

LEC 02

CSE 123

Inheritance & Polymorphism

Questions during Class?

Raise hand or send here


sli.do #cse123



BEFORE WE START

Talk to your neighbors:*What'd you get up to this weekend? Anything fun?*Music: [123 24su Lecture Tunes](#) **Instructor:** Joe Spaniac**TAs:** Andras Eric Sahej Zach
Daniel Nicole Trien


Lecture Outline

- **Announcements/Reminders** 
- Finishing up Points & Lines
- Inheritance
- Polymorphism
 - Declared vs. Actual Type
 - Compiler vs. Runtime Errors
- Points, Lines, and Graphs!


Announcements

- IPL is now open!
 - 12:30-5:30pm M-F, some availability on weekends
- Check-in 1 is tomorrow, June 27th!
 - Just show up to your *registered* quiz section
 - OK if you can't make it, 2/4 -> E, just email me for the activity
- Quiz 1 is this upcoming Tuesday, July 2nd!
 - Topics: Object-Oriented Programming, JUnit, Inheritance, Polymorphism
 - Taken in your *registered* quiz section on paper
 - Allowed 1 sheet of 8.5"x11" paper (double-sided, can be printed)
 - More details on quiz logistics in Ed announcement posted soon!
- Creative Project 1 (C1) is due tonight.
 - Submit what you have so you can get feedback (even if unfinished)
 - Joined class late? Use Resubmission Cycle 1 to submit it!

Lecture Outline

- Announcements/Reminders
- **Finishing up Points & Lines** 
- Inheritance
- Polymorphism
 - Declared vs. Actual Type
 - Compiler vs. Runtime Errors
- Points, Lines, and Graphs!

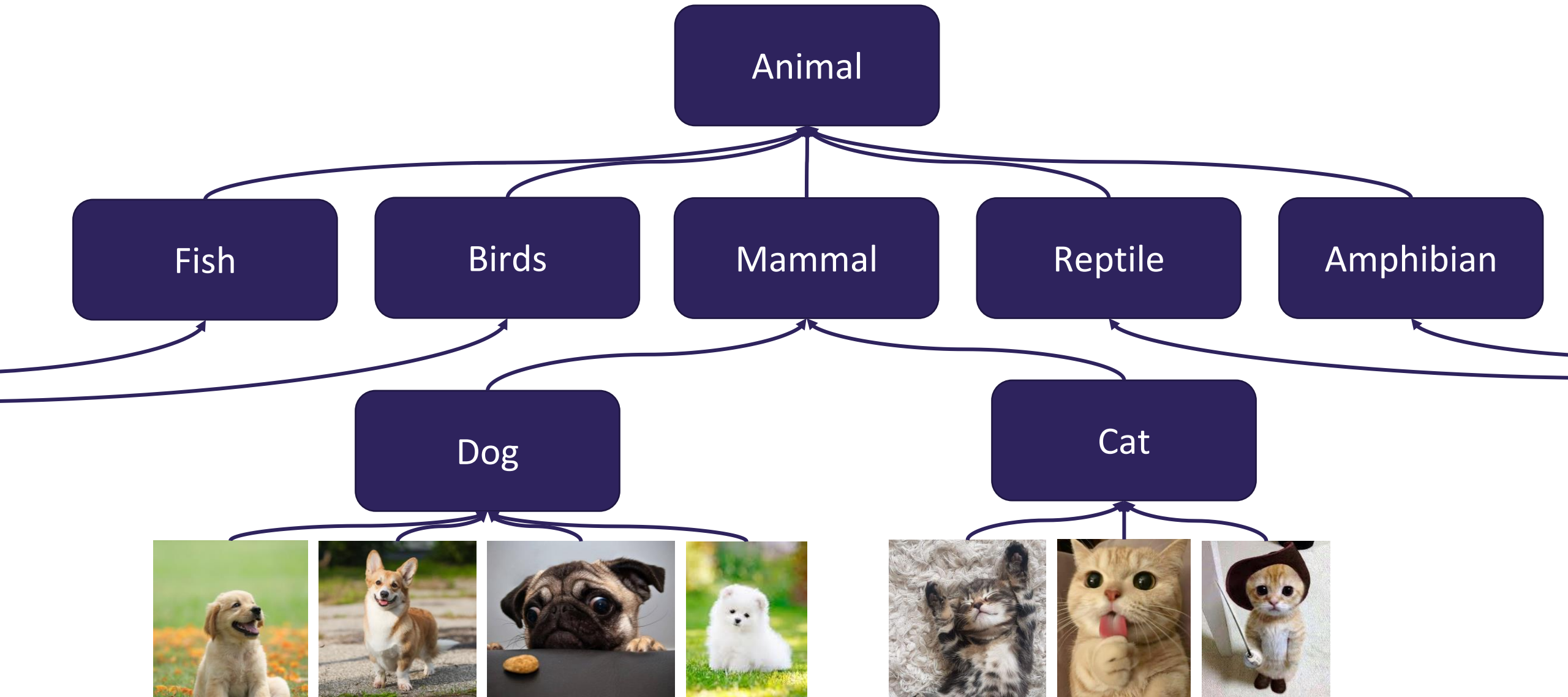
Lecture Outline

- Announcements/Reminders
- Finishing up Points & Lines
- **Inheritance** 
- Polymorphism
 - Declared vs. Actual Type
 - Compiler vs. Runtime Errors
- Points, Lines, and Graphs!


Inheritance

- Connect together a “subclass” and “superclass”
 - Borrow / “inherit” code to reduce redundancy
 - `super()` keyword can be used just like `this()`
- Syntax: `public class Subclass extends Superclass`
- Should Represent “is-a” relationships
 - `public class Chef extends Employee`
 - `public class Server extends Employee`
- In Java, all objects implicitly inherit from the Object class
 - `toString()`, `equals(Object)`, etc.

Is-a Relationships



Lecture Outline

- Announcements/Reminders
- Finishing up Points & Lines
- Inheritance
- **Polymorphism** 
 - Declared vs. Actual Type
 - Compiler vs. Runtime Errors
- Points, Lines, and Graphs!

Polymorphism

- `DeclaredType x = new ActualType()`
 - All methods in `DeclaredType` can be called on `x`
 - We've seen this with interfaces (`List<String>` vs. `ArrayList<String>`)
 - Can also be to inheritance relationships

```
Animal[] arr = {new Dog(), new Cat(), new Bear()};  
for (Animal a : arr) {  
    a.feed();  
}
```

Compiler vs. Runtime Errors

- DeclaredType x = new ActualType()
 - At compile time, Java only knows DeclaredType
 - Compiler error: trying to call a method that isn't present

```
Animal a = new Dog();
a.pet();           // No pet() -> CE
```
 - Can cast to change the DeclaredType of an object

```
((Dog) a).pet();   // No more CE
```
 - Runtime error: attempting to cast to an invalid DeclaredType*

```
Animal a = new Fish();
((Dog) a).pet();   // Can't cast -> RE
```
 - Order matters! Compilation before runtime

Compiler vs. Runtime Errors

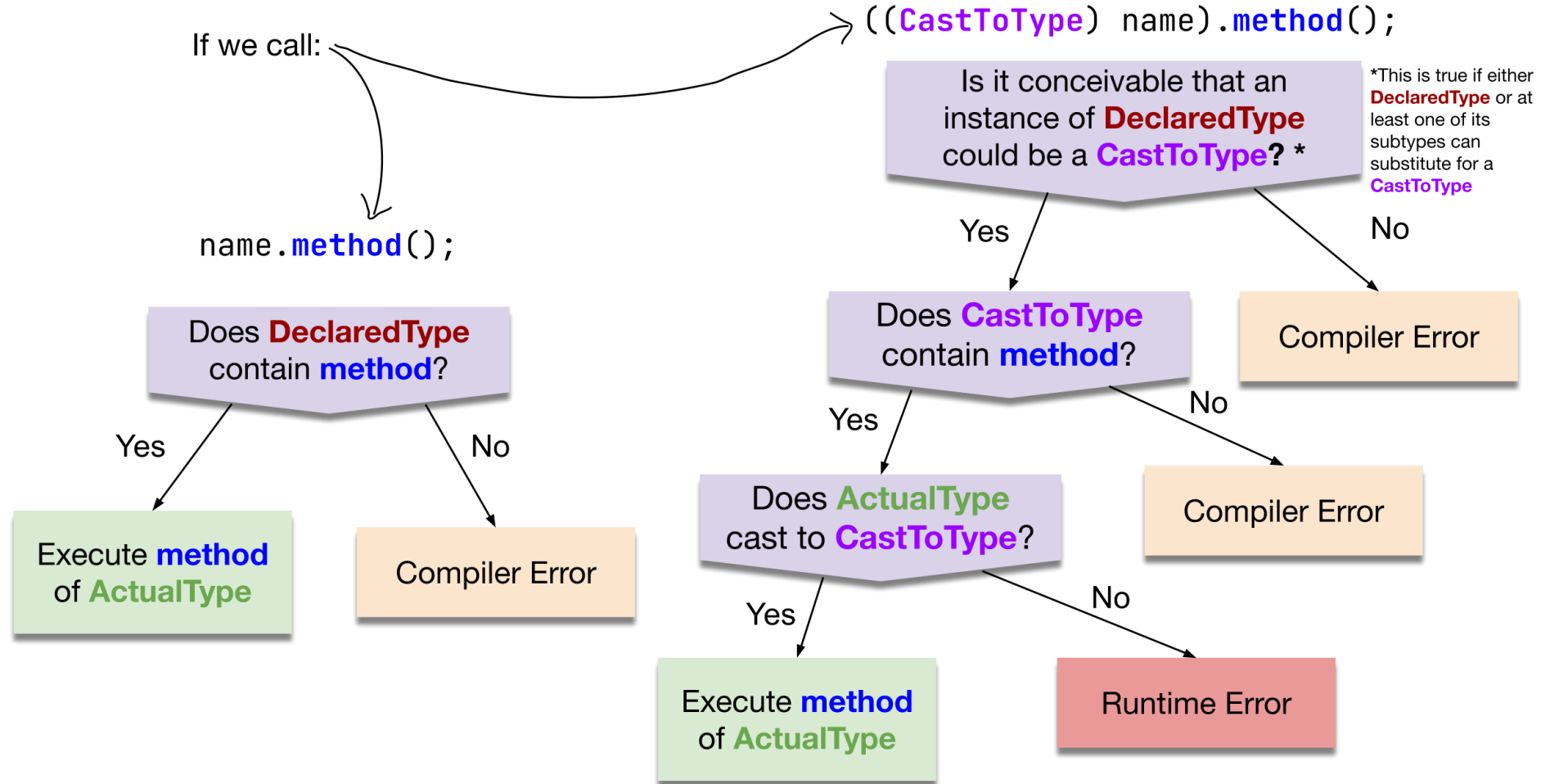
With the following declaration and initialization:

```
DeclaredType name = new ActualType();
```

If we call:

```
name.method();
```

```
((CastToType) name).method();
```



Lecture Outline

- Announcements/Reminders
- Finishing up Points & Lines
- Inheritance
- Polymorphism
 - Declared vs. Actual Type
 - Compiler vs. Runtime Errors
- **Points, Lines, and Graphs!** 