CSSE 220

Linked List Implementation
and
Project Preparation

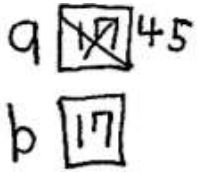
Checkout *LinkedListSimple* project from git

Quiz

- Get into pairs
- Look at/run the code in LinkedList.java main
- Draw a box-and-pointer diagram of what's happening in the main code.
- To figure it out, you'll have to look at the LinkedList constructor and addAtBeginning.
- If you've forgotten how to do box-and-pointer diagrams, checkout the handout on Day 5 of the schedule

Q1

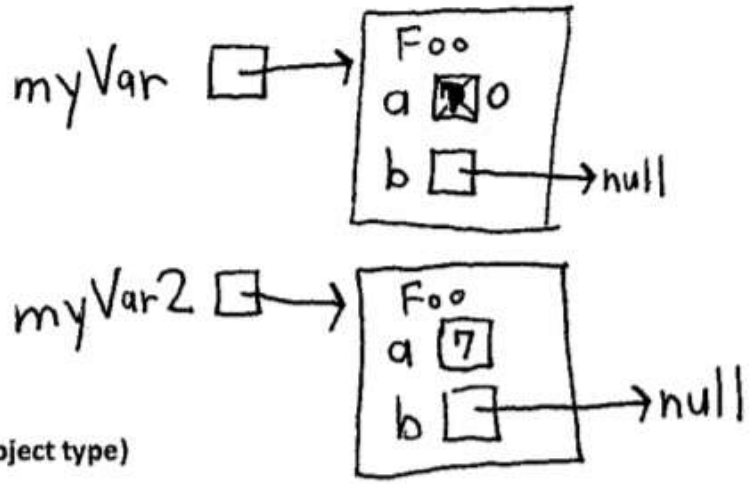
```
int a = 17;
int b = a;
a = 45;
```



A new for a class

A new for a class creates a new instance of a class. You should make a new rectangle for that class, label it with that class's name, and fill in all the fields for the class (according to the constructor of the class). Note that fields follow all the same rules as normal variables. Make the variable being assigned point to that array. *Note that without a "new", no new instances of a class (rectangles) can be created.*

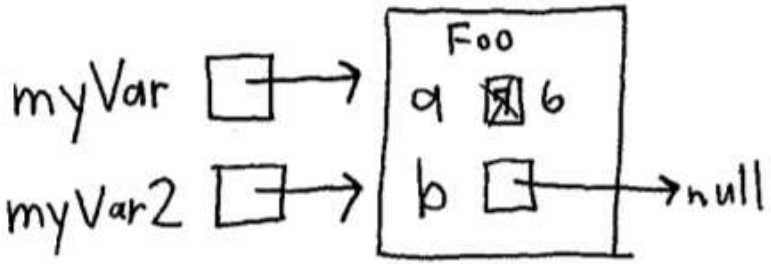
```
class Foo {
   public int a;
   public OtherClass b;
   public Point() {
     a = 7;
     b = null;
   }
}
Foo myVar = new Foo();
Foo myVar2 = new Foo();
myVar.a = 0;
```



An assignment of a non-primitive type (object type)

If you see an assignment of a non-primitive type, that copies the reference (i.e. that makes the variables point to the same object). So the arrow of the assigned object points to whatever the original object pointed to.

```
Foo myVar = new Foo();
Foo myVar2 = myVar;
myVar.a = 6;
```



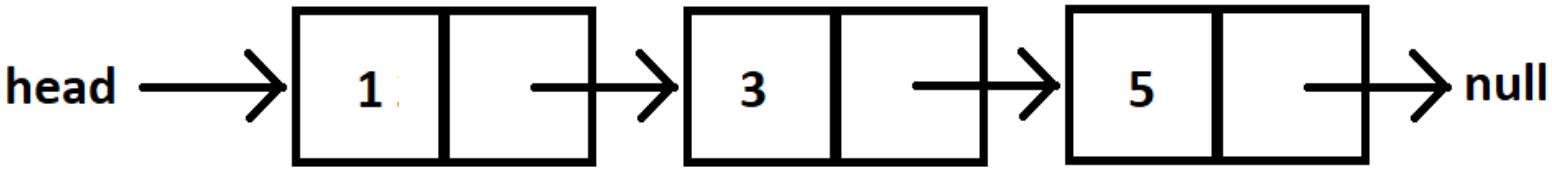
Solve the Other Problems in LinkedListSimple

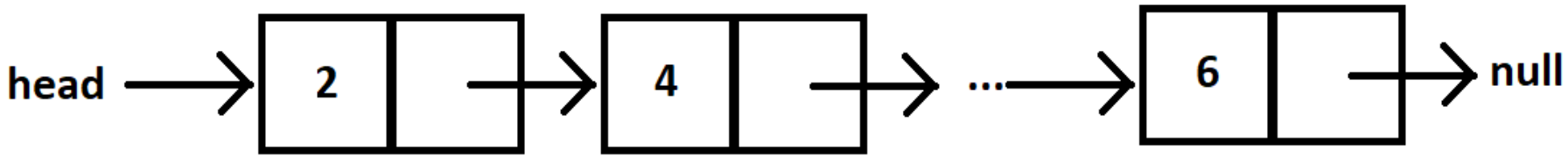
- Look at toString to get an idea of how to do size, then go from there
- They are in approximate difficulty order
- Get help if you get stuck!

- Hold on to your quiz today, we will finish it next class period.

Shorthand Notation

- Using pictures will be extremely helpful
- Can use System.out.println(this) to see what the current list looks like (does it match diagram?)





Loops in Arrays vs. LinkedLists

```
int[]  nums = ……


for (int i=0; i< nums.length; ++) {
        //do stuff with
        //arbitrary element nums[i]
}

Equivalent in while loop

int i=0;
while ( ?  ) {
        //do stuff with
        //arbitrary element nums[i]
        ?
}
```

```
LinkedList list = ……


//Another Day!




Node current = this.head;
while ( ?  ) {
        //do stuff with
        //arbitrary element
        ?
}
```

Loops in Arrays vs. LinkedLists

```
int[]  nums = ……


for (int i=0; i< nums.length; ++) {
        //do stuff with
        //arbitrary element nums[i]
}

Equivalent in while loop

int i=0;
while ( i < nums.length  ) {
        //do stuff with
        //arbitrary element nums[i]
        i++;
}
```

```
LinkedList list = ……


//Another Day!




Node current = this.head;
while ( ?  ) {
        //do stuff with
        //arbitrary element
        ?
}
```

Loops in Arrays vs. LinkedLists

```
int[]  nums = ……


for (int i=0; i< nums.length; ++) {
        //do stuff with
        //arbitrary element nums[i]
}

Equivalent in while loop

int i=0;
while ( i < nums.length  ) {
        //do stuff with
        //arbitrary element nums[i]
        i++;
}
```

```
LinkedList list = ……


//Another Day!


Node current = this.head;
while ( ?  ) {
        //do stuff with
        //arbitrary element
        current = current.next;
}
```

Loops in Arrays vs. LinkedLists

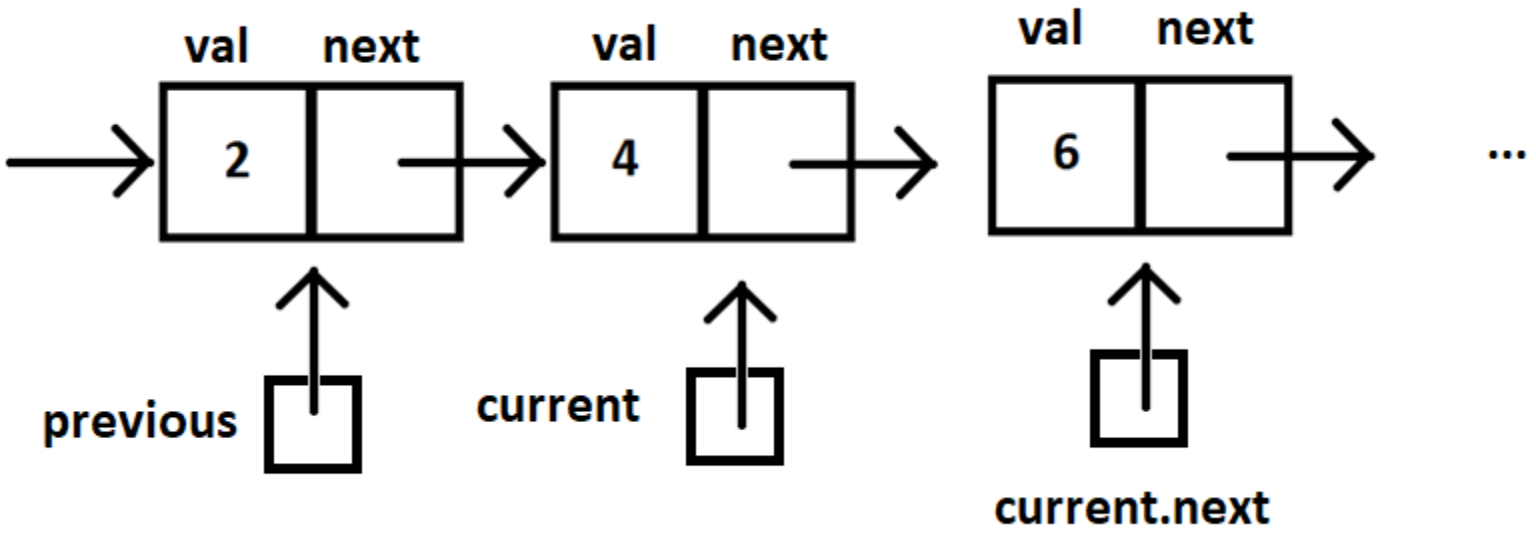
```
int[]  nums = ……


for (int i=0; i< nums.length; ++) {
        //do stuff with
        //arbitrary element nums[i]
}

Equivalent in while loop

int i=0;
while ( i < nums.length  ) {
        //do stuff with
        //arbitrary element nums[i]
        i++;
}
```

```
LinkedList list = ……


//Another Day!




Node current = this.head;
while ( current != null  ) {
        //do stuff with
        //arbitrary element
        current = current.next;
}
```

Solve the Other Problems in LinkedListSimple

- Look at toString to get an idea of how to do size, then go from there
- They are in approximate difficulty order
- Get help if you get stuck!
  - size()
  - add…
  - remove…

Shorthand Notation



Homework

- SinglyLinkedList
  - Requires you to implement a SinglyLinkedList
  - Additional algorithm questions which make use of the SinglyLinkedList
  - Will give time in next class to work on it

TEAM PROJECT WORK TIME

- Move into your groups if not already
- Review comments from Milestone 0 feedback
- Be prepared to ask question of the grader
- You will have ~5 minutes, so use it well