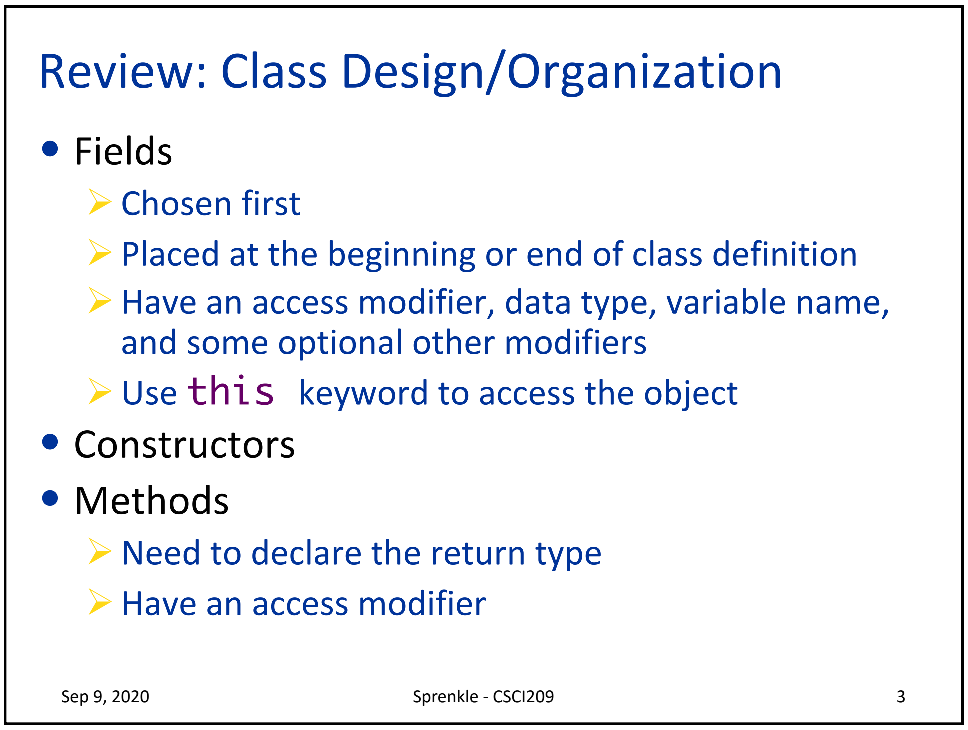
Objectives

- Cloning
- Garbage collection
- Parameter passing

Review

- What is overriding?
- What is overloading?
- How do we make an instance variable unchangeable after construction?
- How do we call a constructor within a constructor?
- What is the root of the Java class hierarchy?
- What method should we implement to allow pretty printing of objects we define?
- What method should we implement for determining if two objects are equivalent?

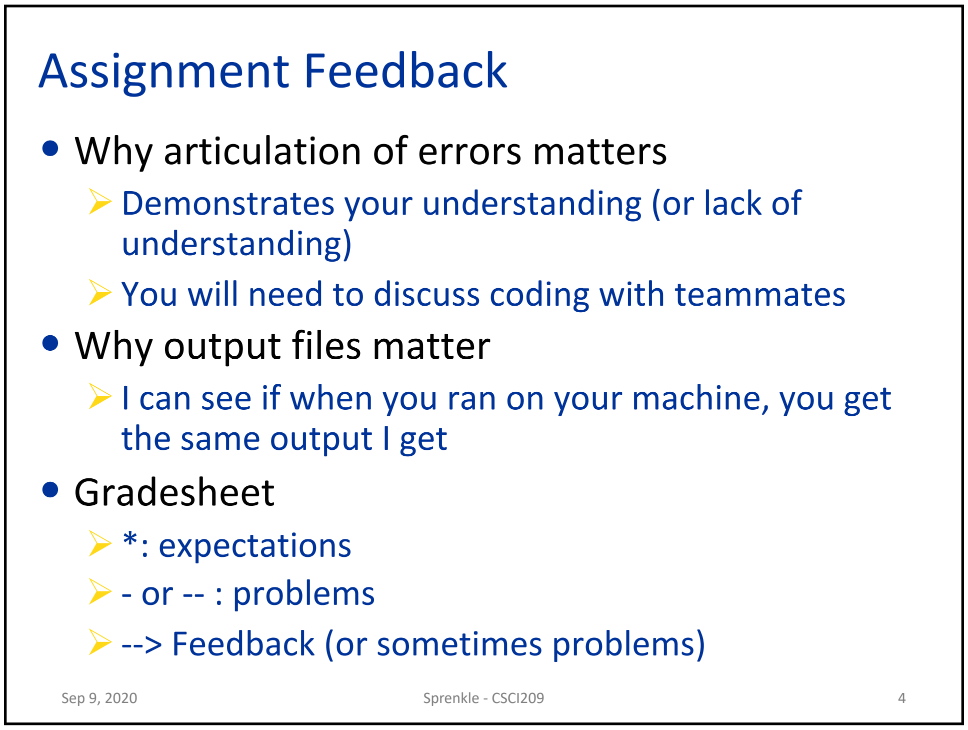


Review: Class Design/Organization

- Fields
  - Chosen first
  - Placed at the beginning or end of class definition
  - Have an access modifier, data type, variable name, and some optional other modifiers
  - Use `this` keyword to access the object
- Constructors
- Methods
  - Need to declare the return type
  - Have an access modifier

Sep 9, 2020 Sprenkle - CSCI209 3

3



Assignment Feedback

- Why articulation of errors matters
  - Demonstrates your understanding (or lack of understanding)
  - You will need to discuss coding with teammates
- Why output files matter
  - I can see if when you ran on your machine, you get the same output I get
- Gradesheet
  - *: expectations
  - - or -- : problems
  - --> Feedback (or sometimes problems)

Sep 9, 2020 Sprenkle - CSCI209 4

4

Assignment 4 Feedback

- Use JavaDocs to describe what methods and constructors do
- Follow examples posted on course page
  - Slide examples are often just snippets
    - Omit comments and other important structure

Review: What is “bad” about this class?

```
public class Farm {
      . . .
      private Chicken headRooster;

      public Chicken getHeadRooster() {
            return headRooster;
      }
      . . .
}
```

Review: What is “bad” about this class?

```
public class Farm {
     . . .
     private Chicken headRooster;

     public Chicken getHeadRooster() {
          return headRooster;
     }
     . . .
}
```

Problem: Giving others access to Farm’s headRooster.
Others can then feed your rooster or change his name!!
(Silly example; understand consequences in design)

7

Fixing the Problem: Cloning

```
public class Farm {
     . . .
     private Chicken headRooster;

     public Chicken getHeadRooster() {
          return (Chicken) headRooster.clone();
     }
     . . .
}
```

Method is available to all objects (inherited from `Object`)

- Another `Chicken` object, with the same data as `headRooster`, is created and returned to the user
- If the user modifies (e.g., feeds) that object, `headRooster` is not affected

8

Cloning

- Cloning is a more complicated topic than it seems from the example
  - Out of scope for this class

What is "bad" about this class?

```
public class Farm {
     . . .
     private Chicken headRooster;

     public Chicken getHeadRooster() {
          return headRooster;
     }
     . . .
}
```

Problem: Giving others access to Farm's headRooster
Others can then feed your rooster or change his name!!
(Silly example; understand consequences in design)

But, then, why is it okay to return a chicken's name, height, or weight?
Similar to Python, primitive types and Strings are ***immutable***.
Since those attributes have data types (String, int, double, respectively) that are immutable, others can't change those attributes.

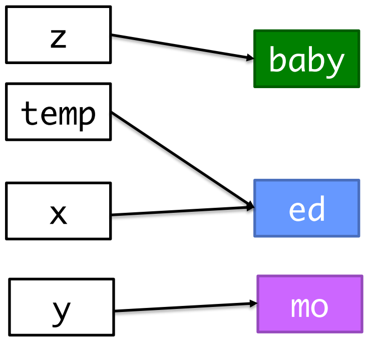
What Happens in This Code?

```
Chicken x, y;
Chicken z = new Chicken("baby", 5, 1.0);
x = new Chicken("ed", 81, 10.3);
y = new Chicken("mo", 63, 6.2);
Chicken temp = x;
x = y;
y = temp;
z = x;
```

1. Think (independently) for 1 minute
2. Share with your neighbor.
3. Discuss as class

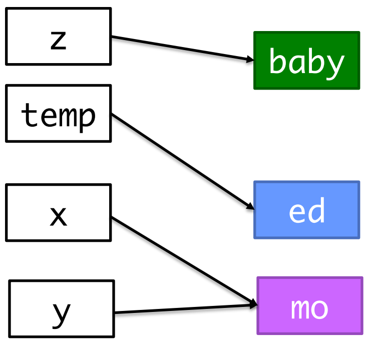
What Happens in This Code?

```
Chicken x, y;
Chicken z = new Chicken("baby", 5, 1.0);
x = new Chicken("ed", 81, 10.3);
y = new Chicken("mo", 63, 6.2);
Chicken temp = x;
x = y;
y = temp;
z = x;
```



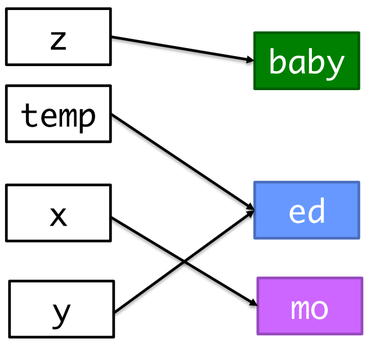
What Happens in This Code?

```
Chicken x, y;
Chicken z = new Chicken("baby", 5, 1.0);
x = new Chicken("ed", 81, 10.3);
y = new Chicken("mo", 63, 6.2);
Chicken temp = x;
x = y;
y = temp;
z = x;
```



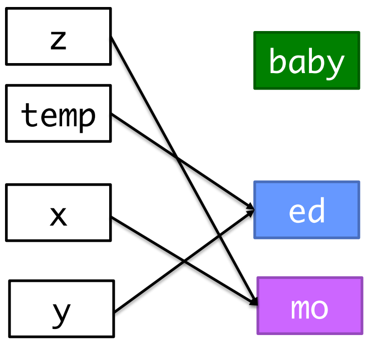
What Happens in This Code?

```
Chicken x, y;
Chicken z = new Chicken("baby", 5, 1.0);
x = new Chicken("ed", 81, 10.3);
y = new Chicken("mo", 63, 6.2);
Chicken temp = x;
x = y;
y = temp;
z = x;
```



What Happens in This Code?

```
Chicken x, y;
Chicken z = new Chicken("baby", 5, 1.0);
x = new Chicken("ed", 81, 10.3);
y = new Chicken("mo", 63, 6.2);
Chicken temp = x;
x = y;
y = temp;
z = x;
```



15

What Happens in This Code?

```
Chicken x, y;
Chicken z = new Chicken("baby", 5, 1.0);
x = new Chicken("ed", 81, 10.3);
y = new Chicken("mo", 63, 6.2);
Chicken temp = x;
x = y;
y = temp;
z = x;
```

Whoops! Lost “baby” chicken!
- No object variable references it
Memory leak!
Luckily Java has **garbage collectors** to clean up the memory leak

16

GARBAGE COLLECTION

Memory Management

- Early languages (e.g., C): free memory when you're done with it
- In C++ and some other OOP languages, classes have explicit *destructor* methods that run when an object is no longer in scope
- Java provides ***automatic garbage collection***
  - Waits until there are no references to an object
  - Reclaims memory allocated for the object that is no longer referenced

Discussion: Benefits and limitations of garbage collection?

Garbage Collector

- Garbage collector is low-priority thread
  - Or runs when available memory gets tight
- Before GC can clean up an object, the object may have opened resources
  - Ex: generated temp files or open network connections that should be deleted/closed first
- GC calls object's `finalize`() method
  - Object's chance to clean up resources

Sep 9, 2020 Sprenkle - CSCI209 19

19

`finalize()`

- Inherited from `java.lang.Object`
- Called before garbage collector sweeps away an object and reclaims its memory
- Should not be used for reclaiming resources
  - i.e., *close resources as soon as possible*
  - Why?
    - *When* method is called is not deterministic or consistent
    - Only know it will run sometime before garbage collection
- Clean up anything that cannot be atomically cleaned up by the garbage collector
  - Close file handles, network connections, database connections, etc.
- Note: no finalizer chaining
  - Must explicitly call parent object's `finalize` method

Sep 9, 2020 Sprenkle - CSCI209 20

20

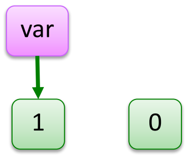
Alternatives to `finalize`

- Recall: unknown when `finalize` will execute—or *if* it will execute
  - Also *heavy performance cost*
- Solution: create your own terminating method
  - User of class terminates when done using object
- Examples: `File`'s or `Window`'s `close` method
- May still want `finalize()` as a safety net if user didn't call the terminate method
  - Log a warning message so user knows error in code

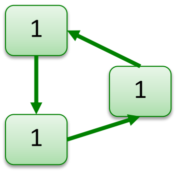
Sep 9, 2020 Do you know what Python does? 21

21

Python Garbage Collection



- Python also does garbage collection
- Python does ***reference counting***
  - On each reference/dereference, update the number of references to the object
    - Can't handle reference cycles



- Python also does ***generational garbage collection*** to handle reference cycles
- Tradeoffs with Java's Garbage Collection
  - Synchronous (not asynchronous) process
  - Cheaper memory costs than Java for keeping track of what can be garbage collected

Sep 9, 2020 Sprenkle - CSCI209 22

22

PARAMETER PASSING

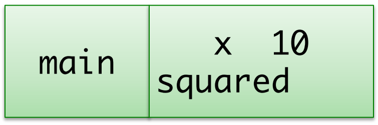
Method Parameters in Java

- Java always passes parameters into methods ***by value***
  - Meaning: the formal parameter becomes a copy of the argument/actual parameter's value
    - caller and callee have two independent variables with the same value
  - Consequence: Methods can**not** change the ***variables*** used as input parameters
  - A subtle point, so we will go through several examples
- Python is something that's not quite pass-by-value—it depends on if the object is mutable or immutable
  - *Pass-by-alias* is one term used

Method Parameters in Java

```java
public static void main(String[] args) {
    int x = 10;
    int squared = square(x);
    System.out.println("The square of " + x + " is " +
                       squared);
}

public static int square(int num) {
    return num*=num;
}
```

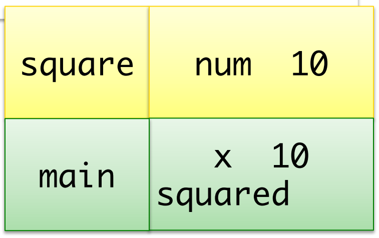
Draw the stack as it changes (similar to Python):



Method Parameters in Java

```java
public static void main(String[] args) {
    int x = 10;
    int squared = square(x);
    System.out.println("The square of " + x + " is " +
                       squared);
}

public static int square(int num) {
    return num*=num;
}
```

num copies the value of x



Method Parameters in Java

```
public static void main(String[] args) {
   int x = 10;
   int squared = square(x);
   System.out.println("The square of " + x + " is " +
                 squared);
}

public static int square(int num) {
   return num*=num;
}
```

main	x 10 squared 100

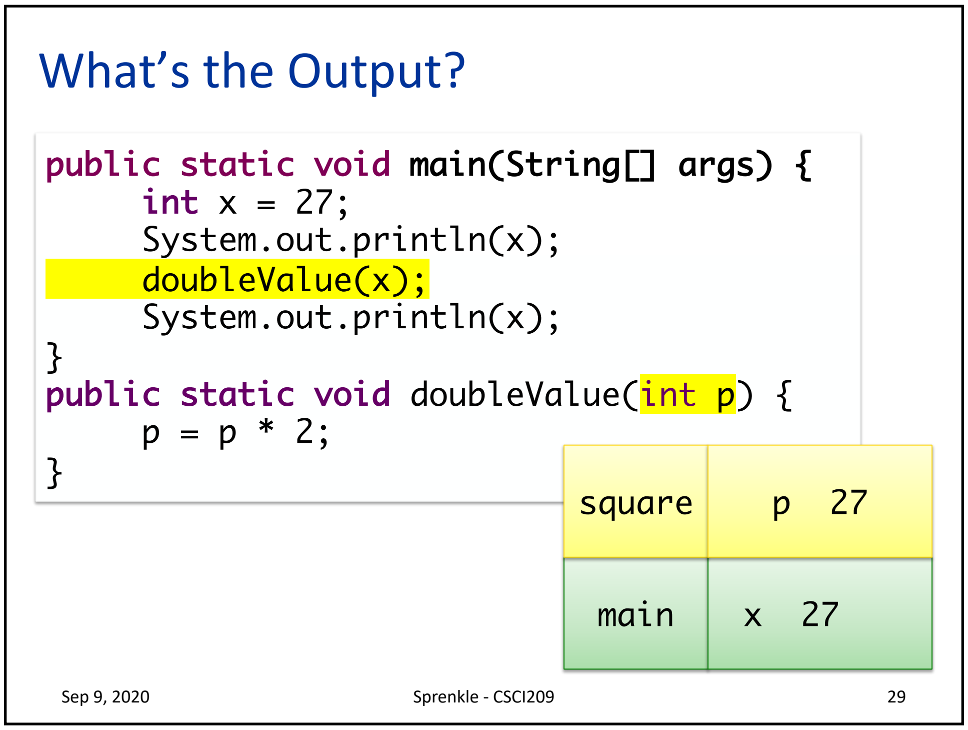
27

What’s the Output?

```
public static void main(String[] args) {
      int x = 27;
      System.out.println(x);
      doubleValue(x);
      System.out.println(x);
}
public static void doubleValue(int p) {
      p = p * 2;
}
```

1. Think (independently) for 1 minute
2. Share with your neighbor.
3. Discuss as class

28



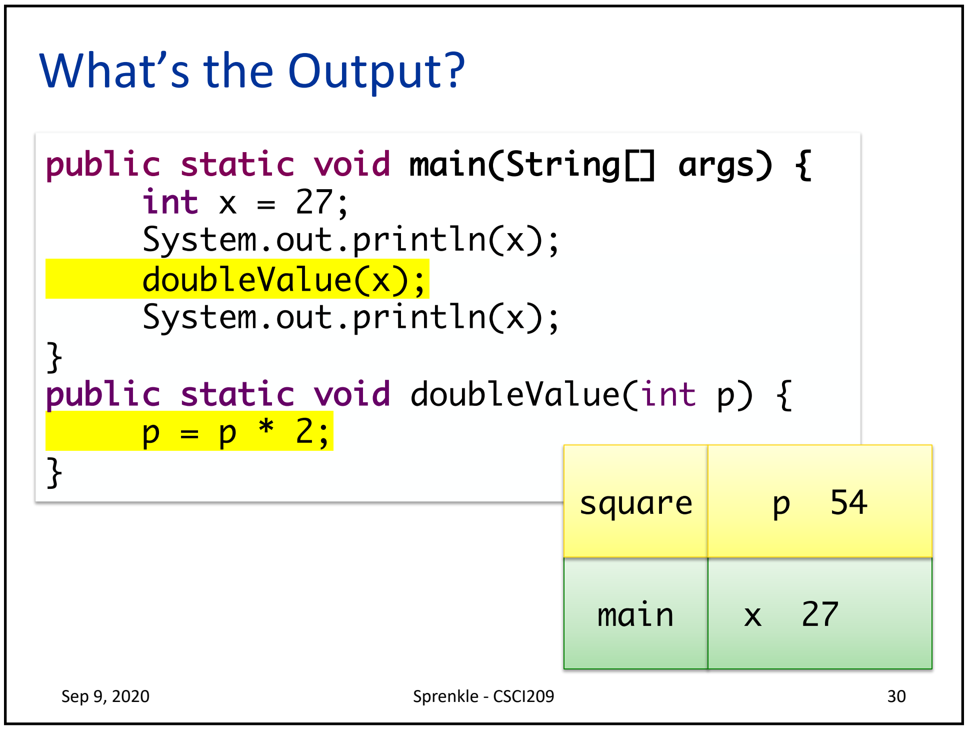
What’s the Output?

```
public static void main(String[] args) {
     int x = 27;
     System.out.println(x);
     doubleValue(x);
     System.out.println(x);
}
public static void doubleValue(int p) {
     p = p * 2;
}
```

square
p 27
main
x 27

Sep 9, 2020 Sprenkle - CSCI209 29

29



What’s the Output?

```
public static void main(String[] args) {
     int x = 27;
     System.out.println(x);
     doubleValue(x);
     System.out.println(x);
}
public static void doubleValue(int p) {
     p = p * 2;
}
```

square
p 54
main
x 27

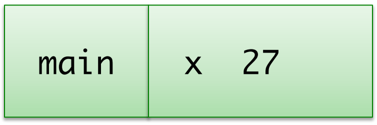
Sep 9, 2020 Sprenkle - CSCI209 30

30

What's the Output?

```
public static void main(String[] args) {
     int x = 27;
     System.out.println(x);
     doubleValue(x);
     System.out.println(x);
}
public static void doubleValue(int p) {
     p = p * 2;
}
```

```
27
27
```



31

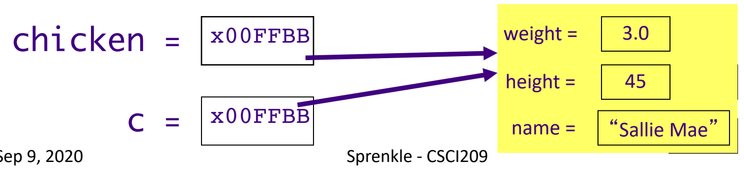
Pass by Value: Objects

- Primitive types are a little more obvious
  - Can't change original variable
- For objects, passing a copy of the parameter looks like

```
public void methodName(Chicken c)
```

Pass Chicken object to `methodName` when calling method

```
methodName(chicken);
```

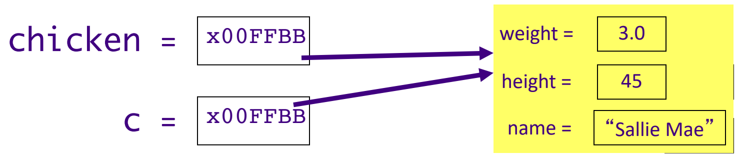


32

Pass by Value: Objects

- What happens in this case?

```
methodName(chicken);
```



```
public void methodName(Chicken c) {
      if( c.getWeight() < MIN ) {
            c.feed();
      }
      …
}
```

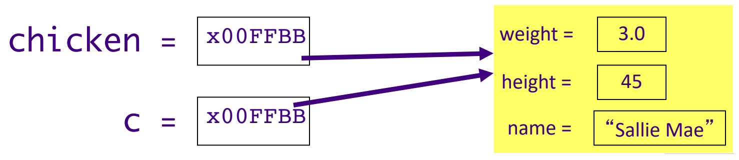
Can the Chicken object be changed in calling method?

33

Pass by Value: Objects

- What happens in this case?

```
methodName(chicken);
```



```
public void methodName(Chicken c) {
      if( c.getWeight() < MIN ) {
            c.feed();
      }
      …
}
```

Can the Chicken object be changed in calling method?

YES! Both chicken and c are pointing to the same Chicken object

34

What's the Output?

```
Farm farm = new Farm("OldMac");
Chicken sal = new Chicken("Sallie Mae", 5, 23.2);
System.out.println(sal.getWeight());
farm.feedChicken(sal);
System.out.println(sal.getWeight());
. . .

// From Farm class
public void feedChicken(Chicken c) {
      c.setWeight( c.getWeight() + .5);
}
```

Sep 9, 2020 Sprenkle - CSCI209 35

35

What's the Output?

```
Farm farm = new Farm("OldMac");
Chicken sal = new Chicken("Sallie Mae", 5, 23.2);
System.out.println(sal.getWeight());
farm.feedChicken(sal);
System.out.println(sal.getWeight());
. . .

// From Farm class
public void feedChicken(Chicken c) {
      c.setWeight( c.getWeight() + .5);
}
```

```
23.2
23.7
```

Sep 9, 2020 Sprenkle - CSCI209 36

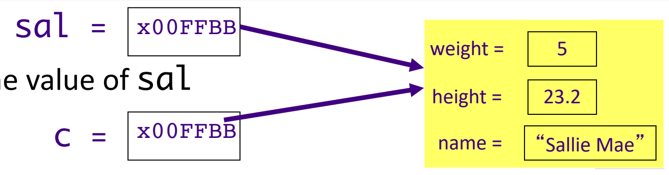
36

What's the Output?

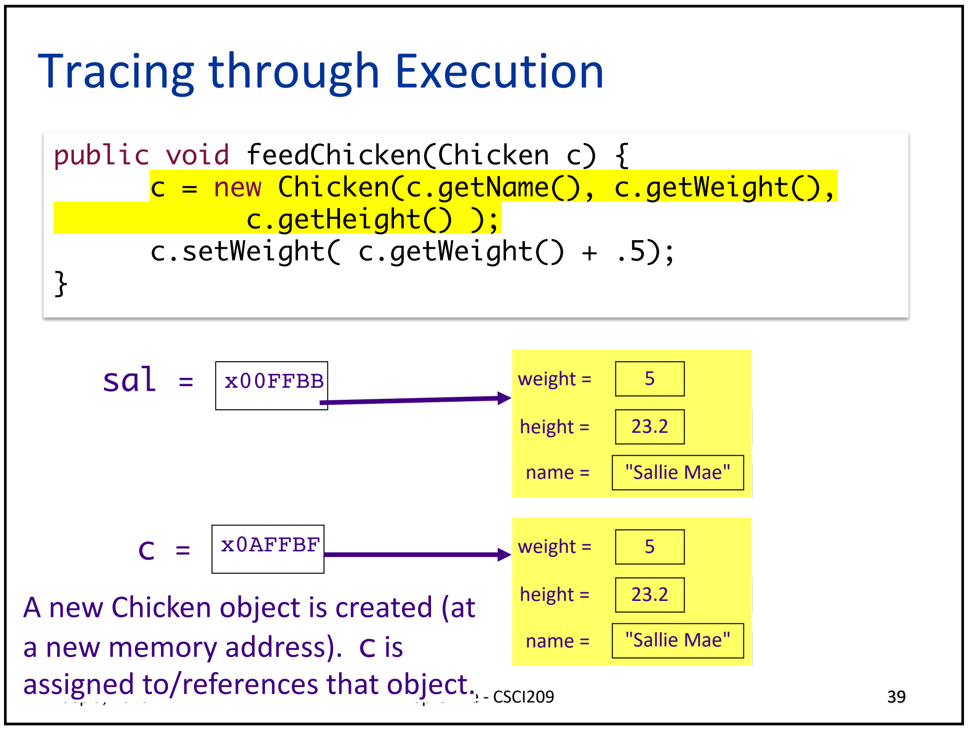
```
Farm farm = new Farm("OldMac");
Chicken sal = new Chicken("Sallie Mae", 5, 23.2);
System.out.println(sal.getWeight());
farm.feedChicken(sal);
System.out.println(sal.getWeight());
. . .

// From Farm class
public void feedChicken(Chicken c) {
      c = new Chicken(c.getName(), c.getWeight(),
            c.getHeight() );
      c.setWeight( c.getWeight() + .5);
}
```

Tracing through Execution

```
Farm farm = new Farm("OldMac");
Chicken sal = new Chicken("Sallie Mae", 5, 23.2);
System.out.println(sal.getWeight());
farm.feedChicken(sal);
System.out.println(sal.getWeight());
. . .
// From Farm class
public void feedChicken(Chicken c) {
      c = new Chicken(c.getName(), c.getWeight(),
            c.getHeight() );
      c.setWeight( c.getWeight() + .5);
}
```



c copies the value of sal



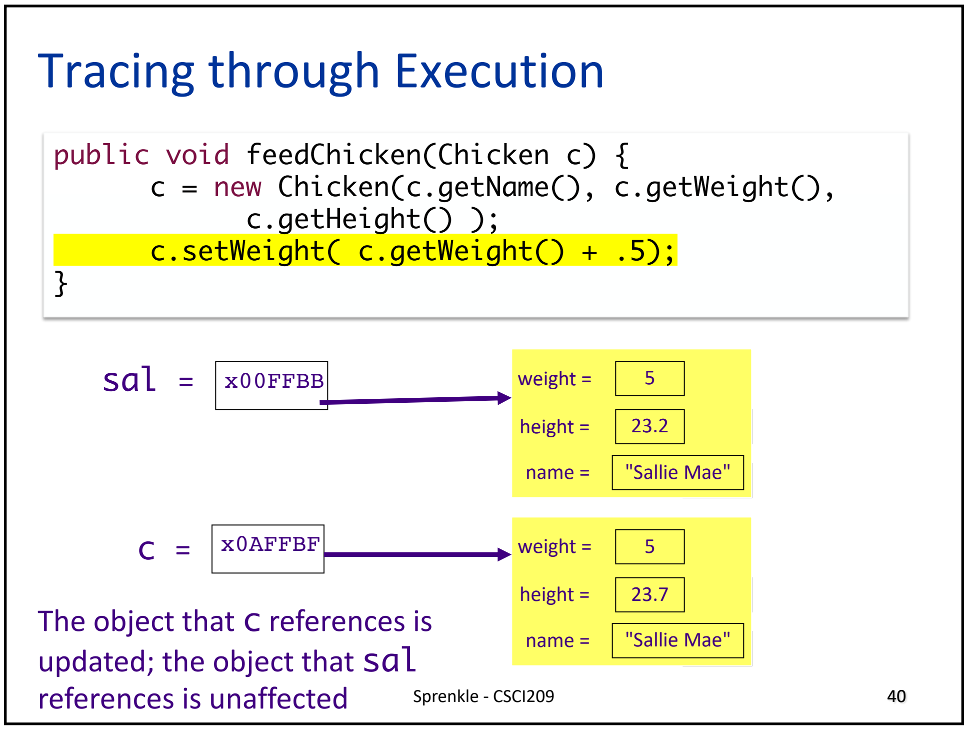
Tracing through Execution

```
public void feedChicken(Chicken c) {
        c = new Chicken(c.getName(), c.getWeight(),
                c.getHeight() );
        c.setWeight( c.getWeight() + .5);
}
```

A new Chicken object is created (at a new memory address). c is assigned to/references that object.

- CSCI209 39

39



Tracing through Execution

```
public void feedChicken(Chicken c) {
        c = new Chicken(c.getName(), c.getWeight(),
                c.getHeight() );
        c.setWeight( c.getWeight() + .5);
}
```

The object that c references is updated; the object that sal references is unaffected

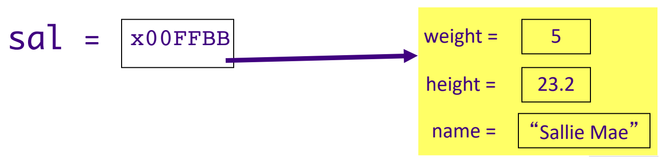
Sprenkle - CSCI209 40

40

Tracing through Execution

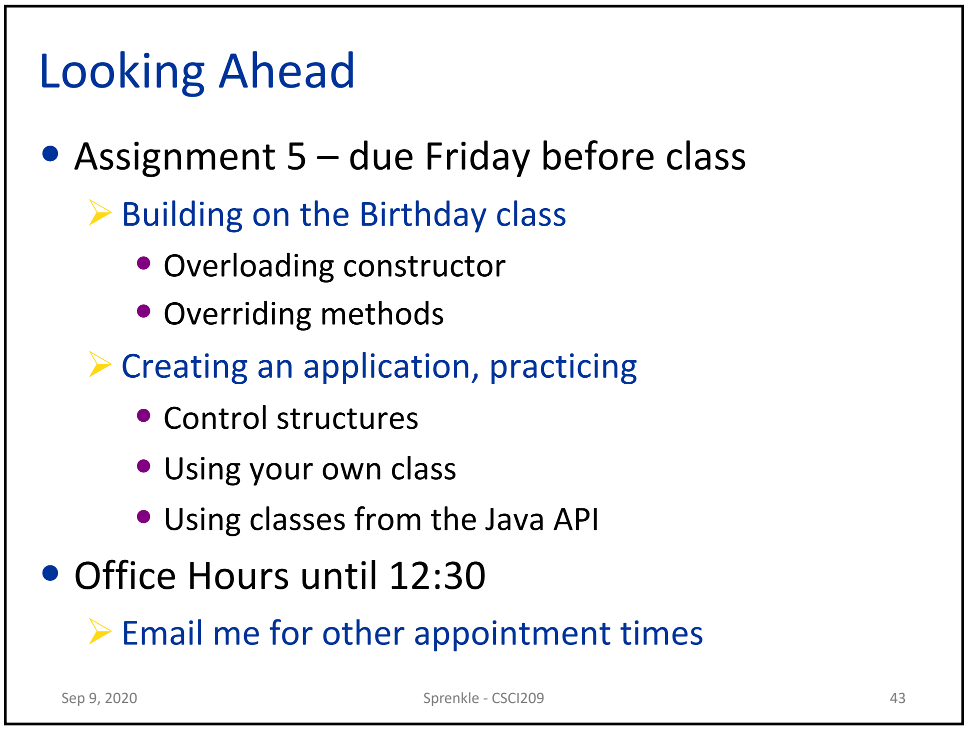
```
Farm farm = new Farm("OldMac");
Chicken sal = new Chicken("Sallie Mae", 5, 23.2);
System.out.println(sal.getWeight());
farm.feedChicken(sal);
System.out.println(sal.getWeight());
. . .
// From Farm class
public void feedChicken(Chicken c) {
      c = new Chicken(c.getName(), c.getWeight(),
            c.getHeight() );
      c.setWeight( c.getWeight() + .5);
}
```

```
23.2
23.2
```



Summary of Method Parameters

- Everything is passed ***by value*** in Java
- An ***object variable*** (not an object) is passed into a method
  - Changing the *state* of an object in a method changes the state of object outside the method
  - Method does not see a copy of the original object



Looking Ahead

- Assignment 5 – due Friday before class
  - Building on the Birthday class
    - Overloading constructor
    - Overriding methods
  - Creating an application, practicing
    - Control structures
    - Using your own class
    - Using classes from the Java API
- Office Hours until 12:30
  - Email me for other appointment times

Sep 9, 2020 Sprenkle - CSCI209 43

43