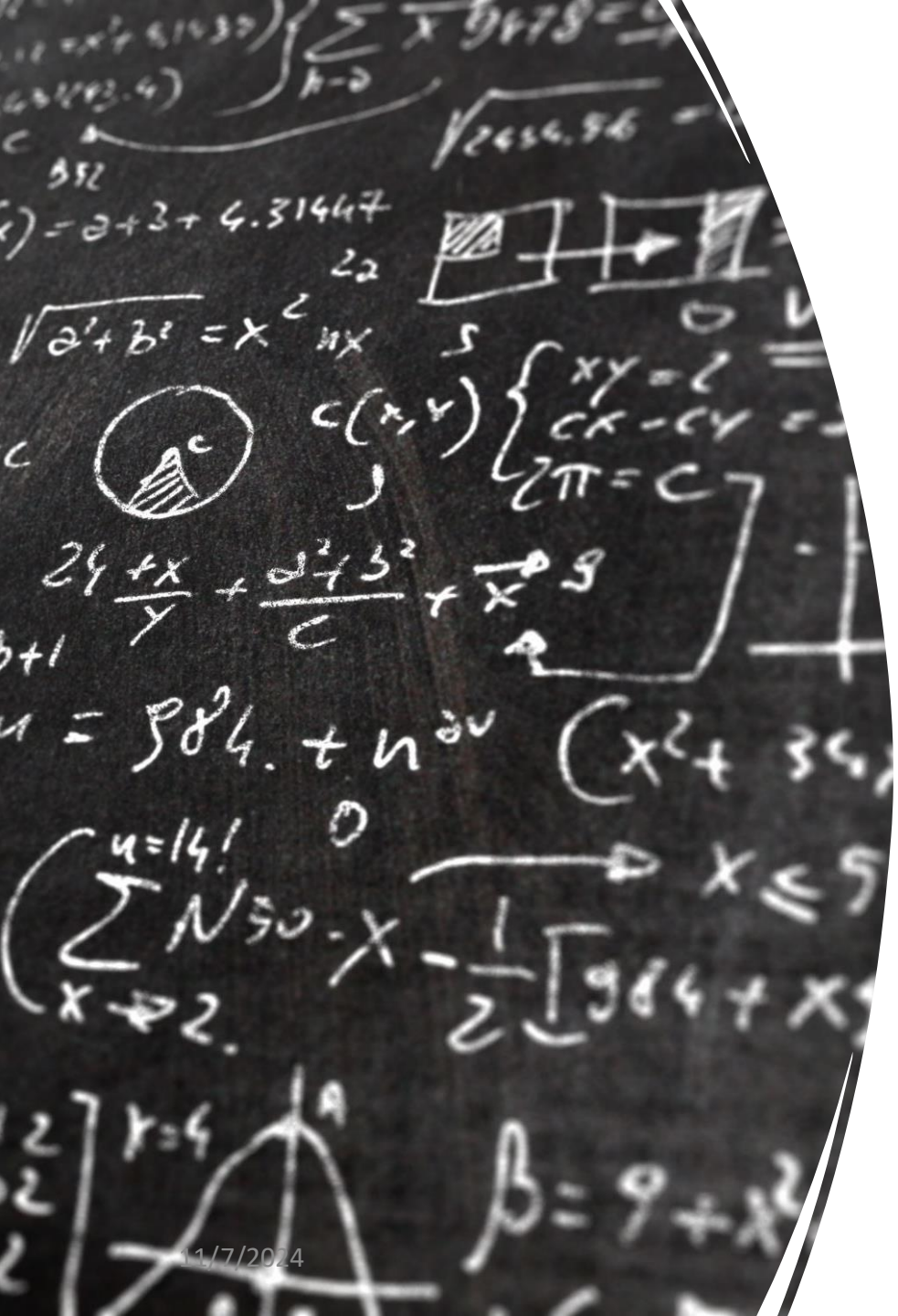


CS 113 – Computer Science I

Lecture 16 – Class Design & Relationships

Tuesday 11/07/2024

Announcements

HW 07 – Due Monday 11/11

- Board game
- longer one
- Lab06 and Lab07 are relevant

Midterm 2: Moving it to Thursday December 5th

Agenda

- Objects & Classes
- Arrays of Objects

Class

A blueprint for a custom data type

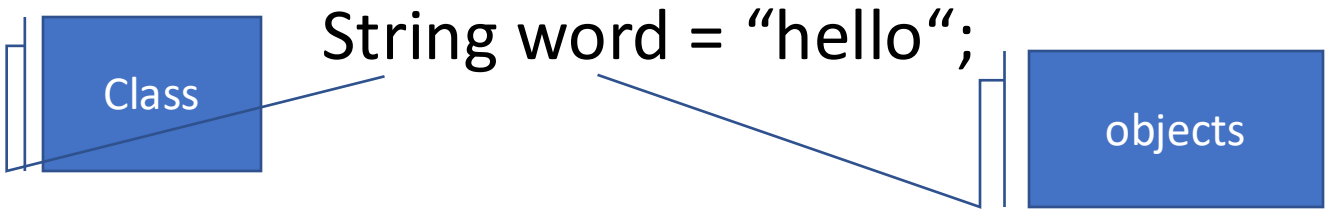
A template for how data/information is stored

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

Classes and objects

A **class** defines the characteristics of a type (data and methods)

An **object** is a particular example of a class



Java is a strict object-oriented programming language, meaning all code must be inside a class!

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

Defining classes

By defining our own classes, we can create our own data types

A class definition contains

- the data contained by the new type (**instance variables**)

- the operations supported by the new type (**instance methods**)

Object-oriented programming (OOP)

Method for designing programs in terms of objects

Recall: Top-down design

- the “nouns” in your feature list correspond to classes/data
- the “verbs” correspond to methods

Bank Actions

How can we find out how much money the bank is holding at once?

How can we find out which account is currently overdraft?

What other questions might the bank want to know?

Exercise: Bank Account

BankAccount should have the following data:

- Name
- Amount

BankAccount should have the following operations:

- currentBalance() // returns current amount in the bank account
- withdraw(float amt) // withdraw the given amount from the account
- deposit(float amt) // deposit the given amount to the account

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

Revisiting the Bank class

```
public class Bank {
    int size;
    String name;
    String[] clients;
    double[] accounts;

    public Bank(String bankName, int numClients) {
        name = bankName;
        size = numClients;
        clients = new String[size];
        accounts = new double[size];

    }
    public String getName() {
        return name;
    }

}
```

Revisiting the Bank class

```
public class Bank {
    int size;
    String name;
    BankAccount[] accounts;

    public Bank(String bankName, int numClients) {
        name = bankName;
        size = numClients;
        accounts = new BankAccount[size];

    }
    public String getName() {
        return name;
    }

}
```

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

Access modifiers

Default in java is public

In this class, make instance data private

Revisiting the Bank class

```
public class Bank {
    private int size;
    private String name;
    private BankAccount[] accounts;

    public Bank(String bankName, int numClients) {
        name = bankName;
        size = numClients;
        accounts = new BankAccount[size];

    }
    public String getName() {
        return name;
    }

}
```

OOP Example & Design: Vending machine

OOP Example: Snack

Name

Cost

Quantity

Defining the snack class

```
public class Snack {
    private int mQuantity;
    private double mCost;
    private String mName;

    public Snack(String name, int quantity, double cost) {
        mQuantity = quantity;
        mCost = cost;
        mName = name;
    }
    public String getName() {
        return mName;
    }

    public void buy() {
       if (mQuantity > 0) {
           mQuantity--;
       }
    }
}
```

Testing the Snack class

```
public static void main(String args[])
{
    Snack snack = new Snack("Slurm", 10, 1.5);
    System.out.println("Snack: "+snack.getName());
}
```

Vending Machine Class

Vending machines have a list of snacks

Designing Classes

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

- Size, color, feathers, fly

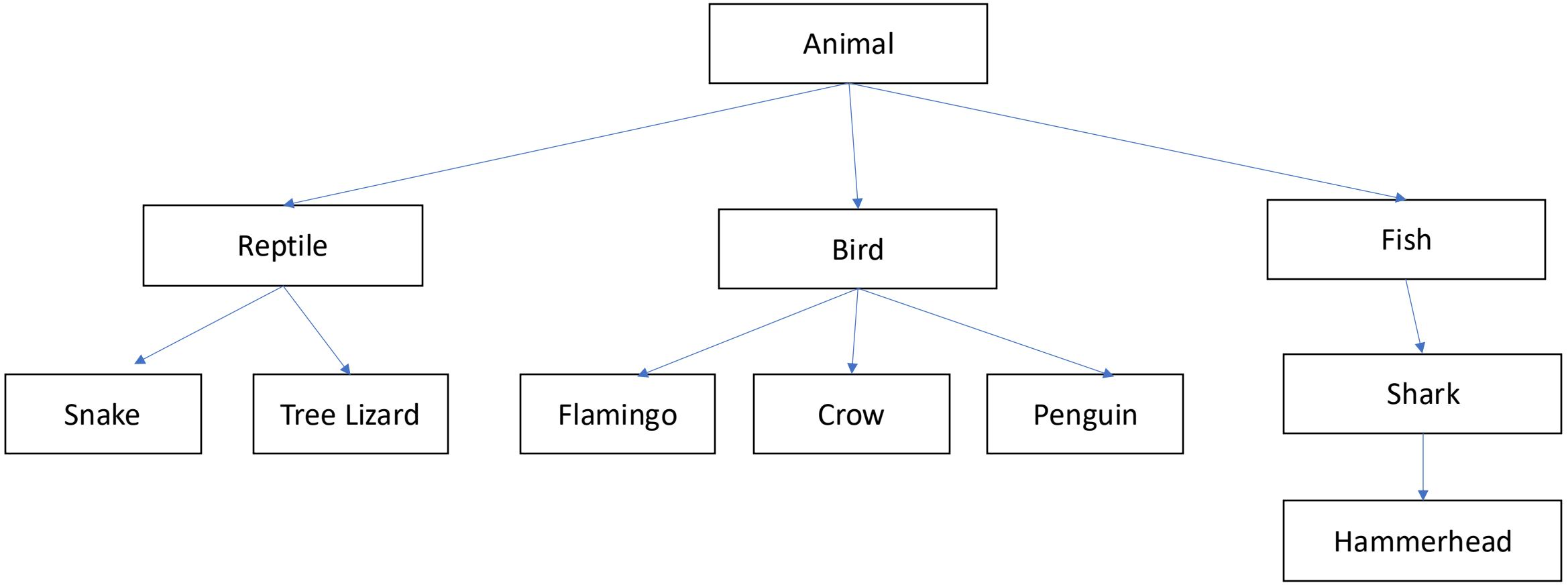
What properties does a lion have and what can it do?

- 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

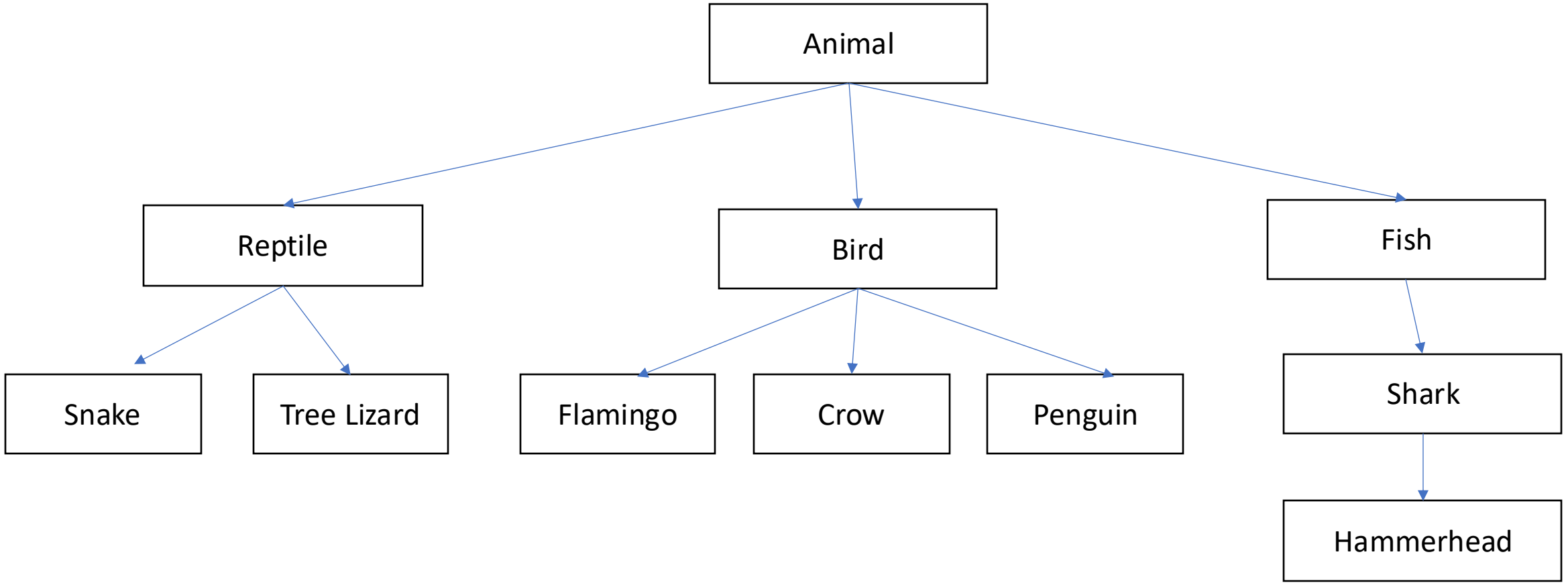


Class inheritance

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?

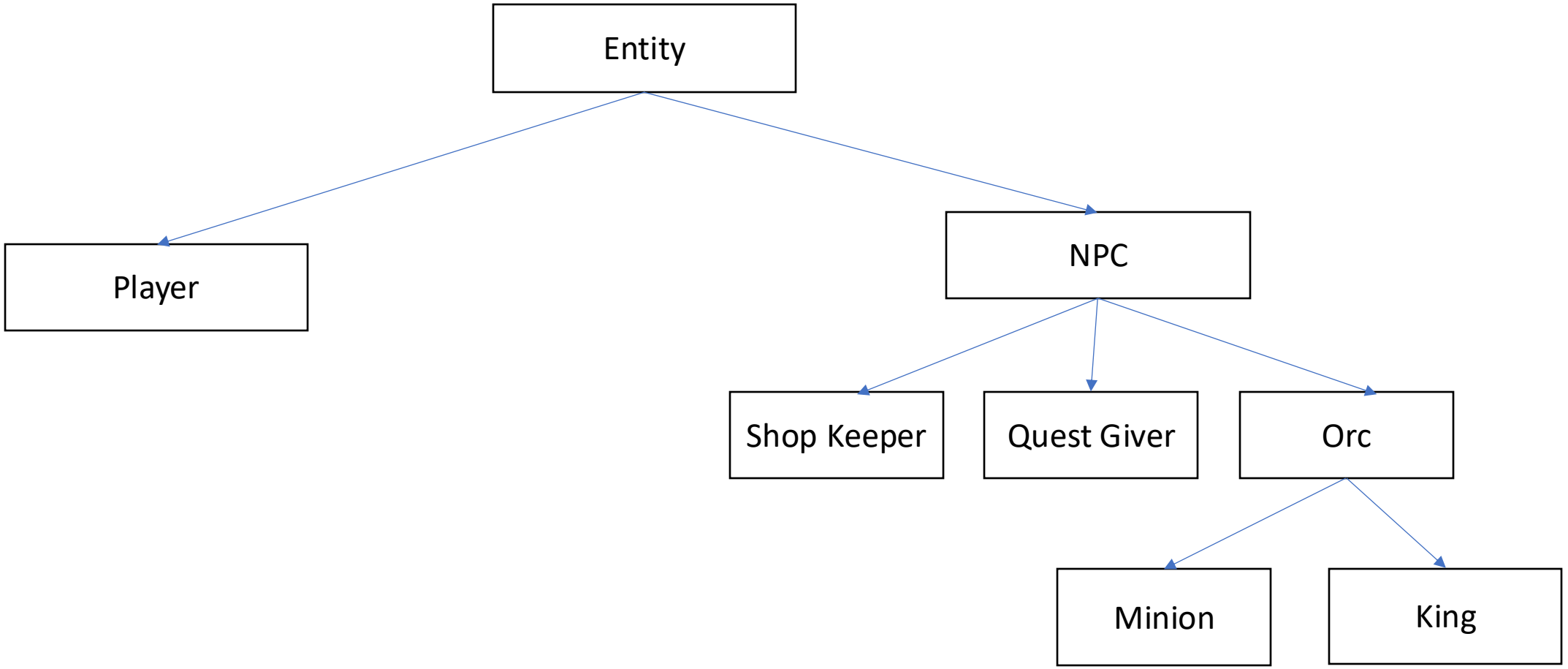
```
public class Zoo {
   public static void main(String[] args) {
      Animal animal1 = new Animal();
      animal1.locomote();

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

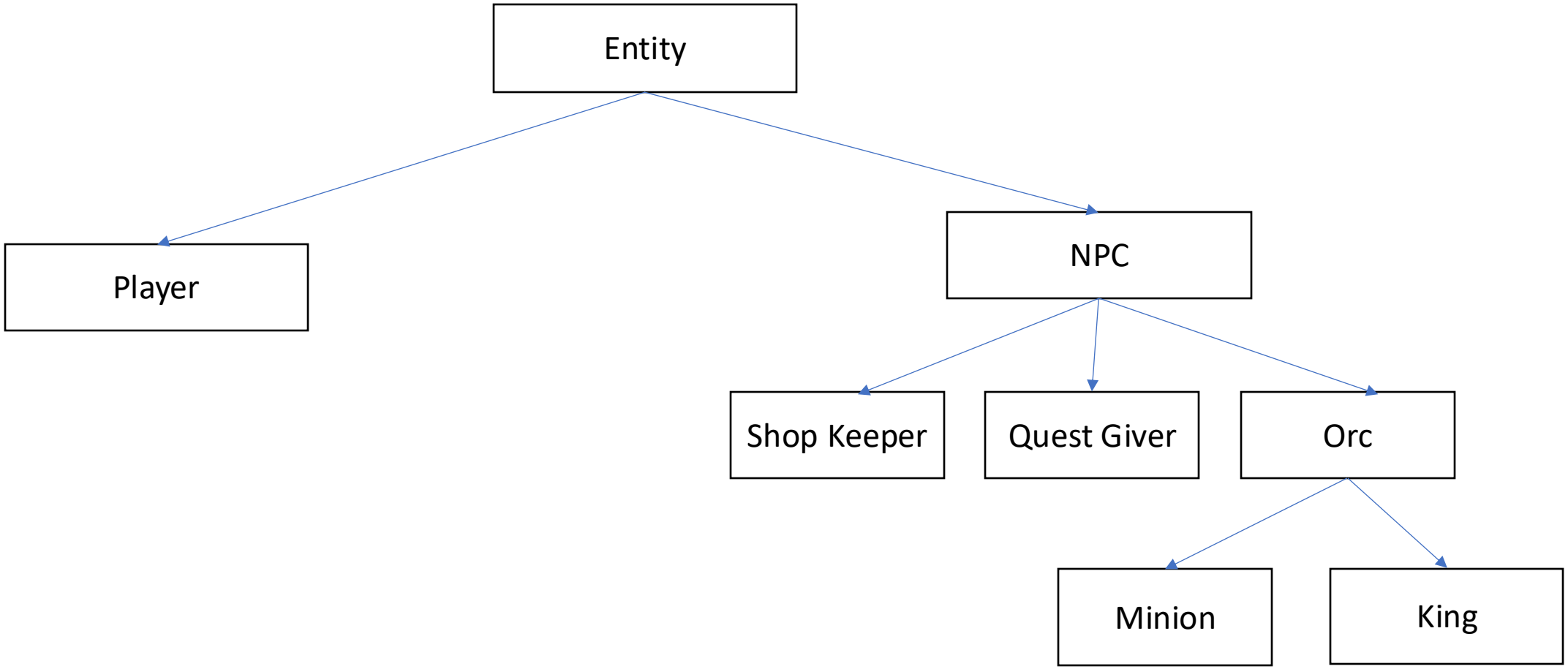
```
public class Animal {
   public Animal() {
   }
   public void locomote() {
      System.out.println("I am moving!");
   }
}
```

```
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