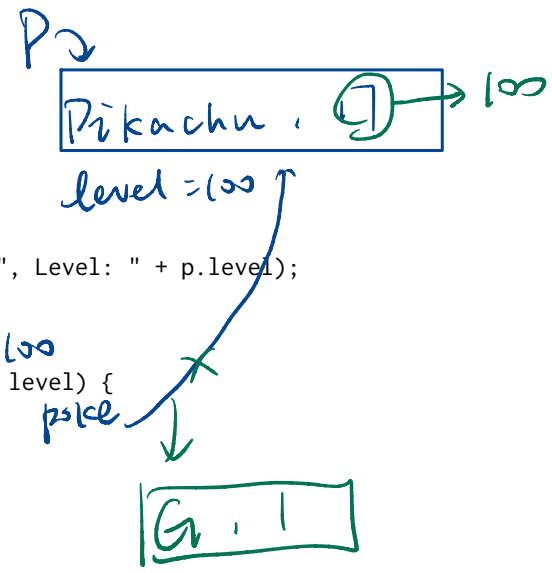
CS 61B Fall 2020

Scope, Pass-by-Value, Static

Discussion 2: August 31, 2020

1 Pass-by-What?

```
1  public class Pokemon {
2      public String name;
3      public int level;
4
5      public Pokemon(String name, int level) {
6          this.name = name;
7          this.level = level;
8      }
9
10     public static void main(String[] args) {
11         Pokemon p = new Pokemon("Pikachu", 17);
12         int level = 100;
13         change(p, level);
14         System.out.println("Name: " + p.name + ", Level: " + p.level);
15     }
16
17     public static void change(Pokemon poke, int level) {
18         poke.level = level;
19         level = 50;
20         poke = new Pokemon("Gengar", 1);
21     }
22 }
```



(a) Draw the box-and-pointer diagram after Java evaluates the `main` method. What would Java print?

change 内的本地变量

(b) On line 19, 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?

2 Static Methods and Variables

```
1  public class Cat {
2     public String name;
3     public static String noise;
4     public Cat(String name, String noise) {
5         this.name = name;
6         this.noise = noise;
7     }
8
9     public void play() {
10       System.out.println(noise + " I'm " + name + " the cat!");
11    }
12
13    public void nickname(String newName) {
14       name = newName;
15    }
16
17    public static void anger() {
18       noise = noise.toUpperCase();
19    }
20
21    public static void calm() {
22       noise = noise.toLowerCase();
23    }
24
25
26 }
```

(a) Write what will happen after each call of `play()` in the following method.

```
1  public static void main(String[] args) {
2      Cat a = new Cat("Cream", "Meow!");
3      Cat b = new Cat("Tubbs", "Nyan!");
4      a.play();
5      b.play();
6      Cat.anger();
7      a.calm();
8      a.play();
9      b.play();
10     a.nickname("Kitty");
11     a.play();
12     b.play()
13 }
```

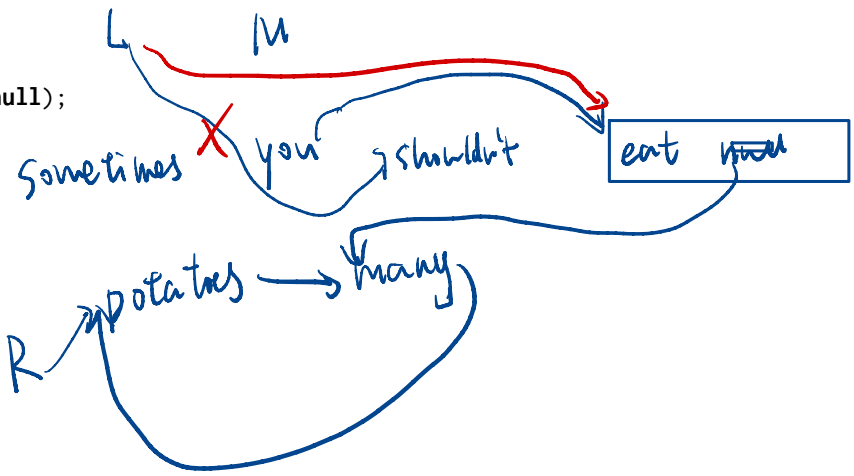
(b) If we were to add `Cat.nickname("KitKat")` to the end of our main function, what would happen?

② Static functions only modify static variables

3 Practice with Linked Lists

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("shouldn't", 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;
```



4 Squaring a List *Extra*

Implement `square` and `squareDestructive` which are static methods that both take in an `IntList L` and return an `IntList` with its integer values all squared. `square` does this non-destructively with recursion by creating new `IntLists` while `squareDestructive` uses an iterative approach to change the instance variables of the input `IntList L`.

```
public static IntList square(IntList L) {
    if (L==null) return null;
    return new (L.first * L.first, square(L.rest));
}
//iterative - des..
public static IntList squareDestructive(IntList L) {
    IntList res = L;
    while (L!=null){
      L.first = Lfirst * L.first;
      L = L.rest;
    }
    return res;
}
```

Extra: Now, implement `square` iteratively, and `squareDestructive` recursively.

四个写法

recursive { destructive
non-des..

iterative { des
non-des