CS 61B Fall 2022

Scope, Static, and Linked Lists

Discussion 3: September 05, 2022

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 = "Cynthia";
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("Luxray", 1);
30         poke.trainer = "Team Rocket";
31     }
32
33     public void printStats() {
34         System.out.println(name + " " + level + " " + trainer);
35     }
36 }
```

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

(b) On line 28, we set `level` equal to `50`. What `level` do we mean?

A. An instance variable of the `Pokemon` object

B. The local variable containing the parameter to the `change` method

C. The local variable in the `main` method

D. Something else (explain)

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

2 Cardinal Directions

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("bananas", L);
3  L = new StringList("never", L);
4  L = new StringList("sometimes", L);
5  StringList M = L.rest;
6  StringList R = new StringList("shredded", null);
7  R = new StringList("wheat", 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;
```

3 Helping Hand

(a) Fill in blanks in the methods below such that they return the index of the first Node with Item `item`, or -1 if there is no such node. Assume that each `Node`'s `item` is not null.

```
1  public class SLList {
2     Node sentinel;
3
4     public SLList() {
5        this.sentinel = new Node();
6     }
7
8     private static class Node {
9        Item item;
10       Node next;
11    }
12
13    private static class Item {
14       // Implementation not shown
15       @Override
16       public boolean equals(Object o) {...}
17
18       // For formality's sake: we'll talk about this later in 61B! Implementation not shown
19       @Override
20       public int hashCode() {...}
21    }
22
23    public int findFirst(Item item) {
24       return ________________________________________;
25    }
26
27    private int findFirstHelper(Item item, int index, Node curr) {
28       if (_____________________) {
29          return -1;
30       }
31       if (_____________________) {
32          return index;
33       } else {
34          return ________________________________________;
35       }
36    }
37 }
```

(b) Why do we use a helper method here? Why can't we just have the signature for `findFirst` also have a pointer to the `curr` node, such that the user of the function passes in the sentinel each time?