Reflection: Introduction

CPSC 501: Advanced Programming Techniques
Winter 2025

Jonathan Hudson, Ph.D
Assistant Professor (Teaching)
Department of Computer Science
University of Calgary

Wednesday, March 5, 2025

Copyright © 2025

UNIVERSITY OF CALGARY

Just the basics

UNIVERSITY OF CALGARY

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

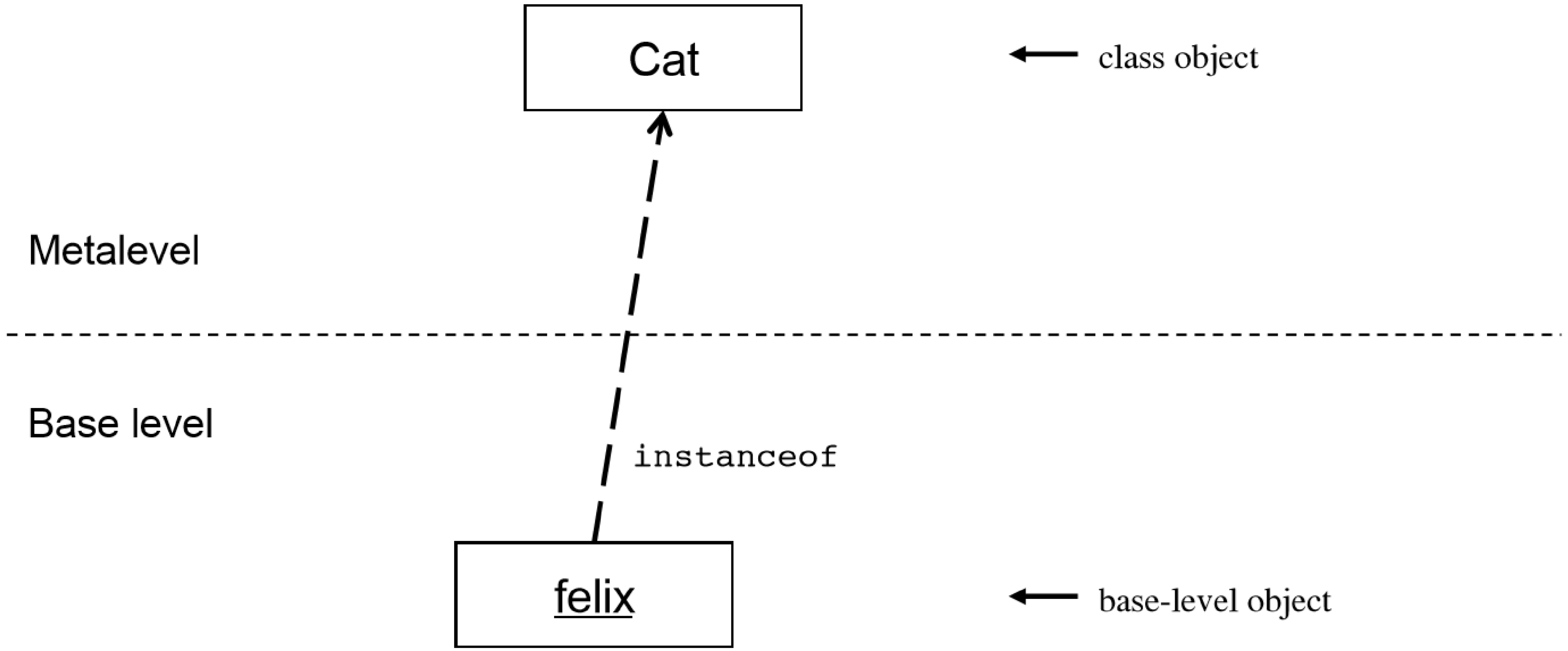
UNIVERSITY OF CALGARY

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

UNIVERSITY OF CALGARY

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)**
       - **Possible in Python**
    2. Operations using metadata
       - E.g. Dynamic method invocation, dynamic class loading, reflective construction
       - **Exists in Java, Python**

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

UNIVERSITY OF CALGARY

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 big problem with Java
      - Has a strong security model
      - Limited intercession
    - Easy to do in Python!!!!

  - Reflective techniques are indirect, thus making code more complex
    - Use reflection only where it makes sense

Do it already

UNIVERSITY OF CALGARY

A Simple Example

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

```
import java.lang.reflect.Method;

public class MainReflect {
   public static void main(String[] args) {
      Object object = null;
      Class classObject = null;
      try {
         // Load the class dynamically
         //1st command line argument
         classObject = Class.forName(args[0]);
         object = classObject.newInstance();
         //Find method by name in
         //2nd command line argument
         Method m = classObject.getMethod(args[1], null);
         m.invoke(object, null);
      } catch (Exception e) {
         e.printStackTrace();
      }
   }
}
```

A Simple Example

java Reflection.MainReflect Reflection.MyClass print

outputs:**Hello, world!**

java Reflection.MainReflect Reflection.MyClass display

outputs: **Goodbye, cruel world!**

Python?

UNIVERSITY OF CALGARY

A Simple Example – Type to Constructor

```python
#Reverse anything that slices
def reverse(sequence):
    #Dynamic constructor
    empty_sequence = type(sequence)()
    if sequence == empty_sequence:
        return empty_sequence
    final_result = reverse(sequence[1:]) + sequence[0:1]
    return final_result
print(reverse([10, 20, 30, 40]))
print(reverse("Jonathan"))
```

```
[40, 30, 20, 10]
nahtanoJ
```

A Simple Example – Constructor by name

```
class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y
class_name = 'Point'
kwargs = {'x': 14.361, 'y': -8.100}
ReflPoint = globals()[class_name]
point = ReflPoint(**kwargs)
print(type(point),point.x,point.y)
```

```
<class '__main__.Point'> 14.361 -8.1
```

A Simple Example – function by name

```python
class Baz:
    def foo(self):
        print("Found me")

w = Baz()
getattr(w,"foo")()
```

```
Found me
```

Same As Java

```
class MyClass:
    def print(self):
        print("Hello, world!")
    def display(self):
        print("Goodbye, cruel world!")

import sys
def main():
    getattr(globals()[sys.argv[1]](), sys.argv[2])()

main()
```

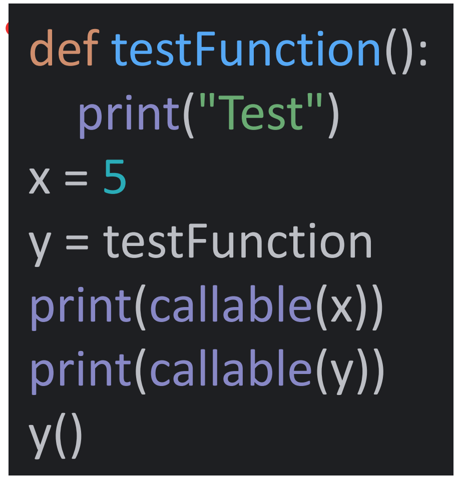
python MainReflect.py MyClass print

outputs:**Hello, world!**

python MainReflect.py MyClass display

outputs: **Goodbye, cruel world!**

A Simple Example – callable



```python
def testFunction():
    print("Test")
x = 5
y = testFunction
print(callable(x))
print(callable(y))
y()
```

```
False
True
Test
```

A Simple Example – attributes

```python
print(dir())
print(dir([1,2,3]))
class Foo():
    name = "Jonathan"
    job = "Assistant Professor"
z = Foo()
print(dir(z))
z.JOB = "Associate Professor"
print(dir(z))
delattr(type(z), "job")
print(dir(z))
print(getattr(z, "name"))
```

```
['__annotations__', '__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__',
'__package__', '__spec__']
['__add__', '__class__', '__class_getitem__', '__contains__', '__delattr__', '__delitem__', '__dir__',
'__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getstate__', '__gt__',
'__hash__', '__iadd__', '__imul__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__',
'__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__reversed__', '__rmul__',
'__setattr__', '__setitem__', '__sizeof__', '__str__', '__subclasshook__', 'append', 'clear', 'copy', 'count',
'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort']
['__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__',
'__getattribute__', '__getstate__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__le__', '__lt__',
'__module__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__sizeof__',
'__str__', '__subclasshook__', '__weakref__', 'job', 'name']
['JOB', '__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__',
'__getattribute__', '__getstate__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__le__', '__lt__',
'__module__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__sizeof__',
'__str__', '__subclasshook__', '__weakref__', 'job', 'name']
['JOB', '__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__',
'__getattribute__', '__getstate__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__le__', '__lt__',
'__module__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__sizeof__',
'__str__', '__subclasshook__', '__weakref__', 'name']
```

Onward to …
more Java reflection.

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

UNIVERSITY OF CALGARY