Reflection: Introduction

CPSC 501: Advanced Programming Techniques
Fall 2020

Jonathan Hudson, Ph.D
Instructor
Department of Computer Science
University of Calgary

Wednesday, August 5, 2020

UNIVERSITY OF CALGARY

Just the basics

Definition: Two parter

- ***Reflection*** is the ability of a running program to:

1. Examine itself and the run-time environment
   - Called introspection

2. Change its behavior, structure, or data depending on what it finds

Introspection via?

- To do **introspection**, a program must have a **representation of itself** available at runtime
  - Called ***metadata***
  - In an OO language, metadata is organized using **metaobjects**
    - In Java, these are typically instances of classes like Class, Method, and Field

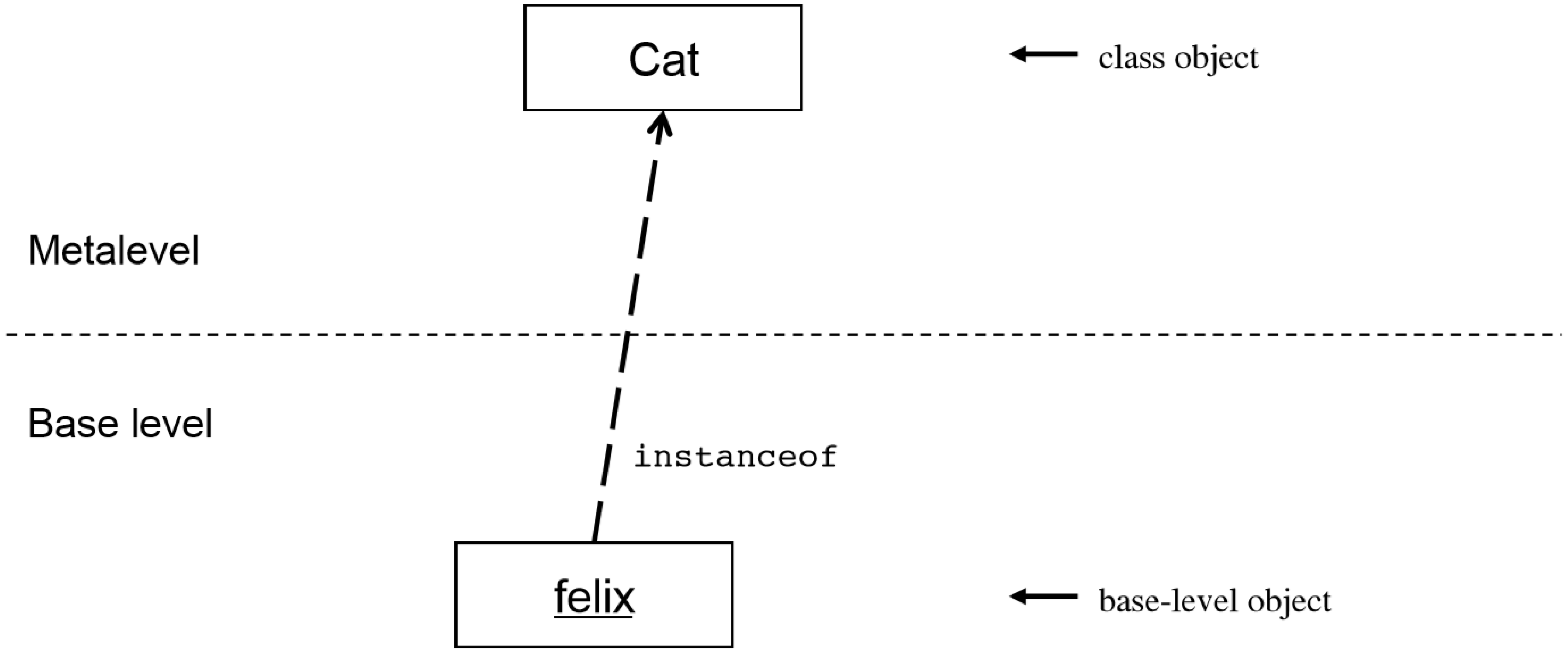
You basic, you meta

You basic, you meta

- The normal, non-reflective part of a program is called the base program
  - Consists of ***base-level objects***

- Each base-level object is an instance of some class
  - The class is represented at the **metalevel** as a **class object** (an example of a metaobject)

Basic Concepts



Basic Concepts

- The fields and methods for a class are represented with **Field** and **Method** metaobjects
  - Are contained within the class object

Flip it, and reverse it

Basic Concepts

- Once introspection is done, you can change a program’s structure, data, or behavior
  - Three general techniques:
    1. Direct metaobject modification
       - E.g. Add methods or fields to an existing class
       - **Not possible in Java (avoids complications)**
    2. Operations using metadata
       - E.g. Dynamic method invocation, dynamic class loading, reflective construction
       - **Exists in Java**

Basic Concepts

3. Intercession
    - Where code intercedes modifies behavior as program runs
    - Typically involves intercepting method calls
    - **In Java, limited to dynamic proxies**

Work it

Basic Concepts

- Growing number of languages support reflection
  - To some degree
  - This list is growing due to the power of it
  - Go, Java, Julia, Lisp, Logo, La, Mathematica, C#, Perl, PHP, Prolog, Python, R, Ruby, Scheme, Smalltalk, Wolfram language

Basic Concepts

- Issues with reflection:
  - Since behavior can be changed dynamically, security could be compromised
    - Not a problem with Java
      - Has a strong security model
      - Limited intercession
  - Reflective techniques are indirect, thus making code more complex
    - Use reflection only where it makes sense

Basic Concepts

- Reflective method calls are slower than normal calls
    - 20x improvement from Java 1.3 to 1.4

Do it already

A Simple Example

```java
import java.lang.reflect.*;

public class Test {

    public static void main(String[] arg) {
        Object object = null;
        Class classObject = null;
        try {                // Load the class dynamically using
            // 1st command-line arg
            classObject = Class.forName(arg[0]);
            // Create an instance of the class
            object = classObject.newInstance();
        } catch (InstantiationException e) {
            System.out.println(e);
            return;
        } catch (IllegalAccessException e) {
            System.out.println(e);
            return;
        } catch (ClassNotFoundException e) {
            System.out.println(e);
            return;
        }
```

A Simple Example

```
        try {
            // Find the no-arg method named by
            // 2nd command line arg
            Method m = classObject.getMethod(arg[1], null);
            // Invoke the method on the object
            m.invoke(object, null);
        } catch (NoSuchMethodException e) {
            System.out.println(e);
        } catch (IllegalAccessException e) {
            System.out.println(e);
        } catch (InvocationTargetException e) {
            System.out.println(e);
        }
    }
}
```

A Simple Example

- Can be used on any class
    - Example:

```
public class MyClass {

    public void print() {
        System.out.println("Hello, world!");
    }

    public void display() {
        System.out.println("Goodbye, cruel world!");
    }
}
```

A Simple Example

java Test MyClass print

outputs:**Hello, world!**

java Test MyClass display

outputs: **Goodbye, cruel world!**

Onward to …
Java reflection.

Jonathan Hudson
jwhudson@ucalgary.ca
https://pages.cpsc.ucalgary.ca/~hudsonj/

UNIVERSITY OF CALGARY