Classes, Objects, and OOP in Java

June 16, 2017

Which is a Class in the below code?

```
Mario itsAMe = new Mario(“Red Hat?”);
```

A. Mario

B. itsAMe

C. new

D. “Red Hat?”

Whats the difference?

`int vs. Integer`

A. One is type, the other is not

B. One is a reference, the other is a Class

C. One is more efficient than the other

D. One is Java, the other is C

Whats the difference?

```
char[] lang = {“j”, “a”, “v”, “a”};
String lang = “java”;
```

A. One is a reference, one is a primitive

B. One is Unicode, the other is ASCII

C. One is an array, the other is type “String”

D. One is type “char”, the others of type “String”

Which evaluate to True?

```
String first = “First”;
String tied = “First”;
```

A. first == tied

B. first.isEqual(tied)

C. first.getClass() == tied.getClass()

D. first == String && second == String

Which are Synonyms?

A. Object and Instance

B. Type and Class

C. int and Integer

D. static and final

Which are Knowable?

```
GIVEN _only_ this information:
1. getPurposeOfLife is a static method
2. DanceGod.getPurposeOfLife() is valid
```

A. getPurposeOfLife() returns an object

B. DanceGod is “final”

C. getPurposeOfLife is “public”

D. DanceGod is a class

Which is a true difference between functions in C, and methods in Java?

A. Functions do not declare their return type, methods do

B. Methods in java can take fewer arguments that functions in C

C. Methods in java are tied to classes, functions in C are tied to structs.

D. Java allows named parameters, C does not

What is knowable about visibility

```
public class PublicClass {
  bool isPublic = true;
  }
```

A. Syntax error

B. is public is visible only to static methods

C. isPublic is visible to other objects

D. isPublic cannot be changed

What is a Constructor

A. An uncommon feature from older versions of java

B. How code configures functionality to trigger when an object is constructed

C. How Java determines the type of the object to be constructed

D. A class that creates other classes

Done!

Housekeeping

- Homework 1 has passed
- Homework 2 over the weekend
- Course topics / schedule
- Class reworking

OOP and Java

- Goals
- History of OOP
- OOP in Java

Why OOP? (Goals)

- “Text Book”
    - Encapsulation (Review today)
    - Polymorphism (Most of today)
    - Inheritance (Today and Monday)
- ¯_(ツ)_/¯

Problem One
Functions and Data

```
struct band {
    char   *  name;
    person    members[10];
    int       num_members;
};
```

```
person band_tallest_member(band *a_band) {…}

// Adding a new member?

// Wanting to do so something else with a “band” (ex, if we
// want to test if “MF Doom” was in the band)

// Default values for bands?
```

Encapsulation

- Binding together functionality and data
- Hiding the inner workings of code
- Prevent “boiler” plate, setup code
- Integrity

Problem Two
Similar Data

```
struct band {
    char   *  name;
    person    members[10];
    int       num_members;
};

person band_tallest_member(band *a_band) {…}

struct company {
    char   *  name;
    person    members[10];
    int       num_members;
    int       age;
};

// Finding tallest member of the company?
```

Polymorphism

- Abstracting over similar things
- Maintaining type information
- From a “consumer” / “client” / “user” perspective

Problem Three
Similar Functionality on Data

```
struct band {
    char   *  name;
    person    members[10];
    int       num_members;
    (some code to calculate tallest person)
};

struct company {
    char   *  name;
    person    members[10];
    int       num_members;
    int       age;
    (some code to calculate tallest person)
};

// How to not write the same thing twice
```

Inheritance

- Defining type hierarchies
- Shoving functionality into parent types
- Common, but not required, in OOP

Java and OOP

- Class is everything
  - Types
  - Functionality taxonomy
  - Data taxonomy
  - Namespaces
  - Functions (sorta…)
- Quirk of Java

Encapsulation in Java

- Define a class
- Define some data the class
- Define some functionality that uses that class

Encapsulation in Java

```
struct band {
    char   *  name;
    person    members[10];
    int       num_members;
};
```

```
person band_tallest_member(band *a_band) {…}
```

Encapsulation.java

Encapsulation in Java

- “this” makes things unambiguous
- public / private controls access
- constructors can create defaults
- Explicit is better than implicit

Polymorphism in Java

- Class hierarchy
- Interfaces (later)
- Generics (even later)

Class Hierarchy

Band

-getTallestMember

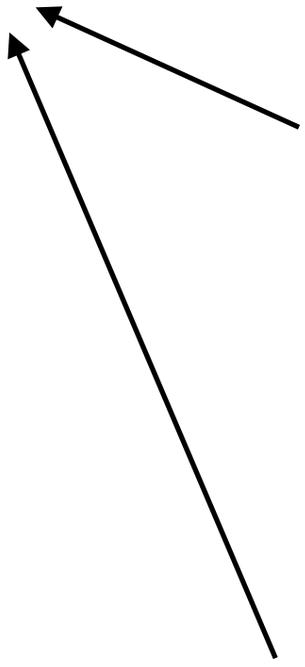
Company

-getTallestMember

-getCompanyAge

Class Hierarchy

PeopleDoingThingsTogether



Band

-getTallestMember

Company

-getTallestMember

-getCompanyAge

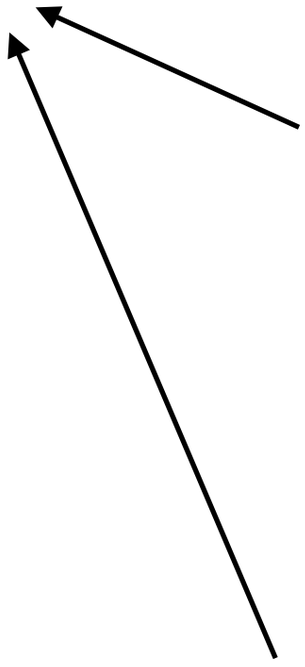
Polymorphism.java

Inheritance in Java

- Sharing code between types
- Check child classes first, then parents, then parents of parents, then…
- “extends"

Class Hierarchy

PeopleDoingThingsTogether



Band

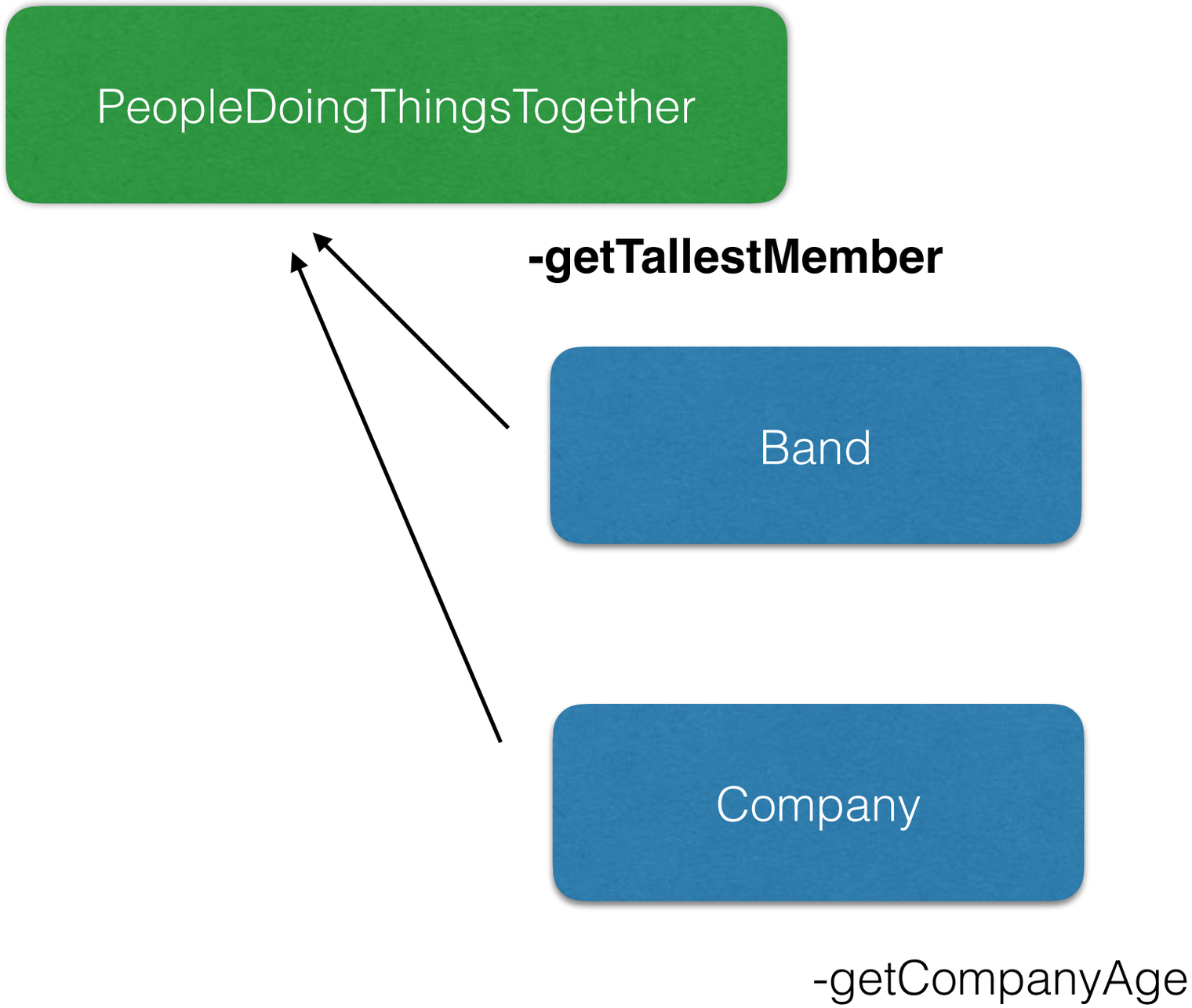
-getTallestMember

Company

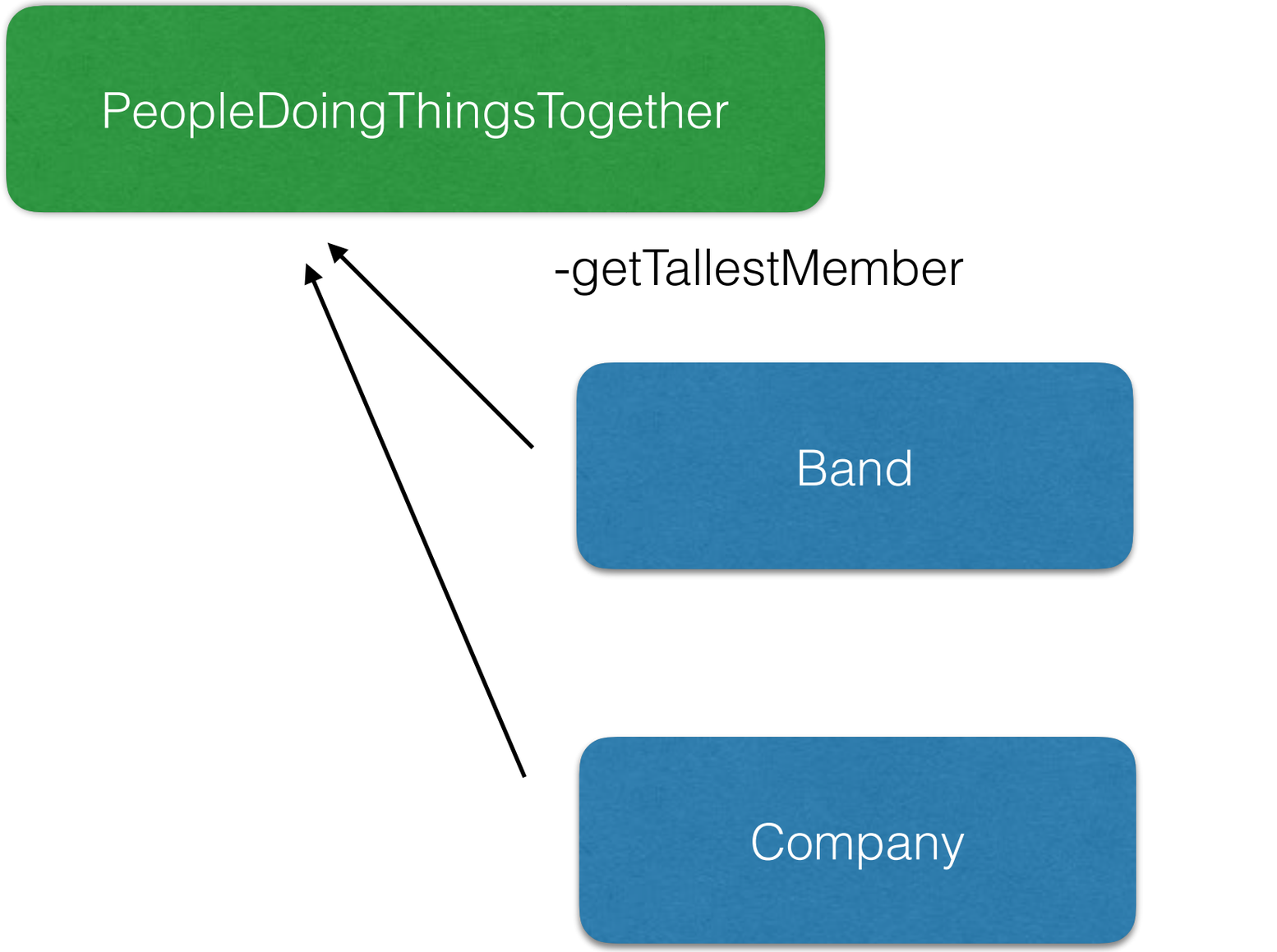
-getTallestMember

-getCompanyAge

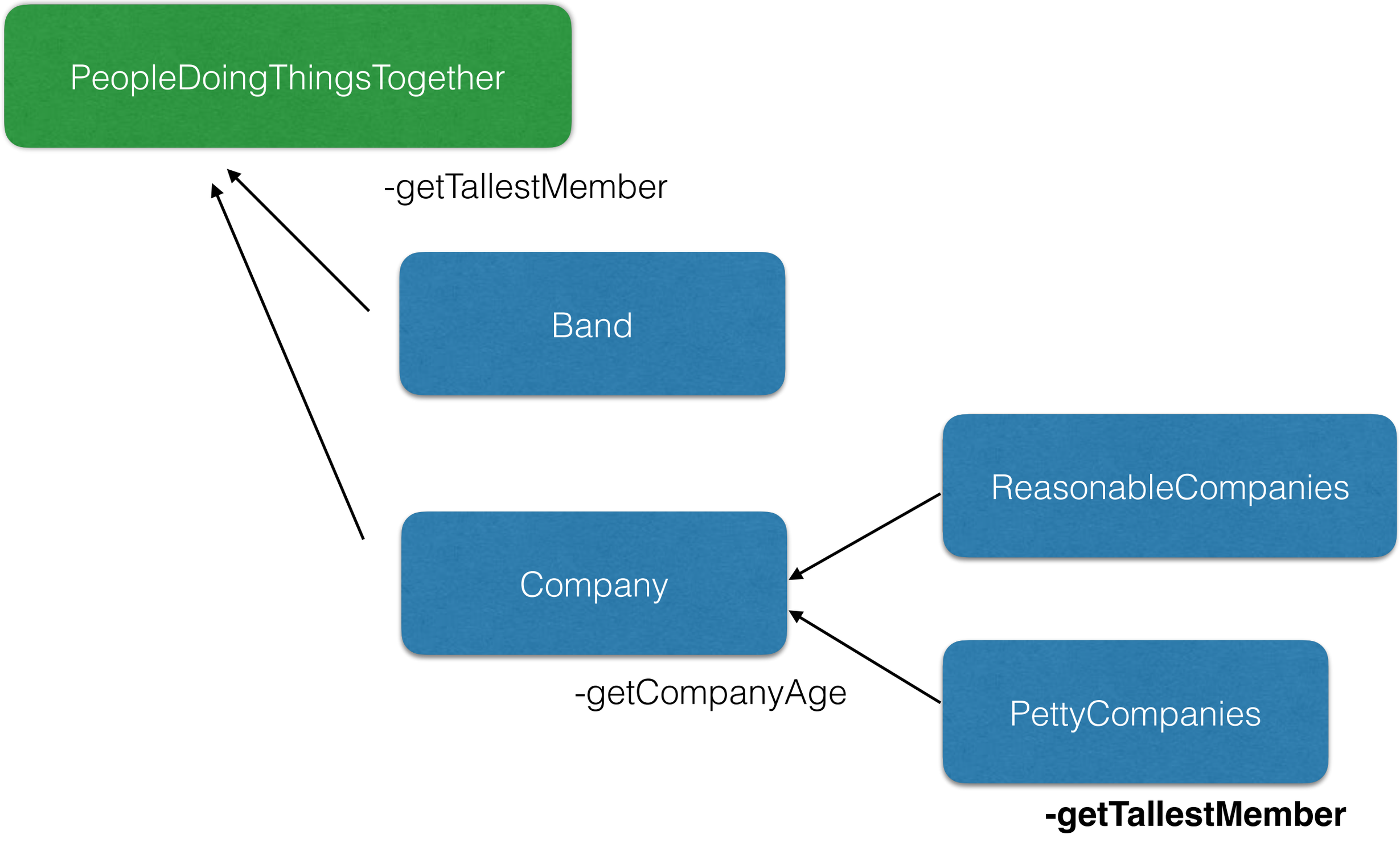
Class Hierarchy



Class Hierarchy



Class Hierarchy



Inheritance in Java

- Start with most specific class
- Keep looking “up” until we find a match… then stop
- `MethodNotFoundException` or compiler error if no chance

Further Code Sharing

- Child methods will often be similar…
- We want D.R.Y. code
- How to “slightly" change parent implementations
- “super” and “this”
- “Abstract”

InheritanceSuper.java

Java OO Particularities

- Arguments are part of a method's signature
- Disambiguation happens at runtime
- Static / final / scoped classes etc.

Programming, Step 1

- Groups
- Brainstorm class hierarchy
- Three tiers deep, at least 4 types
- Examples of three methods, at least one for each level of hierarchy

Programming, Step 2

- Describe a class hierarchy
- Should have at least the following classes
    - Animals
        - Cats
        - Lizards
        - ColdBlooded
        - WarmBlooded
        - Dogs
- Food needs

Programming, Step 3

- Implement simple program that sorts user provided animals into bins, depending on their needs
  - getName()
  - main(String[] args)
  - Print out summary

Wrap Up

- Coding good? Slides better?
- Homework 2
- Reading for Monday
- Office hours… now!