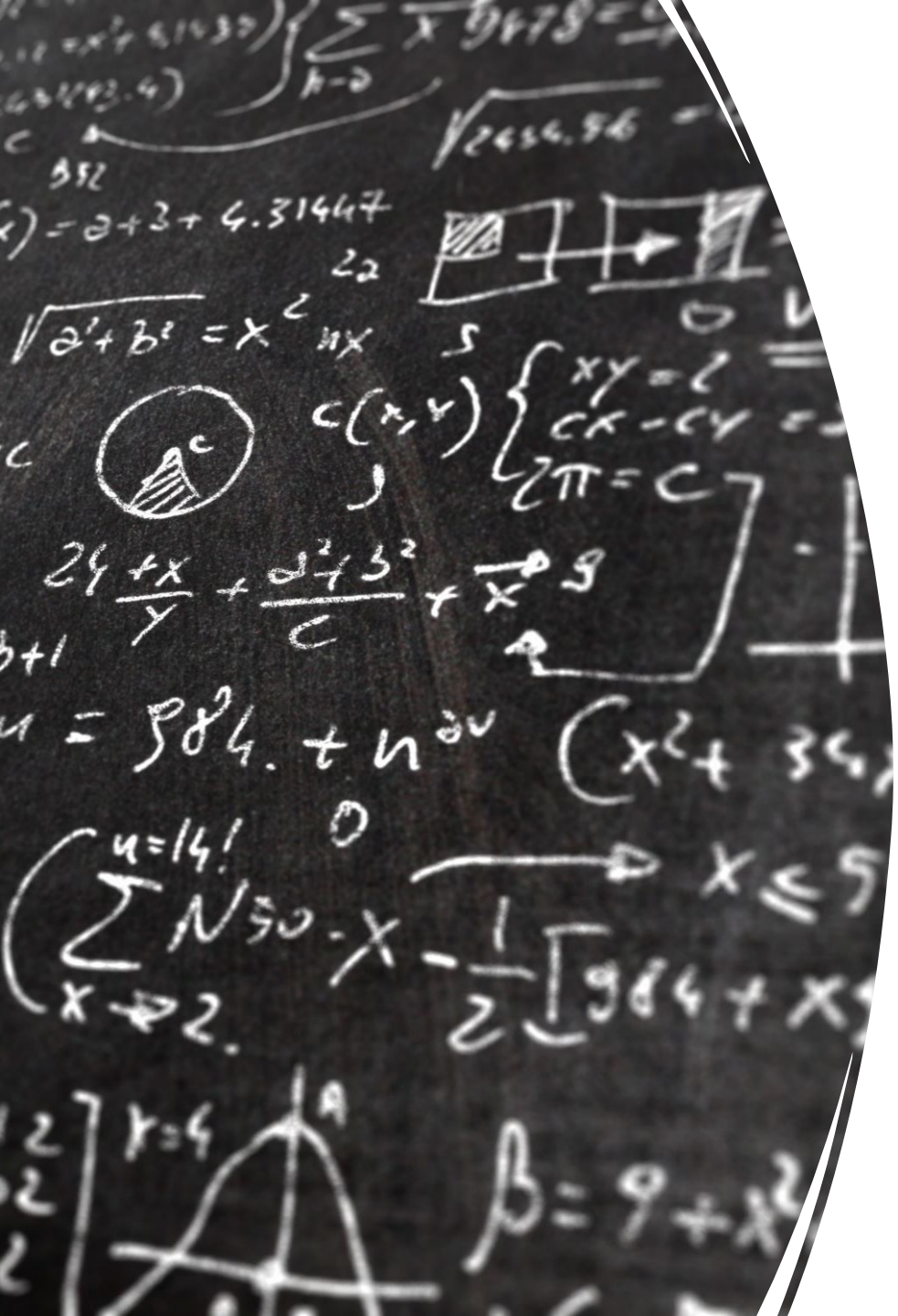


CS 113 – Computer Science I

Lecture 19 – Class Actions & Searching

Adam Poliak

03/28/2023

Announcements

- HW07
  - Due Thursday 03/30
- Midsemester feedback
- Office hours Thursday
- Schedule:
  - Thursday 04/06 – NO CLASS
  - Thursday 04/11 – remote class
  - Thursday 04/13 – Midterm

Outline

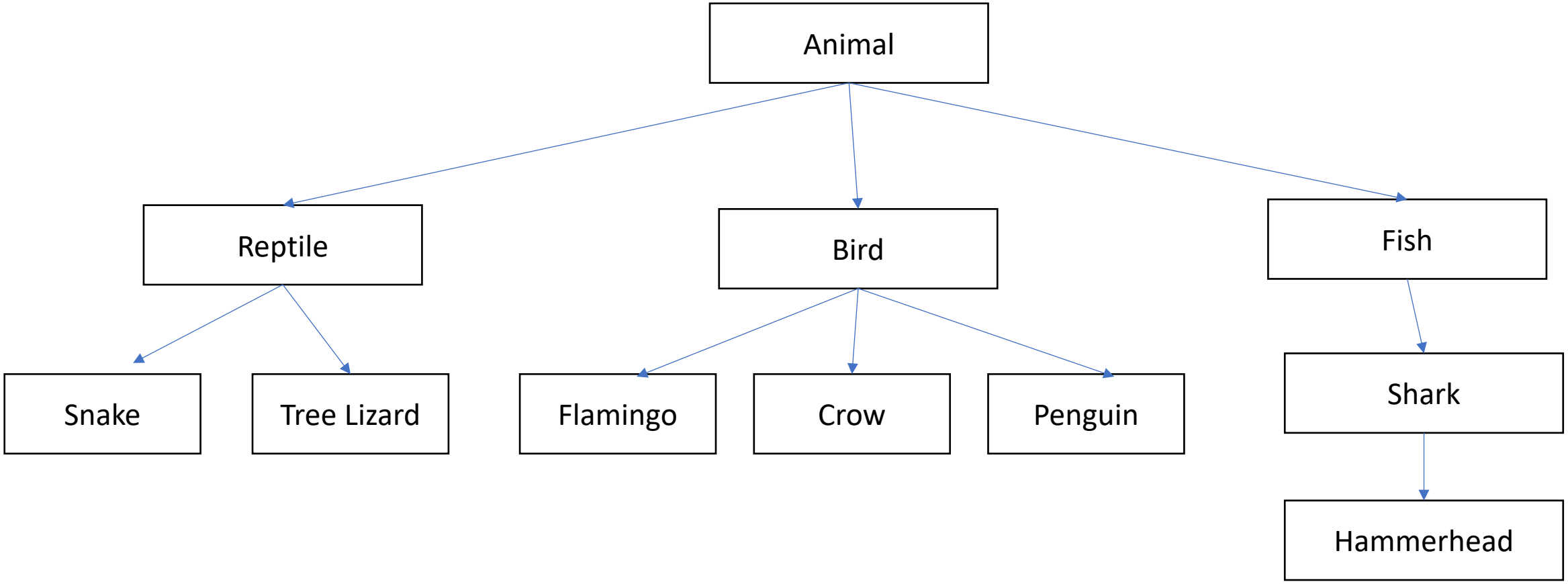
- Review (Inheritance)
- Subtyping (interfaces)
- Searching

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

Polymorphism

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

Java automatically calls the specific methods for each subclass

Polymorphism:

```
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?

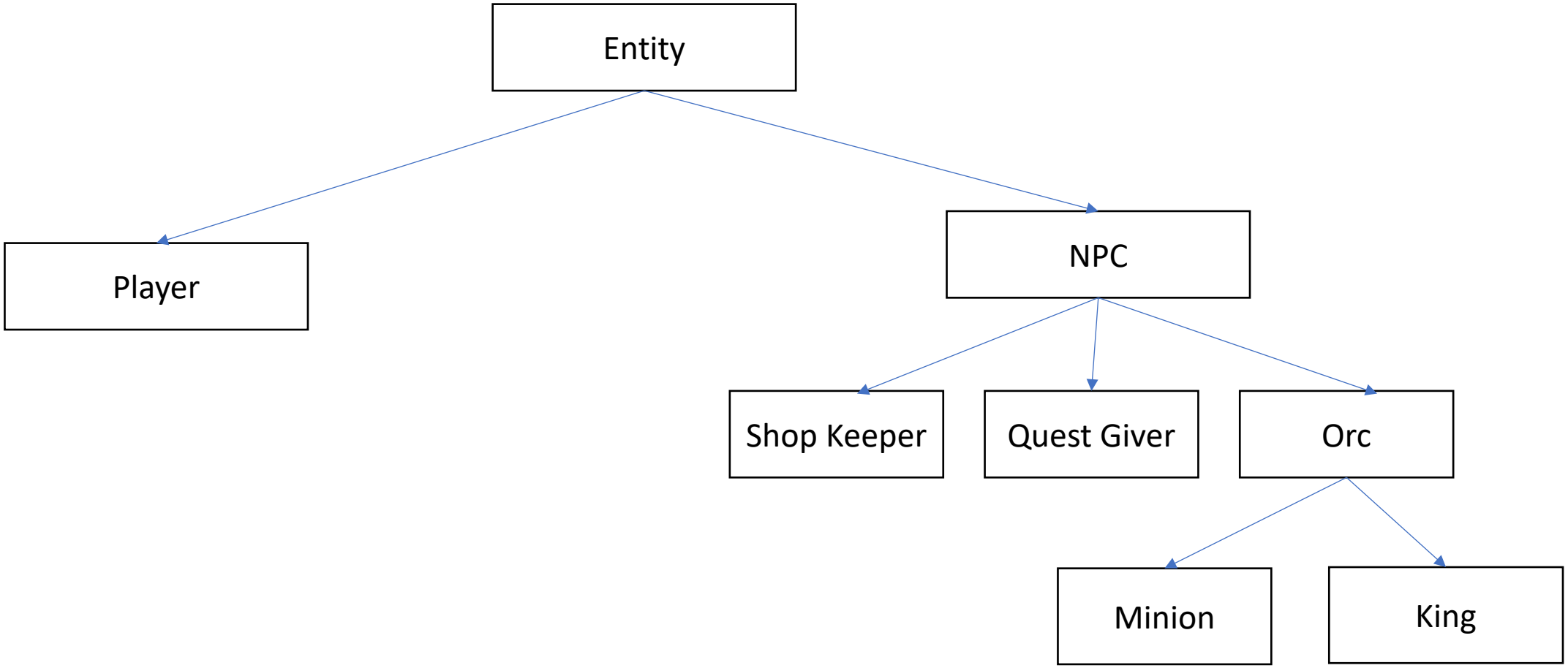
```
public class Zoo {
   public static void main(String[] args) {
      Animal animal1 = new Animal();
      animal1.locomote();

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

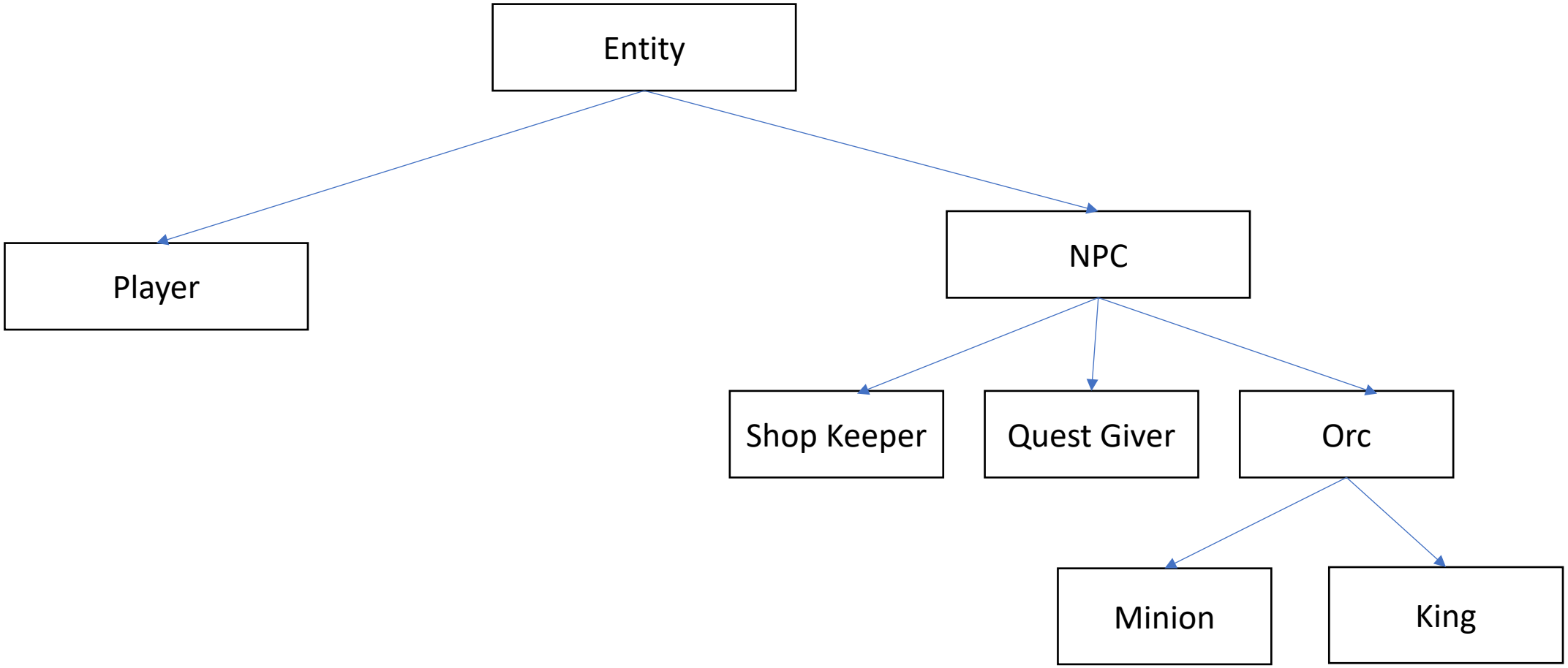
```
public class Animal {
   public Animal() {
   }
   public void locomote() {
      System.out.println("I am moving!");
   }
}
```

```
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();`

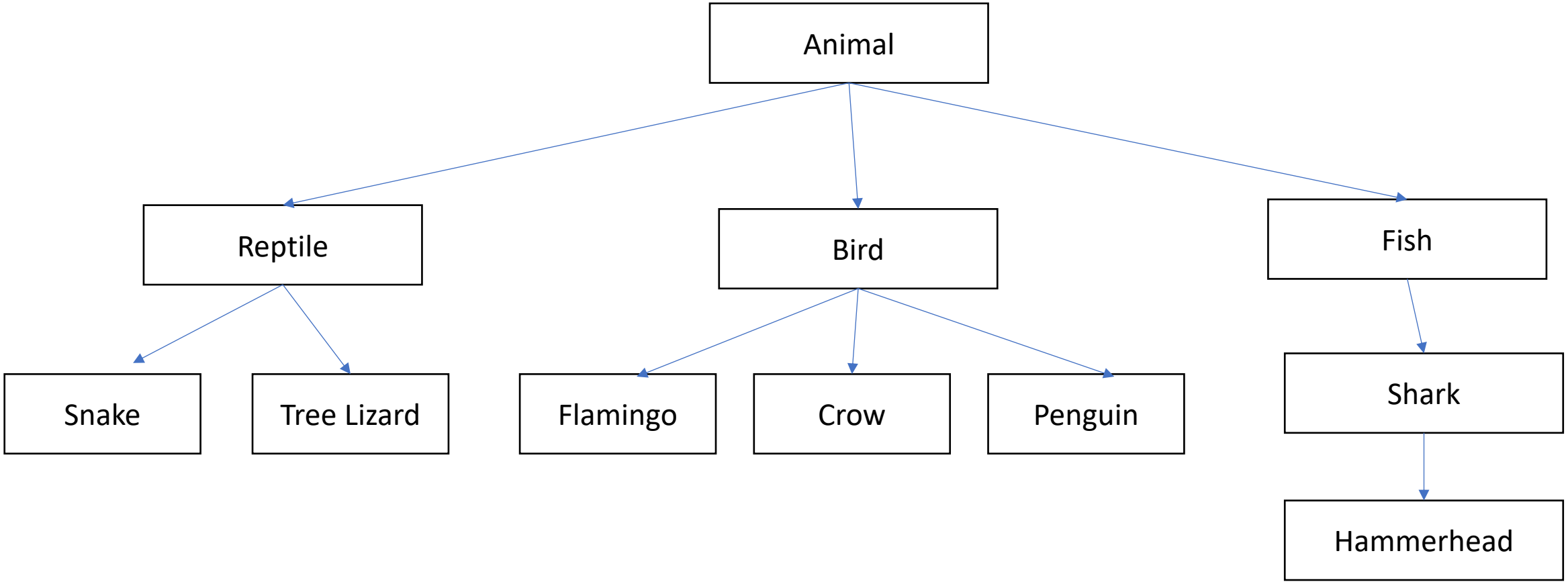
reference variable that is used to refer parent class constructors

Note:

`super:`

reference variable that is used to refer parent class object

Inheritance: feature for organizing classes into hierarchies



What do animals do?

- Eat
- Sleep
- Move
- procreate

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

Outline

- Review (Inhertiance)
- Subtyping (interfaces)
- **Searching**

Searching

Finding whether an item is in a collection

Applications:

- Specific email in an inbox
- Word in a document
- Course in a list of course offerings
- Professor is on RateMyProfessor
- …

Common search problems

- Is an item in an array?
  - Returns: True or False
- Where in an array is the item?
  - Returns: the index (an integer)
    - Standard: -1 if the item is not found
- How many times does the item appear in an array?
  - Returns: a count (an integer)
- What is the min, max, or average value in an array?
  - Returns: the value (double)

Searching in an array of Animals (demo)

Does our collection contain a specific Animal?

Idea:

Iterate through each Animal in the array

Check if the current animal is the same as the animal we are searching for.

If yes:

return True;

return False;

Comparing objects

Recall: variables for objects are references (pointers) to objects

`==` Compares whether the two references are the same

`Object.equals(Object obj)` compares two objects

Every (base) class should implement this

Searching in an array of Animals (demo)

Does our collection contain a specific Animal?

Idea:

Iterate through each Animal in the array

Check if the current animal is the same as the animal we are searching for.

If yes:

return True;

return False;

Searching in our array of Animals (demo)

Where in our collection is the specific Animal?

Idea:

Iterate through each Animal in the array

Check if the current point is the same as the animal we are searching for.

If yes:

return the index;

return False;

Searching in our array of Animals (demo)

How many animals in our collection are less than 5lbs?

Linear Search

These previous approaches are examples of linear search

Check each item in a collection one by one

Why is this call linear search?

Time it takes to search increases *linearly* with the size of the list

Linear Search

What happens (in terms of speed) when the list is very large?

The search becomes slower

In what cases do we do the most work (i.e. perform the most comparisons)?

When the item is not in the list

In what cases do we do the least amount of work?

When the item is the first element in the list