Abstract Classes and Interfaces (?)

June 21, 2017

Reading Quiz

Abstract Classes

A. Abstract classes inherit from multiple parents, standard classes only inherit from one parent.

B. Abstract classes can only have one instance at a time, standard classes can have any number.

C. Abstract classes cannot be directly instantiated, standard classes can be directly instantiated

D. Abstract classes can only contain primitive properties, standard classes can have primitive and reference properties.

Abstract Methods

A. Abstract classes consist of only abstract methods

B. Abstract methods do not specify their implementation

C. Abstract methods are the same thing as a "static final" method

D. Abstract methods perform faster than standard methods

Interfaces vs. Abstract Classes

A. Interfaces always inherit from Abstract Classes

B. Abstract Classes always implement Interfaces

C. Interfaces cannot inherit from Abstract Classes

D. Abstract Classes cannot implement Interfaces

Whats the Result?

```
abstract class Person {
    public String name() {
        return "Person";
    }
}

class Child extends Person {
    public String name() {
        return "Child";
    }
}

Child aKindChild = new Child();
Person aMeanPerson = new Person();

System.out.println(
    aMeanPerson.name() + "💔" +
    aKindChild.name()
);
```

A. "Child💔Person"

B. "Person💔Child"

C. "null💔null"

D. Won't compile

E. Runtime error

Whats the Result?

```
interface Emojier {
    public String asEmoji();
}

class Dog implements Emojier {
    public String asEmoji() {
        return "🐶";
    }
}

class Cat implements Emojier {
    public String asEmoji() {
        return "🐱";
    }
}

Cat toonces = new Cat();
String anEmoji = toonces.asEmoji();
System.out.println(anEmoji);
```

A. "🐱"

B. "🐶"

C. "" (ie empty string)

D. Won't compile

E. Runtime error

Done!

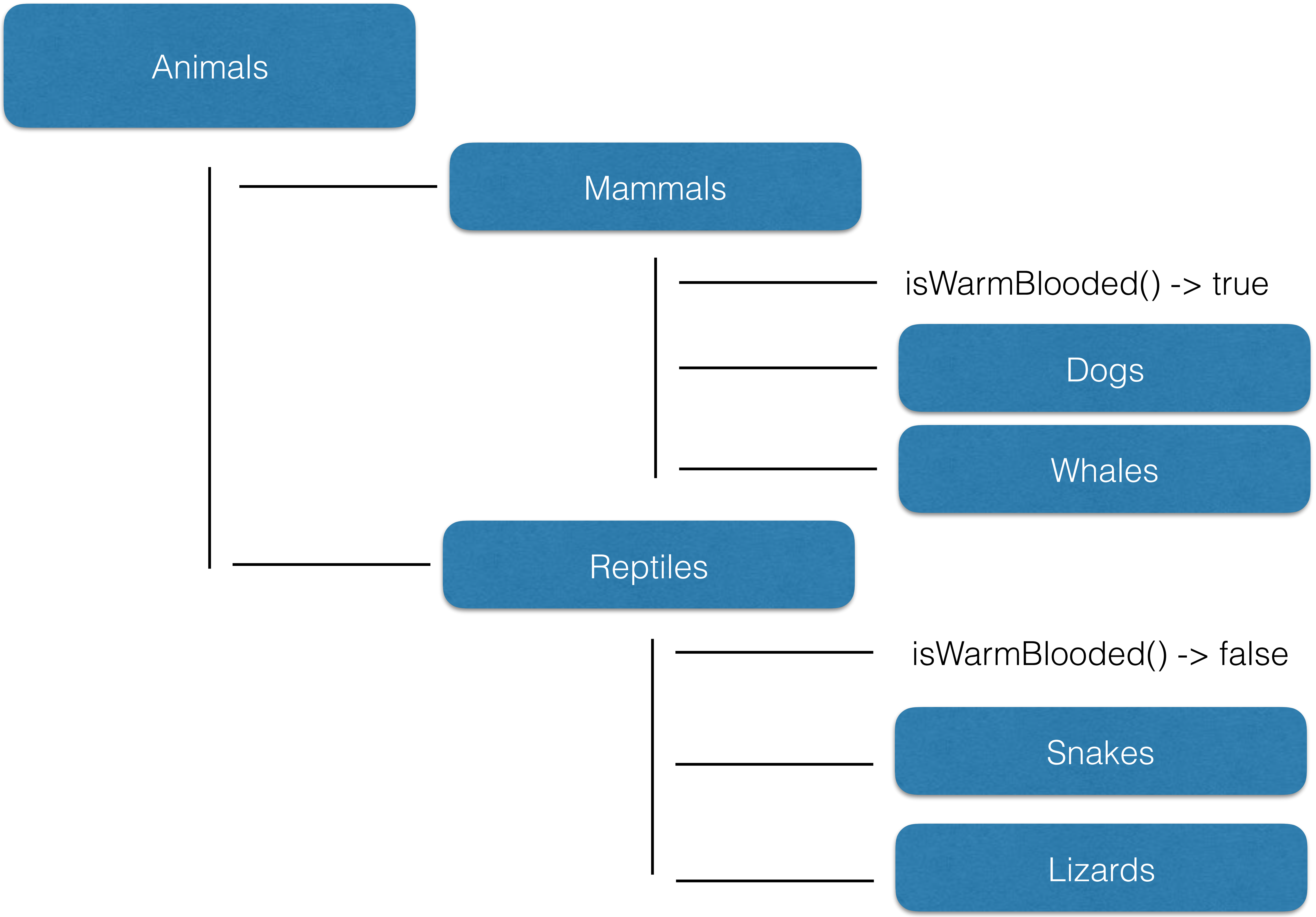
Housekeeping

- Piazza Recap / Java environments
- Homework 2 questions
- Homework 3 incoming

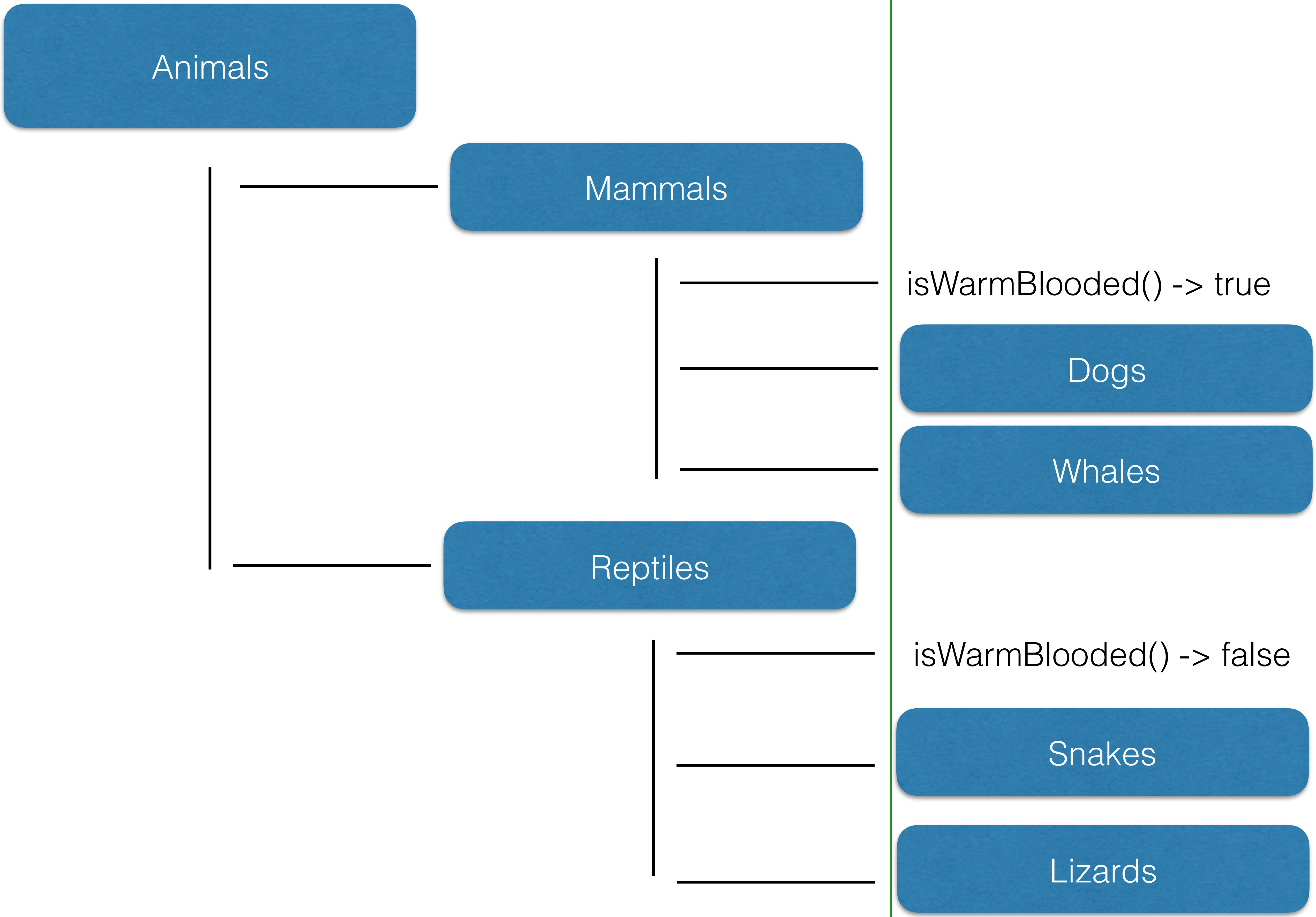
Abstract Classes

Problem

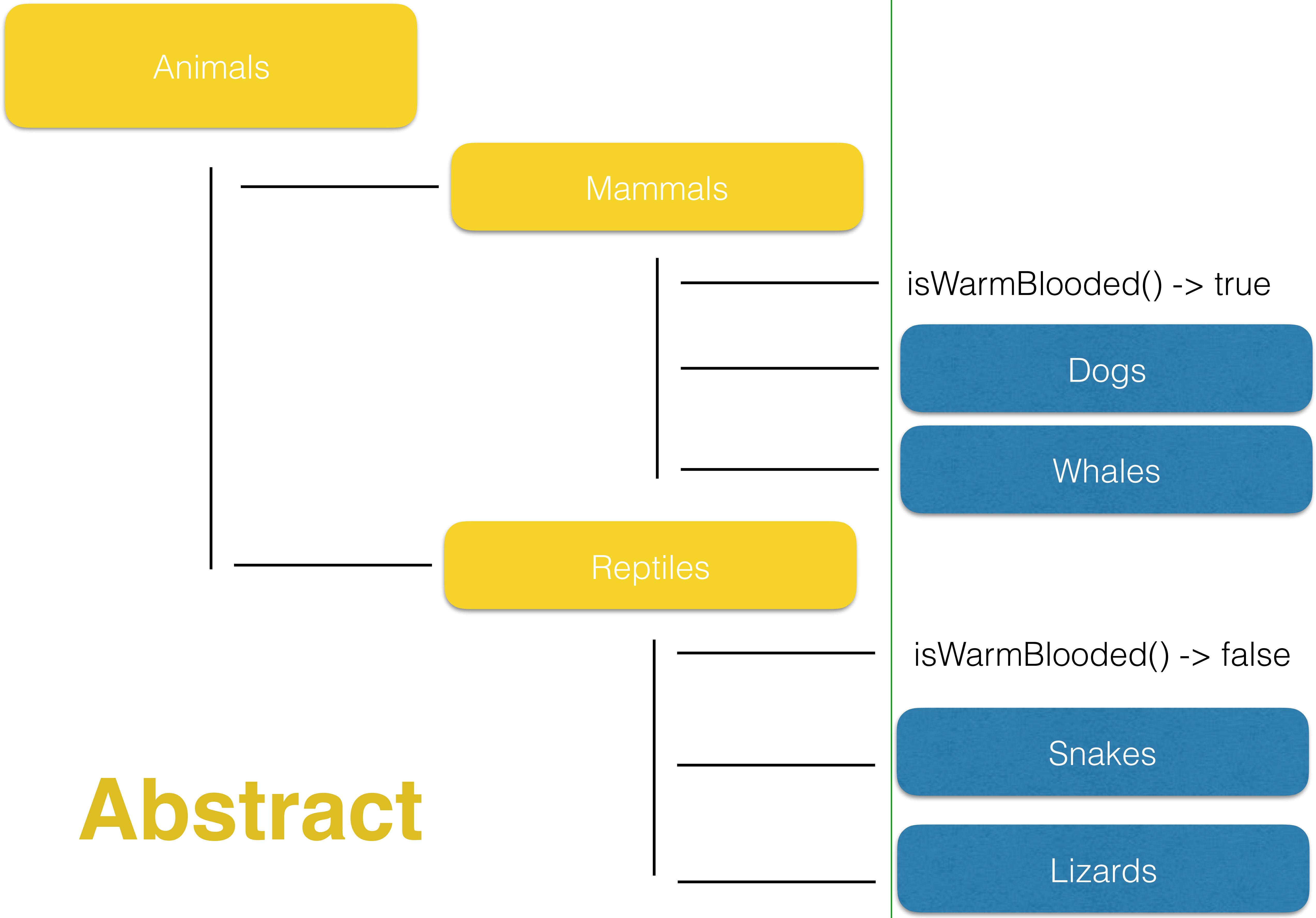
- Types express a Taxonomy **and** data / functionality
- Some types are meant to represent data in a program
- Some types are "only" categorization
- Code should enforce this distinction


Animals
Mammals
isWarmBlooded() -> true
Dogs
Whales
Reptiles
isWarmBlooded() -> false
Snakes
Lizards

Categorization <– | –> Program Logic

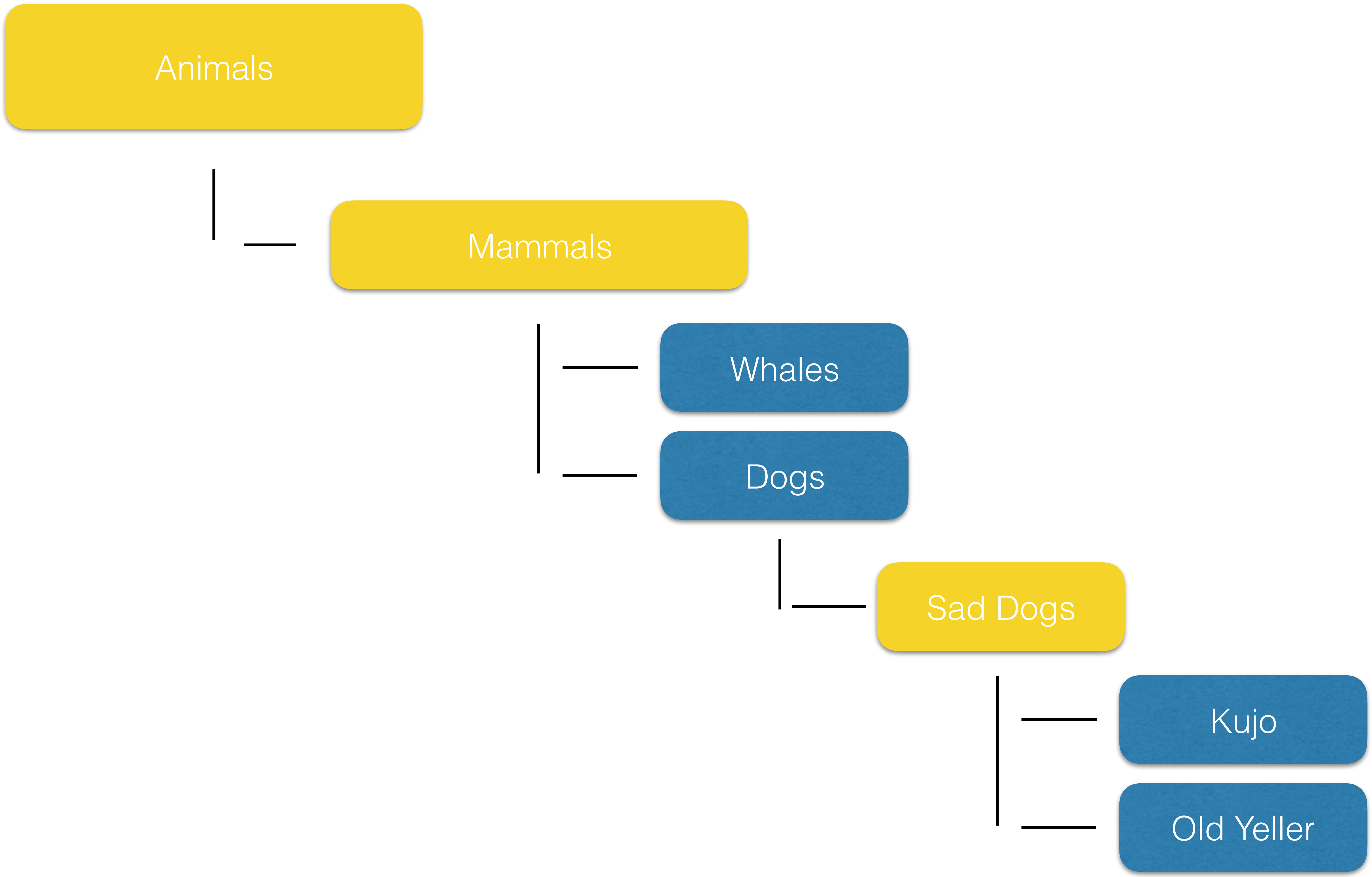


Categorization <– –> Program Logic

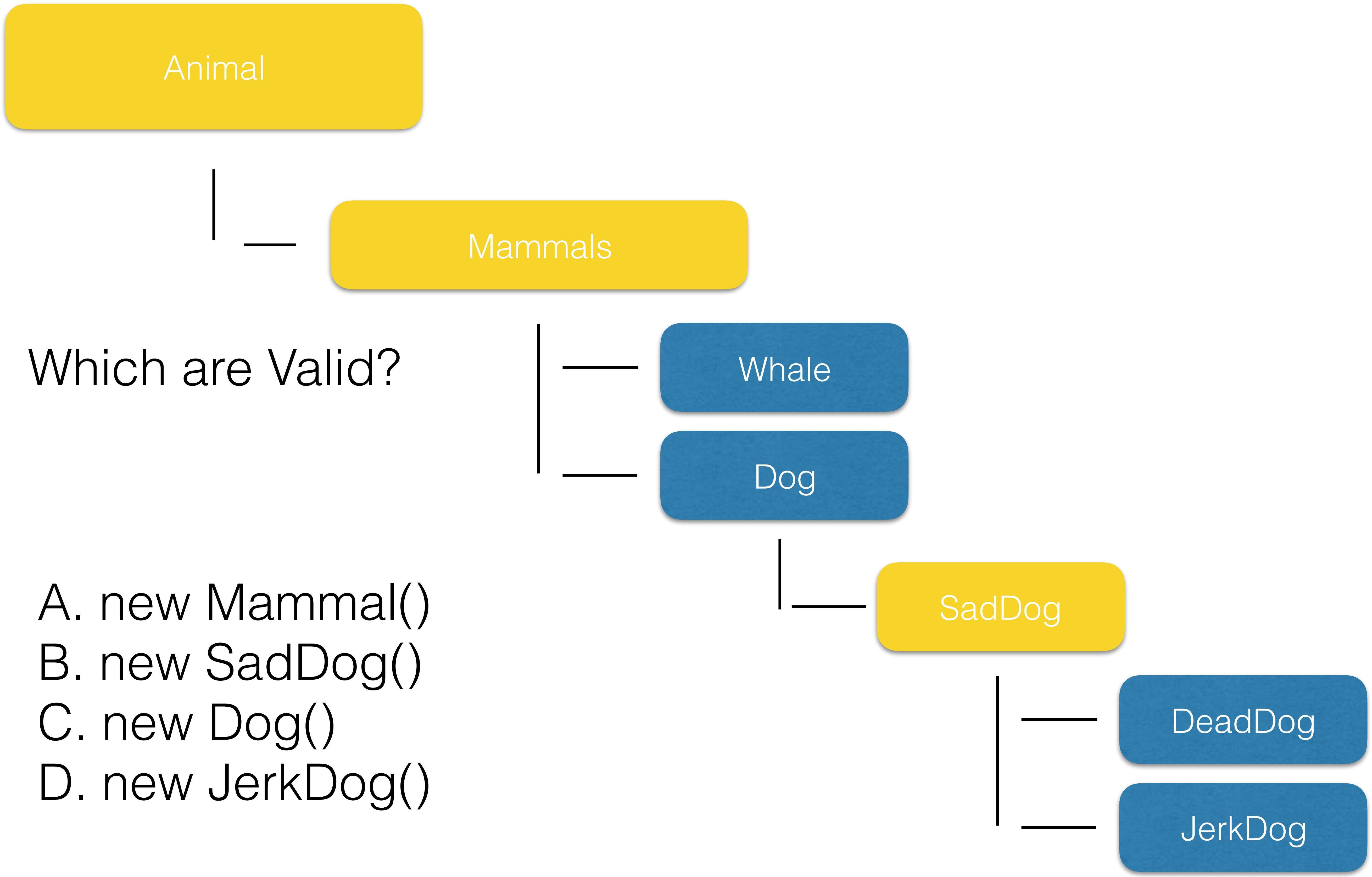


Java Solution: Abstract Classes

- Can not be instantiated, but can be inherited
- Standard inheritance model
- Arbitrarily deep



Animals
Mammals
Whales
Dogs
Sad Dogs
Kujo
Old Yeller



Which are Valid?

A. new Mammal()
B. new SadDog()
C. new Dog()
D. new JerkDog()

Abstract Use Cases?

	Abstract	Concrete Subtype(s)
1	SimpsonCharacter	Bart, Lisa
2	AdminUser	StandardUser, AnonymousUser
3	MembersOfTheRamones	JoeyRamone, TommyRamone
4	CreditCardChargeError	RareCreditCardChargeError

Animal.java –>

Abstract Methods

- Taxonomy can impose "helpful" restrictions too
- Things below me must implement these methods
- Keep types clean

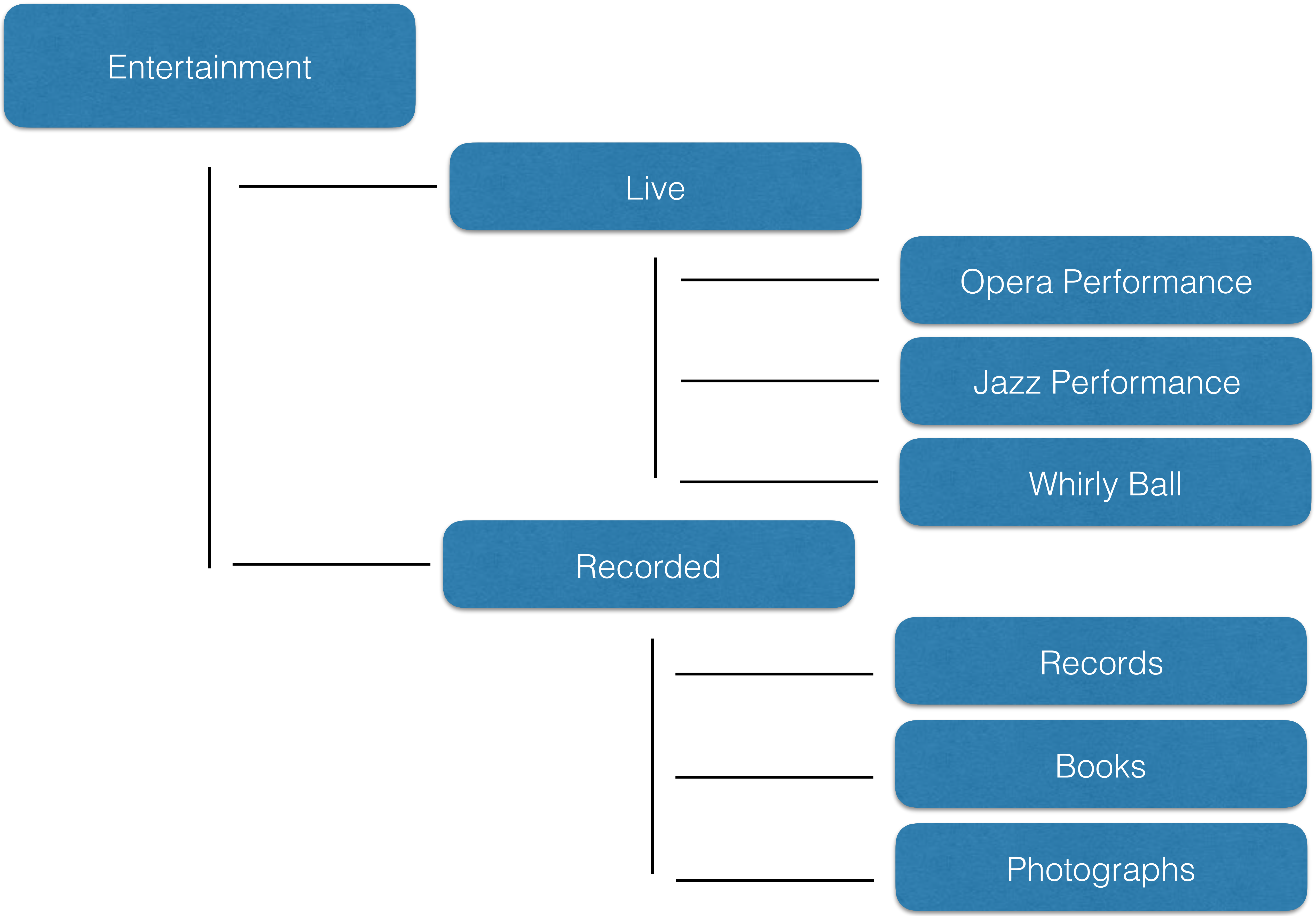
Fur.java –>

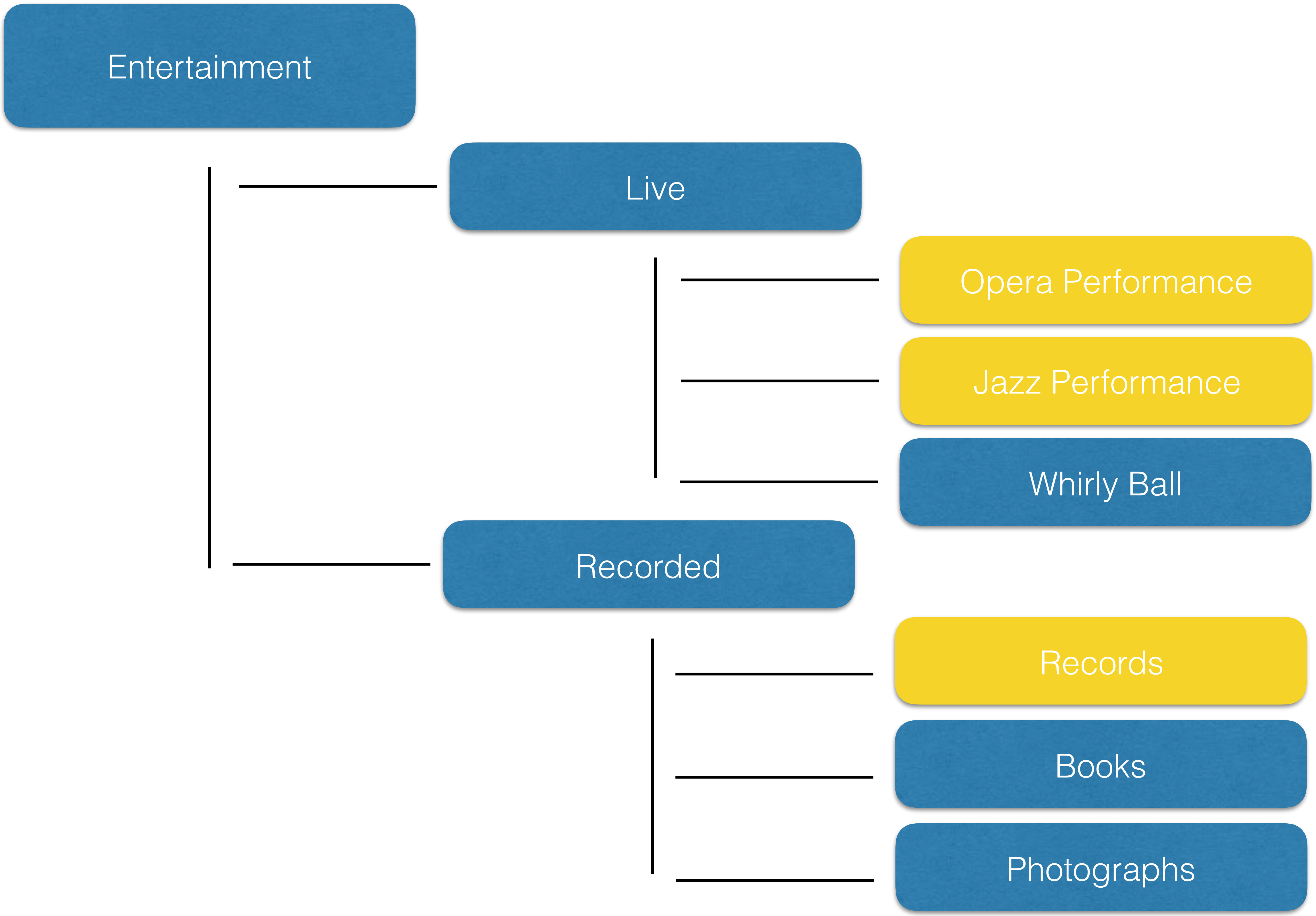
Interfaces

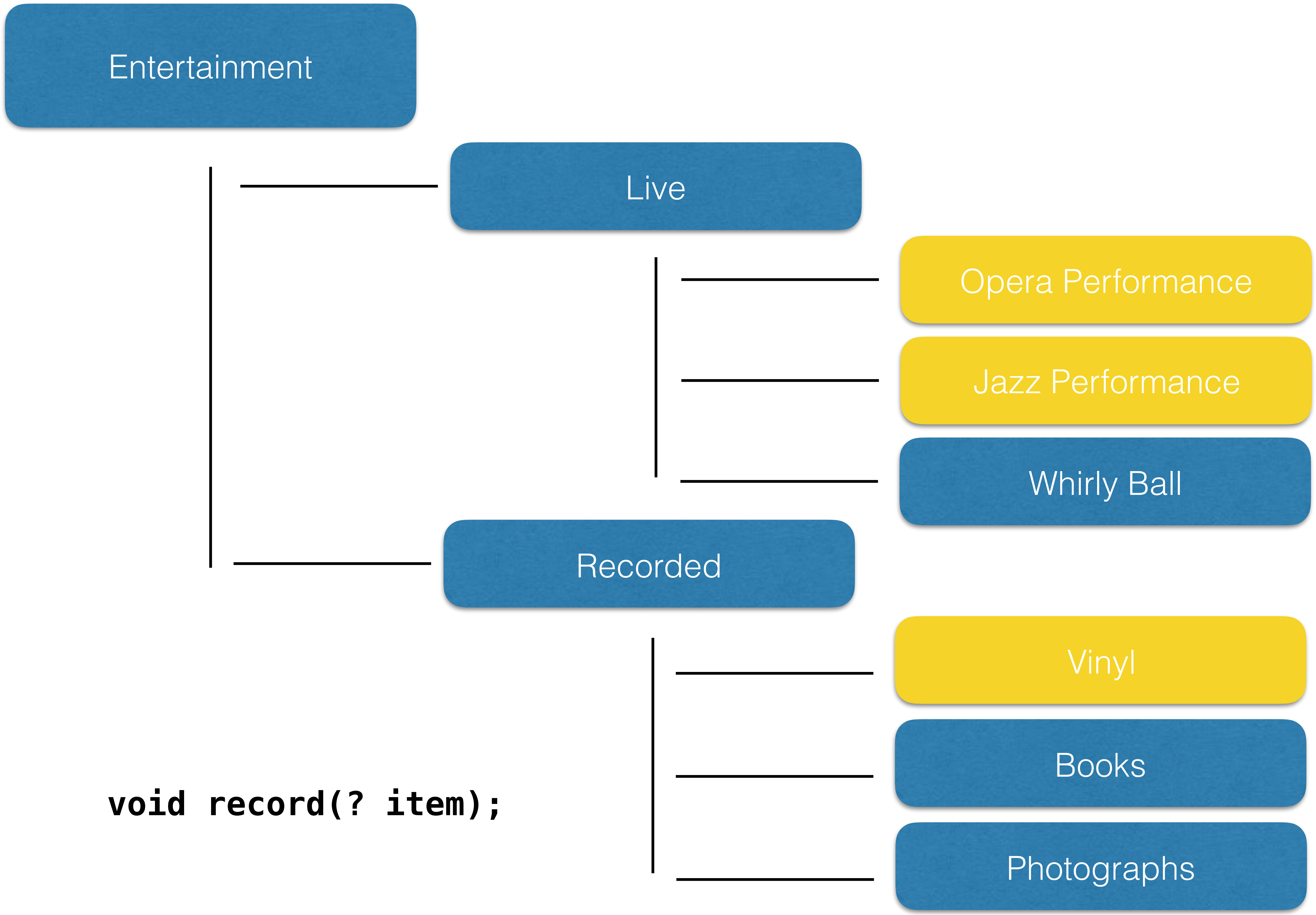
Problem

- Classes categorize data in a single, global way
- Multiple categorizations possible
- Redundancy, or insane hierarchies

TrickyTypes.md –>


Entertainment
Live
Opera Performance
Jazz Performance
Whirly Ball
Recorded
Records
Books
Photographs


Entertainment
Live
Opera Performance
Jazz Performance
Whirly Ball
Recorded
Records
Books
Photographs



```
void record(? item);
```

Java Solution

- Interfaces
- Hierarchical, but separate
- Same modifiers, same inheritance rules

Interfaces vs. Classes

- Classes for data
  - Animals
  - Music Categories
  - Movies
- Interfaces for Functionality
  - Pet-able
  - Danceable
  - Recordable

Interfaces.java –>

Interfaces Specifics

- Classes can implement multiple interfaces
    - ArrayList **implements** Serializable, Iterable, Collection, List
- Small, light
- Similar to Abstract Classes
    - Inheritance, default methods

Taxonomizing File Types

- /etc/mime.types
- Mime is a very simple taxonomy system
- Lets implement in code!