

## 1 Boxes and Pointers II

---

Draw a box and pointer diagram for each code block.

- (a) 

```
int[] x = {1, 2, 3};
int[] y = x;
y[2] = 7;
```

*x and y should both point to an array with values [1, 2, 7].*

- (b) 

```
IntList l = IntList.list(1, 2, 3);
IntList l2 = l;
l.tail.tail.head = 7;
```

*l and l2 should both point to an IntList with values 1, 2, and 7.*

- (c) 

```
IntList[] ll = new IntList[3];
ll[0] = IntList.list(1, 2);
ll[1] = IntList.list(2);
```

*ll should point to an array, where the first two elements point to IntLists and the third is null.*

## 2 Objects Refresher: Does this make sense?

---

- (a) Determine what would be printed after executing the main method of class `Avatar`.

```
1 public class Avatar {
2     public static String electricity;
3     public String fluid;
4
5     public Avatar(String str1, String str2) {
6         Avatar.electricity = str1;
7         this.fluid = str2;
8     }
9
10    public static void main(String[] args) {
11        Avatar foo1 = new Avatar("one ", "two");
12        Avatar foo2 = new Avatar("three ", "four");
13        System.out.println(foo1.electricity + foo1.fluid);
14        foo1.electricity = "I declare ";
15        foo1.fluid = "a thumb war";
16        System.out.println(foo2.electricity + foo2.fluid);
17    }
18 }
```

*The main method will print  
three two  
I declare four*

- (b) Consider swapping `Avatar` and `this` in lines 6 and 7. Which swaps, if any would cause errors if we tried to compile and run the code?

Both `Avatar` and `this` would work on line 6, but only `this` will work for line 7. Changing `this` to `Avatar` on line 7 will cause a compile-time error because we cannot reference instance variables using a static class reference.

- (c) Will adding the following method to class `Avatar` cause any errors during compilation or execution?

```
public static String getFluid() {
    return fluid;
}
```

The method will cause a compile-time error because we can not reference an instance variable (in this case, `fluid`) from inside a static context.

When the object is not specified (the thing before the period) in a field access or method call, Java will use `this`. by default. However, since the new method is static, `this` does not exist and therefore an error is thrown.

### 3 Min/Max

---

Given an array `A`, return a 2 element array `B` where `B[0]` is the minimum element of `A` and `B[1]` is the maximum element of `A`.

```
import static java.lang.Math.max; // max(a, b) returns max of a, b
import static java.lang.Math.min; // min(a, b) returns min of a, b

public static int[] minMax(int[] A) {
    int maxVal = Integer.MIN_VALUE; // smallest int in Java
    int minVal = Integer.MAX_VALUE; // largest int in Java

    int[] B = new int[2];

    for (int i = 0; i < A.length; i+= 1) {
        maxVal = max(maxVal, A[i]);
        minVal = min(minVal, A[i]);
    }
    B[0] = minVal;
    B[1] = maxVal;
    return B;
}
```

## 4 Reverse

---

Given an array A, reverse its elements in place (i.e. do not create any new arrays; this should be a destructive method).

```
public static void reverse(int[] A) {  
  
    for (int i = 0; i < A.length / 2; i++) {  
        int temp = A[A.length - i - 1];  
        A[A.length - i - 1] = A[i];  
        A[i] = temp;  
    }  
  
}
```