Introduction to Java (cs2514)

Lecture 1: Syllabus and Introduction

M. R. C. van Dongen

January 16, 2017

Syllabus

- Credit weighting 5.
- 24 x 1 hour lectures.
- Assessment: 100 marks.
  - Written exam 80 marks (1.5 hours).
  - Continuous assessment 20 marks: test and/or lab assignments.
  - Work that is submitted late shall be awarded zero marks.
- 40% pass rate.
- 1.5 hour repeat exam (mark for CA is carried forward).

Lectures

lecture Monday 12 m. – 1 p.m. in WGB G26;
lecture Friday 11 a.m. – 12 m. in WGB G01;

Assignment Submission

- Submit your assignments using http://cs4.ucc.ie/moodle.
- The official deadline is on the assignment sheet.
- There usually is a *grace* period, with a grace deadline.
  - Avoids problems for people who were "just" too late.
- When you too late for the official deadline,
  - You may still submit until the grace deadline.

Assignment Submission

- Submit your assignments using http://cs4.ucc.ie/moodle.
- The official deadline is on the assignment sheet.
- There usually is a *grace* period, with a grace deadline.
  - Avoids problems for people who were "just" too late.
- When you too late for the official deadline,
  - You may still submit until the grace deadline.
- When you too late for the grace deadline

Assignment Submission

- Submit your assignments using `http://cs4.ucc.ie/moodle`.
- The official deadline is on the assignment sheet.
- There usually is a *grace* period, with a grace deadline.
  - Avoids problems for people who were “just” too late.
- When you too late for the official deadline,
  - You may still submit until the grace deadline.
- When you too late for the grace deadline,
  - You’re too late.

Assignment Submission

- Submit your assignments using `http://cs4.ucc.ie/moodle`.
- The official deadline is on the assignment sheet.
- There usually is a *grace* period, with a grace deadline.
  - Avoids problems for people who were "just" too late.
- When you too late for the official deadline,
  - You may still submit until the grace deadline.
- When you too late for the grace deadline,
  - You're too late.
  - No submission allowed.

Assignment Submission

- Submit your assignments using http://cs4.ucc.ie/moodle.
- The official deadline is on the assignment sheet.
- There usually is a *grace* period, with a grace deadline.
  - Avoids problems for people who were "just" too late.
- When you too late for the official deadline,
  - You may still submit until the grace deadline.
- When you too late for the grace deadline,
  - You're too late.
  - No submission allowed.
  - Sorry about that.

Module Content

- Class definitions;
- Procedural abstraction and data abstraction;
- Associations between objects;
- Class hierarchies and inheritance;
- Polymorphism and dynamic method binding.

Learning Outcomes

On Successful Completion Students should be able to

- Interpret a set of requirements for a software system;
- Construct Java programs in a good object oriented style;
- Design medium-sized software in a disciplined manner;
- Examine an existing software system for quality criteria;
- Employ object oriented abstractions such as encapsulation and inheritance in an appropriate way.

Required Literature

title	Head First Java.
edition	Second;
author	Kathy Sierra & Bert Bates;
publisher	O'Reilly;
year	2004;
ISBN	978-0-596-00712-6.

Optional Literature

Loads of Good Books

title	Effective Java;
author	Joshua Bloch;
publisher	Addison–Wesley;
year	2008;
ISBN	978-0-321-35668-0.

Optional Literature

Loads of Good Books

title	Java Puzzlers *Traps, Pitfalls, and Corner Cases;*
author	Joshua Bloch and Neal Gafter;
publisher	Addison–Wesley;
year	2005;
ISBN	0-321-33678-x.

Optional Literature

Loads of Good Books

title	Head First Object-Oriented Analysis & Design;
author	Brett D. McLaughlin, Gary Pollice, and David West;
publisher	O'Reilly;
year	2007;
ISBN	978-0-596-00867-3.

Optional Literature

Loads of Good Books

title	Design Patterns *Elements of Reusable Object-Oriented Software* (36th Printing);
author	Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides;
publisher	Addison–Wesley;
year	2008;
ISBN	0-201-63361-2.

Plagiarism

Presenting Someone Else's Work as your Own

- When done deliberately, it is cheating.
- Plagiarism applies to:
  - Text;
  - Software;
  - Graphics;
  - Tables;
  - Formulae; or
  - Any representation of ideas in print, electronic or any other media.

UCC Policy on Plagiarism

- All students are required to read, to understand, and to comply with the UCC Policy on Plagiarism:
  - http://www.ucc.ie/en/exams/procedures-regulations/.

Submitting Coursework

- In general, you should write all coursework in your own words.
- Coursework includes but is not limited to:
  - Programming assignments;
  - Literature reviews;
  - Abstracts and summaries;
  - Final-year projects.

Submitting Existing Software?

As a General Rule

assignments You are usually *not* allowed to submit existing software unless the lecturer clearly indicates that this is allowed.

- Consult with your course lecturer if you are unsure whether you are allowed to submit existing software for assignments.

project You are usually allowed to submit (small) parts of existing software.

- Consult with your project supervisor if you are unsure whether you are allowed to re-use existing software for your thesis.

Submitting Work from Others

- If you wish to quote small portions of software, text, include images, software, or other work created by others, you need to make it clear that you are doing so.
- You usually do this by putting quotation marks around quoted text and by including citations.
- Note that pictures and diagrams in books and papers may be copyrighted, in which case you need explicit permission from the copyright holder.

Acknowledging Existing Work

- If you acknowledge the original source, your lecturers/examiners will know that you are aware of the source.
  - You can receive credit for this in the form of marks.
- If you fail to acknowledge the source, your lecturers/examiners cannot give you any credit for using the source.
  - When failing to acknowledge the source is deliberate, this is a form of cheating, which may result in awarding a zero mark.

Citing Existing Software

- If you submit (parts of) existing software as part of your coursework, you should always give proper credit to the original author(s).
- In addition, you should clearly indicate which parts of these software are yours and which are not.
  - In a program listing you should indicate this using comments;
  - In a final-year project report you should also indicate the source of the software in the running text.
    - This should include a proper citation.

What is Java?

What is Java?

island: A holiday destination.

What is Java?

island: A holiday destination.

coffee: Nice one too.

What is Java?

island: A holiday destination.

coffee: Nice one too.

language: The programming language Java.

- We write our Java programs in text (source) files.

What is Java?

island: A holiday destination.

coffee: Nice one too.

language: The programming language `Java`.

- We write our `Java` programs in text (source) files.

compiler: We compile our source files with the `javac` compiler.

- Creates Java byte code file(s).
- Java byte code is an abstract machine language.
- Can be "run" with a (Java) *virtual machine* (VM).

What is Java?

island: A holiday destination.

coffee: Nice one too.

language: The programming language `Java`.

- We write our `Java` programs in text (source) files.

compiler: We compile our source files with the `javac` compiler.

- Creates Java byte code file(s).
- Java byte code is an abstract machine language.
- Can be "run" with a (Java) *virtual machine* (VM).

VM: We execute/interpret the byte code with the `java` VM.

- This "runs" the program.

How Does that Work?

Unix Session

```
$
```

How Does that Work?

Unix Session

```
$ ls
```

How Does that Work?

Unix Session

```
$  ls
Hello.java
$
```

How Does that Work?

Unix Session

```
$  ls
Hello.java
$  javac Hello.java
```

How Does that Work?

Unix Session

```
$  ls
Hello.java
$  javac Hello.java
$
```

How Does that Work?

Unix Session

```
$  ls
Hello.java
$  javac Hello.java
$  ls
```

How Does that Work?

Unix Session

```
$  ls
Hello.java
$  javac Hello.java
$  ls
Hello.java    Hello.class
$
```

How Does that Work?

Unix Session

```
$  ls
Hello.java
$  javac Hello.java
$  ls
Hello.java    Hello.class
$  java Hello
```

How Does that Work?

Unix Session

```
$  ls
Hello.java
$  javac Hello.java
$  ls
Hello.java    Hello.class
$  java Hello
Hello world!
$
```

State and Behaviour

- Classes are used to *define* your objects.
- Each class corresponds to an *object type*.
- When you create a class, you think about:

  Knowing: What the object knows.

  Doing: What the object does.

State and Behaviour

- Classes are used to *define* your objects.
- Each class corresponds to an *object type*.
- When you create a class, you think about:

  Knowing: What the object knows.
  - The object's *instance variables* define its memory.

  Doing: What the object does.

State and Behaviour

- Classes are used to *define* your objects.
- Each class corresponds to an *object type*.
- When you create a class, you think about:

  Knowing: What the object knows.
  - The object's *instance variables* define its memory.
  - Instance variables are also known as *attributes*.

  Doing: What the object does.

State and Behaviour

- Classes are used to *define* your objects.
- Each class corresponds to an *object type*.
- When you create a class, you think about:

  Knowing: What the object knows.
  - The object's *instance variables* define its memory.
  - Instance variables are also known as *attributes*.
  - The object's attributes define the object's *state*.

  Doing: What the object does.

State and Behaviour

- Classes are used to *define* your objects.
- Each class corresponds to an *object type*.
- When you create a class, you think about:

  Knowing: What the object knows.
  - The object's *instance variables* define its memory.
  - Instance variables are also known as *attributes*.
  - The object's attributes define the object's *state*.

  Doing: What the object does.
  - What object do "are" its (instance) *methods*.

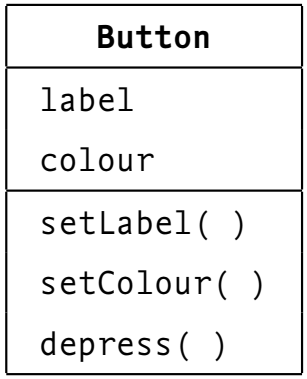
State and Behaviour

- Classes are used to *define* your objects.
- Each class corresponds to an *object type*.
- When you create a class, you think about:

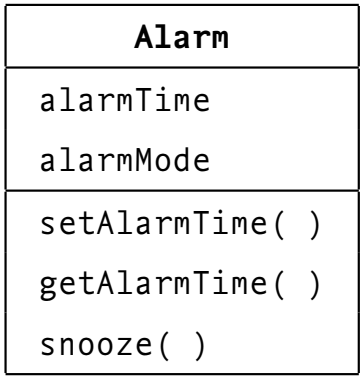
  Knowing: What the object knows.
  - The object's *instance variables* define its memory.
  - Instance variables are also known as *attributes*.
  - The object's attributes define the object's *state*.

  Doing: What the object does.
  - What object do "are" its (instance) *methods*.
  - The object's instance methods define its *behaviour*.

State and Behaviour



State and Behaviour



How Does it Work in Java?

Java

```
public class Song {
    private String title;  // Attribute: knowing.
    private String artist; // Attribute: knowing.

    public String getTitle( ) { // Method: doing.
        return title;
    }

    public void playSong( ) {   // Method: doing.
        // play song.
    }
    ⋮
}
```

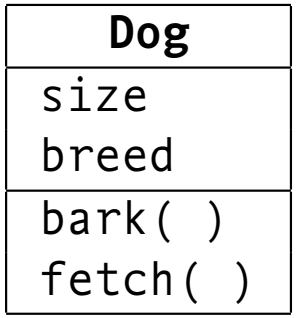
Class versus Object

- A class is not an object.
- A class is used to construct objects at run time.
- The class acts as a *blueprint* for the object.
- One class may define several objects (a.k.a. instances).

Class versus Object

One Class

- A class is not an object.
- A class is used to construct objects at run time.
- The class acts as a *blueprint* for the object.
- One class may define several objects (a.k.a. instances).

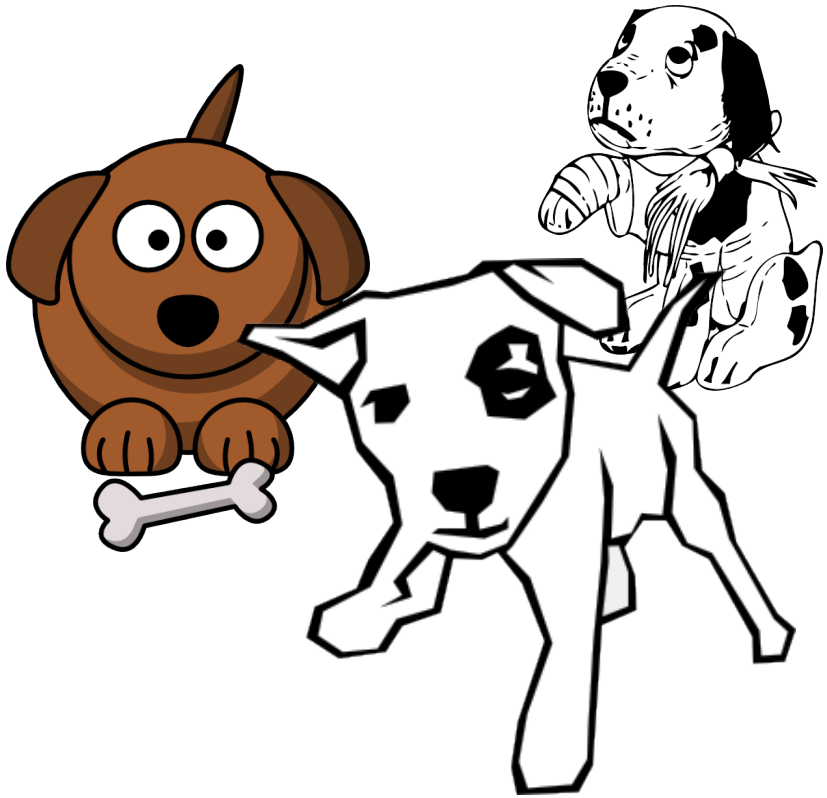


Class versus Object

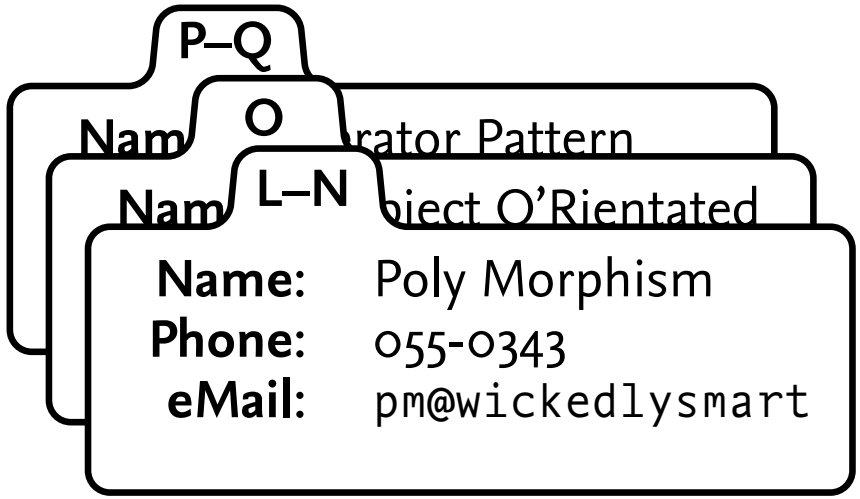
Many Instances

- A class is not an object.
- A class is used to construct objects at run time.
- The class acts as a *blueprint* for the object.
- One class may define several objects (a.k.a. instances).

Dog
`size` `breed`
`bark( )` `fetch( )`

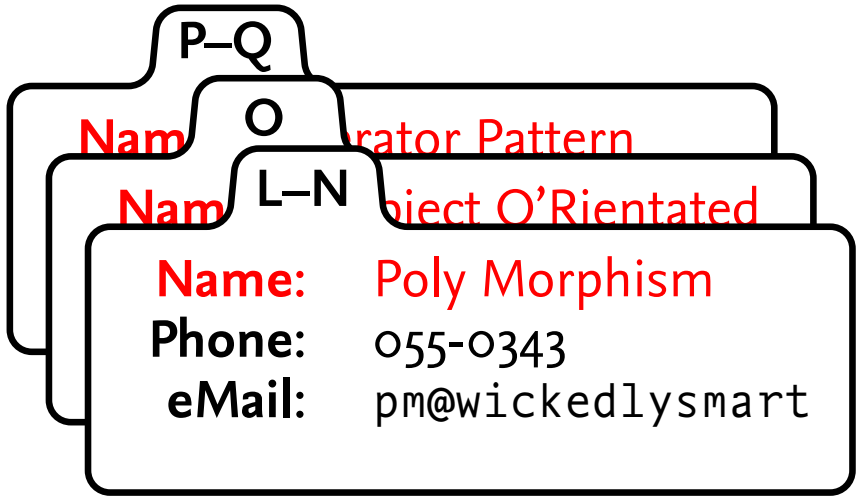


Objects are like Rolodex Cards



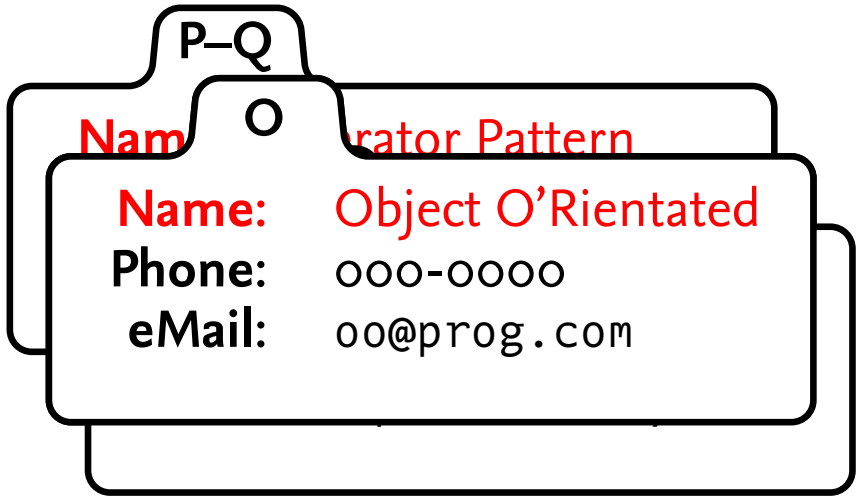
Objects are like Rolodex Cards

Name



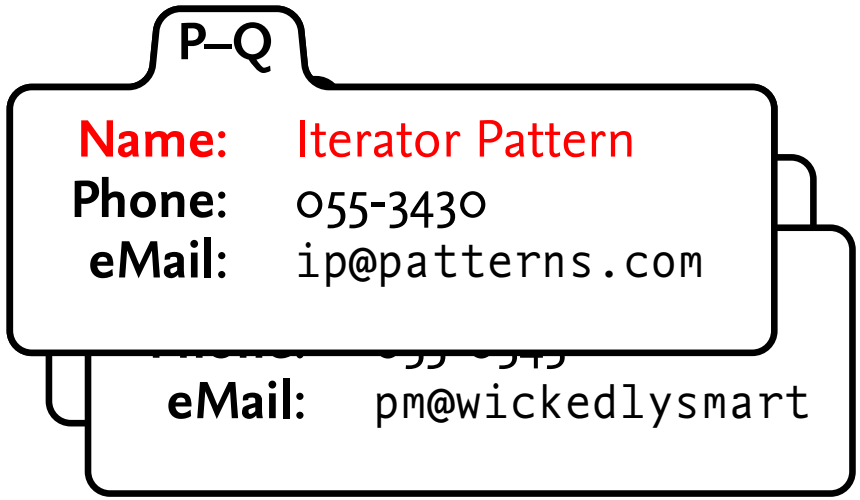
Objects are like Rolodex Cards

Name



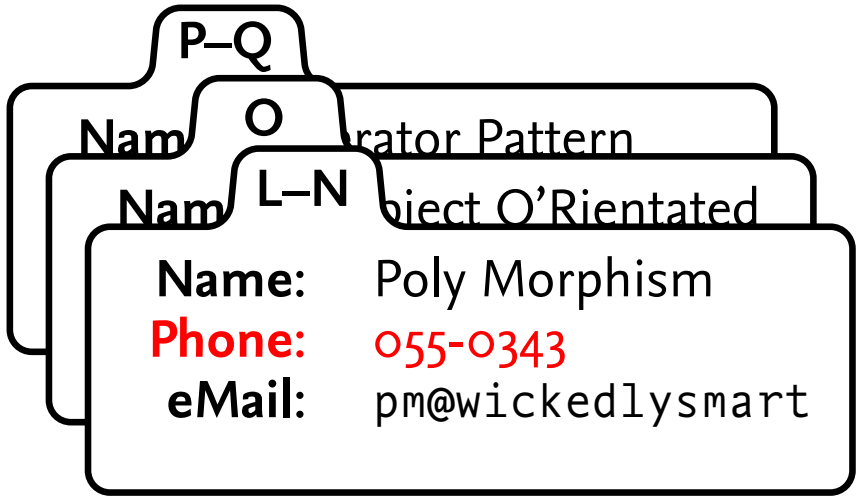
Objects are like Rolodex Cards

Name



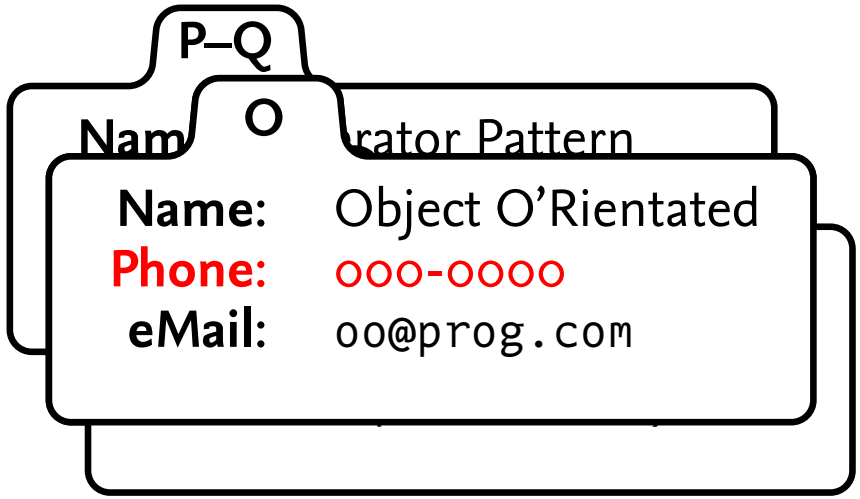
Objects are like Rolodex Cards

Phone



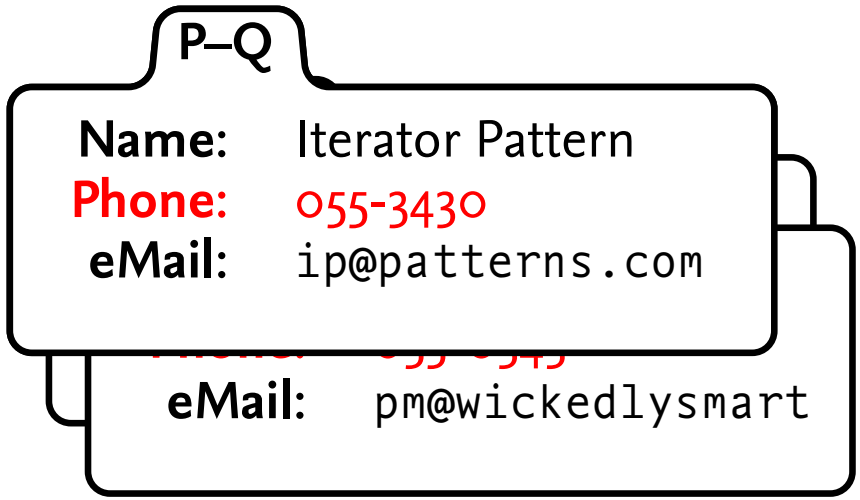
Objects are like Rolodex Cards

Phone



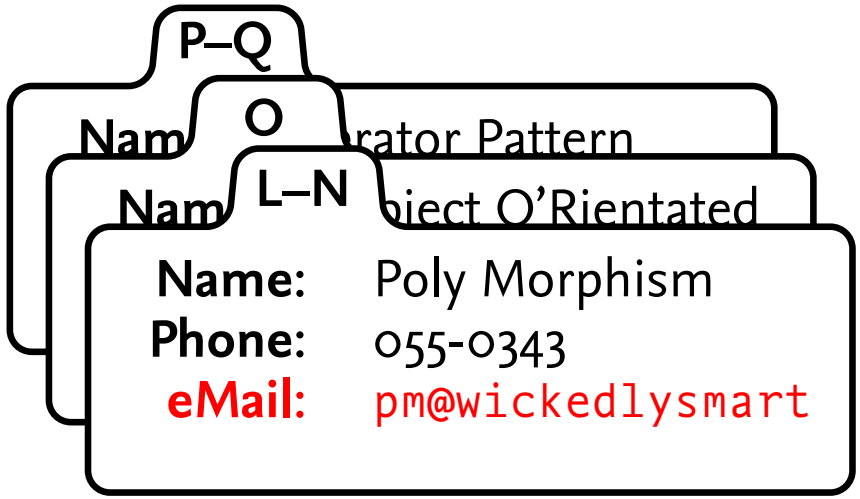
Objects are like Rolodex Cards

Phone



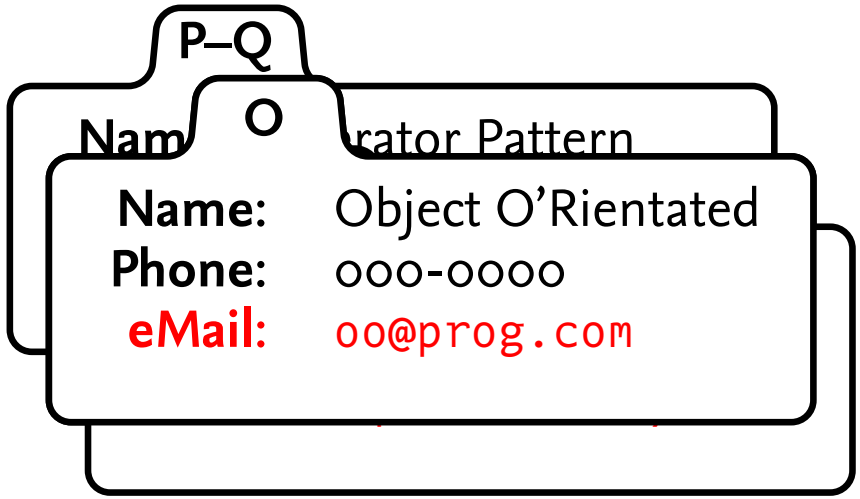
Objects are like Rolodex Cards

eMail



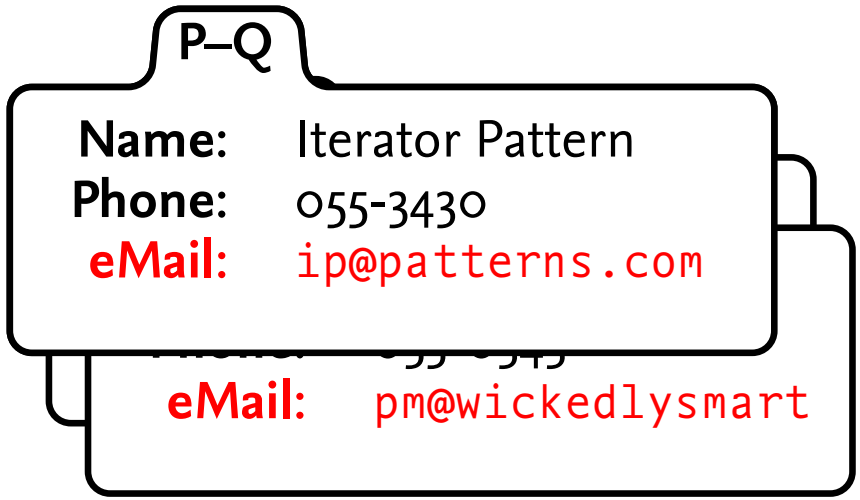
Objects are like Rolodex Cards

eMail



Objects are like Rolodex Cards

eMail



How Does it Work in Java?

The main() Defined the Main Execution Thread

Java

```
public class Dog {
  public String breed;
  public double size;


  public void bark( ) {
    if (size > 1.0) {
      System.out.println( "Ruff" );
    } else {
      System.out.println( "Bark" );
    }
  }
}
```

Java

```
public class Main {
  public static void main( String[] args ) {
    Dog john = new Dog( );
    john.breed = "Bulldog";
    john.size = 2.0;
    Dog lucky = new Dog( );
    lucky.breed = "Terrier";
    lucky.size = 0.5;

    john.bark( ); // Ruff
    lucky.bark( ); // Bark
  }
}
```

How Does it Work in Java?

Create new Dog

Java

```
public class Dog {
  public String breed;
  public double size;


  public void bark( ) {
    if (size > 1.0) {
      System.out.println( "Ruff" );
    } else {
      System.out.println( "Bark" );
    }
  }
}
```

Java

```
public class Main {
  public static void main( String[] args ) {
    Dog john = new Dog( );
    john.breed = "Bulldog";
    john.size = 2.0;
    Dog lucky = new Dog( );
    lucky.breed = "Terrier";
    lucky.size = 0.5;

    john.bark( ); // Ruff
    lucky.bark( ); // Bark
  }
}
```

How Does it Work in Java?

Set John's Breed

Java

```
public class Dog {
  public String breed;
  public double size;


  public void bark( ) {
    if (size > 1.0) {
      System.out.println( "Ruff" );
    } else {
      System.out.println( "Bark" );
    }
  }
}
```

Java

```
public class Main {
  public static void main( String[] args ) {
    Dog john = new Dog( );
    john.breed = "Bulldog";
    john.size = 2.0;
    Dog lucky = new Dog( );
    lucky.breed = "Terrier";
    lucky.size = 0.5;

    john.bark( ); // Ruff
    lucky.bark( ); // Bark
  }
}
```

How Does it Work in Java?

Set John's Size

Java

```
public class Dog {
  public String breed;
  public double size;


  public void bark( ) {
    if (size > 1.0) {
      System.out.println( "Ruff" );
    } else {
      System.out.println( "Bark" );
    }
  }
}
```

Java

```
public class Main {
  public static void main( String[] args ) {
    Dog john = new Dog( );
    john.breed = "Bulldog";
    john.size = 2.0;
    Dog lucky = new Dog( );
    lucky.breed = "Terrier";
    lucky.size = 0.5;

    john.bark( ); // Ruff
    lucky.bark( ); // Bark
  }
}
```

How Does it Work in Java?

Create new Dog

Java

```
public class Dog {
  public String breed;
  public double size;


  public void bark( ) {
    if (size > 1.0) {
      System.out.println( "Ruff" );
    } else {
      System.out.println( "Bark" );
    }
  }
}
```

Java

```
public class Main {
  public static void main( String[] args ) {
    Dog john = new Dog( );
    john.breed = "Bulldog";
    john.size = 2.0;
    Dog lucky = new Dog( );
    lucky.breed = "Terrier";
    lucky.size = 0.5;

    john.bark( ); // Ruff
    lucky.bark( ); // Bark
  }
}
```

How Does it Work in Java?

Set Lucky's Breed

Java

```
public class Dog {
  public String breed;
  public double size;


  public void bark( ) {
    if (size > 1.0) {
      System.out.println( "Ruff" );
    } else {
      System.out.println( "Bark" );
    }
  }
}
```

Java

```
public class Main {
  public static void main( String[] args ) {
    Dog john = new Dog( );
    john.breed = "Bulldog";
    john.size = 2.0;
    Dog lucky = new Dog( );
    lucky.breed = "Terrier";
    lucky.size = 0.5;

    john.bark( ); // Ruff
    lucky.bark( ); // Bark
  }
}
```

How Does it Work in Java?

Set Lucky's Size

Java

```
public class Dog {
  public String breed;
  public double size;


  public void bark( ) {
    if (size > 1.0) {
      System.out.println( "Ruff" );
    } else {
      System.out.println( "Bark" );
    }
  }
}
```

Java

```
public class Main {
  public static void main( String[] args ) {
    Dog john = new Dog( );
    john.breed = "Bulldog";
    john.size = 2.0;
    Dog lucky = new Dog( );
    lucky.breed = "Terrier";
    lucky.size = 0.5;

    john.bark( ); // Ruff
    lucky.bark( ); // Bark
  }
}
```

How Does it Work in Java?

Make John Bark

Java

```
public class Dog {
  public String breed;
  public double size;


  public void bark( ) {
    if (size > 1.0) {
      System.out.println( "Ruff" );
    } else {
      System.out.println( "Bark" );
    }
  }
}
```

Java

```
public class Main {
  public static void main( String[] args ) {
    Dog john = new Dog( );
    john.breed = "Bulldog";
    john.size = 2.0;
    Dog lucky = new Dog( );
    lucky.breed = "Terrier";
    lucky.size = 0.5;

    john.bark( ); // Ruff
    lucky.bark( ); // Bark
  }
}
```

How Does it Work in Java?

Make Lucky Bark

Java

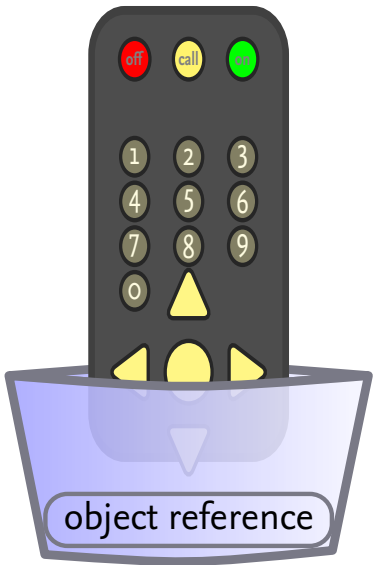
```
public class Dog {
  public String breed;
  public double size;


  public void bark( ) {
    if (size > 1.0) {
      System.out.println( "Ruff" );
    } else {
      System.out.println( "Bark" );
    }
  }
}
```

Java

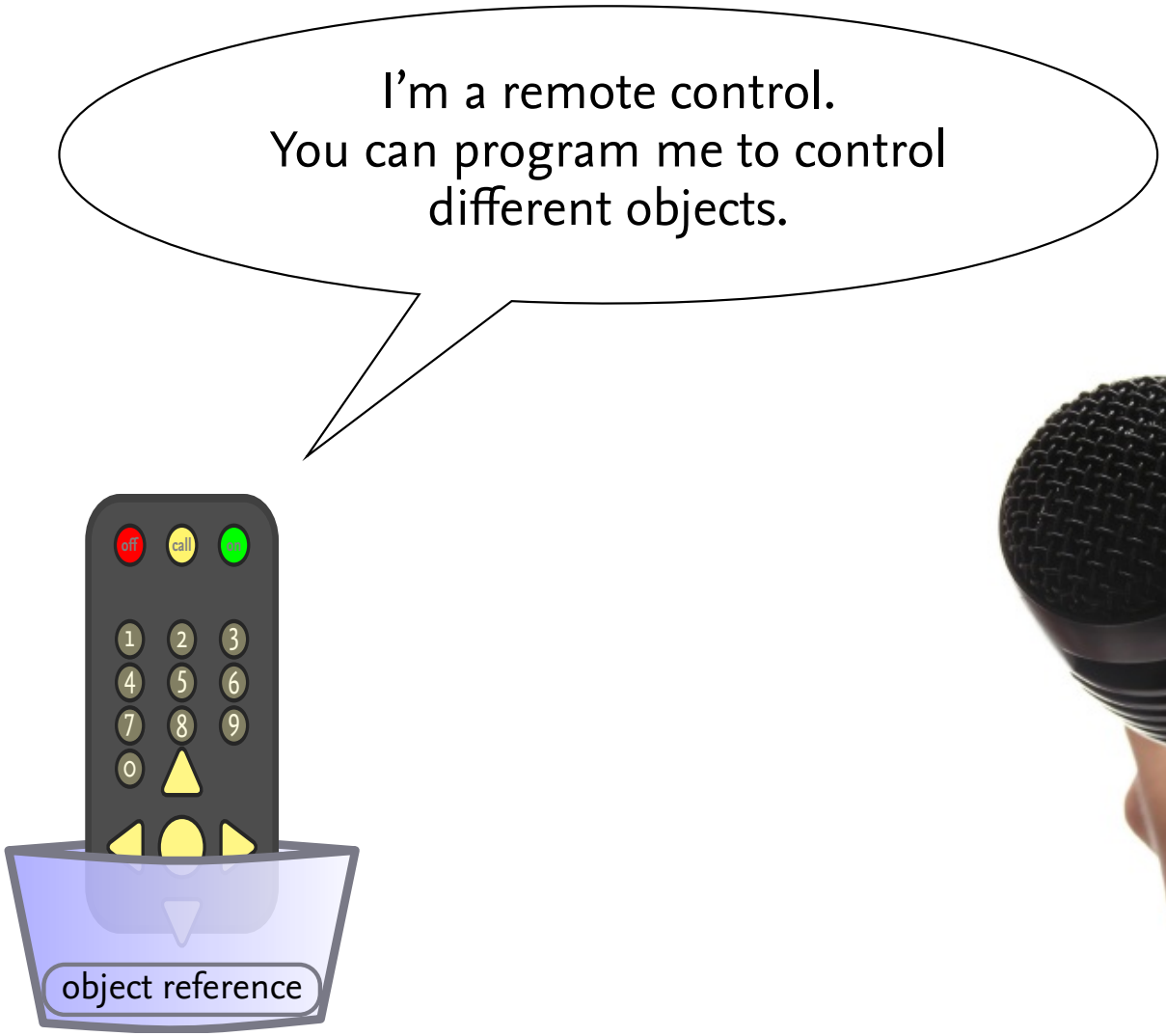
```
public class Main {
  public static void main( String[] args ) {
    Dog john = new Dog( );
    john.breed = "Bulldog";
    john.size = 2.0;
    Dog lucky = new Dog( );
    lucky.breed = "Terrier";
    lucky.size = 0.5;

    john.bark( ); // Ruff
    lucky.bark( ); // Bark
  }
}
```

Interview with an Object Reference Variable

So, what's it like to be an object reference?

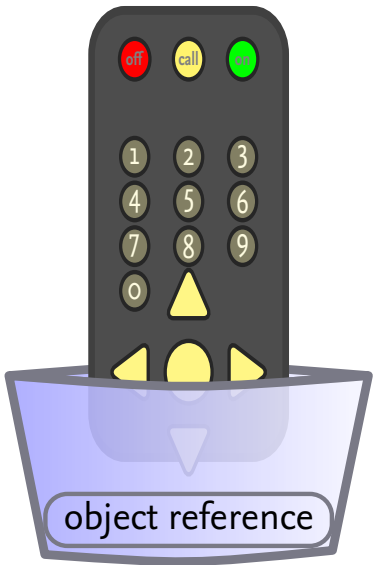


Interview with an Object Reference Variable



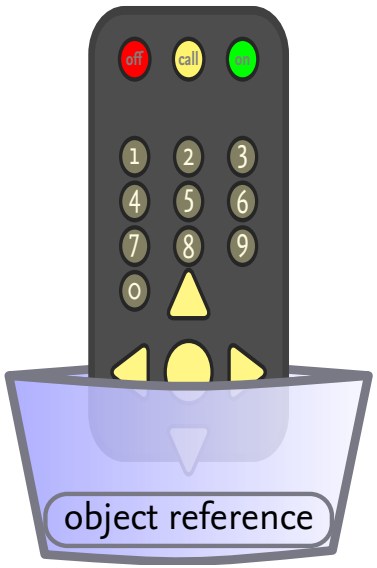
Interview with an Object Reference Variable

Cool. Do you mean different kinds of objects?
For example, a `Cat` object and a `House` object?



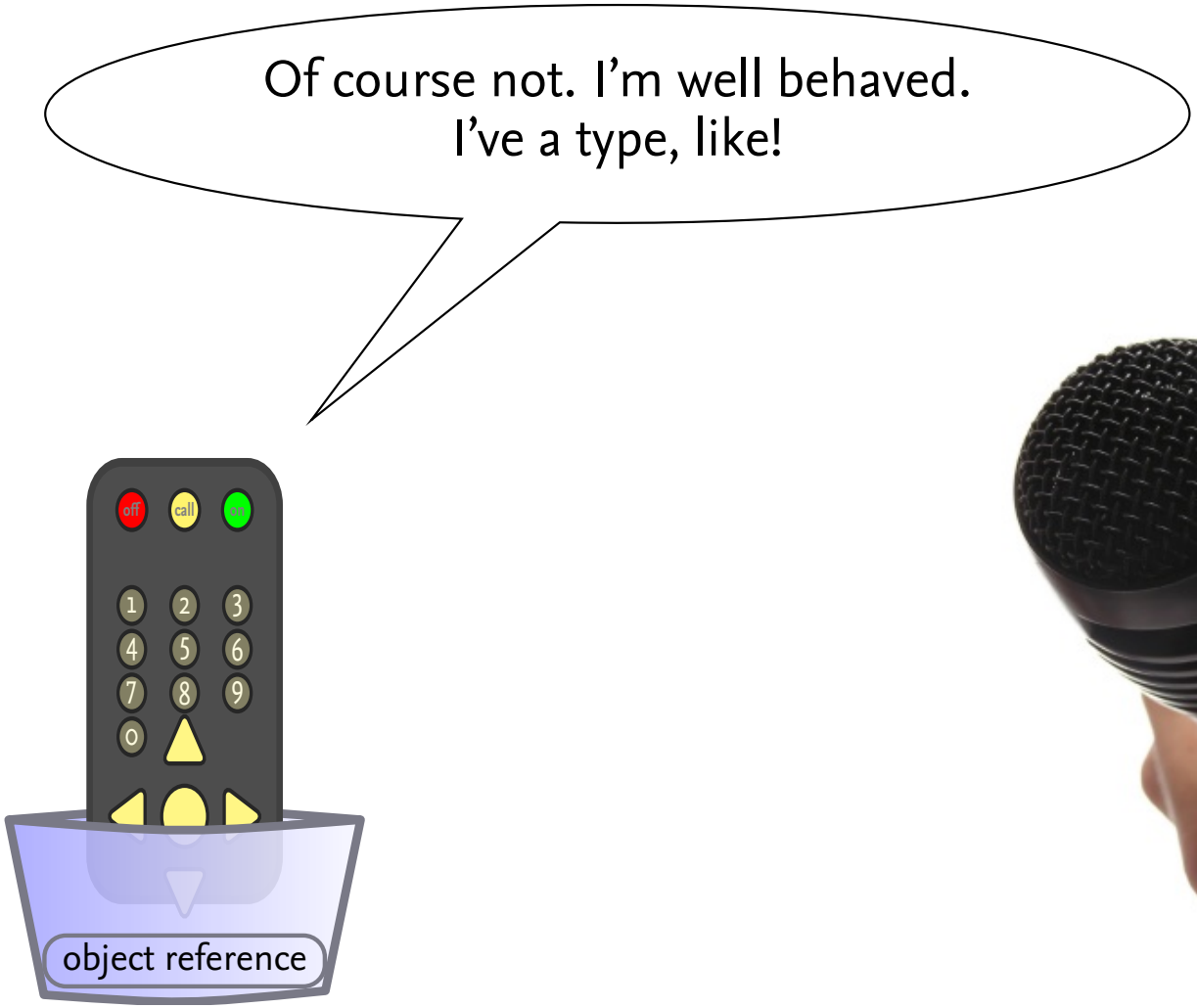
Interview with an Object Reference Variable

```
remote = new Cat( );
remote = new House( );?
```



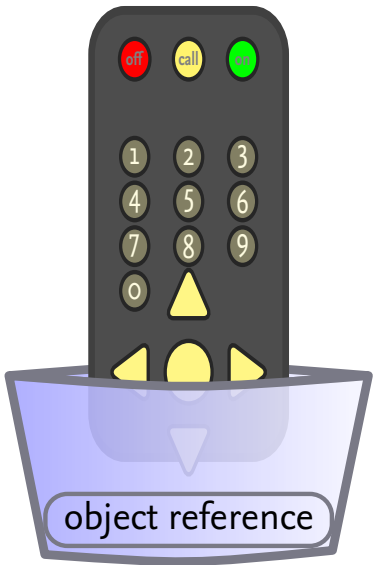
Interview with an Object Reference Variable

Interview with an Object Reference Variable



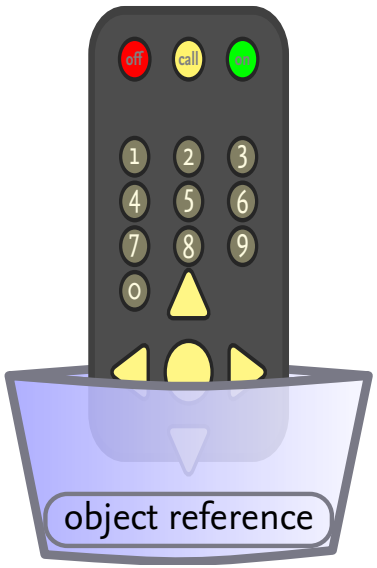
Interview with an Object Reference Variable

```
Cat remote;
remote = new House( ); // not allowed
```



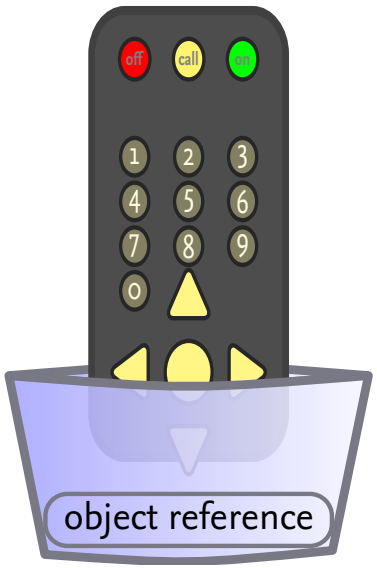
Interview with an Object Reference Variable

But I don't understand.
They told me you can control
several, different, objects.

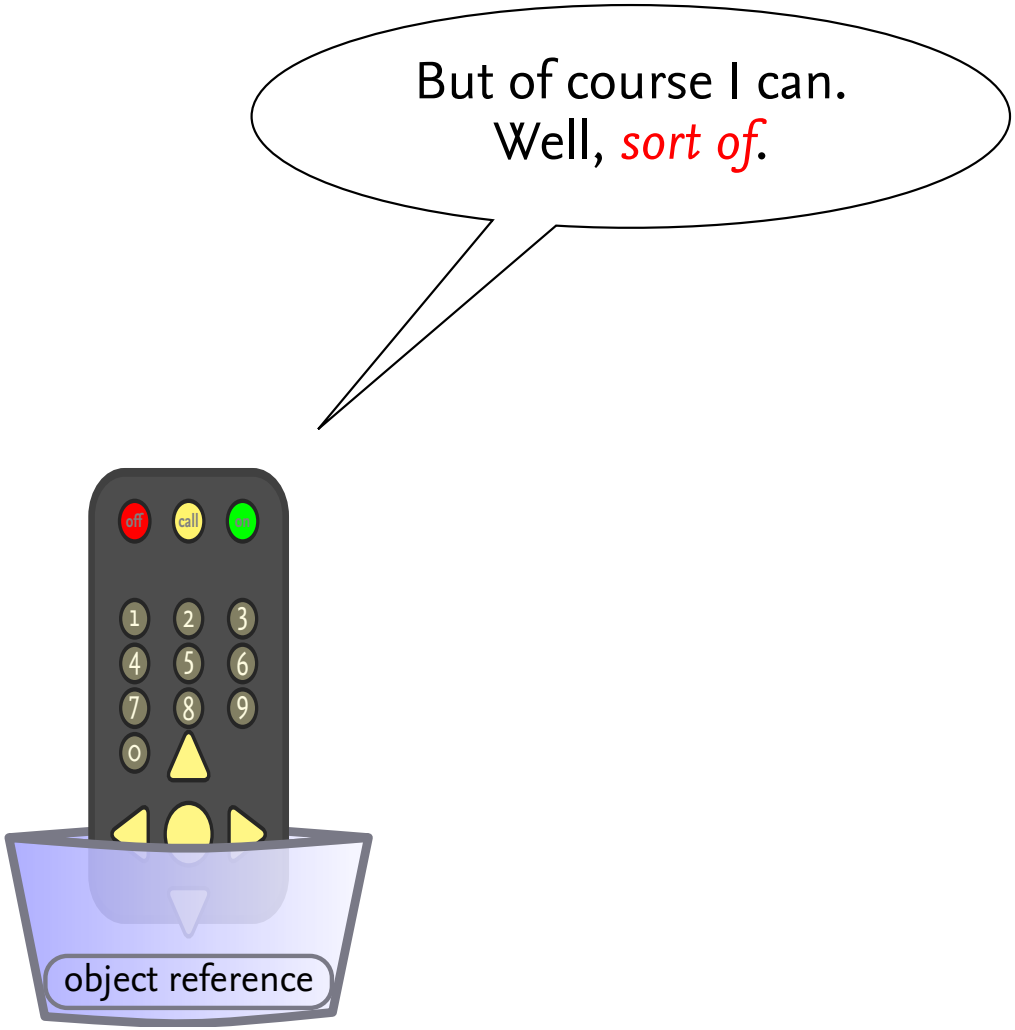


Interview with an Object Reference Variable

The people they send out
for interviews these days....

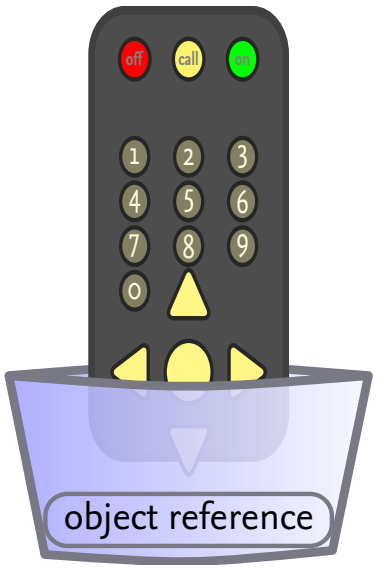


Interview with an Object Reference Variable



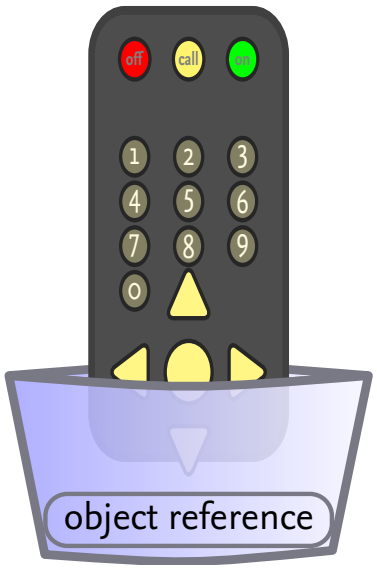
Interview with an Object Reference Variable

```
Cat remote = new Cat( );
Cat other = new Cat( );
remote = other;
```



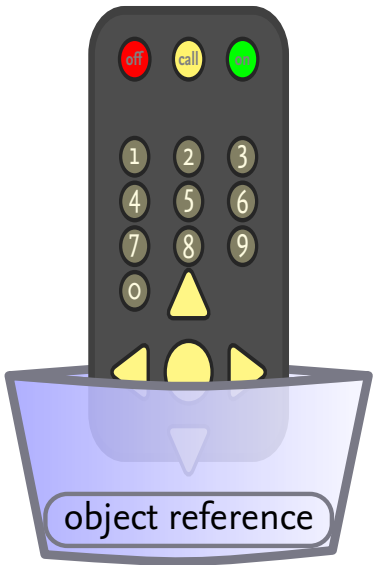
Interview with an Object Reference Variable

Ah, I see, but what about that “sort of?”



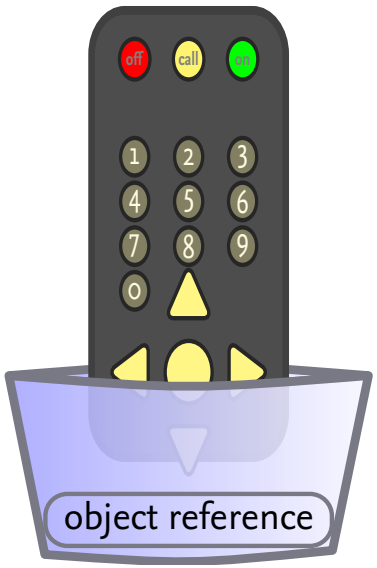
Interview with an Object Reference Variable

There are two things really. The first is when I'm `final`: `final` variables can only take one value.



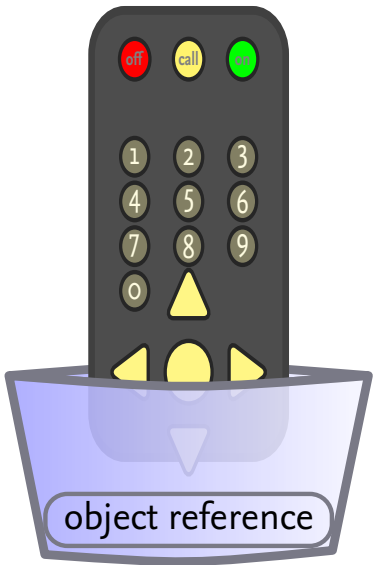
Interview with an Object Reference Variable

```
final Dog remote = new Dog( );
remote = new Dog( ); // not allowed
```



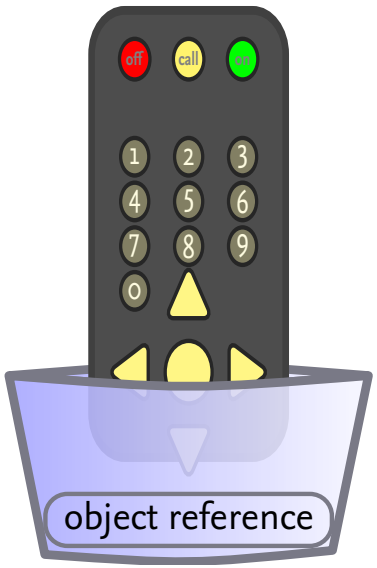
Interview with an Object Reference Variable

The second thing is when my type is `Object`.
As `Object` I can take *any object* reference value.



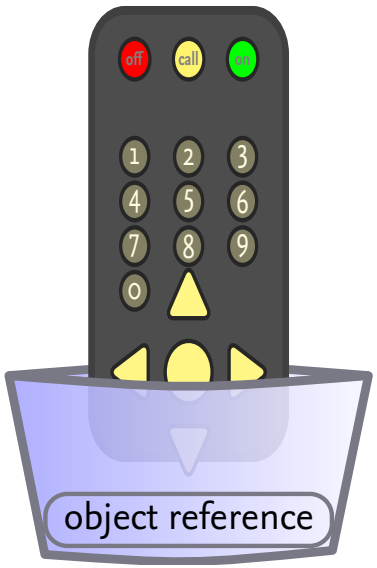
Interview with an Object Reference Variable

```
Bun bun = new Bun( "Crunchy" );
Object filler = new Beef( "Guaranteed Irish" );
filler = new Horse( "Mr Ed" );
bun.fill( filler ); // beef crisis
```



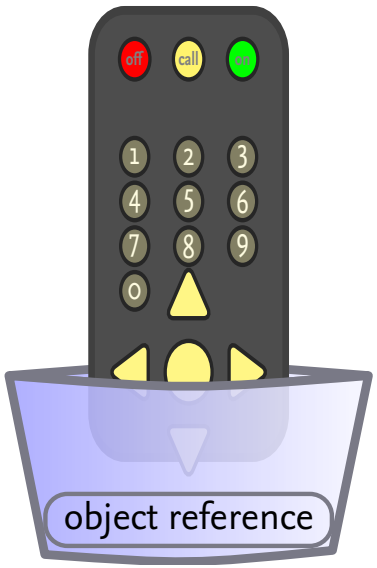
Interview with an Object Reference Variable

The `Object` type makes me paranoid. It's just not safe. You can be assigned any object reference value. Including reference values that aren't safe.

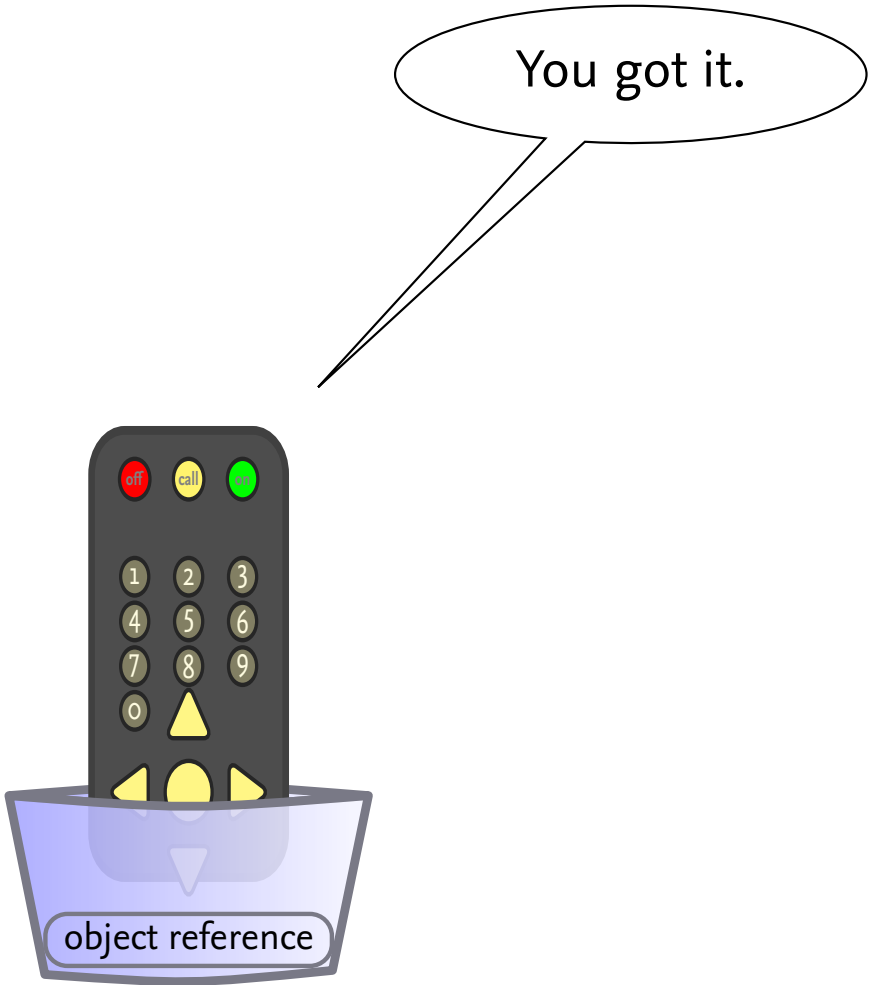


Interview with an Object Reference Variable

I get it. You're saying that Java is stronly typed. Its types help prevent certain errors.

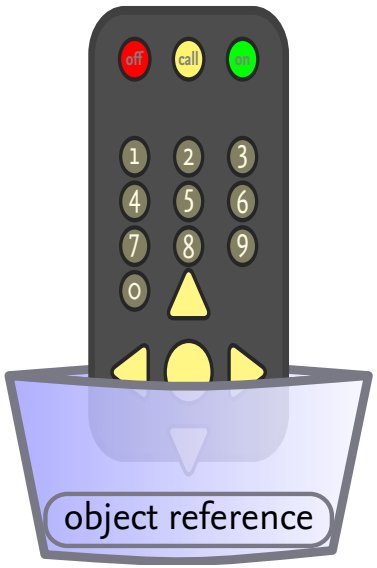


Interview with an Object Reference Variable



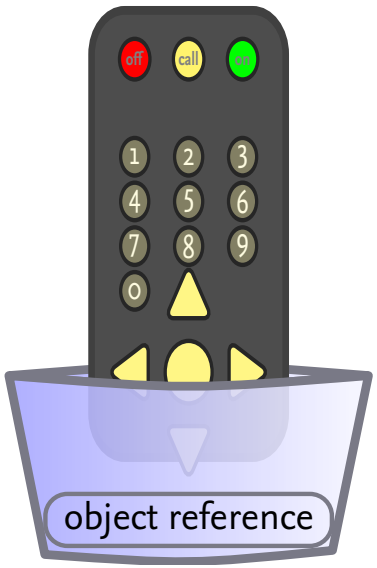
Interview with an Object Reference Variable

This is the right moment.
I've got to ask him now.



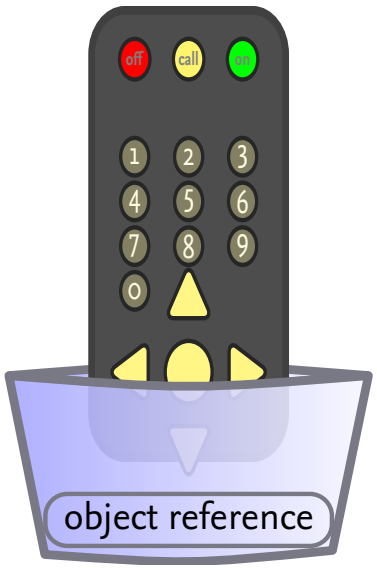
Interview with an Object Reference Variable

Can you say something about those `null` values? I've heard they're a bit of a taboo subject among object reference variables.



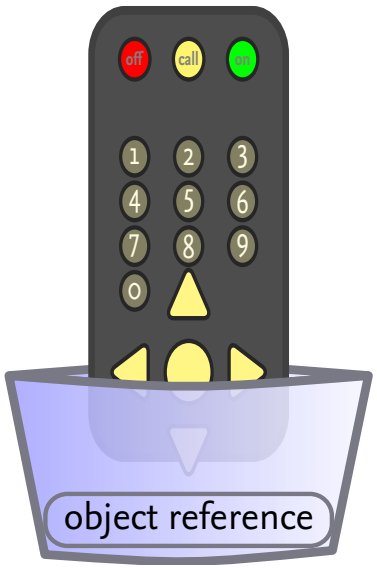
Interview with an Object Reference Variable

I'd rather not talk about it.
It's just too embarassing.



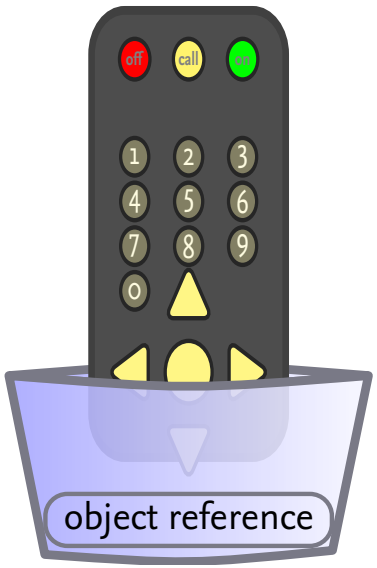
Interview with an Object Reference Variable

I'm sorry to hear that, but wouldn't you at least give it a try? Our audience would be really interested in this.



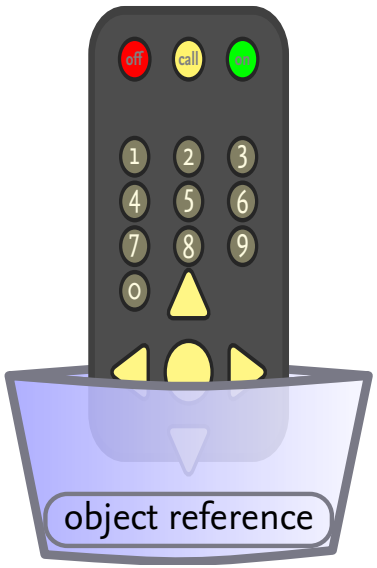
Interview with an Object Reference Variable

Well, *any* object reference variable
can be assigned the value `null`.
It's the worst thing that can happen to us.

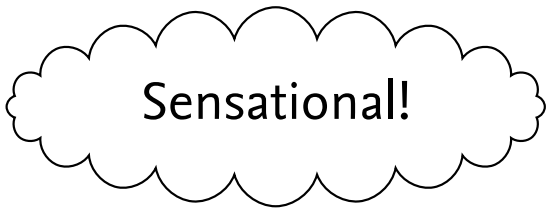


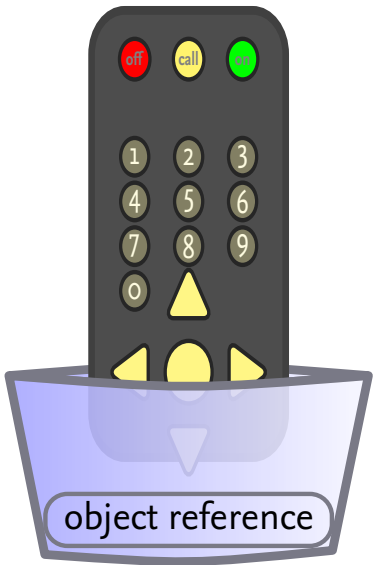
Interview with an Object Reference Variable

```
Dog remote = null;
Cat cat = null;
```

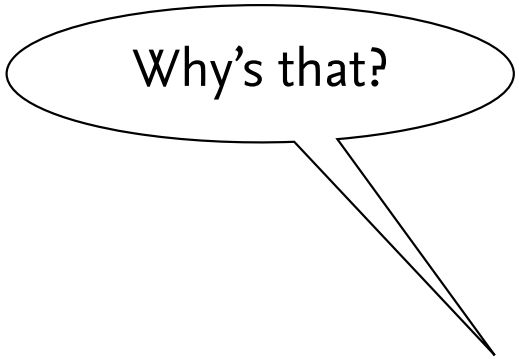


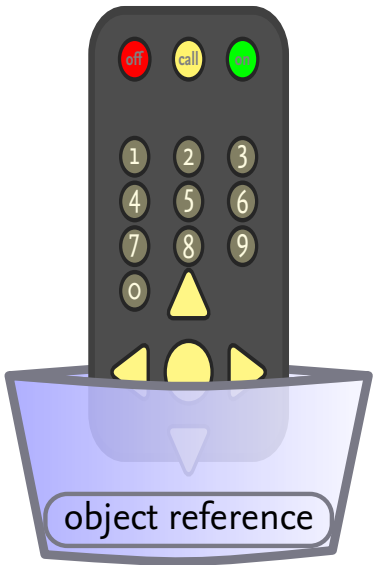
Interview with an Object Reference Variable





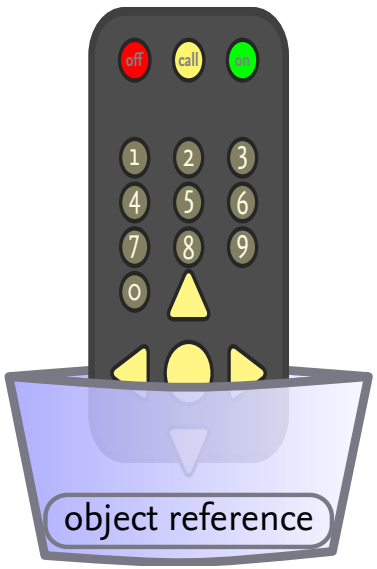
Interview with an Object Reference Variable





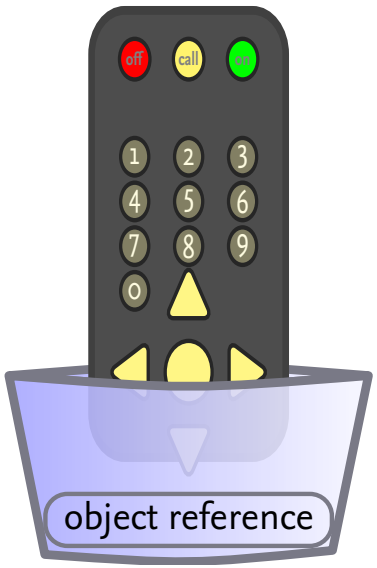
Interview with an Object Reference Variable

Well, when that happens, you're nothing. It's like unsubscribing from SKY: When you do that, your SKY remote becomes useless.



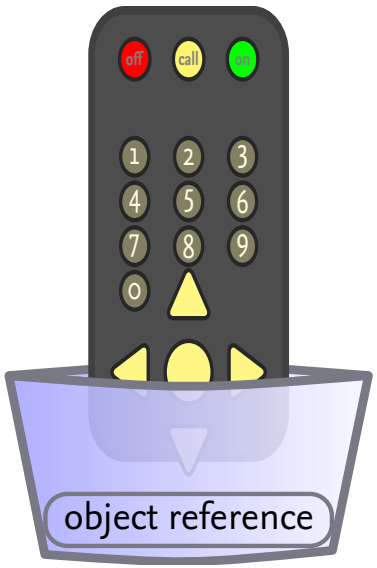
Interview with an Object Reference Variable

For us object reference variables it's even worse. When we are `null` we're not even allowed to *try* control objects.



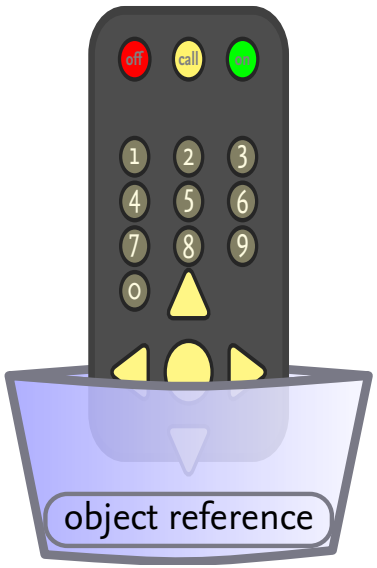
Interview with an Object Reference Variable

```
Dog remote = null;
remote.bark( ); // run-time error.
```



Interview with an Object Reference Variable

Well thanks. That was very insightful.
Have a nice day.



Interview with an Object Reference Variable

Question Time

Questions Anybody?

For Next Friday

- Get the book.
- Study Chapter 1.
- Read Chapter 2.

Acknowledgements

- This lecture is partially based on
  - [Sierra, and Bates 2004, Chapter 1].

Bibliography I


Sierra, Kathy, and Bert Bates [2004]. *Head First Java*. O'Reilly. ISBN: 978-0-596-00712-6.

About this Document

- This document was created with `pdflatex`.
- The LaTeX document class is `beamer`.