# CS 61B Spring 2021

# Scope, Static, and Linked Lists

Discussion 3: February 01, 2021

## 1 Static Electricity

```
1  public class Pokemon {
2      public String name;
3      public int level;
4      public static String trainer = "Ash";
5      public static int partySize = 0;
6
7      public Pokemon(String name, int level) {
8          this.name = name;
9          this.level = level;
10         this.partySize += 1;
11     }
12
13     public static void main(String[] args) {
14         Pokemon p = new Pokemon("Pikachu", 17);
15         Pokemon j = new Pokemon("Jolteon", 99);
16         System.out.println("Party size: " + Pokemon.partySize);
17         p.printStats()
18         int level = 18;
19         Pokemon.change(p, level);
20         p.printStats()
21         Pokemon.trainer = "Ash";
22         j.trainer = "Brock";
23         p.printStats();
24     }
25
26     public static void change(Pokemon poke, int level) {
27         poke.level = level;
28         level = 50;
29         poke = new Pokemon("Voltorb", 1);
30         poke.trainer = "Team Rocket";
31     }
32
33     public void printStats() {
34         System.out.print(name + " " + level + " " + trainer);
35     }
36
37 }
```

(a) Write what would be printed after the main method is executed.

```
Party Size: 2
Pikachu 17 Ash
Pikachu 18 Team Rocket
Pikachu 18 Brock
```

(b) On line 28, we set `level` equal to `50`. What `level` do we mean? An instance variable of the `Pokemon` object? The local variable containing the parameter to the `change` method? The local variable in the `main` method? Something else?

It is the local variable in the `change` method and does not have any effect on the other variables of the same name in the `Pokemon` class or the `main` method.

(c) If we were to call `Pokemon.printStats()` at the end of our main method, what would happen?

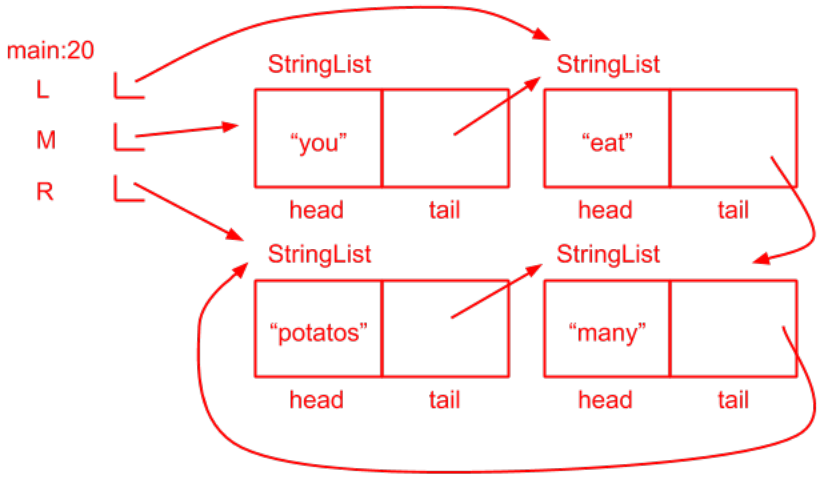
If we were to add this line to our main method, it would error. In the class, `printStats()` is an instance method. What would it mean to print the name and level of the class `Pokemon`, as opposed to a specific Pokemon's name? It doesn't really make sense. So when we try to run this method on our class, it errors.

One more thing to note is the method `change` is declared static itself. Static methods can be called using the name of the class, as in line 19, whereas non-static methods cannot. The golden rule for static methods to know is that **static methods can only modify static variables**.

# 2 To Do List

Draw the box-and-pointer diagram that results from running the following code. A `StringList` is similar to an `IntList`. It has two instance variables, `first` and `rest`.

```
1  StringList L = new StringList("eat", null);
2  L = new StringList("should", L);
3  L = new StringList("you", L);
4  L = new StringList("sometimes", L);
5  StringList M = L.rest;
6  StringList R = new StringList("many", null);
7  R = new StringList("potatoes", R);
8  R.rest.rest = R;
9  M.rest.rest.rest = R.rest;
10 L.rest.rest = L.rest.rest.rest;
11 L = M.rest;
```




For a step by step walkthrough of this box and pointer diagram, see https://tinyurl.com/y38jkzpj

# 3 Helping Hand *Extra*

(a) Fill in blanks in the methods `findFirst` and `findFirstHelper` below such that they return the first `IntNode` with item `n`, or null if there is no such node containing that item.

```
1  public class SLList {
2     Node sentinel;
3
4     public SLList() {
5        this.sentinel = new Node();
6     }
7
8     private static class Node {
9        int item;
10       Node next;
11    }
12
13    ...
14
15    public int findFirst(int n) {
16       return findFirstHelper(n, 0, sentinel.next);
17    }
18
19    private int findFirstHelper(int n, int index, Node curr) {
20       if (curr == null) {
21          return -1;
22       }
23       if (curr.item == n) {
24          return index;
25       } else {
26          return findFirstHelper(n, index + 1, curr.next);
27       }
28    }
29
30 }
```

(b) Why do we use a helper method here? Why can't we just have the signature for `findFirst` take in the start index and `curr`, such that the user of the function passes in 0 and `sentinel.next` each time?

It's not intuitive for the user to have to pass in 0 and `sentinel.next` every single time they're calling `findFirst()`, as it is unrelated to what they're actually requesting. Additionally, it is breaking the abstraction barrier, as it requires our user to understand how this method works under the hood. Finally, if the user didn't understand what to pass in (because again, it's not quite intuitive), they could pass in some random values that will result in an incorrect answer.

Thus, it is bad programming practice to make the user pass in those extra arguments every time. However, we do need a way of keeping track of which node and index we're on as we recurse, so we must make a helper method that can keep track of all that information.