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 }
```

p → name p
level 18

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

PartySize: 2
Pikachu 17 Ash
Pikachu 18 TeamRocket
Pikachu 18 Brock

没有歧义 因为静态方法
不能修改非静态

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

变量

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

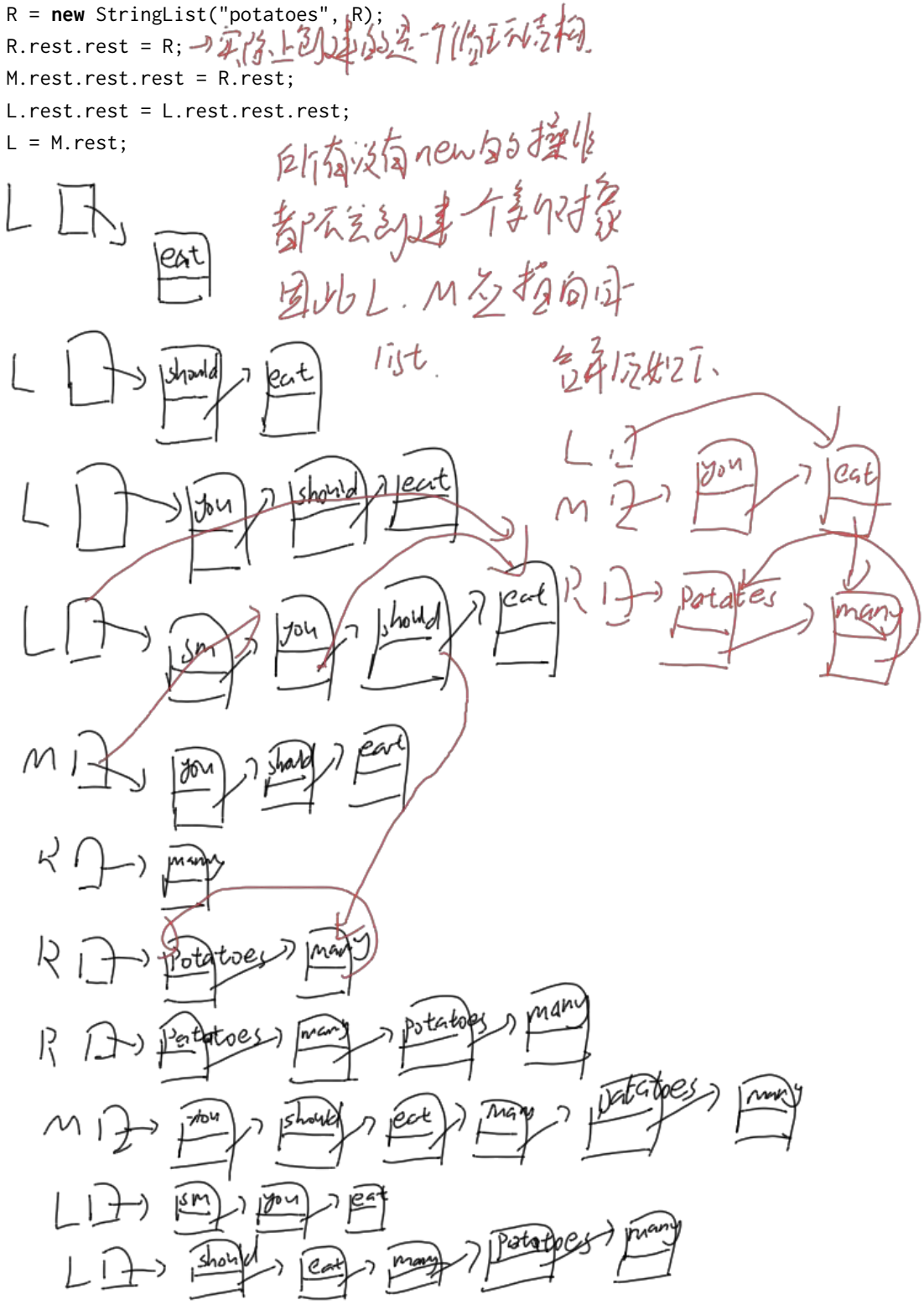
Error.

此外 注意 change 的标记(2). change是 static,
无法修改非static成员. 但此处修改的是实例对象的
level. 可以做到. static方法无法访问非static成员是由于通过类名调用的方法, 其非static成员因为没有实例而无确定值. 此处修改的level是参数. 所以也可以.

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;
```



3 Helping Hand *Extra*

(a) Fill in blanks in the methods `findFirst` and `findFirstHelper` below such that they return the index of the first Node with item `n`, or -1 if there is no such node containing that item.

```
public class SLList {
   Node sentinel;

   public SLList() {
      this.sentinel = new Node();
   }

   private static class Node {
      int item;
      Node next;
   }

   public int findFirst(int n) {
      return findFirstHelper(n, 0, sentinel.next);
   }

   private int findFirstHelper(int n, int index, Node curr) {
      if (curr.next==null && curr.item != n) {    [handwritten correction: -> curr == null]
         return -1;
      }
      if (curr.item == n) {
         return index;
      } else {
         return (n, index+1, curr.next);
      }
   }

}
```

(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?

用户不会在调用函数时使用节点参数. 如果这么做,需要用户了解函数原理,打破函数的抽象。让用户传递索引参数是一个不好的习惯。需要一个辅助的方法来隐藏创建的索引.