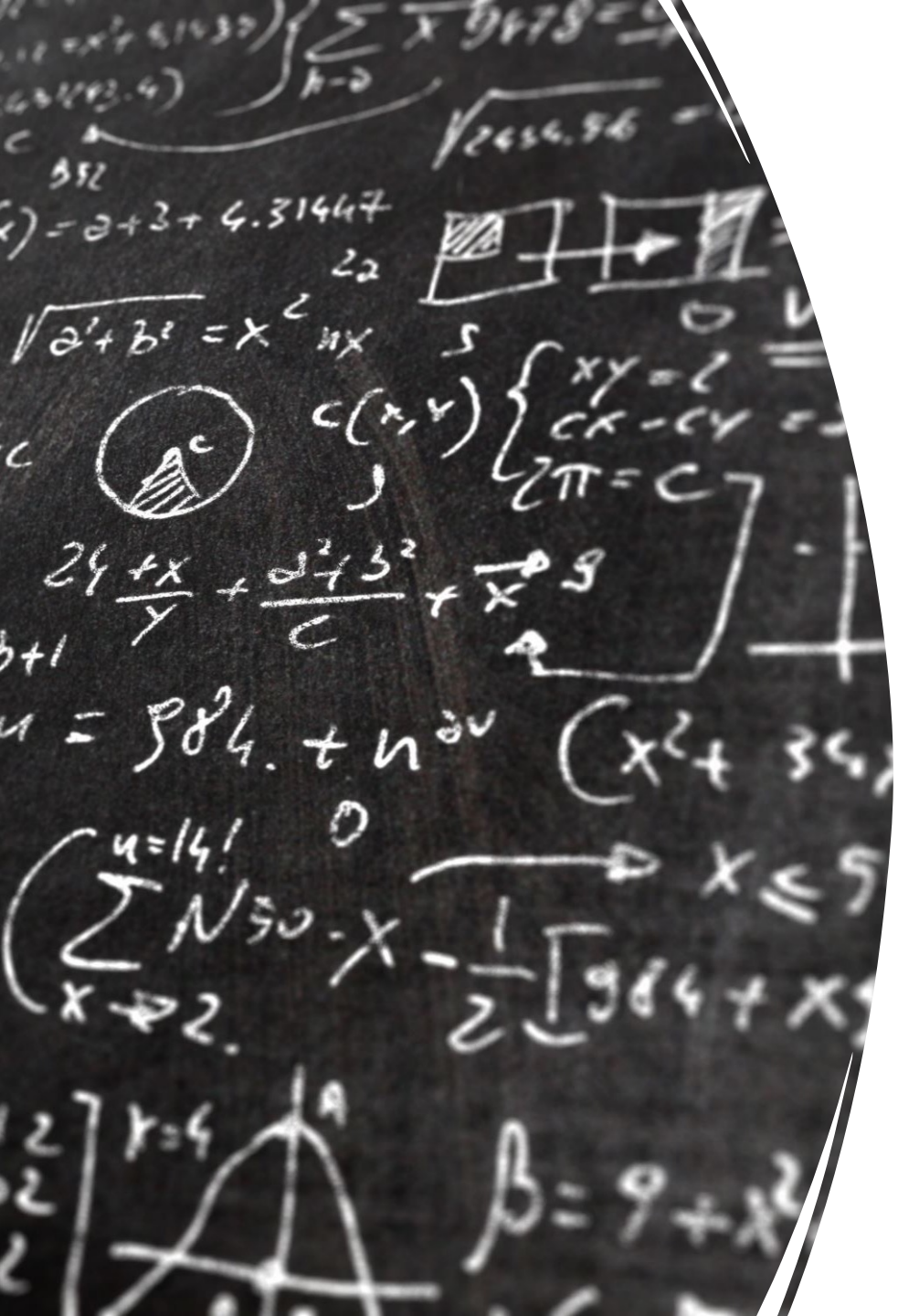


CS 113 – Computer Science I

Lecture 14 – More Objects, and some sorting

Adam Poliak

11/01/2022

Announcements

- Assignment 07
  - Due Thursday 11/03
- Mid-semester feedback
- Pythontutor.com - https://pythontutor.com/java.html
- Office hours:
  - Wednesday 10:30-11:30am & 3:45pm-4:45pm (just this week)

Homeworks Comments - `break;`

exit a loop without letting it run to completion

```
while (condition) {
	if (choice == 1) {
		// do something
	}
	else {
		break;
	}
}
```

Homeworks Comments – `System.exit()`

Exit the current program, terminate the program

```
System.exit(0);
System.exit(1);
```

Class inheritance

Review:

- Classes are like categories
- Objects are like examples of the categories

Classes can be arranged hierarchically where,

a child class ”inherits” from a parent class

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 constructors

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();`

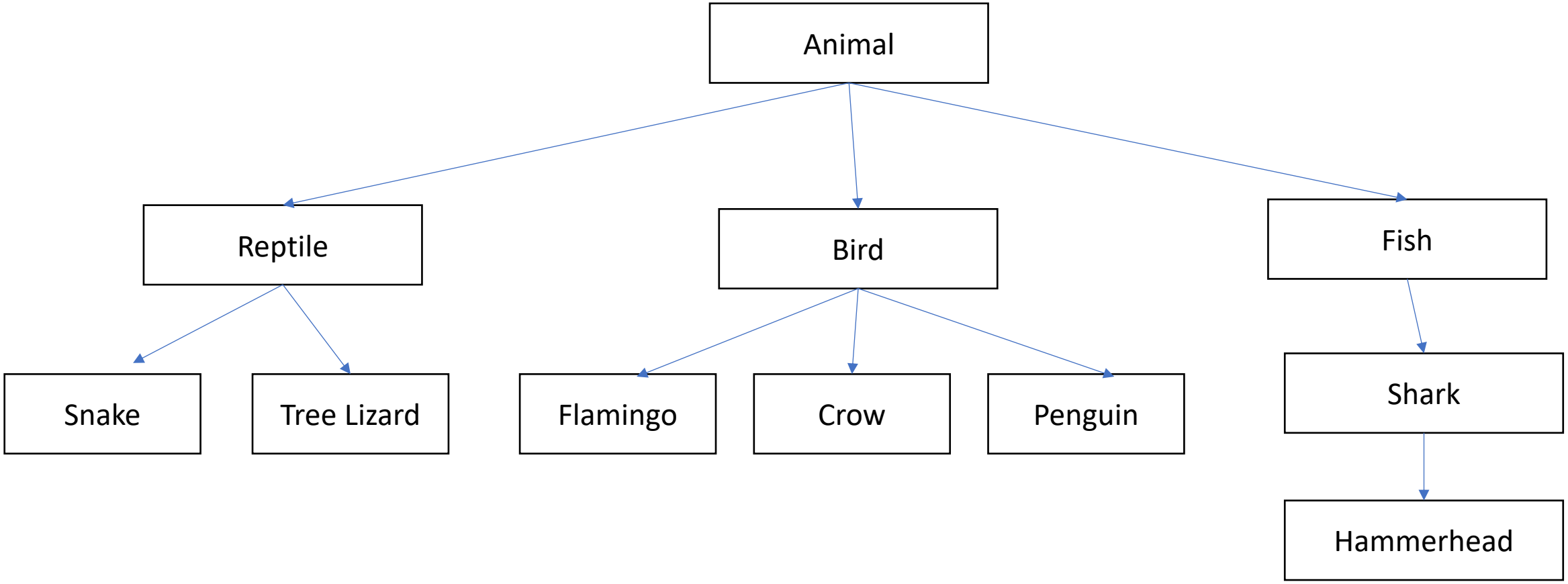
reference variable that is used to refer parent class constructors

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

Sorting & 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
- …

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 2D Points (demo)

Does our collection contain a specific Point?

Idea:

Iterate through each Point in the array

Check if the current point is the same as the point 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 our array of 2D Points (demo)

Does our collection contain a specific Point?

Idea:

Iterate through each Point in the array

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

If yes:

return True;

return False;

Searching in our array of 2D Points (demo)

Where in our collection is the specific Point?

Idea:

Iterate through each Point in the array

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

If yes:

return the index;

return False;

Searching in our array of 2D Points (demo)

How many points in our collection are less than 5 distances away from the origin?

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 elist

Sorting 2D Points (demo)

How can we sort an array of 2D Points?

Compute the distance of each point to the origin (Point(0, 0))

Make a new array of these distances

Sort the array of distances