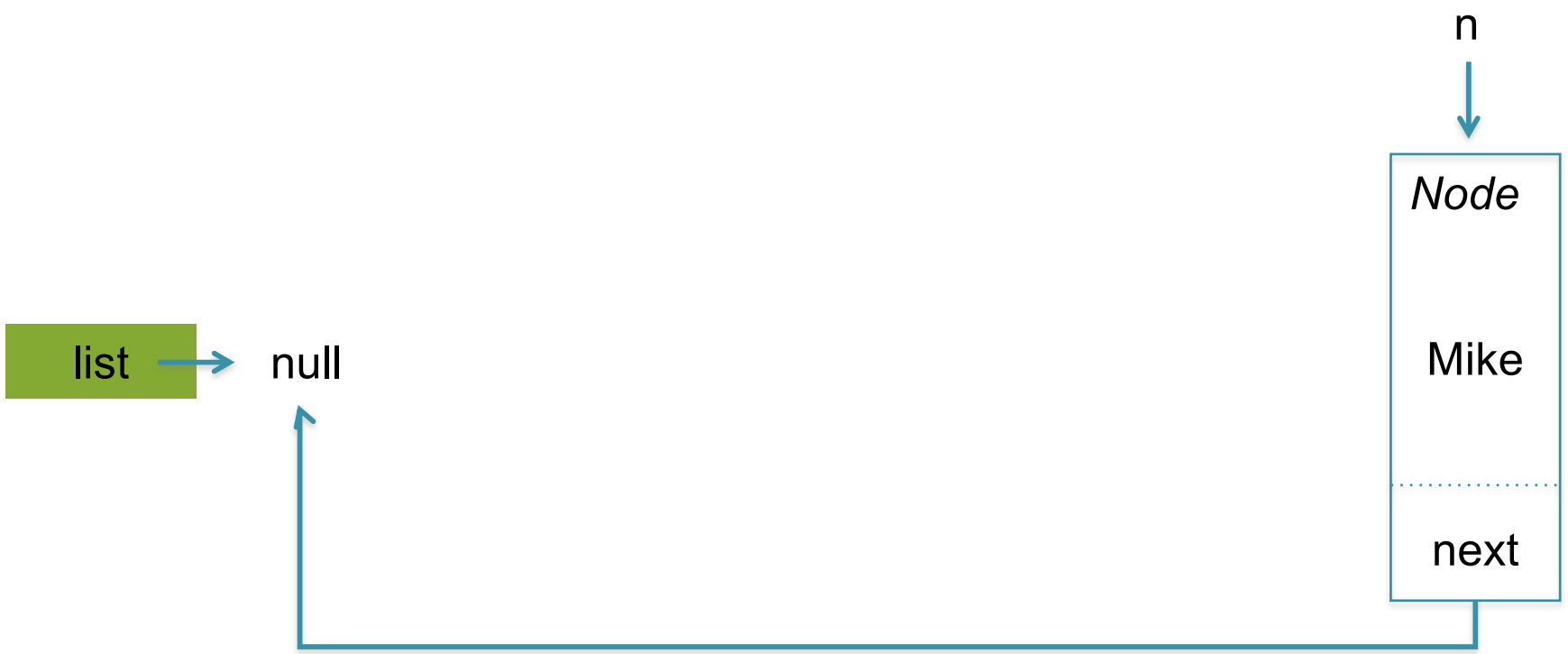
Linked List Construction


```
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

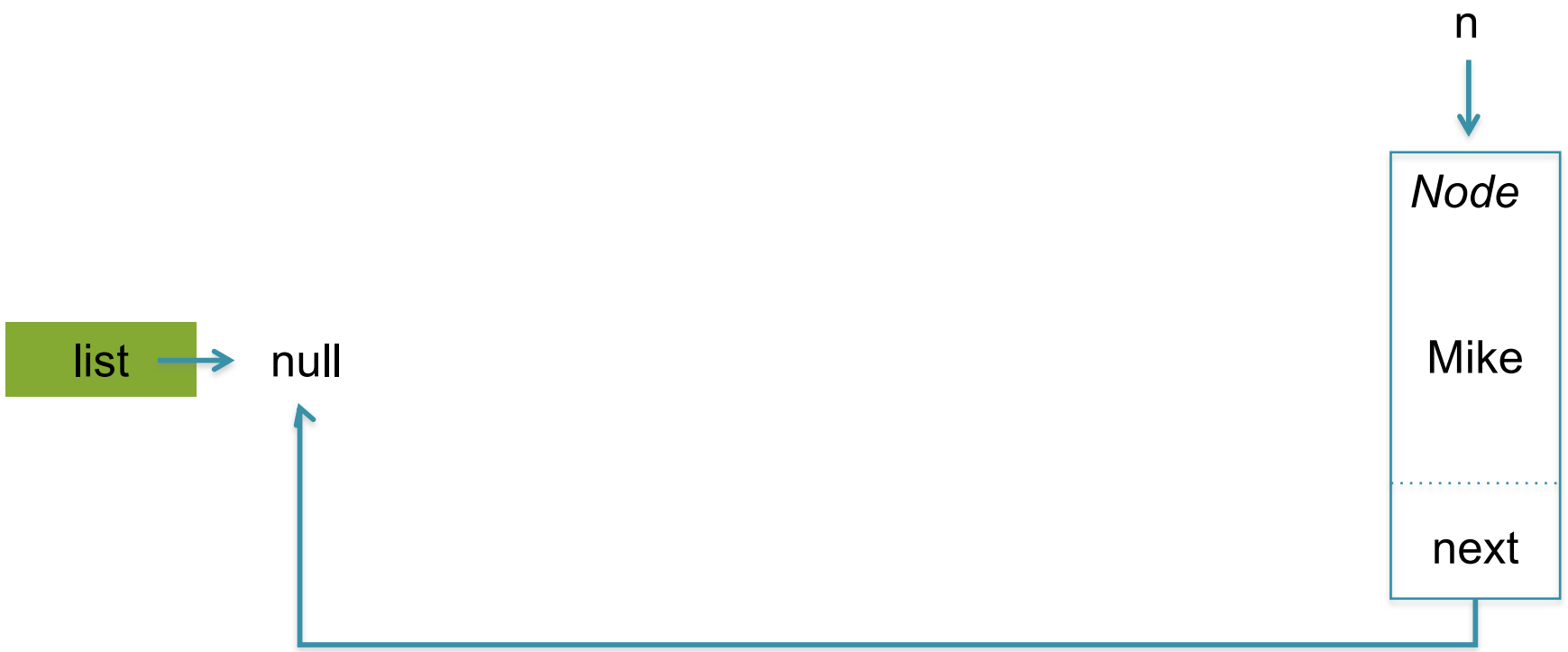



```
mylist.add("Mike");
```

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;
 }
```

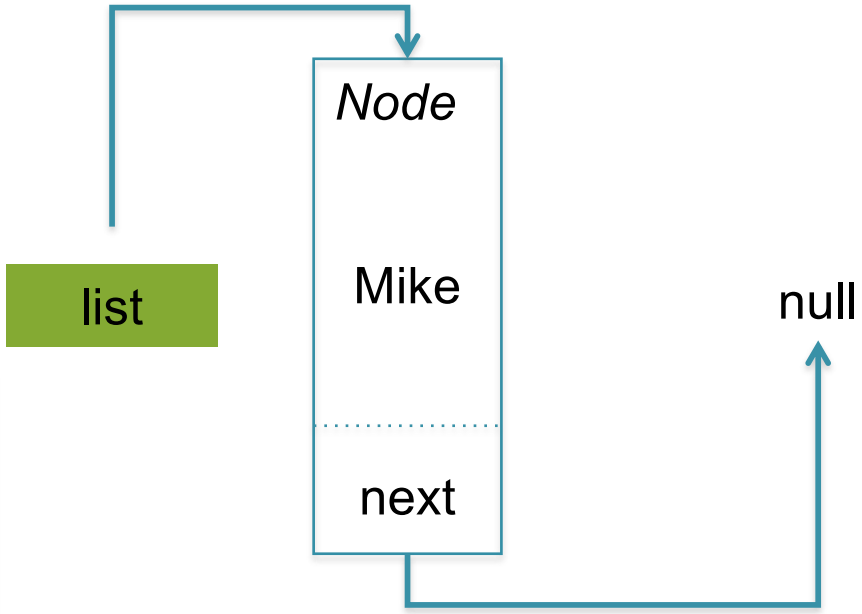
Linked List Construction


```
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

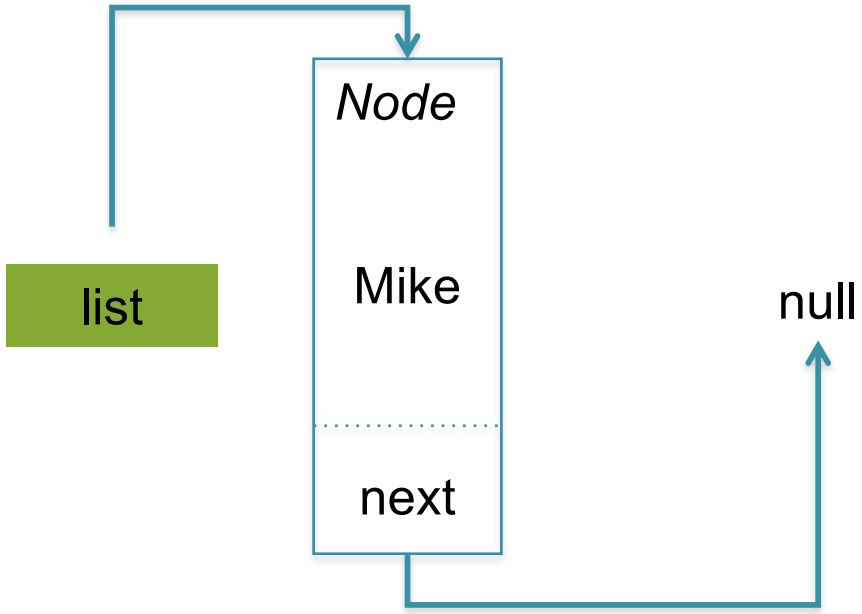



```
mylist.add("Mike");
```

List.java

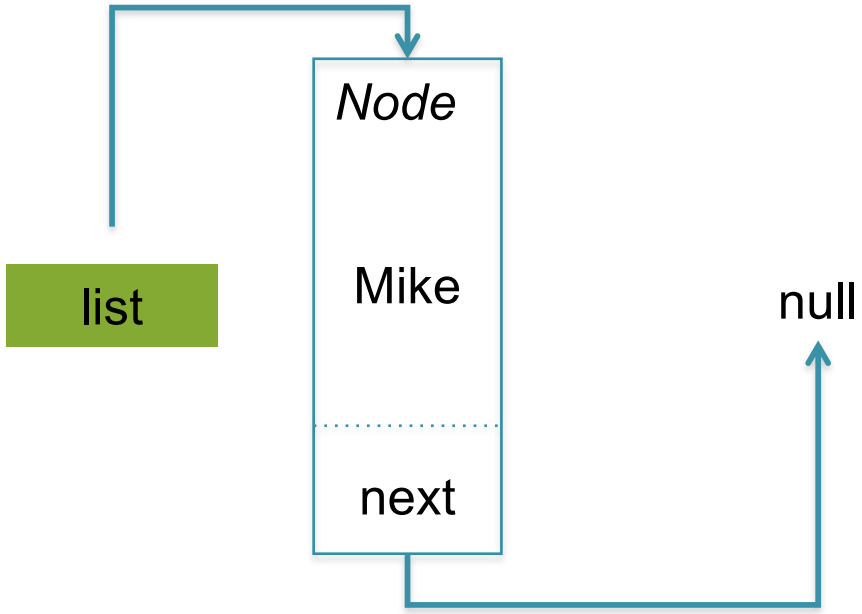
```
public void add(String value){
   StringNode n = new StringNode(value);
   n.setNext(this.list);
   this.list = n;
 }
```

Linked List Construction



First node successfully added (prepend)
Questions?

Linked List Construction

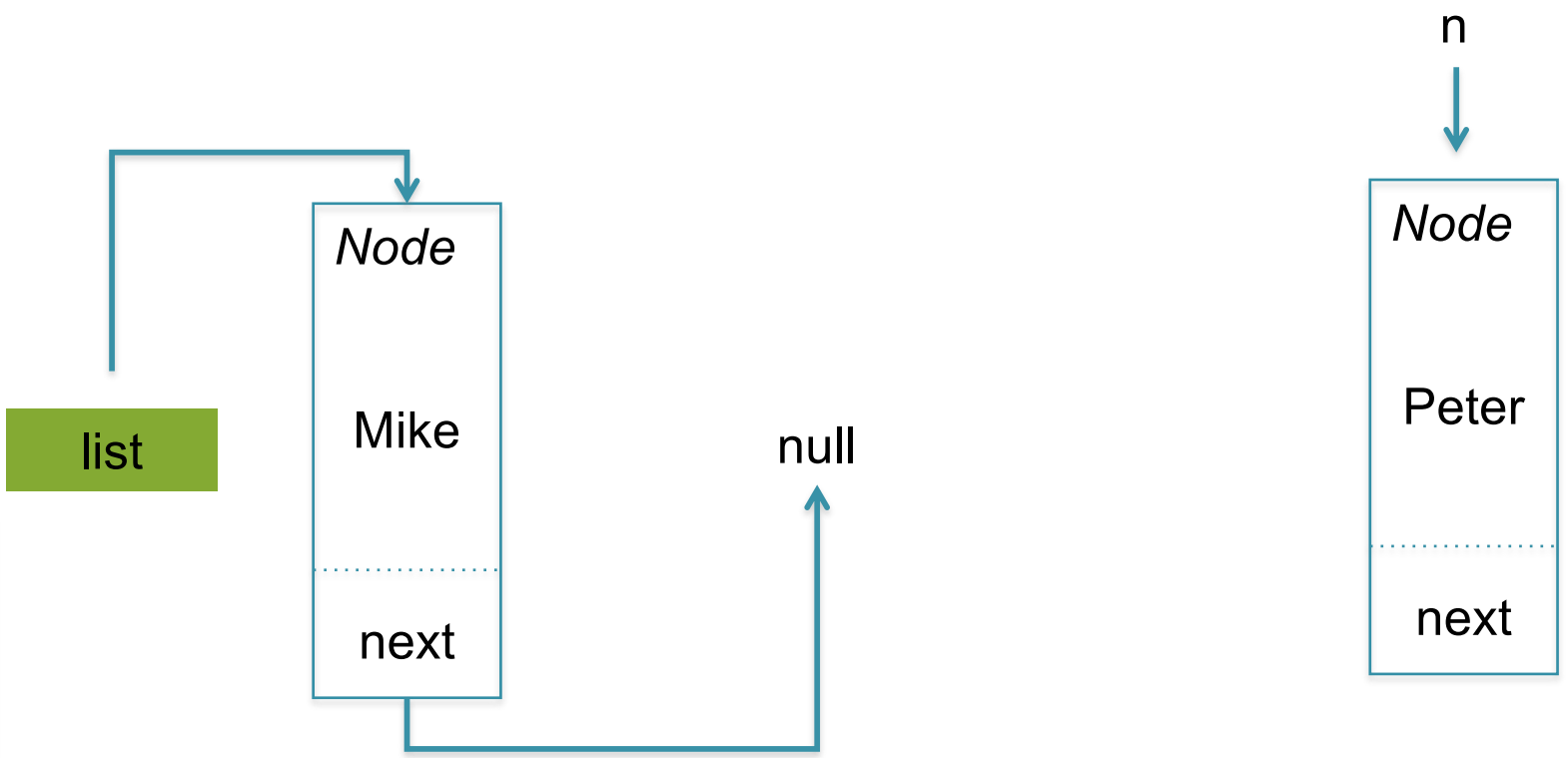



```
mylist.add("Peter");
```

List.java

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

Linked List Construction

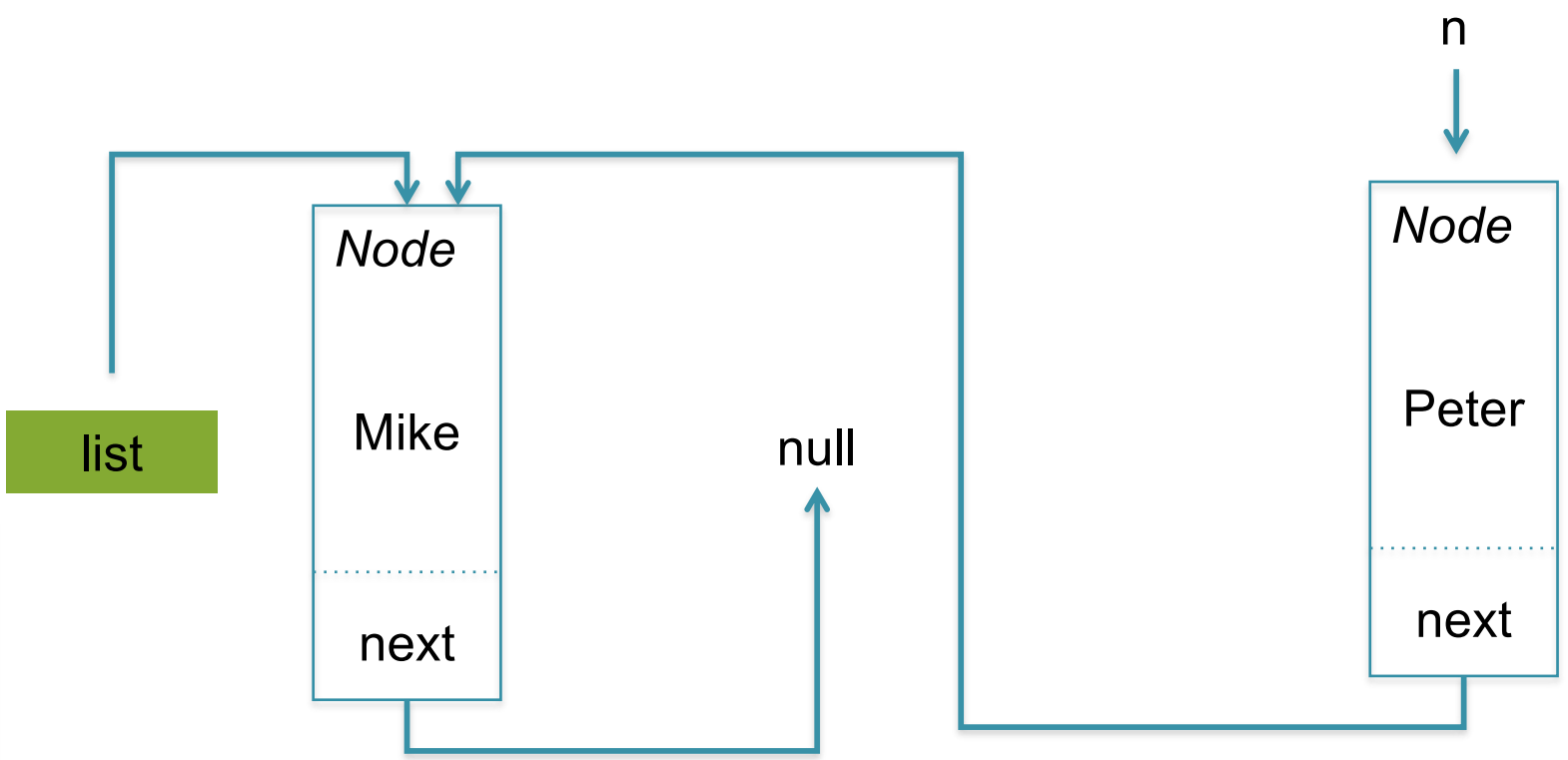



```
mylist.add("Peter");
```

List.java

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

Linked List Construction

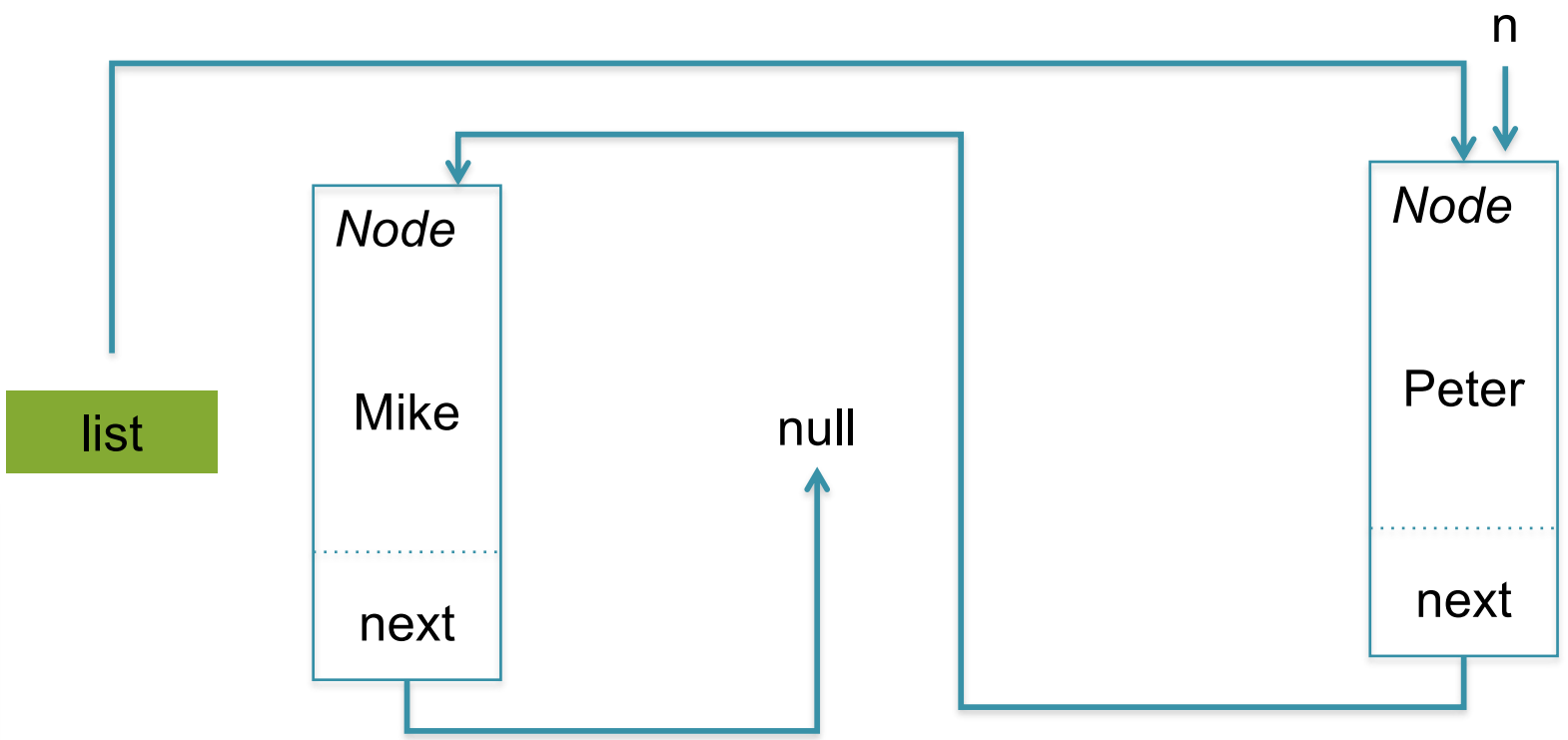



```
mylist.add("Peter");
```

List.java

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

Linked List Construction

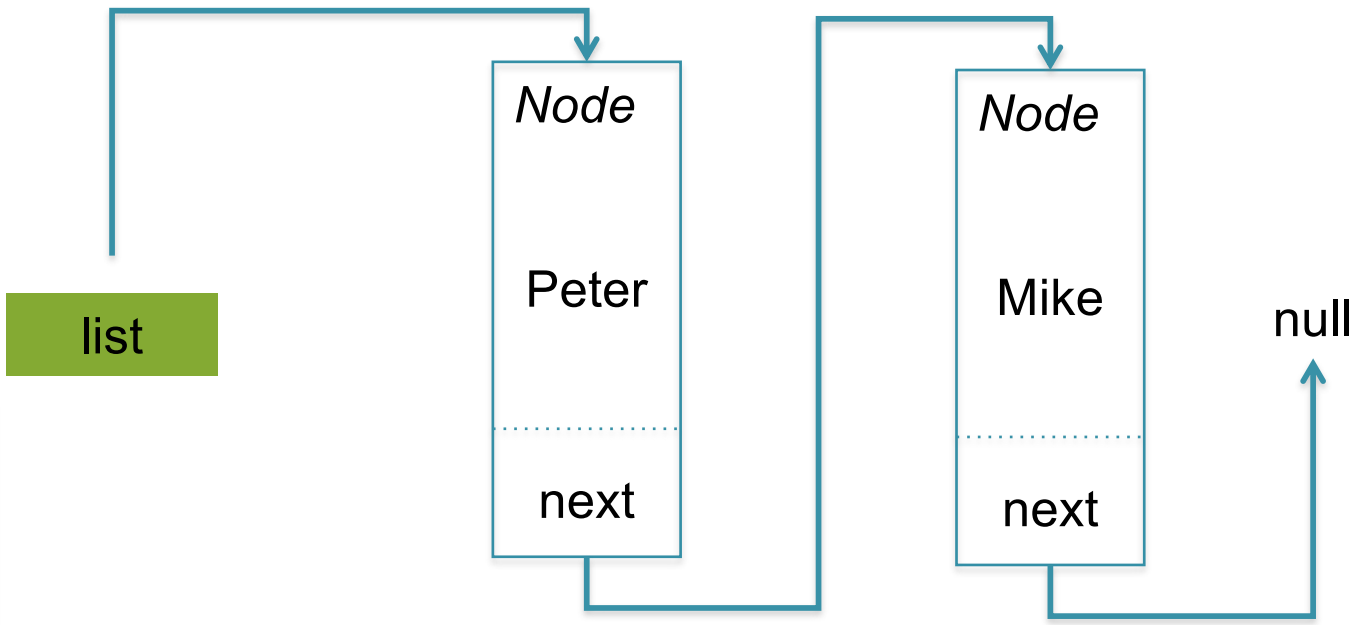



```
mylist.add("Peter");
```

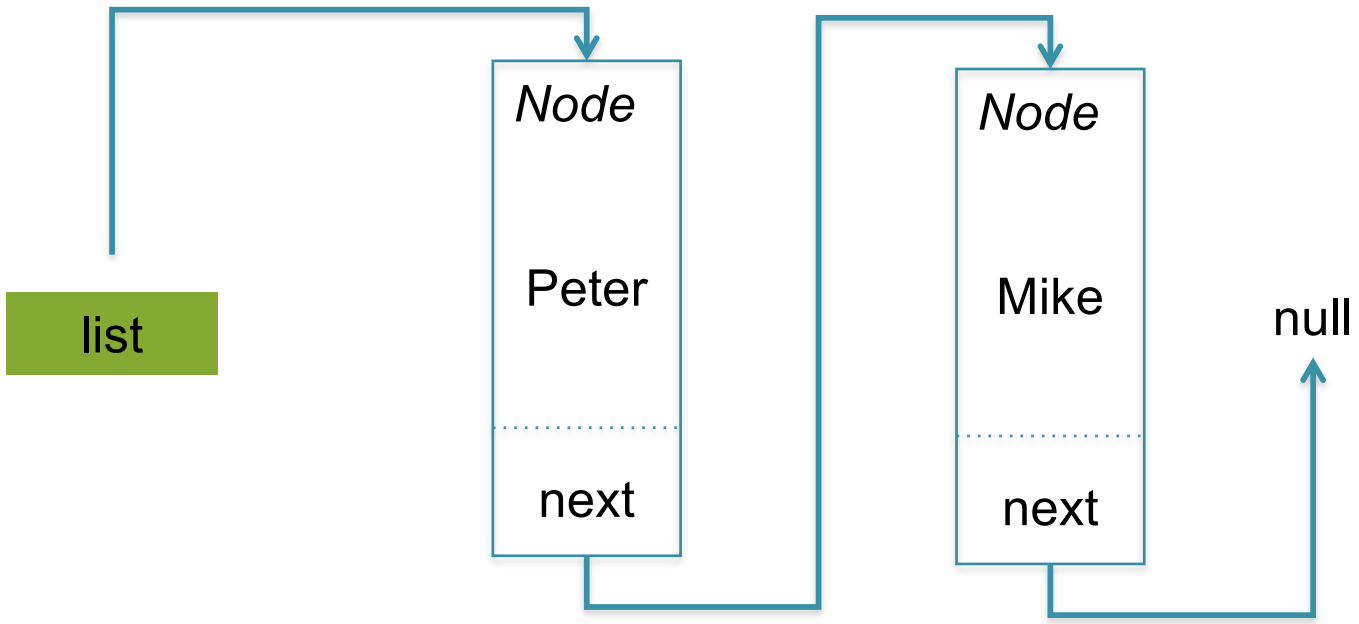
List.java

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

Linked List Construction



Linked List Construction

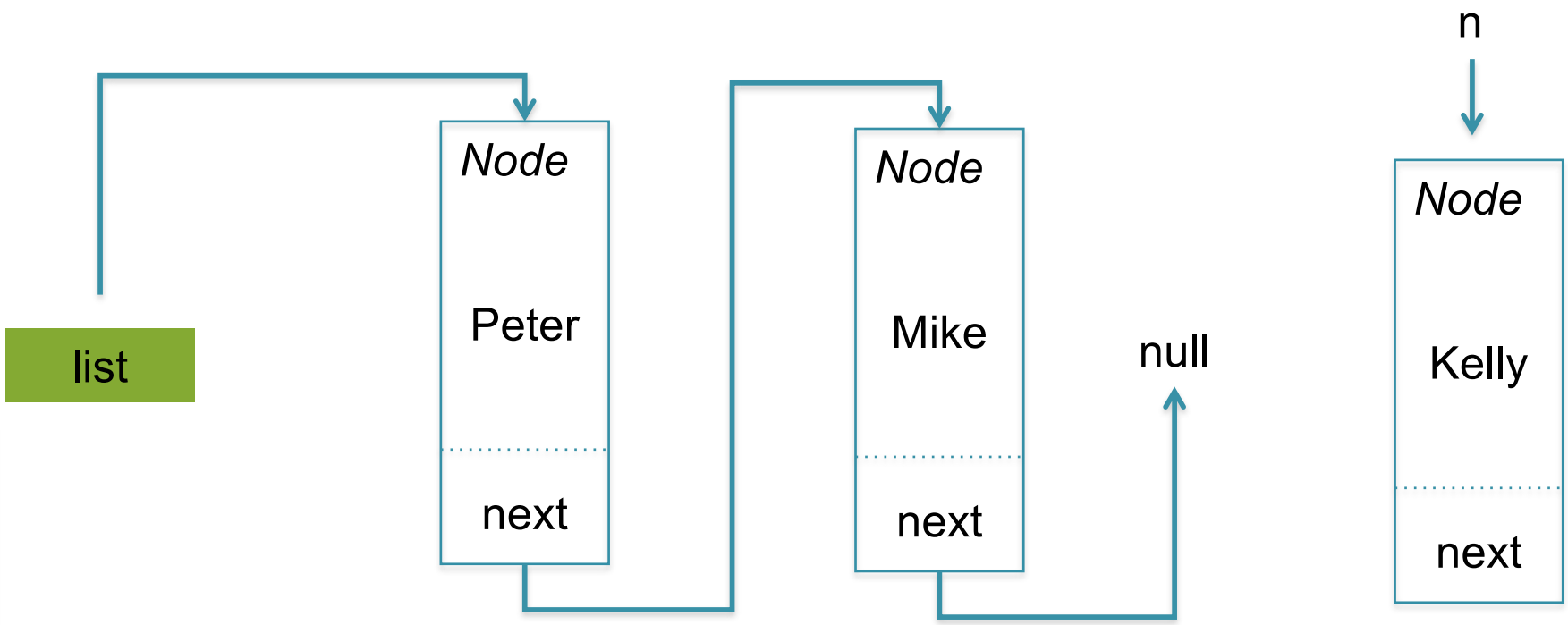



```
mylist.add("Kelly");
```

List.java

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

Linked List Construction

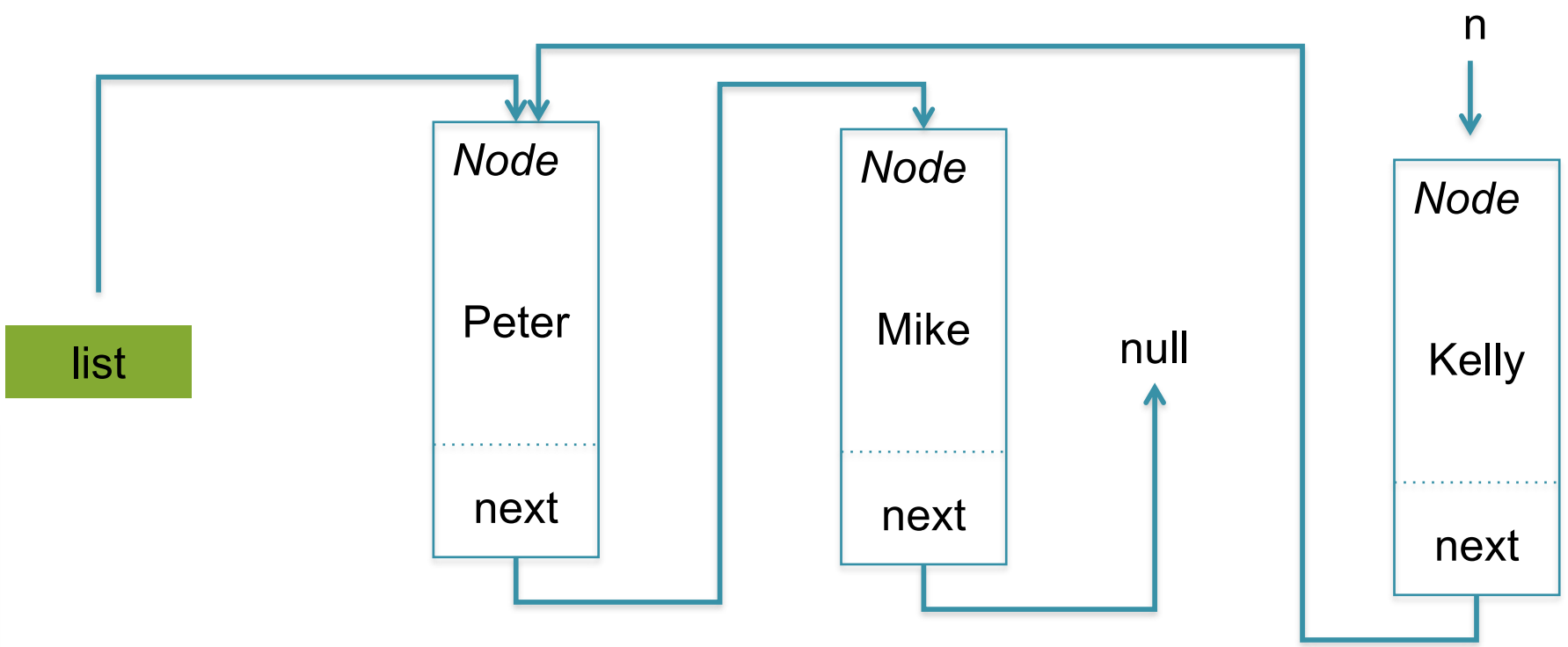



```
mylist.add("Kelly");
```

List.java

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

Linked List Construction

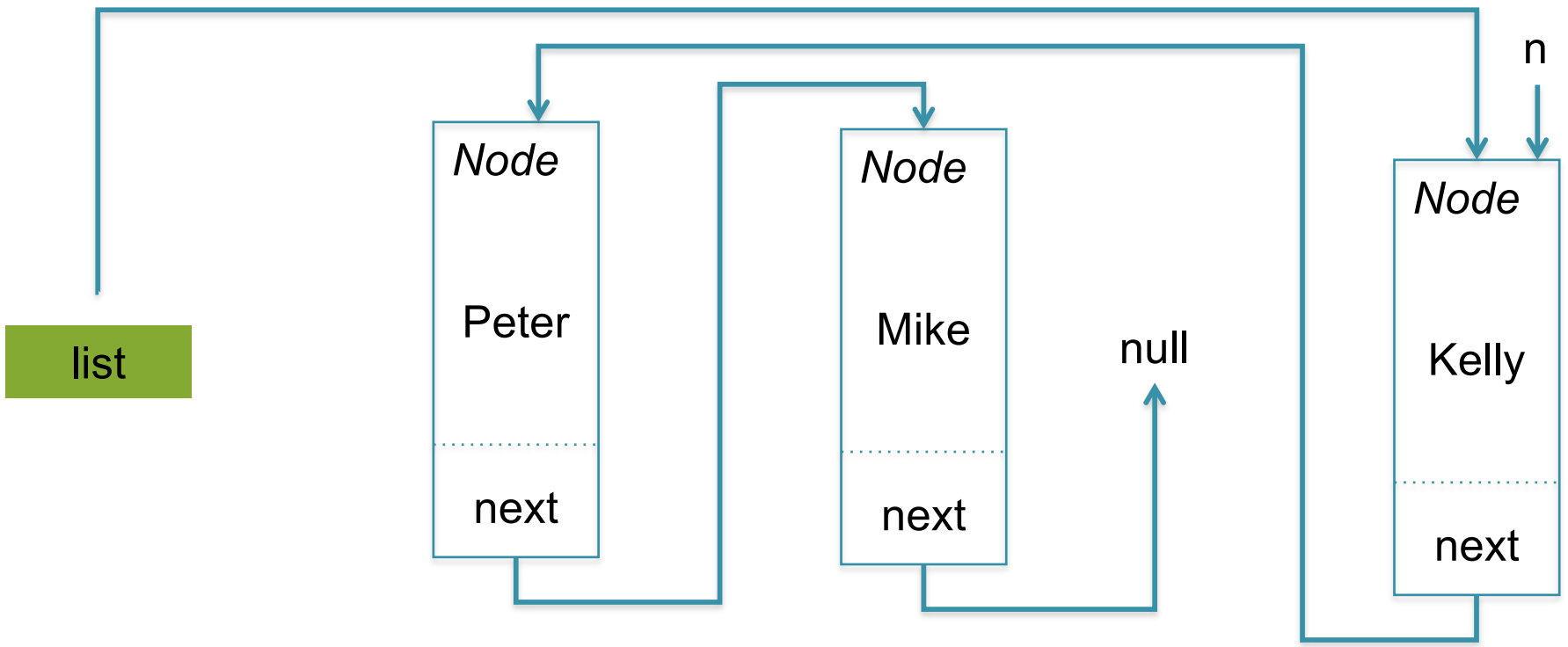



```
mylist.add("Kelly");
```

List.java

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

Linked List Construction

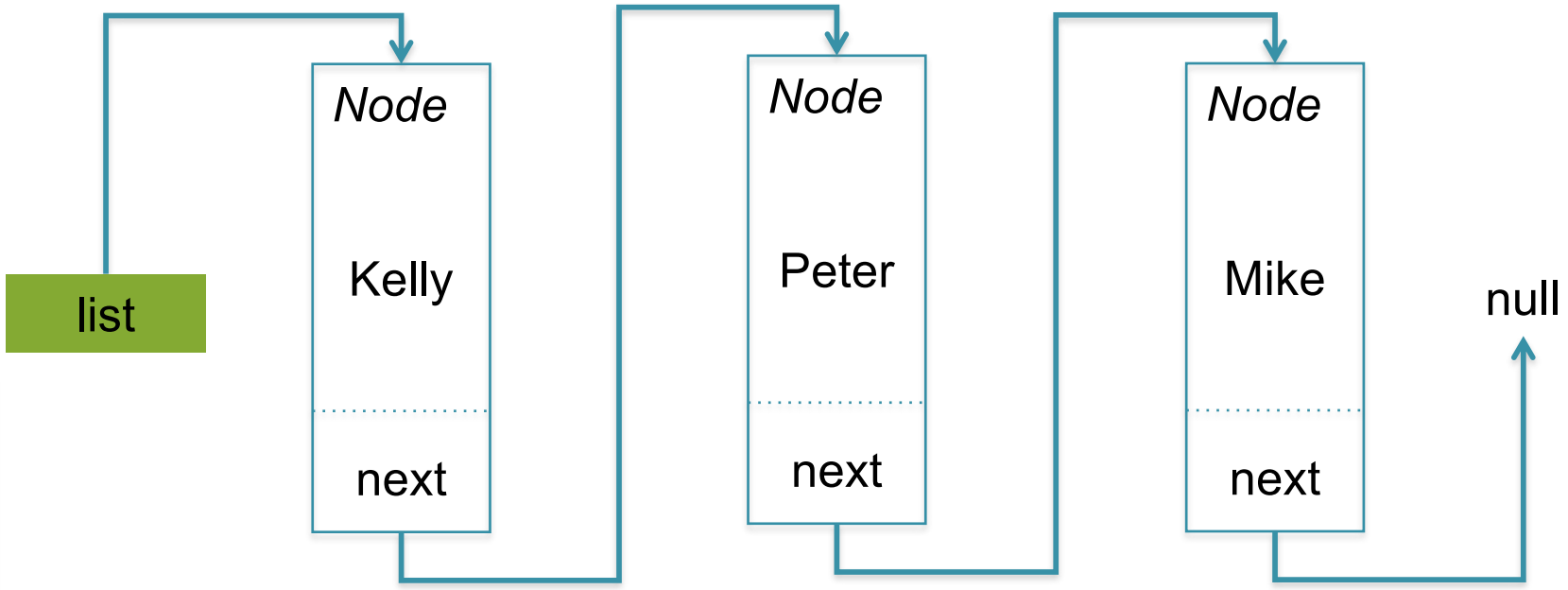



```
mylist.add("Kelly");
```

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

list

null

Wh *end)?*

Next Steps

1. Work on HW1

2. Check on Piazza for tips & corrections!