Objectives

- Object-oriented programming in Java
  - Encapsulation
  - Access modifiers
  - Using others' classes
  - Defining own classes

Assign 1

- Problems?
- Tips or tricks for others?
  - Read: what mistakes will you vow never to do again but probably will?

Review (from Wed)

- What is the keyword for a constant value?
- What does `static` mean?
- What Java classes did we discuss?
- What do the following control structures look like in Java?
  - If, While, For
- What is the syntax for logic operators in Java?
- How do you create an array?
- How do you determine the size of an array?
- How can you sort an array?

Assign0 Feedback

- Terminology clarification
  - Declaration: `int x = 3;`
  - Definition: `x = 3;`
- Comment for author: `@author Dr. Seuss`
  - Syntax will make more sense when we talk more about JavaDocs

What does this code do?

```
if ( x > 4 );
     System.out.println("x is " + x);
```


What does this code do?

```
if ( x > 4 );
     System.out.println("x is " + x);
```

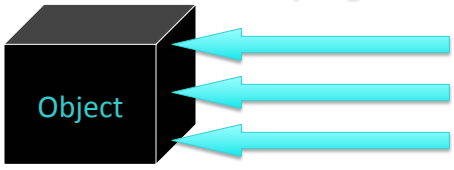
- ; is a valid statement
- Print statement *always* executes
- Indentation doesn't matter

Review: Object-Oriented Programming

- What is OO programming?
  - Components?
- Benefits?

Review: Objects

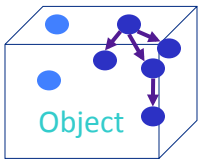
- ***How*** object does something doesn't matter
  - Example: if object *sorts*, does not matter if uses merge or quick sort
- ***What*** object does matters (its **functionality**)
  - What object *exposes* to other objects
  - Referred to as "**black-box programming**"



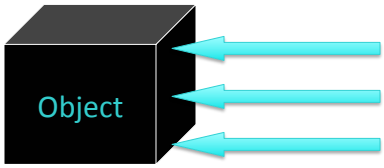
- Has public **interface** that others can use
- Hides state from others

Discussion

- What is the problem with white-box programming?



Can see and manipulate object's internals



Java's structure helps us enforce black-box programming

Classes & Objects

- **Classes** define template from which objects are made
  - "Cookie cutters"
  - Define **state** – data, usually private
  - Define **behavior** – an object's methods, usually public
    - Exceptions?
- Many objects can be created for a class
  - Object: the cookie!
  - Ex: Many Mustangs created from Ford's "blueprint"
  - Object is an instance of the class

Access Modifiers

- A `public` method (or instance field) means that any object *of any class* can directly access the method (or field)
  - Least restrictive
- A `private` method (or instance field) means that any object *of the same class* can directly access this method (or field)
  - Most restrictive
- Additional access modifiers will be discussed with inheritance

In general, what access modifiers will we use for methods? For instance fields?

Constructors

- **Constructor:** a special method that constructs and initializes an object
  - After construction, can call methods on object
- Constructors have the same name as their classes

Using Other's Classes: Random

- Problem: write a Java program that prints "heads" or "tails" at random.
- Look at API of Random
  - What functionality is available?
  - How do you use the class?

CREATING YOUR OWN CLASSES

General Java Class Structure

```
public class ClassName {

   // --------- INSTANCE VARIABLES ---------------
   // define variables that represent object's state
   private int inst_var;

   // --------- CONSTRUCTORS ---------------
   public ClassName() {
      // initialize data structures
   }

   // ----------- METHODS ------------
   public int getInfo() {
      return inst_var;
   }
}
```

Note: instance variables are `private` and methods are `public`

Chicken.java

```
public class Chicken {

   // --------- INSTANCE VARIABLES ---------------
   private String name;
   private int height; // in cm

   // --------- CONSTRUCTORS ---------------
   public Chicken(String name, int h,
                  double weight) {

      this.name = name;
      this.height = h;
      this.weight = weight;
   }
   …
```

Constructor name same as class's name

Type and name for each parameter

Params don't need to be same names as instance var names

`this`: Special name for the constructed object, like `self` in Python (differentiate from parameters)

Chicken.java

```
public class Chicken {

   // --------- INSTANCE VARIABLES ---------------
   private String name;
   private int height; // in cm
   private double weight;

   // --------- CONSTRUCTORS ---------------
   public Chicken(String name, int h,
                                 double weight) {
      this.name = name;
      this.height = h;
      this.weight = weight;
   }
   …
```


Methods: Chicken.java

```
   …
// --------- Getter Methods ---------------
   public String getName() {
      return name;
   }

// --------- Mutator Methods ---------------
   public void feed() {
      weight += .2;
      height += 1;
   }
   …
}
```

Type the method returns

Chicken object's instance variables

Note that you don't *have* to use `this` when variables are unambiguous

Constructing objects

- Given the `Chicken` **constructor**

```
Chicken( String name, int height, double
  weight )
```

create three chickens

- "Fred", height: 38, weight: 2.0
- "Sallie Mae", height: 45, weight: 3.0
- "Momma", height: 83, weight: 6.0

Using Classes You Wrote

- In `Chicken.java`,
  - Construct chickens
  - Call methods on the constructed objects

TODO

- Assignment 2:
  - Part 1: Debugging
  - Part 2: Writing a Birthday class (will build on later)
  - Due Monday before class