Objectives

- Object-oriented programming in Java
  - Object references
  - Static methods, fields
  - Constructors
  - Default constructors

Assign 1 Discussion

- Java Conventions:
  - Class names: begin with capital letter
  - Class constants: name with all capital letters, e.g., `DIFFICULTY_SCORE`

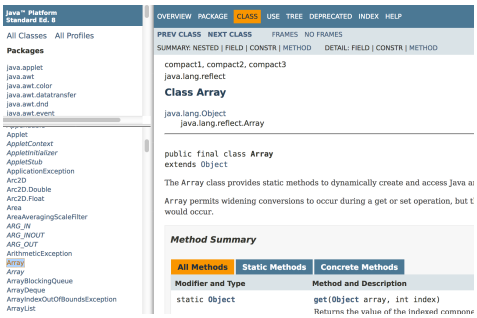
- Can fully-specify the class instead of importing class

```
java.util.Arrays.sort(myArray);
```

Danger of a Large Library

- Lots of classes that seem like they're what we want but aren't

`java.lang.reflect.Array`
`javax.sql.rowset.serial.Array`



An array (e.g., `int[] array`) is **not** an instance of a class, so we cannot call methods on it.

Benefits of Static Typing

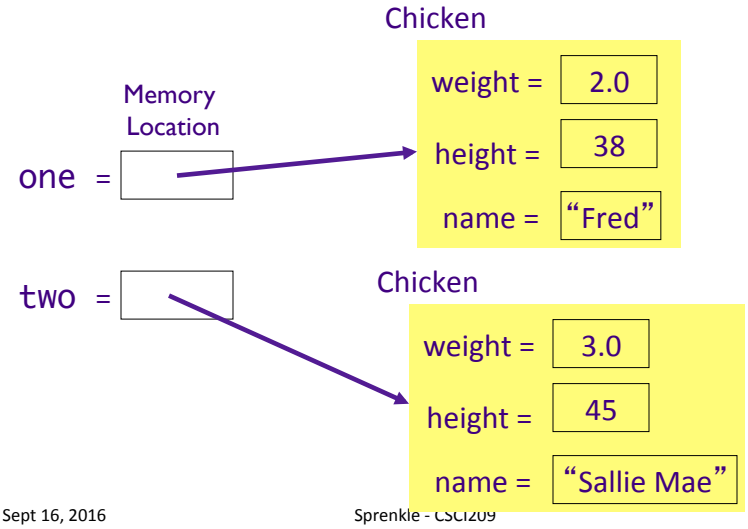
- Easier to remember type of variable
  - Know operations that can be executed on a variable of a certain type
- Compiler can check that you're only using valid operations for this type
- More benefits later this semester

Review

- Why OO programming?
  - What are its components?
- What's wrong with "white-box" programming?
  - How does Java help to enforce black-box programming?
- What is the syntax for defining a constructor?
- What is the syntax for defining a method?
- What is the Java equivalent of None?

Object References

- Variable of type object: value is memory location



Object References

- Variable of type object: value is memory location

one =

two =

If I haven't called the constructor, only *declared* the variables:

```
Chicken one;
Chicken two;
```

Both one and two are equal to `null`

Null Object Variables

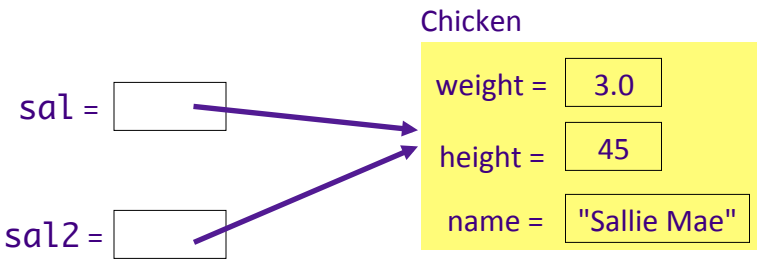
- An object variable can be explicitly set to `null`
  - Means that the object variable does not currently refer to any object
- Can test if an object variable is set to `null`

```
Chicken chick = null;
… … …
if (chick == null) {
    . . .
}
```

Multiple Object Variables

- More than one object variable can refer to the same object

```
Chicken sal = new Chicken("Sallie Mae");
Chicken sal2 = sal;
```



What happens here?

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

What happens here?

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

Whoops! Lost "baby" chicken!
Memory leak!
Luckily Java has **garbage collectors** to clean up the memory leak

STATIC METHODS AND FIELDS

Review

- What does static mean?
- How do we make a *class constant*?

Static Methods/Fields

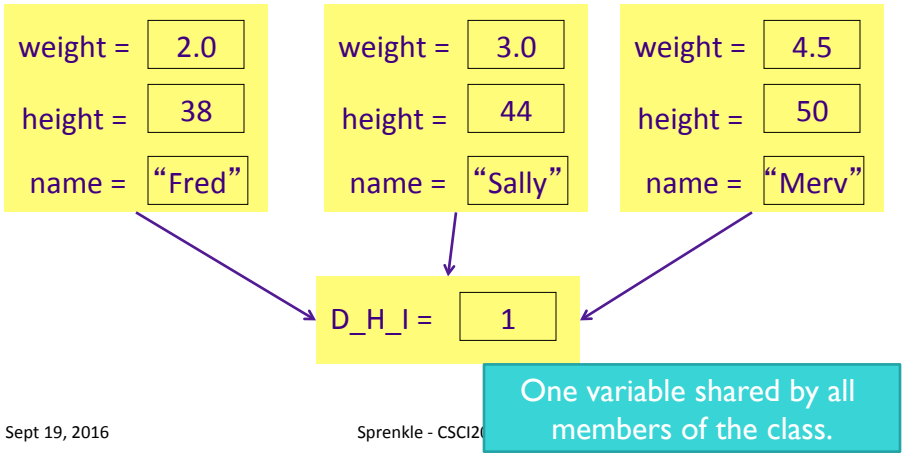
- For related functionality/data that isn't specific to any particular object

- `java.lang.Math`
  - No constructor (what does that mean?)
  - Static fields: PI, E
  - Static methods:
    - `static double sin(double a)`

Chicken static field example

```
static int DEFAULT_HEIGHT_INCREASE = 1;
```

A bunch of Chicken objects



Static Methods

- Do **not** operate on objects
  - Can**not** access instance fields of their class
- Can access *static fields* of their class
- Similar to Python *functions* that are associated with the class

Why is `main` static?

`main()`

- Most common static method
- `main()` does not operate on any objects
  - Runs when a program starts…there are no objects yet
- `main()` executes and constructs the objects the program needs and will use
  - Like the *driver function* for the program

Analyzing `java.lang.String`

- `String toUpperCase()`
  - Converts all of the characters in this String to upper case
- `static String valueOf(boolean b)`
  - Returns the string representation of the `boolean` argument

Why can the second method be `static`?

Static Summary

- Static fields and methods are part of a class and **not** an object
  - Do not require an object of their class to be created in order to use them
- When would we make a method `static`?
  - When a method does not have to access an object's state (fields) because all needed data are passed into the method
  - When a method only needs to access static fields in the class

MORE ON CONSTRUCTORS

More on Constructors

- A class can have **more than one** constructor
  - Whoa! Let that sink in for a bit
- A constructor can have zero, one, or multiple parameters
- A constructor has no return value
- A constructor is always called with the new operator

Constructor Overloading

- Allowing > 1 constructor (or any method) with the same name is called ***overloading***
  - Constraint: Each of the methods that have the same name must have different parameters so that compiler can distinguish between them
    - "different" → *Number* and/or *type*
- Compiler handles ***overload resolution***
  - Process of matching a method call to the correct method by matching the parameters
- No function overloading in Python

Why isn't overloading possible in Python?

`final` keyword

- An instance field can be `final`
- `final` instance fields **must** be set in the constructor or in the field declaration
  - Cannot be changed *after object is constructed*

```
private final String dbname = "invoices";
private final String id;
…
public MyObject( String id ) {
      this.id = id;
}
```

Default Constructor

- ***Default constructor*: constructor with no parameters**
- If class has *no constructors*
  - **Compiler** provides a default constructor
    - Sets all instance fields to their default values
- If a class has at least one constructor and no default constructor
  - Default constructor is NOT provided

Default Constructor

- Chicken class has one constructor:

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

➡ No default constructor

```
Chicken chicken = new Chicken();
```

- Is a compiler error

Constructors Calling Constructors

- Can call a constructor from inside another constructor
- The **first** statement of constructor must be

```
this( . . . );
```

to call another constructor of the same class

  - `this` refers to the object being constructed

Why would you want to call another constructor?

Constructors Calling Constructors

- Why would you call another constructor?
  - Reduce code size/reduce duplicate code
- Ex: if name not provided, use default name

```
Chicken( int height, double weight ) {
      this( "Bubba", height, weight);
}
```

- Another example:

```
Chicken( int height, double weight ) {
      this();
      this.height = height;
      this.weight = weight;
}
```

Not in example code online

TODO

- Assignment 3
  - Modifying the `Birthday` class
  - Static method practice