

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 }
```

- 1.1 (a) What would Java display?

Name: Pikachu, Level: 100

- (b) Draw the box-and-pointer diagram after Java evaluates the main method.

```
https://cscircles.cemc.uwaterloo.ca/java_visualize/#code=public+
class+Pokemon+%7B%0A++++public+String+name%3B%0A++++public+int+
level%3B%0A%0A++++public+Pokemon(String+name,+int+level)+%7B%0A+++
++++this.name+%3D+name%3B%0A++++++++this.level+%3D+level%3B%0A+++
+%7D%0A%0A++++public+static+void+main(String%5B%5D+args)+%7B%0A+++
++++Pokemon+p+%3D+new+Pokemon(%22Pikachu%22,+17)%3B%0A++++++++int+
level+%3D+100%3B%0A++++++++change(p,+level)%3B%0A++++++++System.
out.println(%22Name%3A+%22+%2B+p.name+%2B+%22,+Level%3A+%22+%2B+p.
level)%3B%0A+++++%7D%0A++++%0A++++public+static+void+change(Pokemon+
poke,+int+level)+%7B%0A++++++++poke.level+%3D+level%3B%0A++++++++
+level+%3D+50%3B%0A++++++++poke+%3D+new+Pokemon(%22Gengar%22,+1)
%3B%0A+++++%7D%0A%7D&mode=display&curInstr=0
```

- (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? 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.

2 Static Methods and Variables

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

2.1 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 }
```

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

3.1 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;
```

[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](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 Squaring a List *Extra*

- 4.1 Implement `square` and `squareMutative` 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-mutatively with recursion by creating new `IntLists` while `squareMutative` uses a recursive approach to change the instance variables of the input `IntList L`.

```
1 public static IntList square(IntList L) {  
  
1     if (L == null) {  
2         return L;  
3     } else {  
4         IntList rest = square(L.rest);  
5         IntList M = new IntList(L.first * L.first, rest);  
6         return M;  
7     }  
8 }  
  
1 public static IntList squareMutative(IntList L) {  
  
1     IntList B = L;  
2     while (B != null) {  
3         B.first *= B.first;  
4         B = B.rest  
5     }  
6     return L;  
7 }
```

4.2 *Extra:* Now, implement `square` iteratively, and `squareMutative` recursively.

```
1 public static IntList square(IntList L) {
2     if (L == null) {
3         return L;
4     }
5     IntList B = L.rest;
6     IntList LSquared = new IntList(L.first * L.first, null);
7     IntList C = LSquared;
8     while (B != null) {
9         C.rest = new IntList(B.first * B.first, null);
10        B = B.rest;
11        C = C.rest;
12    }
13    return LSquared;
14 }

1 public static IntList squareMutative(IntList L) {
2     if (L == null) {
3         return L;
4     } else {
5         L.first = L.first * L.first;
6         squareMutative(L.rest);
7     }
8     return L;
9 }
```