Reference Semantics
See Piazza.
public static void listMangler(ArrayList<String> list) {
    for (int i = 0; i < list.size(); i++) {
        list.set(i, "");
    }
}

public static void main(String[] args) {
    ArrayList<String> animals = new ArrayList<String>();
    animals.add("Elephant");
    animals.add("Bunny");
    animals.add("Zebra");
    
    ArrayList<String> animals2 = animals;
    listMangler(animals);
    System.out.println(animals);
    System.out.println(animals2);
}
public static void listMangler(ArrayList<String> list) {
    for (int i = 0; i < list.size(); i++) {
        list.set(i, "");
    }
}

public static void main(String[] args) {
    ArrayList<String> animals = new ArrayList<String>();
    animals.add("Elephant");
    animals.add("Bunny");
    animals.add("Zebra");

    ArrayList<String> animals2 = animals;

    listMangler(animals);
    System.out.println(animals);
    System.out.println(animals2);
}
What Are We Doing. . . ?
We're trying to understand how Java passes arguments to our methods.

Today’s Main Goals:
- To understand how Java passes arguments to methods
- To understand what `null` is
- To understand the difference between primitive and Object types
Talk to the person next to you and try to answer the following two questions: Unambiguously describe...

- ...the day of the month you were born.
- ...every detail of the house you grew up in.

Your descriptions should be good enough that the person next to you could (given enough time) completely recreate the answer.
Talk to the person next to you and try to answer the following two questions: Unambiguously describe...

- ...the day of the month you were born.
- ...every detail of the house you grew up in.

Your descriptions should be good enough that the person next to you could (given enough time) completely recreate the answer.

Bottom Line: The first one is really easy; the second one is hard.
A Silly Question

Talk to the person next to you and try to answer the following two questions: Unambiguously describe...

- ...the day of the month you were born.
- ...every detail of the house you grew up in.

Your descriptions should be good enough that the person next to you could (given enough time) completely recreate the answer.

Bottom Line: The first one is really easy; the second one is hard.

What's a better way of describing the house?

Give your address which allows the person to look at the house!
A **house** is:
- Built on a **plot of land**
- Identified by its **address**
- Made up of **rooms** which are decorated a **color**

### Java Lane

| 100: yellow bdrm[0] | 200: yellow bdrm[0] | 300: blue bdrm[0] red bdrm[1] |

### Some Questions
- Are any of these houses on the same piece of land?
- What if 100 Java Lane and 200 Java Lane are decorated identically? They’re kind of the same now, right?
Suppose House `h100` refers to “100 Java Lane”. What does the following code do?

```java
House hx = h100;
```
Suppose House h100 refers to “100 Java Lane”. What does the following code do?

```java
House hx = h100;
```

It doesn’t actually change the plots!
The statement makes h100 refer to the same house as hx, which is “100 Java Lane”.

public class Room {
    // BAD STYLE (public field)
    String decorations;
}

public class House {
    // BAD STYLE (public field)
    Room[] bedrooms;
}
var1 = new Object() **builds a house.**

var2 = var1 **makes the variables refer to the same address.**

Draw a picture of the “street” after each line of this code. How many houses were built? How many addresses were given out?

1. Object o1 = `new` Object();
2. Object o2 = o1;
3. Object o3 = `new` Object();
4. o3 = o2;
Object o1 = new Object();

Object o2 = o1;

Object o3 = new Object();

o3 = o2;
We want to make a list of the color of every bedroom on Java Lane. Write down a procedure for doing this.

1. Go to a house
2. Go to each bedroom in the house and write down its color

In code:

```java
public void decorations(House h) {
    int num = h.bedrooms.length;
    for (int i = 0; i < h.bedrooms.length; i++) {
        System.out.println(h.bedrooms[i].decorations);
    }
}
```
public class Room {
    String decorations;
}

public class House {
    Room[] bedrooms;
}

We want to make a list of the color of every bedroom on Java Lane. Write down a procedure for doing this.

- Go to a house
- Go to each bedroom in the house and write down its color

In code:

```java
public void decorations(House h) {
    int num = h.bedrooms.length;
    for (int i = 0; i < h.bedrooms.length; i++) {
        System.out.println(h.bedrooms[i].decorations);
    }
}
```

In Java, the "." means "use the address to locate the thing"
public class Room {
    String decorations;
}

public class House {
    Room[] bedrooms;
}

Room myRoom = new Room();
myRoom.decorations = "black";

Room[] rooms = new Room[1];
room[0] = myRoom;

House h = new House();
h.bedrooms = rooms;

/* This method should repaint all the rooms of
the specified house to be green. */
public static void repaintHouse(House house) {
}
public class Room {
  String decorations;
}

public class House {
  Room[] bedrooms;
}

Room myRoom = new Room();
myRoom.decorations = "black";

Room[] rooms = new Room[1];
//room[0] = myRoom;

House h = new House();
h.bedrooms = rooms;

/* This method should repaint all the rooms of 
 * the specified house to be green. */
public static void repaintHouse(House house) { }
public class Room {
    String decorations;
}

public class House {
    Room[] bedrooms;
}

Room myRoom = new Room();
myRoom.decorations = "black";

Room[] rooms = new Room[1];
/* This is like saying the house was supposed to have a *
 * bedroom, but it was never made. */
room[0] = null;

House h = new House();
h.bedrooms = rooms;

/* This method should repaint all the rooms of *
 * the specified house to be green. */
public static void repaintHouse(House house) {
}
Consider the following code:

```java
ArrayList<Integer> list = new ArrayList<Integer>();
```

**Q:** The house is the `ArrayList`, what are the rooms?

Describe in the house analogy what each of the following lines of code do:

```java
1. ArrayList<Integer> list2 = list;
2. ArrayList<Integer> list3 = new ArrayList<Integer>();
3. list2.add(5);
4. list3.add(7);
5. append9000(list2);
6. public void append9000(ArrayList<Integer> list) {
   7.     list.add(9000);
   8. }
```
public void youGuess(int theAnswer) {
    theAnswer = 1000;
    System.out.println("I guess " + theAnswer);
}

public static void main(String[] args) {
    int myNumber = 42;
    youGuess(myNumber);
    System.out.println("The right answer is " + myNumber);
}
public class TheAnswer {
    int answer;
}

public void youGuess(TheAnswer ans) {
    ans.answer = 1000;
    System.out.println("I guess " + ans.answer);
}

public static void main(String[] args) {
    TheAnswer ans = new TheAnswer();
    ans.answer = 42;
    youGuess(ans);
    System.out.println("The right answer is " + ans.answer);
}
Consider the following. It’s your birthday, and...

- I ask you “how old were you before today?”
- You answer `prevYearsOld`
- In my head, I increment that value (`prevYearsOld += 1`).
- I shout out “You’re `prevYearsOld` now!”

If I were to ask you again, would you answer differently?

```java
public shoutOnBirthday(int prevYearsOld) {
    prevYearsOld += 1;
    System.out.println("You’re " + prevYearsOld + " years old now!");
}

public static void main(String[] args) {
    int prevYearsOld = 350; /* Loch Ness Monster */
    shoutOnBirthday(prevYearsOld);
}
```
public void doit(int[] array) {
    if (array == null) {
        array = new int[1];
        array[0] = 9999;
    } else {
        array[0] = 101;
    }
}

public static void main(String[] args) {
    int[] array = null;
    doit(null);
    doit(null);
    doit(array);
    doit(array);
}