Royal University of Bhutan

Unit I

Introduction to Object-Oriented Programming (OOP) Concepts

Tutor: Pema Galey


O
O
P
Object
Oriented
Programming

Learning Outcomes

In this session, you will learn about:

- 2 paradigms to programming
- Object Oriented Programming
- Objects, class, interface
- Features of OOP
- Advantages of OOP
- Applications of OOP

2 Paradigms to Programming

- Conceptually organizing a program :
  - *Around code (Procedural Programming)*
  - *Around data (Object Oriented Programming)*

Object Oriented Programming

- **Basic Object-Oriented Programming Metaphor: Interacting Objects** — Any object oriented program is a collection of interacting objects that models a collection of real-world objects.

- OOP is a programming methodology that helps organize complex programs through the use of :
  - ***Inheritance***
  - ***Encapsulation***
  - ***Abstraction***
  - ***Polymorphism***

OOP Principles

Objects

- Like in the real world, an object can be anything. Eg:- dog
  - *An object has state, behavior and identity.*
- Meaning that an object can have
  - **data** (which gives it state- *attribute-value* pair),
  - **methods** (to produce behavior), and
  - each object can be **uniquely distinguished** from every other objects, i.e, each object has a unique address in memory.

- Objects are sometimes referred to as ***instances*** *of a class*

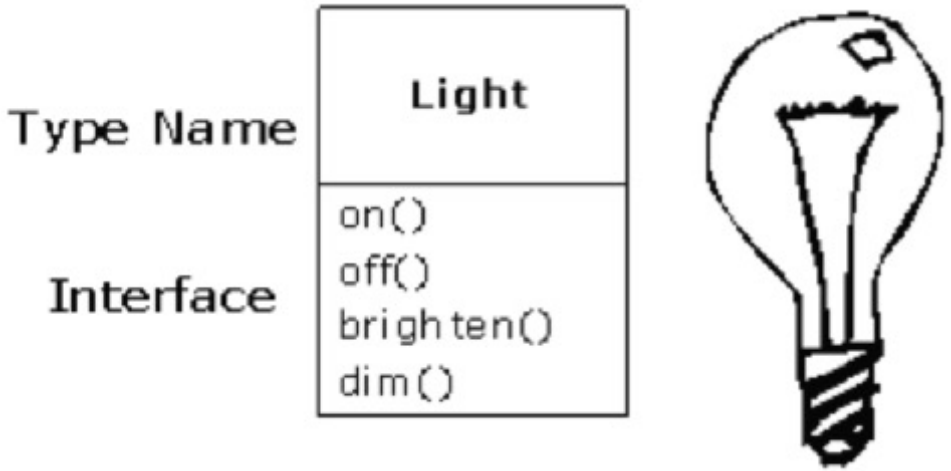
Class

- A class is an abstract data type.
- A *class* defines the structure and behavior (data and code/ functionality) that will be shared by a set of objects.
- Each object of a given class contains the structure and behavior defined by the class.
- You can create **variables** of a **type** (called *objects* or *instances*) and manipulate those variables (called *sending messages* or *requests;* you send a message and the object figures out what to do with it).

Interface

- But how do you get an object to do useful work for you? The requests you can make of an object are defined by its *interface*, and the type is what determines the interface.
- A simple example might be a representation of a light bulb:

```
Light lt = new Light();
lt.on();
```



Interface

- A type has a method associated with each possible request, and when you make a particular request to an object, that method is called.
- This process is usually summarized by saying that you “send a message” (make a request) to an object, and the object figures out what to do with that message (it executes code).
- Here, the name of the type/class is **Light**, the name of this particular **Light** object is **lt**, and the requests that you can make of a **Light** object are to turn it on, turn it off, make it brighter, or make it dimmer.
- You create a **Light** object by defining a “reference” (**lt**) for that object and calling **new** to request a new object of that type.
- To send a message to the object, you state the name of the object and connect it to the message request with a period (dot).

Features of OOP

- All object-oriented programming languages provide mechanisms that help you implement the object-oriented model:
  1. **Abstraction**
  2. **Encapsulation**
  3. **Inheritance and**
  4. **Polymorphism**

Abstraction

- Abstraction is ***the process of reducing an object to its essence so that only the necessary elements are represented***, in which the user is interested.
- Humans *manage complexity* through abstraction. Meaning, it deals with the outside view of an object. Eg. Cars, computers.
- The objects we design in our Java programs will be abstractions because they ignore many of the attributes that characterize the real objects and focus only on those attributes that are essential for solving a particular problem.

Encapsulation

- Encapsulation is **the process of providing relevant information of an object & hiding the non-essential information.** *AKA information hiding.*
- The mechanism that binds together code and the data it manipulates, and keeps both safe from outside interference and misuse.
- The power of encapsulated code is that everyone knows how to access it and thus can use it regardless of the implementation details—and without fear of unexpected side effects.

Encapsulation

- In Java, the basis of encapsulation is the **class**.
- Since the purpose of a class is to encapsulate complexity, there are mechanisms for hiding the complexity of the implementation inside the class.
- Each method or variable in a class may be marked private or public.

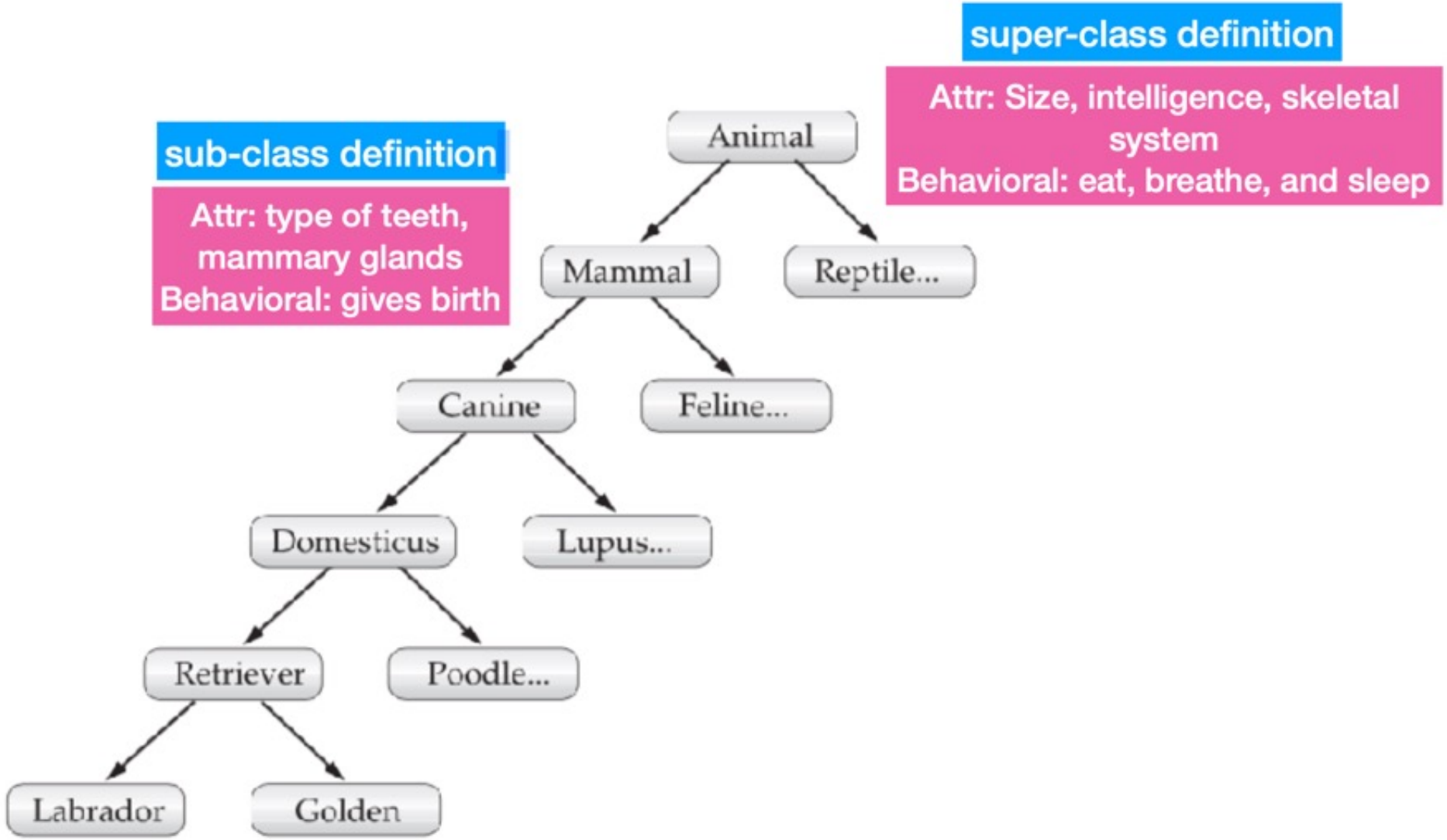
Access Modifiers

- A*ccess modifiers* determine who can use the definitions that follow. Java uses three explicit keywords to set the boundaries in a class:
  - **public**, **private**, and **protected**.
- **public:** the following element is available to everyone
- **private:** no one can access that element except the class creator, inside methods of that type/class. Someone who tries to access a private member will get a compile-time error.
- **protected:** acts like private, with the exception that an inheriting class has access to **protected** members, but not **private** members.
- **Default:** AKA *package access* because classes can access the members of other classes in the same *package* (library component), but outside of the package those same members appear to be **private**.

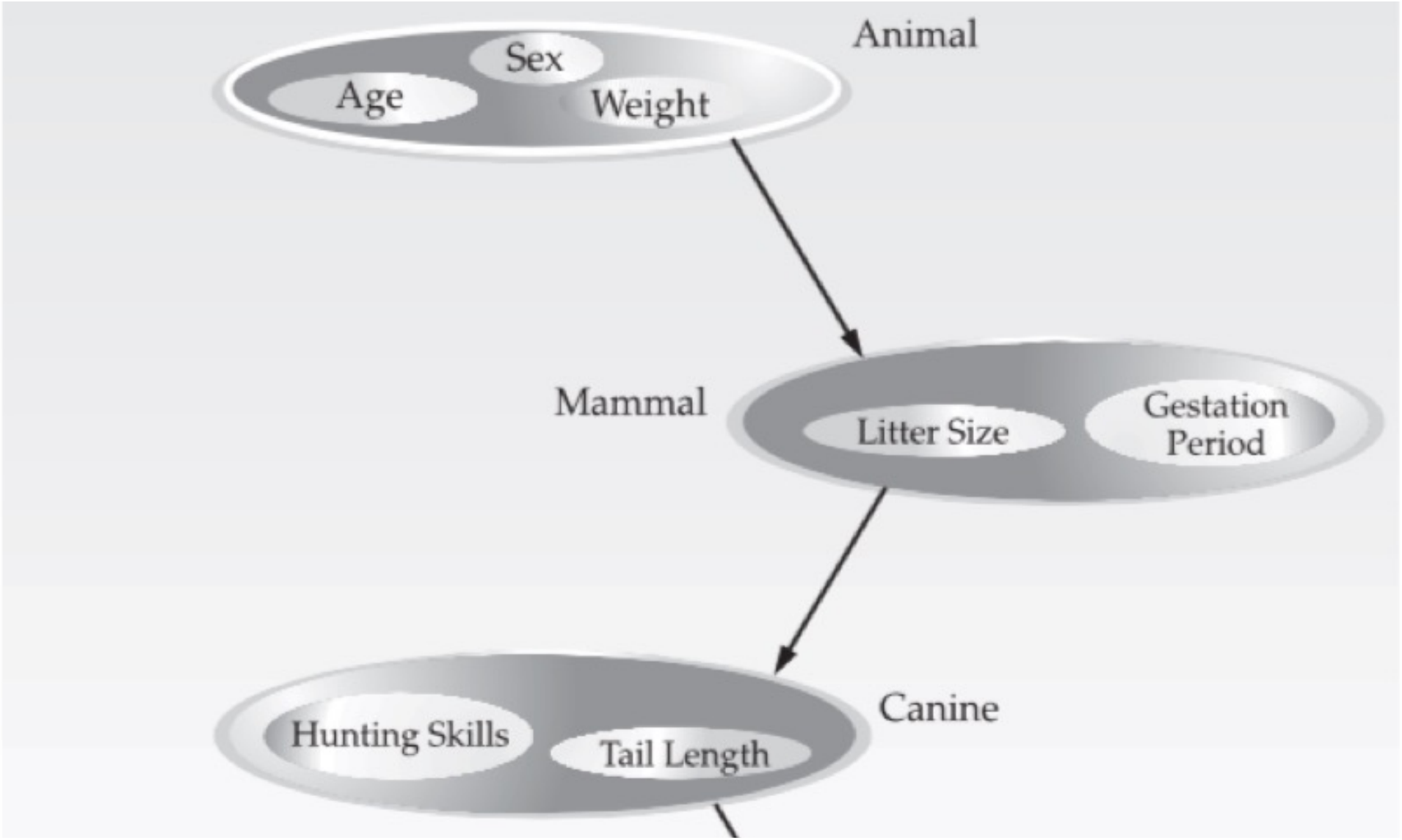
Inheritance

- *Inheritance* is **the process by which one object acquires the properties of another object thereby supporting the concept of hierarchical classification.**
- Each object can inherit its general attributes from its parent.
- But it need only define those qualities that make it unique within its class.
- Eg- Animals > Mammals > Dogs [subclass & superclass]

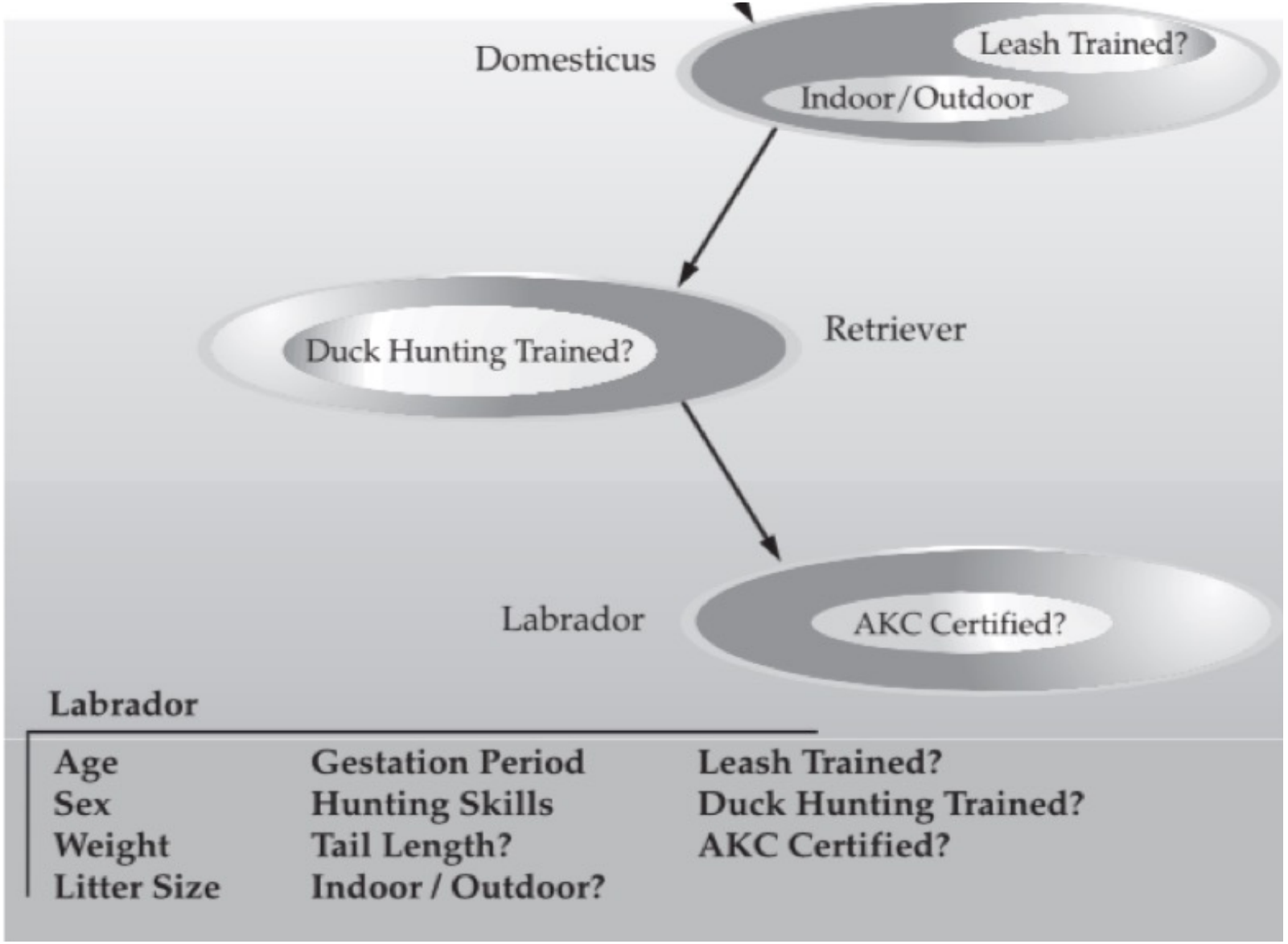
Inheritance



Inheritance



Inheritance



Polymorphism

- from Greek, meaning “many forms”
- Polymorphism describes **the ability of an object to exist in various forms with a unique and specific behavior, each time.**
- “one interface, multiple methods.”
- This means that it is possible to design a generic interface to a group of related activities.
- Dog’s nose-> smells cat ; smells food
- Man-> son, father, husband depending upon the relation at hand.

Advantages of OOP

- **Realistic modeling**: OOP models the real world objects more accurately
- **Reusability**: Object-oriented languages divide programs into separate classes which can be reused. (a class ideally represents a useful unit of code)
- **Resilience to change:** Systems can be allowed to evolve without re-building it from scratch.

Applications of OOP

OOP is used extensively in the following application areas (but not limited):

1. Multiple data structures
2. Data in multiple programs
3. Parallel programming
4. Simulation and modeling
5. Artificial Intelligence (AI)
6. Expert systems
7. Neural networks and
8. Computer-aided Design (CAD) systems.

Summary

- What is OOP? - a programming paradigm that models real world objects using abstraction, encapsulation, inheritance and polymorphism.
- **Writing an object-oriented program** is largely a matter of **designing classes** and writing definitions for those classes in Java.
- **Designing a class** is a matter of specifying all of the **attributes** and **behaviors** that are characteristic of that type of object.

Thank you!