# What Am I? – Part 1

Here's a class:

```
public class WhatAmI {
    public String name;
    public WhatAmI another;
}
```

Right now the class doesn't involve very much: just two fields. Your job for today is to figure out what this class is doing and how we can use it.

1. Draw a picture of what memory looks like (i.e., what objects exist and what values they have) after the following code has been executed (from somewhere outside of the `WhatAmI` class):

```
WhatAmI one = new WhatAmI();
one.name = "Alice";

one.another = new WhatAmI();
one.another.name = "Bob";
one.another.another = null;
```

Show me your picture and ask for the next page.

# What Am I? – Part 2

2. Suppose we add the following method to the class `WhatAmI` (if you haven't seen the word `this` before, this is how we refer to the object in which we are currently sitting):

```
public WhatAmI getSomething() {
    WhatAmI toReturn = this;
    while(toReturn.another != null) {
        toReturn = toReturn.another;
    }
    return toReturn;
}
```

Now suppose you call `one.getSomething()` from outside the `WhatAmI` class. What is returned?

3. Now suppose we add another new method to `WhatAmI`:

```
public void doesSomething(String n) {
    WhatAmI found = getSomething();
    found.another = new WhatAmI();
    found.another.name = n;
}
```

What does the call `one.doesSomething("Carol")` do to our picture of memory? What happens if we then call `one.another.doesSomething("Dennis")`? What does this do in general?

4. Here's another method that we could add to `WhatAmI`:

```
public WhatAmI getSomethingElse() {
    if(this.another == null) return this;
    else return this.another.getSomethingElse();
}
```

What does this do?

5. What does this method do?

```
public int getAnInt() {
    if(this.another == null) return 1;
    else return 1 + this.another.getAnInt();
}
```

6. And what about this one?

```
public int getAnotherInt() {
    int count = 1;
    WhatAmI moving = another;
    while(moving != null) {
        count = count + 1;
        moving = moving.another;
    }
    return count;
}
```

Ask me for the next page.

# What Am I? – Part 3

7. Write a method to return the `n`th element of a "linked list." The first element is element 1. If the element does not exist (i.e., if there are fewer than `n` elements in the list) return `null`. Do not use recursion.

8. Now do the same thing, this time using recursion.

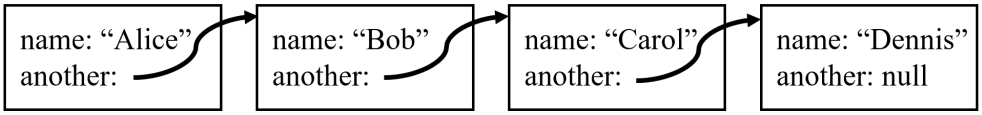
Show me both answers, and ask for the next page.

# What Am I? – Part 4

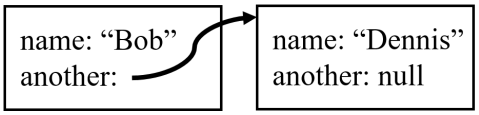
9. Write a method in `WhatAmI` with the following header:

```
public WhatAmI splitMe()
```

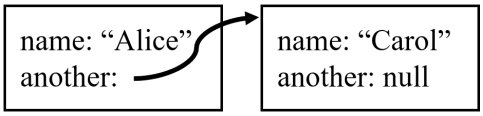
This method should split the linked list into two lists, one containing the odd-numbered positions and one the even. The original starting point should contain the odd-numbered list, and the even-numbered list should be returned. For example, if the list starts like this:




Then the returned list should be this:




And the original head (first element of the list) should now point to this:




For simplicity, you can assume that the list contains an even number of elements. Draw pictures to help yourself think about this!

# Think about for Wednesday

The `getSomethingElse()` method finds and returns the last element in the list. Suppose that we wanted to find and *remove* and return the last element in the list.

- How would you modify `getSomethingElse()` so that you remove the last element from the list before returning it? (Hint: you might need to add a parameter to the method.)
- What if we didn't want to add extra parameters to our remove method? What field could you add to `WhatAmI` that would accomplish the same thing?
- What if there is only one element left in the list, and you want to delete this final element?
- How long does your removal process take in terms of `n`, the number of elements in the list?
- What if you wanted to make removing the last element more efficient?