1 Pass by what?

Consider the following code block:

```
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) What will be printed out by the code above?

Name: Pikachu, Level: 100

b) Draw a box and pointer diagram to illustrate what happened.

Visualizer Link

c) 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? Or 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 (such as in the Pokemon class or in the main method)

2 Static Methods and Variables

```
public class Cat {
   public String name;
   public static String noise;

   public Cat(String name, String noise) {
       this.name = name;
       this.noise = noise;
   }

   public void play() {
      System.out.println(noise + " I'm " + name + " the cat!");
   }

   public static void anger() {
      noise = noise.toUpperCase();
   }
   public static void calm() {
      noise = noise.toLowerCase();
   }
}
```

Write what willehappen after each call of `play()` in the following method.

```
   public static void main(String[] args) {
       Cat a = new Cat("Cream", "Meow!");
       Cat b = new Cat("Tubbs", "Nyan!");
       a.play();
       b.play();
       Cat.anger();
       a.calm();
       a.play();
       b.play();
   }
```

Nyan! I'm Cream the cat!
Nyan! I'm Tubbs the cat!
nyan! I'm Cream the cat!
nyan! I'm Tubbs the cat!

3 Practice with Linked Lists

Draw a box and pointer diagram to represent the `StringList`s after each statement. A `StringList` is similar to an `IntList`. It has two instance variables, `String head` and `StringList tail`.

```
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.tail;
6         StringList R = new StringList("many", null);
7         R = new StringList("potatoes", R);
8         R.tail.tail = R;
9         M.tail.tail.tail = R.tail;
10        L.tail.tail = L.tail.tail.tail;
11        L = M.tail;
```

http://cscircles.cemc.uwaterloo.ca/java_visualize/#code=public+class+StringList+%7B%0A+++String+head%3B%0A+++StringList+tail%3B%0A+++public+StringList(String+head,+StringList+tail)+%7B%0A++++++this.head+%3D+head%3B%0A++++++this.tail%3Dtail%3B%0A+++%7D%0A+++public+static+void+main(String%5B%5D+args)+%7B%0A+++StringList+L+%3D+new+StringList(%22eat%22,+null)%3B%0A%09L+%3D+new+StringList(%22shouldn't%22,+L)%3B%0A%09L+%3D+new+StringList(%22you%22,+L)%3B%0A%09L+%3D+new+StringList(%22sometimes%22,+L)%3B%0A%09StringList+M+%3D+L.tail%3B%0A%09StringList+R+%3D+new+StringList(%22many%22,+null)%3B%0A%09R+%3D+new+StringList(%22potatoes%22,+R)%3B%0A%09R.tail.tail+%3D+R%3B%0A%09M.tail.tail.tail+%3D+R.tail%3B%0A%09L.tail.tail+%3D+L.tail.tail.tail%3B%0A%09L+%3D+M.tail%3B%0A++++%7D%0A%7D%0A&mode=display&showStringsAsObjects=&curInstr=52

4 Bonus Problems: Squaring a List

Write the following methods to destructively and non-destructively square a linked list. Destructively squaring means modifying the items in the `IntList` passed as an argument. Non-destructively means all values in the original list are unaffected by the function.

```
/** Destructively squares each element of the given IntList L.
 * Don't use 'new'; modify the original IntList.
 * Should be written iteratively. */
    public static IntList SquareDestructive(IntList L) {
        IntList B = L;
        while (B != null) {
            B.head *= B.head;
            B = B.tail
        }
        return L;
    }

/** Non-destructively squares each element of the given IntList L.
 * Don't modify the given IntList.
 * Should be written recursively*/
    public static IntList SquareNonDestructive(IntList L) {
            if (L == null) {
                return L;
        } else {
                IntList tail = SquareNonDestructive(L.tail);
                IntList M = new IntList(L.head * L.head, tail);
                return M;
        }
}
```

Even more bonus problems: Write `SquareDestructive` recursively. Write `SquareNonDestructive` iteratively.

```
    public static IntList SquareDestructive(IntList L) {
        if (L == null) {
            return L;
        } else {
            L.head = L.head*L.head;
            SquareDestructive(L.tail);
        }
        return L;
    }
    public static IntList SquareNonDestructive(IntList L) {
        if (L == null) {
            return L;
        }
        IntList B = L.tail;
        IntList LSquared = new IntList(L.head*L.head, null);
        IntList C = LSquared;
        while (B != null) {
            C.tail = new IntList(B.head*B.head, null);
            B = B.tail;
            C = C.tail;
        }
```

```
            return LSquared;
        }
    }
```