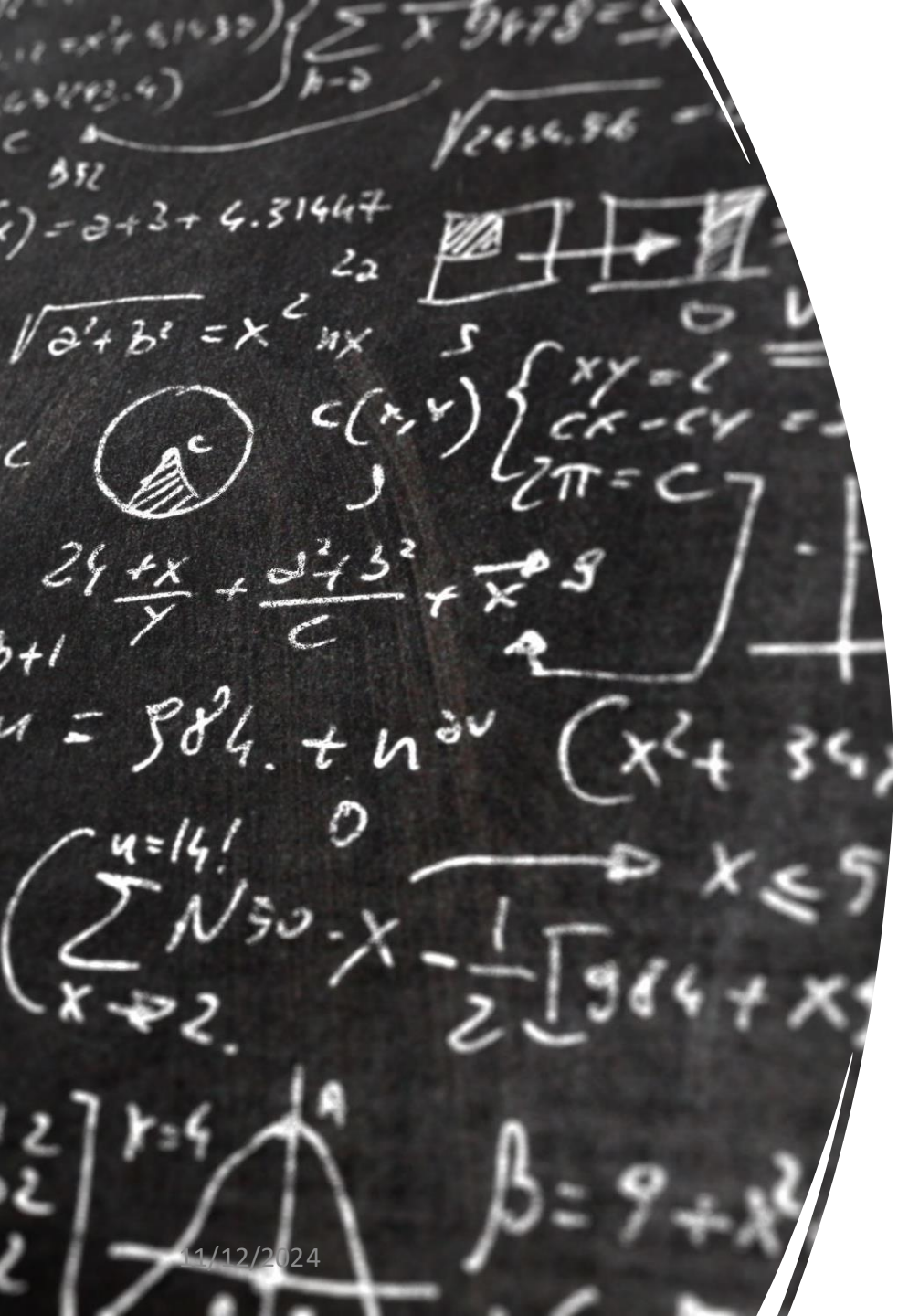


CS 113 – Computer Science I

Lecture 17 – Relationships

Tuesday 11/12/2024

Announcements

HW 07 – Due ~~Monday 11/11~~ tonight

Board game

longer one

Lab06 and Lab07 are relevant

Mid-semester feedback form: https://forms.gle/Ed7G9oe74QQBT5sy5

Midterm 2: Thursday December 5th

Designing Classes

What properties does a bird have and what can it do?

- Size, color, feathers, fly

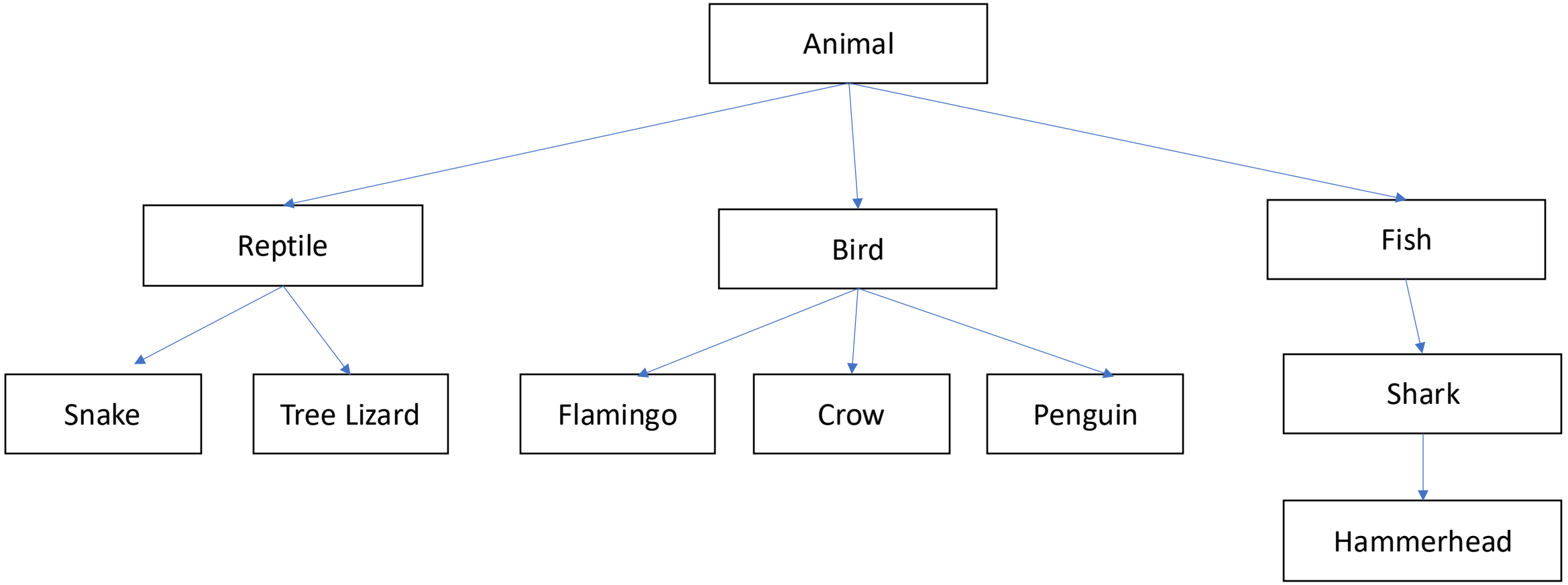
What properties does a lion have and what can it do?

- Size, color, hair, runs

What properties does a kangaroo have and what can it do?

- Size, color, arms, jumps

Inheritance: feature for organizing classes into hierarchies

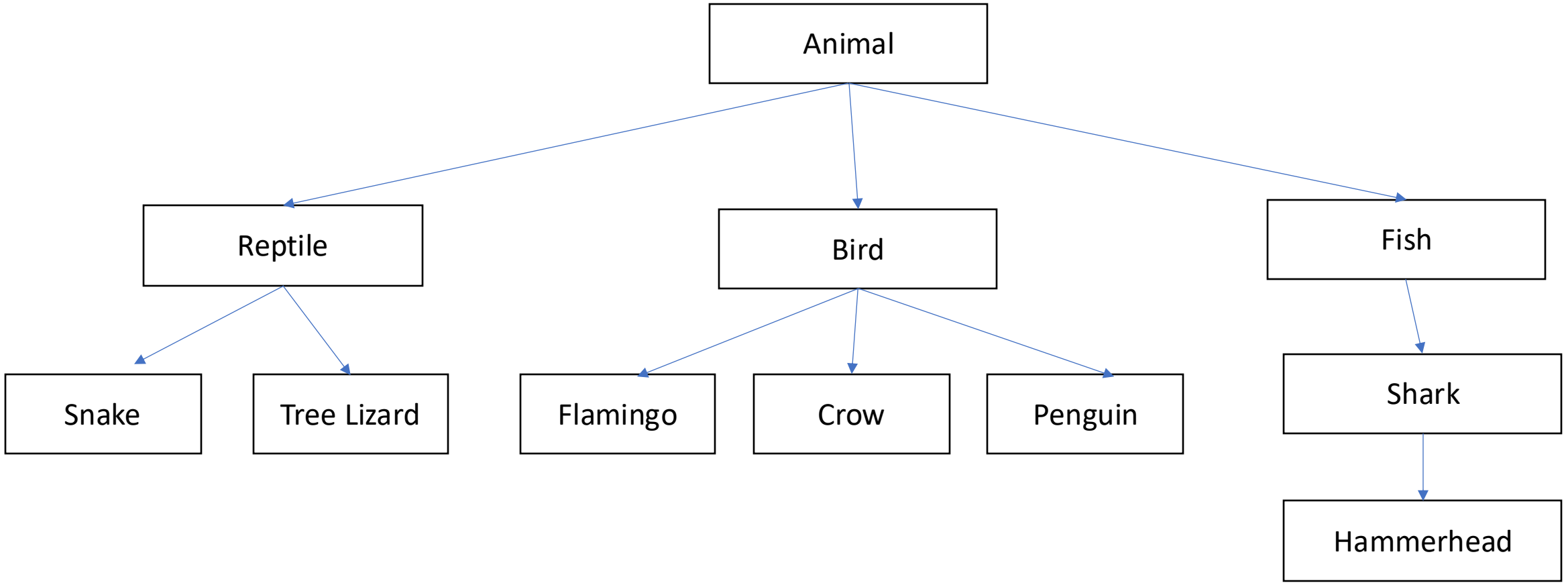


Class inheritance

Classes can be arranged hierarchically where,

a child class ”inherits” from a parent class

Inheritance: feature for organizing classes into hierarchies



Inheritance: subclasses refine behavior/state

Subclasses can override methods from parent class

Exercise

1. Implement getter functions for instance variables inside Animal

2. In Zoo.java, call the getters and output the values to console

Polymorphism

Program can treat all objects that extend a base class the same

Java automatically calls the specific methods for each subclass

Polymorphism: Demo

```
public class Zoo {
   public static void main(String[] args) {
      Animal animal1 = new Animal();
      animal1.locomote();

      Animal animal2 = new Reptile();
      animal2.locomote();
   }
}
```

```
public class Animal {
   public Animal() {
   }
   public void locomote() {
      System.out.println("I am moving!");
   }
}
```

```
public class Reptile extends Animal {
   public Reptile() {
   }
   public void locomote() {
      System.out.println("I am walking!");
   }
}
```

Exercise: What is the output of this program?

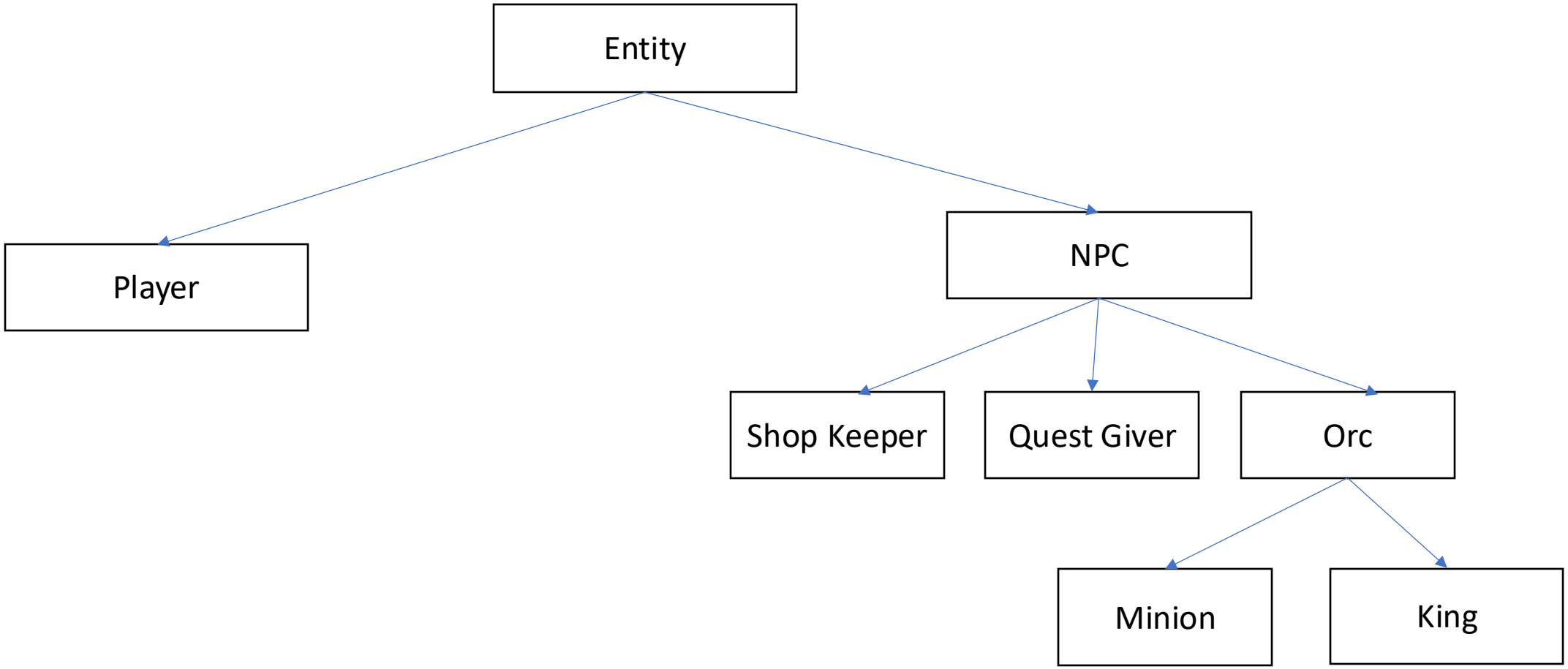
```
public class Zoo {
   public static void main(String[] args) {
      Animal animal1 = new Animal();
      animal1.locomote();

      Animal animal2 = new Fish();
      animal2.locomote();
   }
}
```

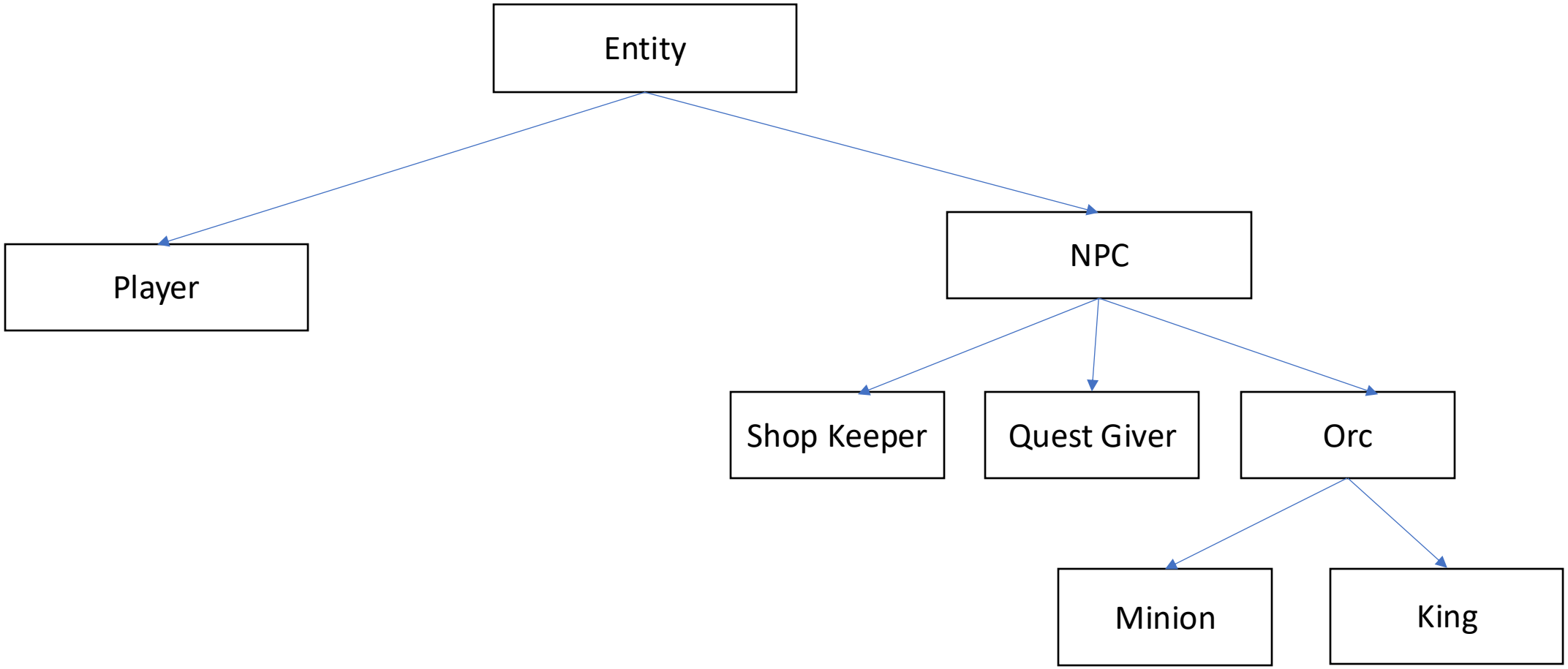
```
public class Animal {
   public Animal() {
   }
   public void locomote() {
      System.out.println("I am moving!");
   }
}
```

```
public class Fish extends Animal {
  public Fish() {
  }
  public void locomote() {
    System.out.println("I am swimming!");
  }
}
```

Question: How would we implement Minion?



Inheritance



Inheritance: subclasses refine behavior/state

Subclasses can override methods from parent class

```
class Animal {

    public Animal(String name, boolean hasHair,
                  int numberLegs, boolean swimable) {
        this.hasHair = hasHair;
        this.numberLegs = numberLegs;
        this.name = name;
        this.swimable = swimable;
    }
```

```
public class Fish extends Animal {

    public Fish(String name, boolean hasHair,
                int numLegs, boolean swimable) {
        this.name = name;
        this.hasHair = hasHair;
        this.numberLegs = numLegs;
        this.swimable = swimable;
    }
```

Inheritance: constructors - super();

super();

reference variable that is used to refer parent class constructor

Inheritance: subclasses refine behavior/state

Subclasses can override methods from parent class

```
class Animal {

    public Animal(String name, boolean hasHair,
                  int numberLegs, boolean swimable) {
        this.hasHair = hasHair;
        this.numberLegs = numberLegs;
        this.name = name;
        this.swimable = swimable;
    }
```

```
public class Fish extends Animal {

    public Fish(String name, boolean hasHair,
                int numLegs, boolean swimable) {
        this.name = name;
        this.hasHair = hasHair;
        this.numberLegs = numLegs;
        this.swimable = swimable;
    }
```

Inheritance: constructors - super();

```
class Animal {

    public Animal(String name, boolean hasHair,
                  int numberLegs, boolean swimable) {
        this.hasHair = hasHair;
        this.numberLegs = numberLegs;
        this.name = name;
        this.swimable = swimable;
    }
```

```
public class Fish extends Animal {

    public Fish(String name, boolean hasHair,
                int numLegs, boolean swimable) {
        this.name = name;
        this.hasHair = hasHair;
        this.numberLegs = numLegs;
        this.swimable = swimable;
    }
```

```
public class Fish extends Animal {

    public Fish(String name, boolean hasHair,
                int numLegs, boolean swimable) {
        super();
    }
```

Inheritance: constructors - super();

super();

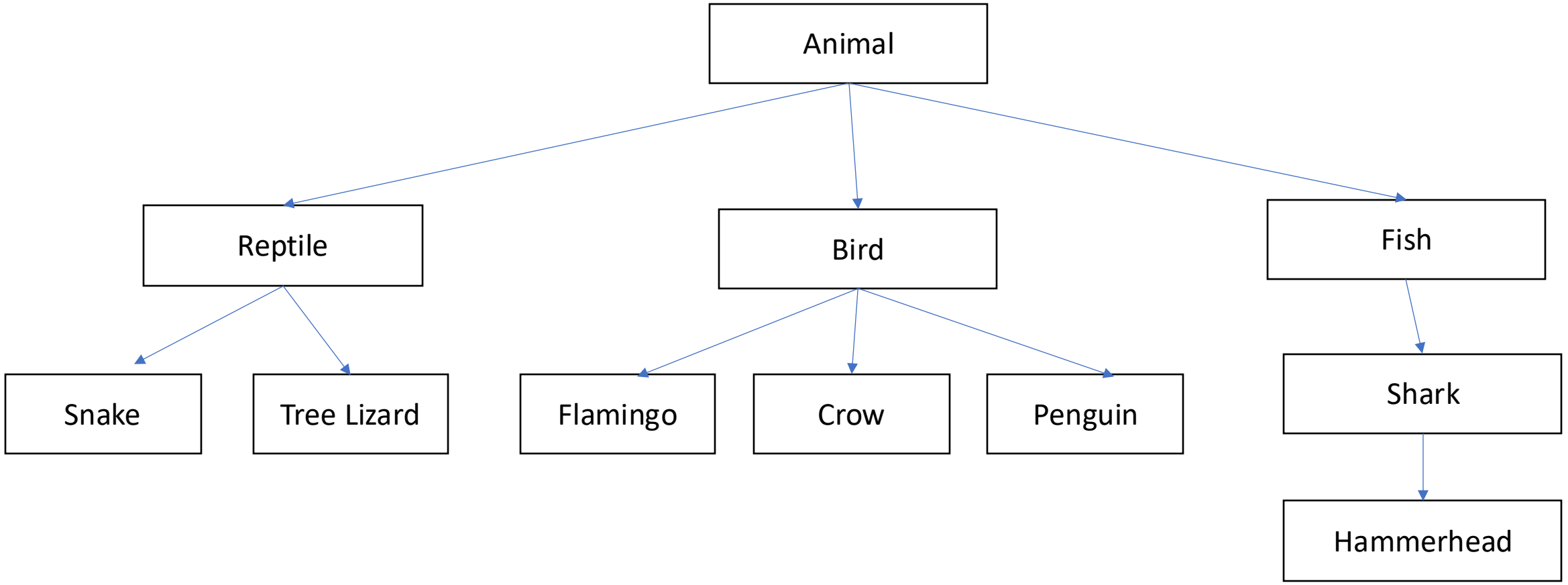
reference variable that is used to refer parent class constructors

Note:

super:

reference variable that is used to refer parent class object

Inheritance: feature for organizing classes into hierarchies



interfaces

A common set of methods that each implementing class must include (like a blueprint)

Contract for a class to implement a certain set of methods

Implementing class *inherits* a list of functions from the interface

methods in an interface are abstract
- declared method without an implementation
- contains just method signature

Define an interface using the interface keyword

Implementing an interface

1. Use implements keyword instead of extends (demo)

2. Implement the functions

Inheritance vs Extends

Interfaces (subtyping)

- implements
- Guarantees same types have same functions
  - Though the same functions are implemented differently
- A class can implement multiple interfaces
- An interface can extend another interface

Inheritance (subclassing)

- extends
- Reuses implementations
- Consequences:
  - Dependent on base class
  - Changes in superclass affects all subclasses
  - Can re-use code inside classes
- A class can extend just one parent class