CSC 7322 : Object Oriented Development

J Paul Gibson, A207

paul.gibson@telecom-sudparis.eu

http://www-public.telecom-sudparis.eu/~gibson/Teaching/CSC7322/

Reflection (in Java)

…/~gibson/Teaching/CSC7322/L9-Reflection.pdf

Reflection

Reflection is the process by which a program can read its own **metadata** (data about data).

A program is said to reflect on itself, extracting metadata from its assembly and using that metadata either to inform the user or to modify its own behavior.

In an object-oriented world, metadata is organized into objects, called **metaobjects**. The runtime self-examination of the metaobjects is called **introspection**.

Reflection is important since it lets you write programs that do not have to "know" everything at compile time, making them more **dynamic**, since they can be tied together at **runtime**.

Applications programmed (cleanly) with reflection **adapt** more easily to **changing requirements**. Well programmed reflective components are more likely to be **reused** flawlessly in other applications.

Reflection is dangerous

You can use reflection to access private attributes and methods:

private/public/protected – for scoping (not security)

Use of reflection methods is normally checked by the security manager:

Applets are always run with the security manager, but mostly Java code is not (unless specified)

Question: why/when would you like to invoke a private method?

Reflection: Some Background/Further Reading

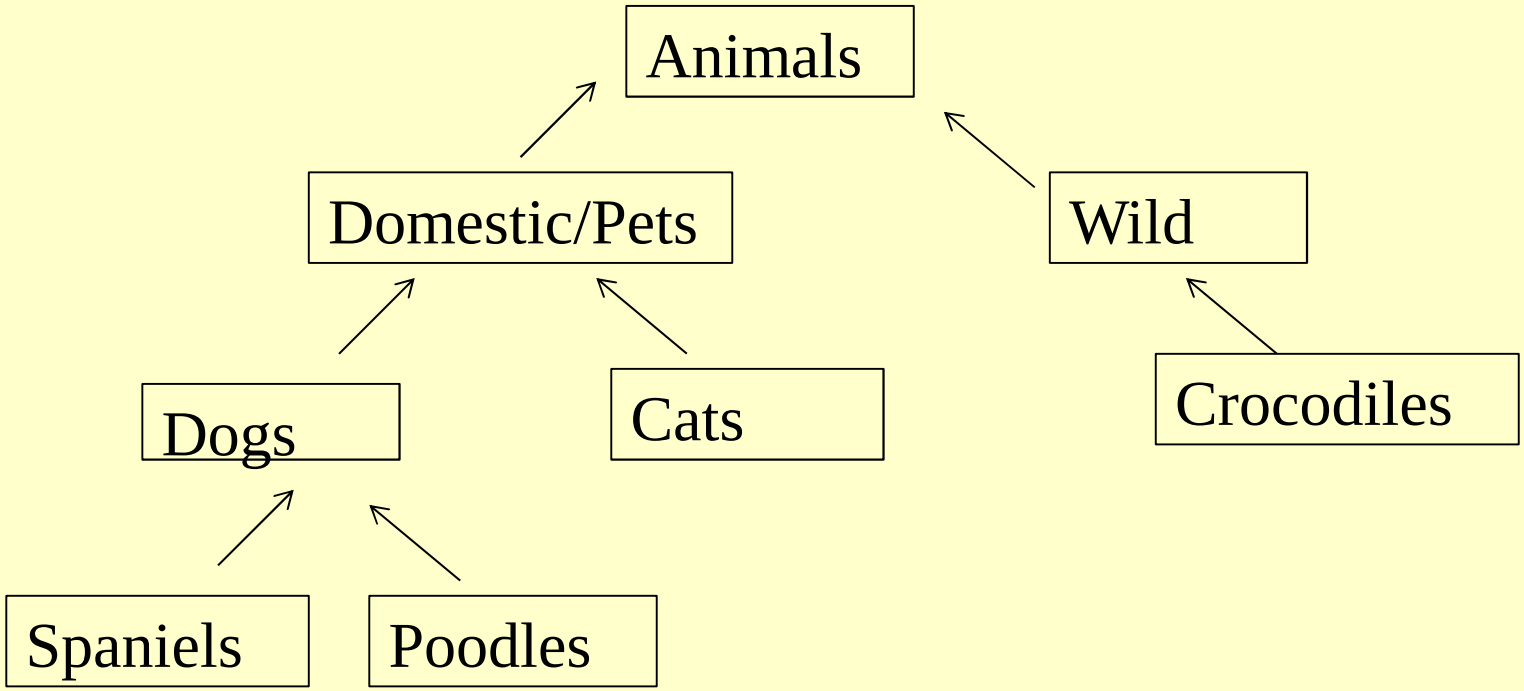
Reflection and semantics in LISP. **Brian Cantwell Smith. Proceedings of the 11th ACM SIGACT-SIGPLAN symposium on Principles of programming languages (POPL '84). ACM**

Reflection in logic, functional and object-oriented programming: a Short Comparative Study, **Francois-Nicola Demers and Jacques Malenfant, Proc. of the IJCAI'95 Workshop on Reflection and Metalevel Architectures and their Applications in AI. pp. 29–38. August 1995**

Java Reflection in Action (In Action Series). **Ira R. Forman and Nate Forman. 2004 Manning Publications Co., Greenwich, CT, USA.**

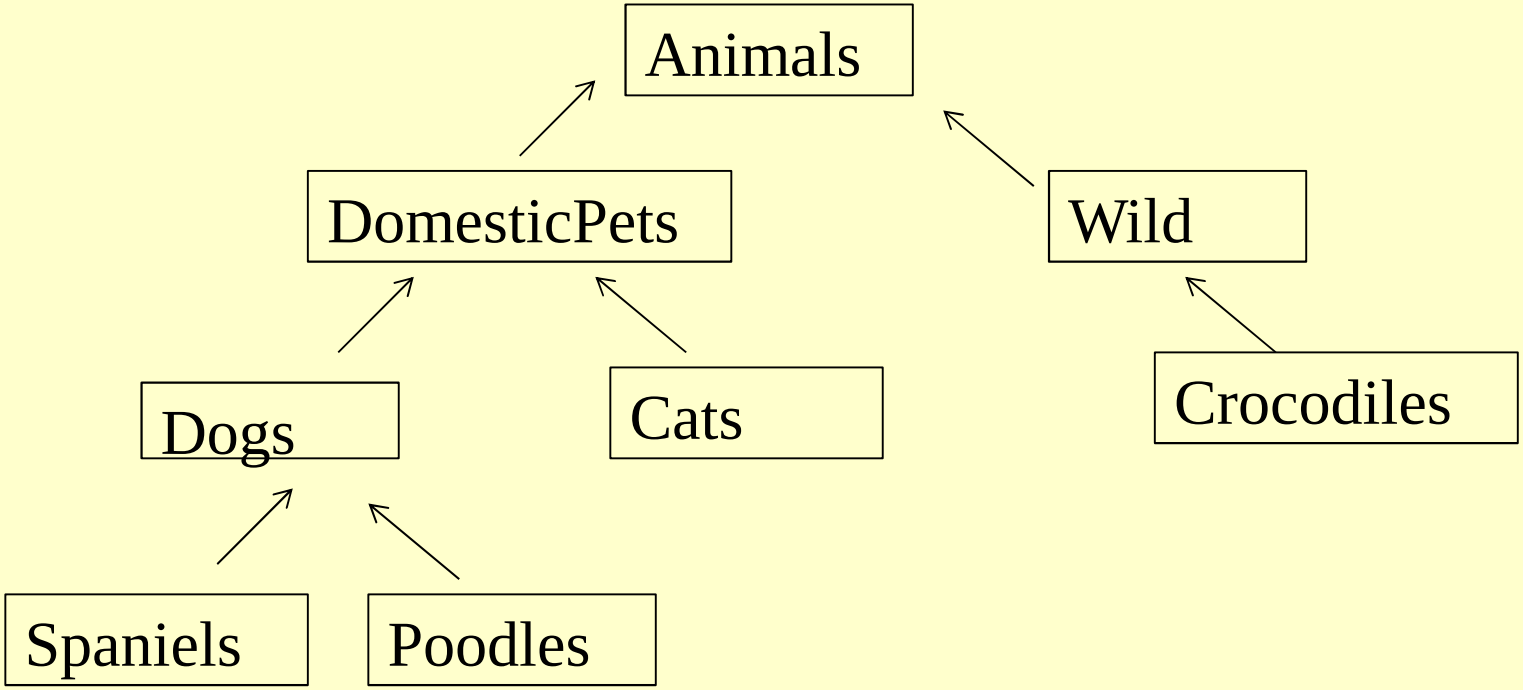
Reflection PBL – LeastAbstractCommonSuperclass (LACS)

Consider the following class hierarchy



Imagine if we had a generic collection of Animals and that we wished to examine all elements of the collection and find the least abstract subclass to which all these Animals belonged

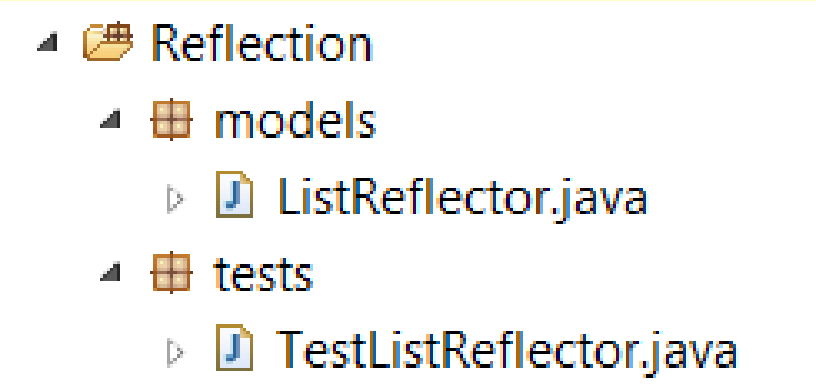
Reflection PBL - LeastAbstractCommonSuperclass



LACS {spaniel, poodle1, poddle2, cat, spaniel} = DomesticPets
LACS {spaniel, poodle, dog} = Dogs
LACS {spaniel, poodle, crocodile} = Animals

TO DO: Test the developed code on these 3 example cases

Reflection PBL - LeastAbstractCommonSuperclass



TO DO: Download the Reflection outline code from the website and try to understand what it is doing

The `ListReflector` class provides to methods that require the use of reflection:

void models.ListReflector.reflect(List<T> list)

Parameters:

<T> is the generic type of list elements

list is the list elements whose information (gathered using reflection) will be printed to the screen

Class<? extends Object> models.ListReflector.lowestCommonSuperclass(List<T> listOfObjects)

Parameters:

<T> is the abstract type/class of the List objects

listOfObjects

Returns:

the most concrete class of which all the list elements are members

@todo

The students are to write this code so that it functions correctly as tested in `TestListReflector`.

Reflection PBL - LeastAbstractCommonSuperclass

The `TestListReflector` should test the that the LACS requirement is correctly fulfilled

tests.TestListReflector

Author:

J Paul Gibson Template test code for reflection problem (OOD)

@todo

The students are to improve the test to show that the method `ListReflector.lowestCommonSuperclass` is currently not working correctly

They are then to fix the method `ListReflector.lowestCommonSuperclass` and show that their fix is correct (by executing the updated test)

Objective: You will learn about reflection from trying to solve this problem