CS 113 – Computer Science I

Lecture 18 – Class Design & Relationships

Tuesday 11/07/2023

Announcements

HW07 – Due Monday 11/13

      Board game

      longer one

      Lab06 and Lab07 are relevant

HW08 – Due Monday 11/20

      Class design

Mid-semester feedback survey

Midterm 2

Tuesday 11/21

Material:

      Midterm 1 material

      Loops

      Classes & OOP

Do we want to move the midterm to be after Thanksgiving?

Outline

- Review
- Access modifiers
- Inheritance

Class

A blueprint for a custom data type

A template for how data/information is stored

Contains a set of methods for how to interact/operate on the stored data

Using objects: some special methods

The **constructor method** is called when you do a `new`

accesors (aka getters)
> return the values of instance variables

mutators (aka setters)
> set the values of instance variables

toString()
> returns a string representation of an object

this

`this` is a special keyword that refers to the object inside an instance method

Allows us to access other instance variables within an instance method

Access modifiers

Specify the access-level of instance variables/methods

- `public`
  - code outside of the class can access the variable/method

- `private`
  - code outside of the class cannot access the variable/method

Default in java is `public`

In this class, make instance data private

Designing Classes

What properties does a bird have and what can it do?
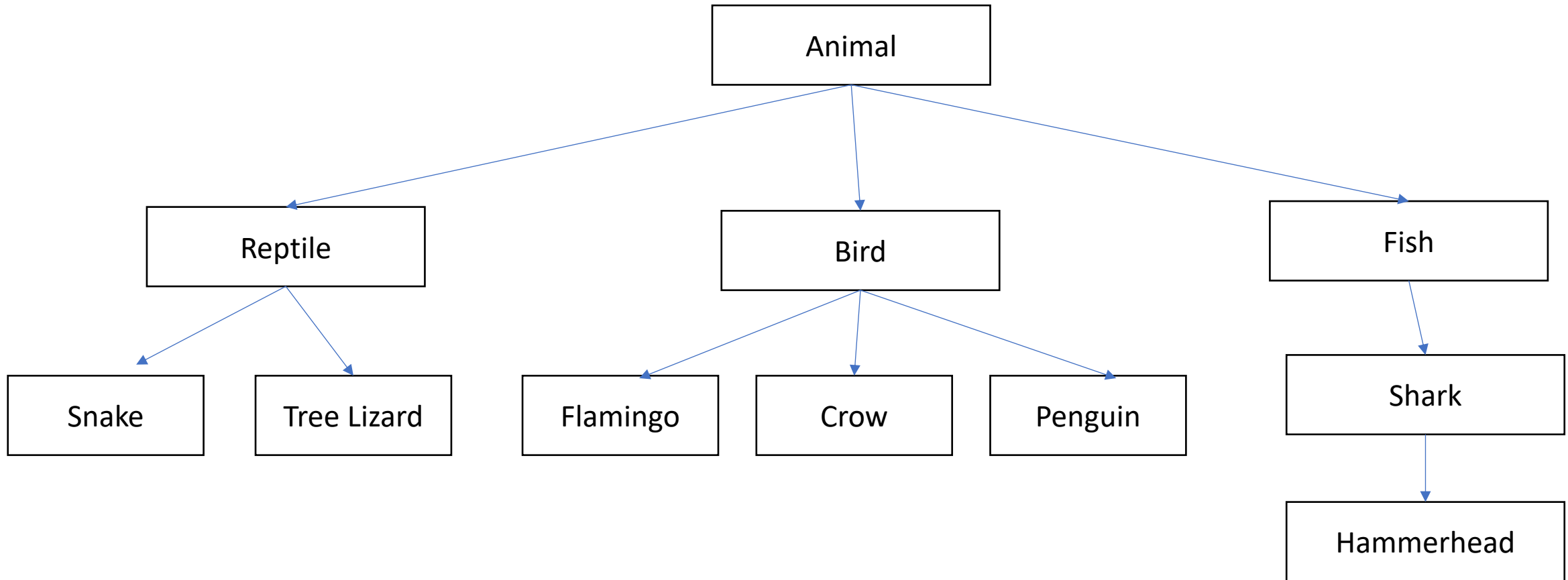
• Size, color, feathers, fly

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?
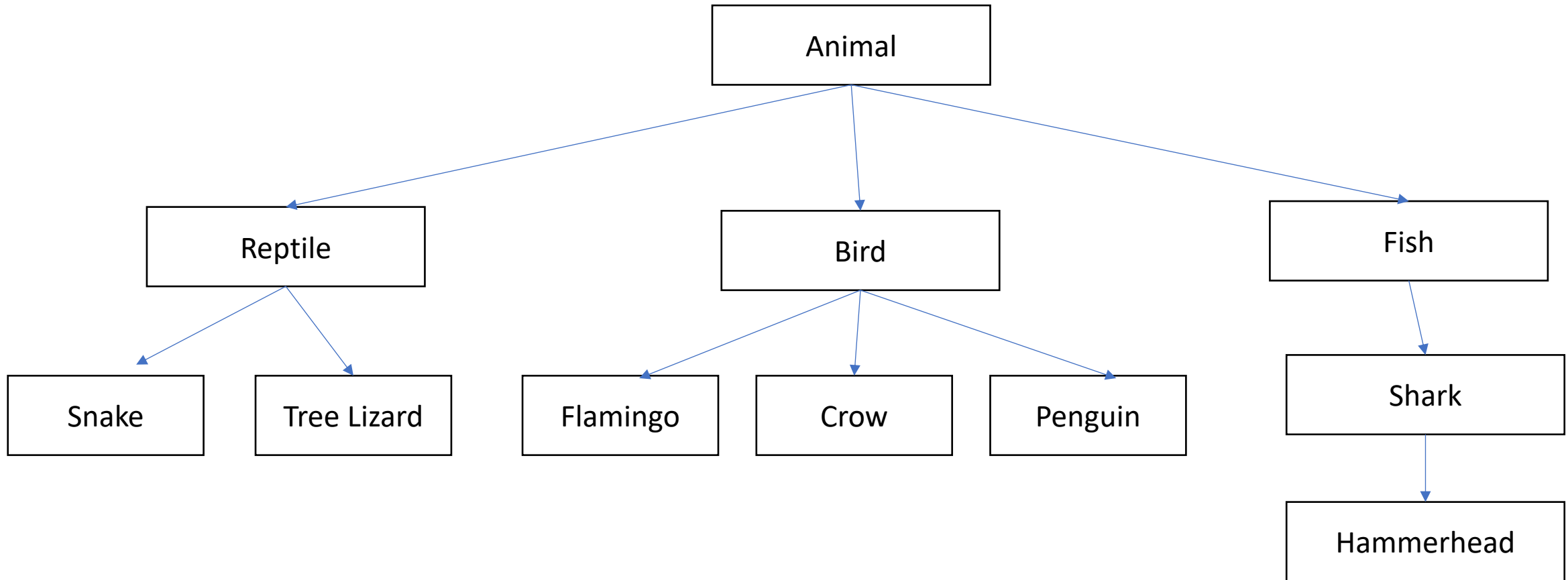
• Size, color, arms, jumps

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

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

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

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

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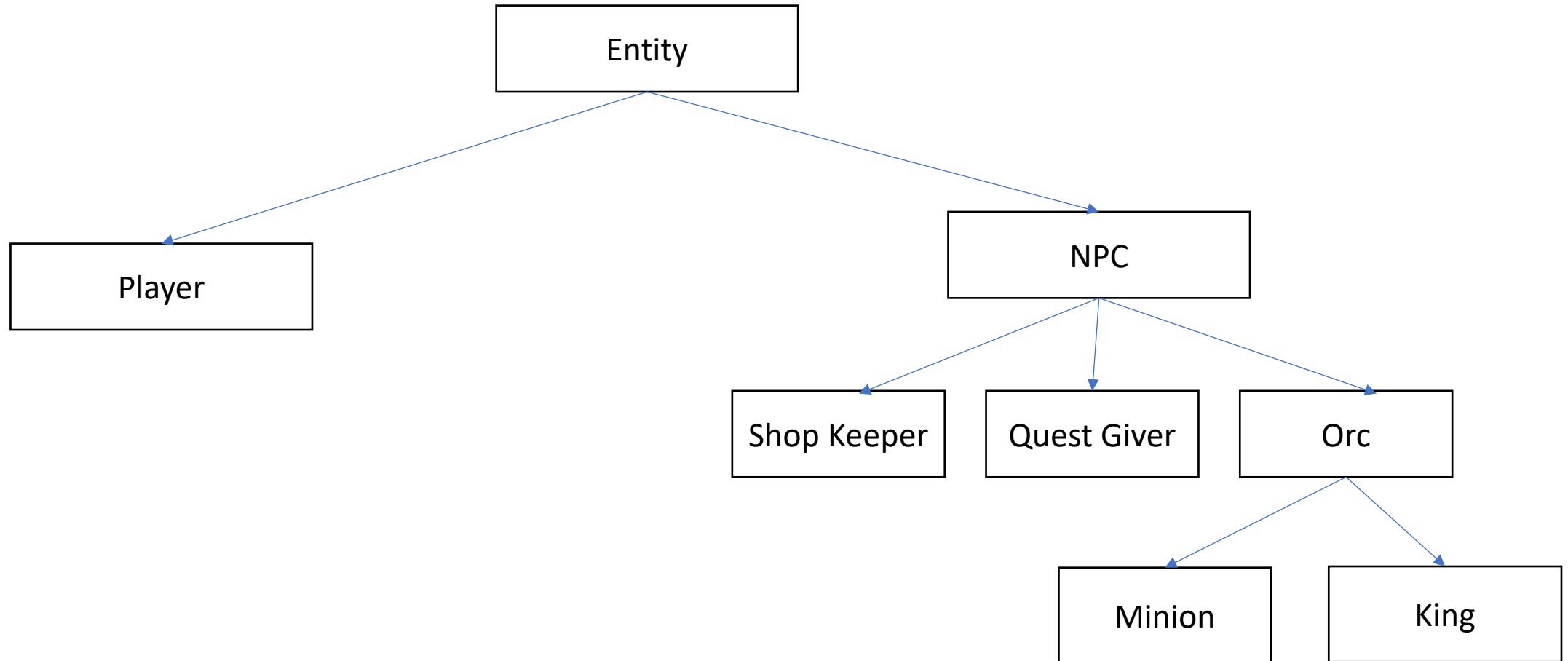
```
public class Animal {
    public Animal() {
    }
    public void locomote() {
        System.out.println("I am moving!");
    }
}
```

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

Question: How would we implement Minion?
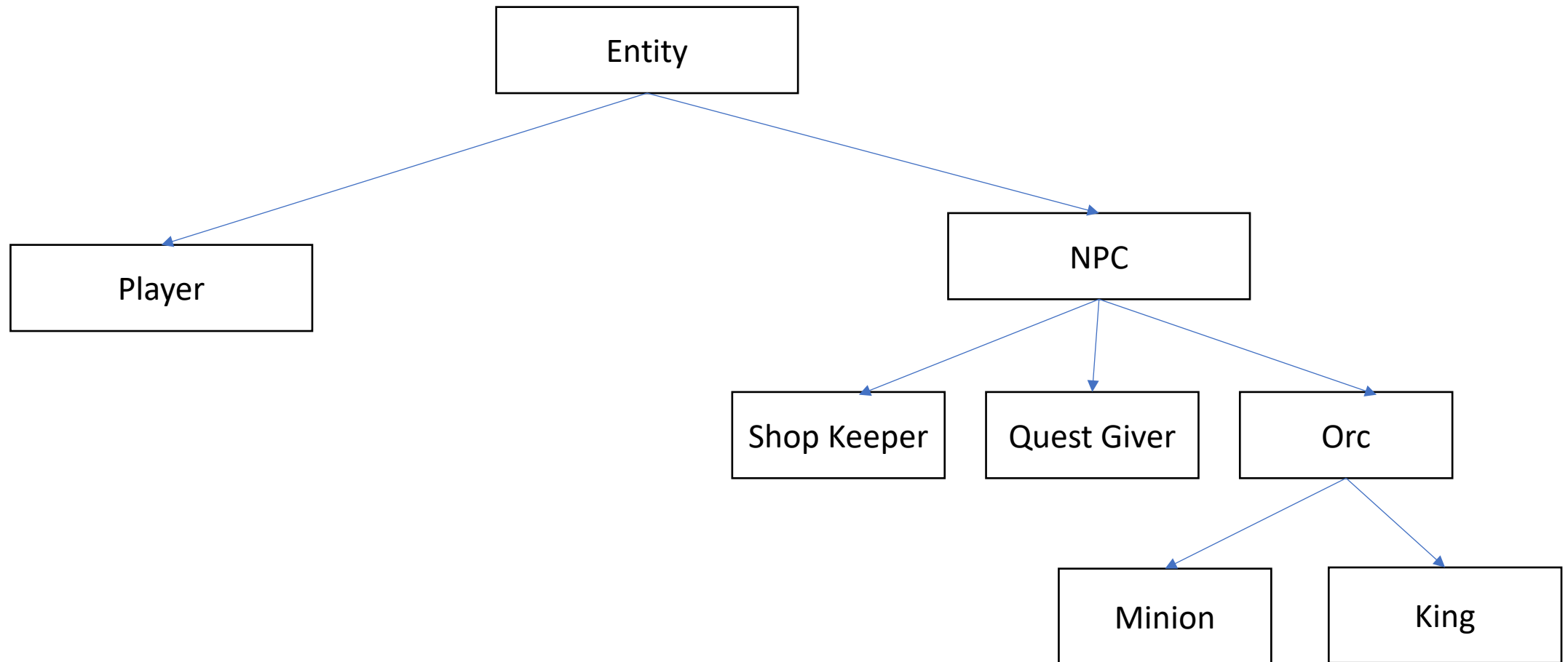
Inheritance

Inheritance: subclasses refine behavior/state

Subclasses can override methods from parent class

```java
class Animal {

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

```java
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

```java
class Animal {

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

```java
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();`

```java
class Animal {

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

}
```

```java
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;
    }

}
```

```java
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