Objectives

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

Review

- 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?

Assign 0 Feedback

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

Assign 1

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

What does this code do?

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

What does this code do?

```java
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?

**Property: Encapsulation**

- *Encapsulation*: Combining data and behavior (functionality) into one package (the object) and hiding the implementation of the data from the user of the object.

- Java’s characteristics allow us to enforce encapsulation better than Python.

**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.

**Discussion**

- What is the problem with white-box programming?

**Classes, Objects, Methods**

- An object’s state is stored in **instance fields**

- **Method**: sequence of instructions that access/modify an object’s data
  - Accessor: accesses (doesn’t modify) object
  - Mutator: changes object’s data
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

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

Constructing objects using `new`

- Given the File **constructor**
  ```java
  File(String pathname)
  ```
- Create a new File object using `new` keyword
  - Recall `new` means allocates memory

```java
    File myFile = new File("debug.out");
    Type/Classname
```

Calling Methods

- Similar to Python
  ```java
  <objectname>..<methodname>(<parameters>);
  ```
- Examples with String and System classes
- To call **static** methods, use
  ```java
  <ClassName>..<methodname>(<parameters>);
  ```

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
**Classes and Objects**
- Java is pure object-oriented programming
  - All data and methods in a program must be contained within a class
- But, for data, can use objects as well as primitive types (e.g., int, float, char)

**Example: Chicken class**
- **State**
  - Name, weight, height
- **Behavior**
  - **Accessor methods**
    - `getWeight`, `getHeight`, `getName`
    - Convention: "get" for "getter" methods
  - **Mutator methods**
    - `feed`: adds weight and height when bird eats
    - `setName`

**General Java Class Structure**

```java
public class ClassName {
    // ----------- INSTANCE VARIABLES -----------
    // define variables that represent object's state
    private int inst_var;
    // ----------- CONSTRUCTORS -------------
    public ClassName() {
        // initialize data structures
    }
    // ----------- METHODS ------------
    public return inst_var;
}
```

**Example: Chicken class**
- **State**
  - Name, weight, height
- **Behavior**
  - **Accessor methods**
    - `getWeight`, `getHeight`, `getName`
    - Convention: "get" for "getter" methods
  - **Mutator methods**
    - `feed`: adds weight and height when bird eats
    - `setName`

**Discussion:** data types for state variables?

**Instance Variables: Chicken.java**
```
public class Chicken {
    private String name;  // in cm
    private int height;   // in cm
    private double weight; // in lbs
}
```

**Constructor: Chicken.java**
```
public class Chicken {
    private String name;  // in cm
    private int height;   // in cm
    private double weight;
    public Chicken(String name, int h, double weight) {
        this.name = name;
        this.height = h;
        this.weight = weight;
    }
}
```
Constructor: Chicken.java

```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 w) {
        this.name = name;
        this.height = h;
        this.weight = w;
    }
    // ----------- GETTER METHODS -------------
    public String getName() {
        return name;
    }
    // ----------- MUTATOR METHODS -------------
    public void feed() {
        weight += .2;
        height += 1;
    }
}
```

Example: Chicken class

- **State**
  - Name, weight, height
- **Behavior**
  - Accessor methods
    - `getName`, `getHeight`, `getName`
  - Convention: "get" for "getter" methods
  - Mutator methods
    - `feed`: adds weight, height
    - `setName`

**Discussion**: What are the methods' **input** (parameters) and **output** (what is returned)?

Constructing objects

- Given the Chicken constructor
  ```java
  Chicken( String name, int height, double weight )
  ```
  create three chickens
  - "Fred", weight: 2.0, height: 38
  - "Sallie Mae", weight: 3.0, height: 45
  - "Momma", weight: 6.0, height: 83

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