Managing Inheritance

Today's Objectives

- **`final`** classes & methods
- Review of constructor inheritance
- **`abstract`** classes & methods
- **`public`** vs. **`private`** vs. **`protected`**

Sleeping Animals...

final Methods & Classes

- Making methods **`final`** means that children can't override what the parent has defined
    - Prevents logic that is integral to the application from being overridden by a poorly behaving subclass
    - Just a design decision that should have a good reason for using
- Making classes **`final`** means that another class can't inherit from it
    - Again, just a design decision. Should have a good reason for doing it

Constructor Inheritance Review

- In **`FarmAnimal`**, we have a constructor, but have to implement that in the sub classes
- Constructors aren't inherited and must be redefined
- If the default constructor is not defined in the base class, you must call a valid constructor in the subclass constructor
    - Note that you **only have to call <u>one</u> constructor** from you subclass

FarmAnimals Redux:

Abstract Classes & Methods

Abstraction & Classes/Methods

- **Abstraction** is the principle of handling complexity by hiding unnecessary details from the user. Its goal is to enable the user to implement more complex logic on top of the provided abstraction without understanding or even thinking about all the hidden complexity.
    - Abstraction is sometimes referenced as a fourth principle of Object-Oriented Programming but it is often not included as one of the principles because it motivates the other three principles in one way or another.
- An **abstract class** is a class that cannot be instantiated. It exists solely for purposes of inheritance and polymorphism.
- An **abstract method/function** is a method/function that does not have an implementation and must be overridden by subclasses.

Abstract Class & Methods: Rules

- Abstract classes cannot have objects created from them, but they can provide logic and structure to their subclasses.
- Abstract methods are methods with no logic that must be implemented by concrete subclasses.
- If a class has an abstract method, it must be an abstract class.
- If a class does not override an abstract method from its parent, it must also be an abstract class

How to Choose Access Modifiers

- **`public`** is for any set in stone methods that you want other programmers to rely on to use your object.

- **`protected`** is for building connections between inherited classes. It lets you have methods in a parent that are accessible by its children, and vice versa, but not anyone else. **Be careful** with this though, because in Java **`protected`** properties and methods are **not only accessible to subclasses but also to any class in the same same package!!!!**

- **`private`** is for unstable or helper methods that may change and only have use inside the class itself.