

# CSC7336 : Advanced Software Engineering...

**J Paul Gibson, D311**

`paul.gibson@telecom-sudparis.eu`

`http://www-public.telecom-sudparis.eu/~gibson/Teaching/CSC7336/`

## **Reflection (in Java)**

`.../~gibson/Teaching/CSC7336/L4-Reflection.pdf`

# Reflection

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

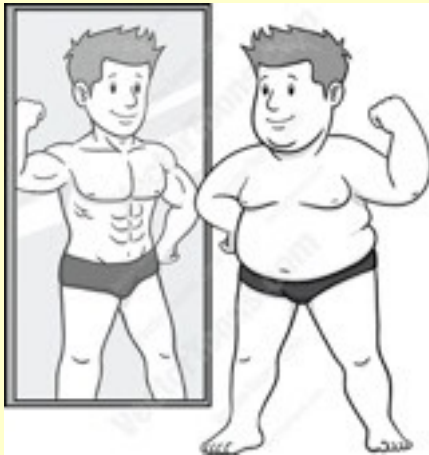
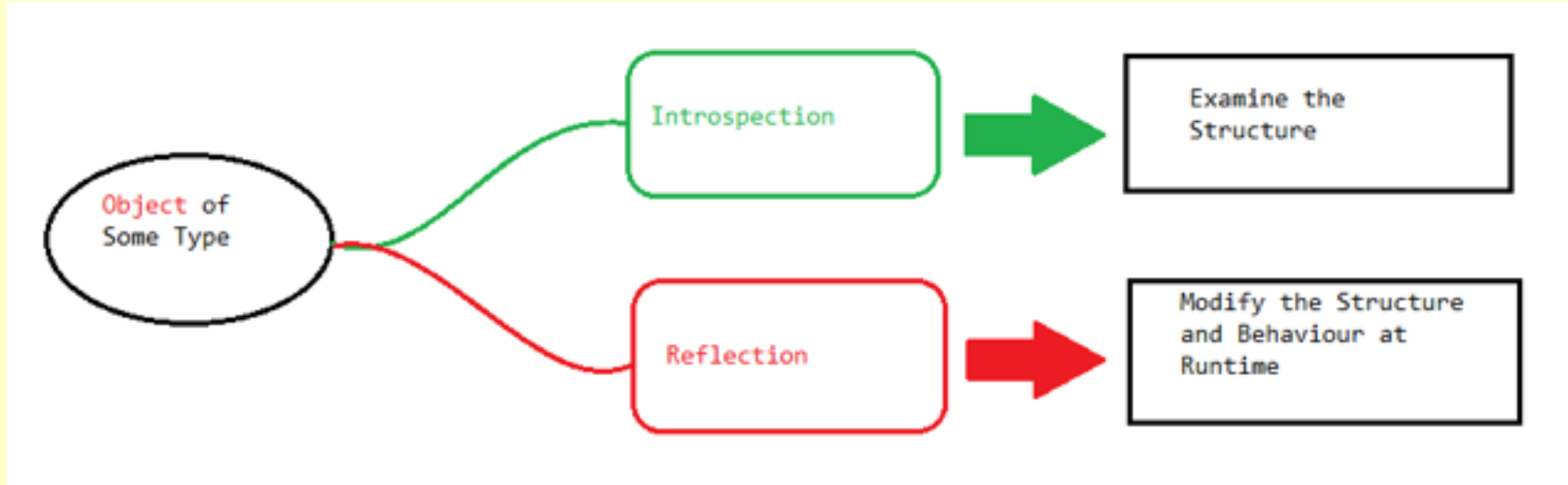
A program is said to reflect on itself, extracting metadata from its assembly and using that metadata either to inform the user or to modify its own behaviour.

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 and Introspection



**Can we trust what we see in the reflection?**

# 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 (in Java) is powerful

Using reflection, you can:

- Convert strings and others into classes and objects at runtime.
- Ask detailed questions in code about the abilities of a type.
- Dynamically compile, load, and add classes to a running program.
- Pass function pointers (via Method objects)

Reflection is used internally by many Java technologies including IDEs/compiler, debuggers, serialization, Java Beans, RMI, ...

# Reflection in self\* systems

Self-organizing

Self-healing

Self-managing

Self-stabilizing

Self-optimizing

Self-\* ???

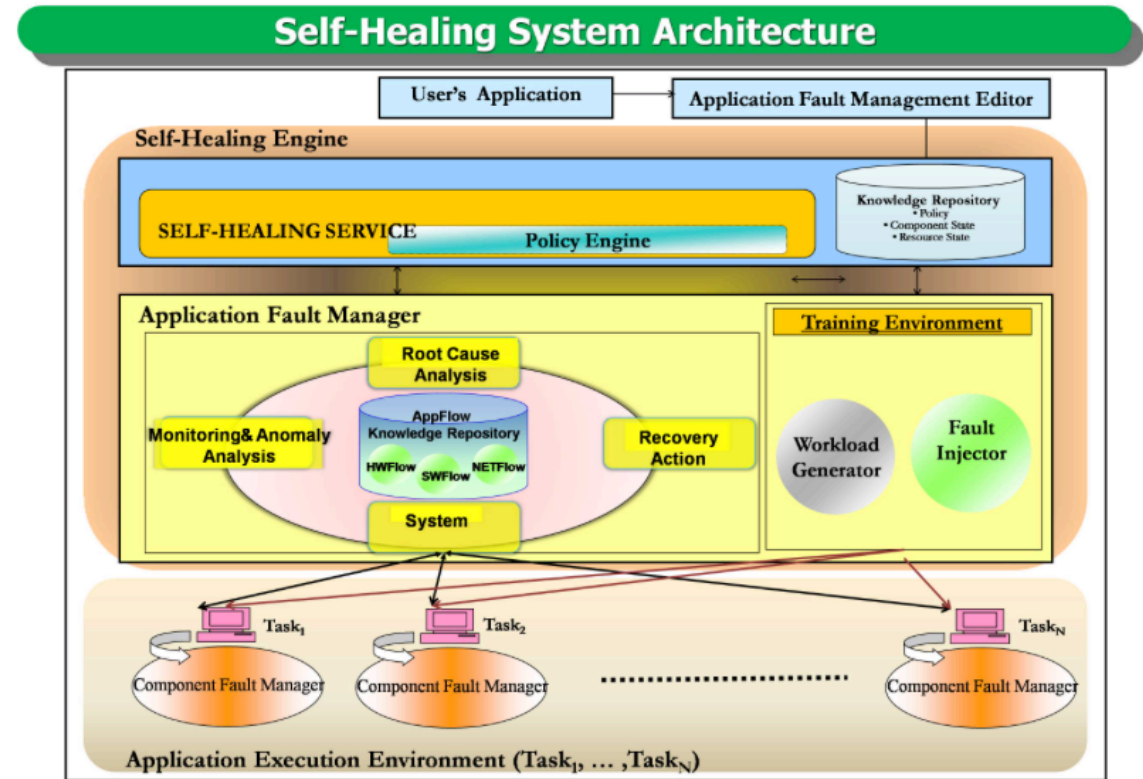
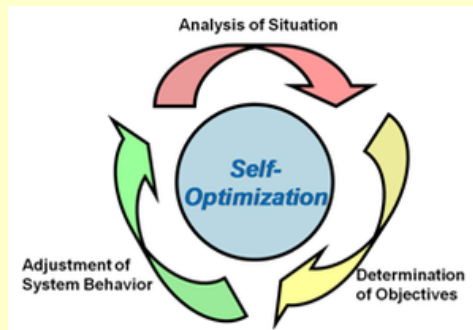
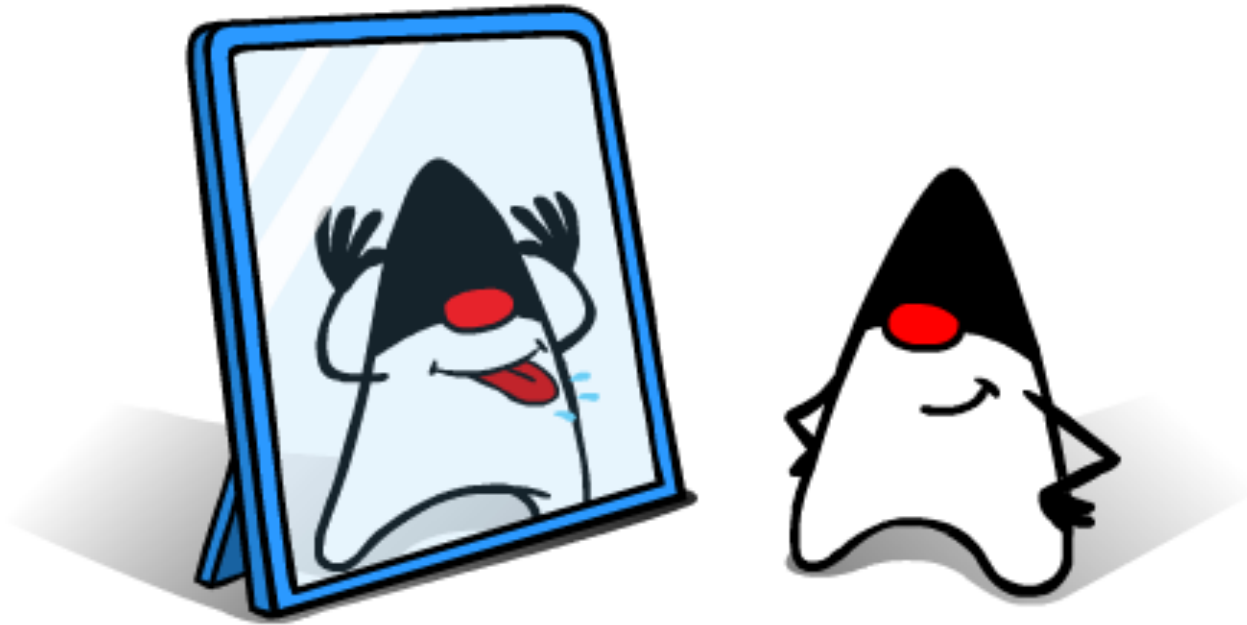


Figure 1: Self-Healing System

<http://acl.ece.arizona.edu/projects/old/Self-Healing/index.html>

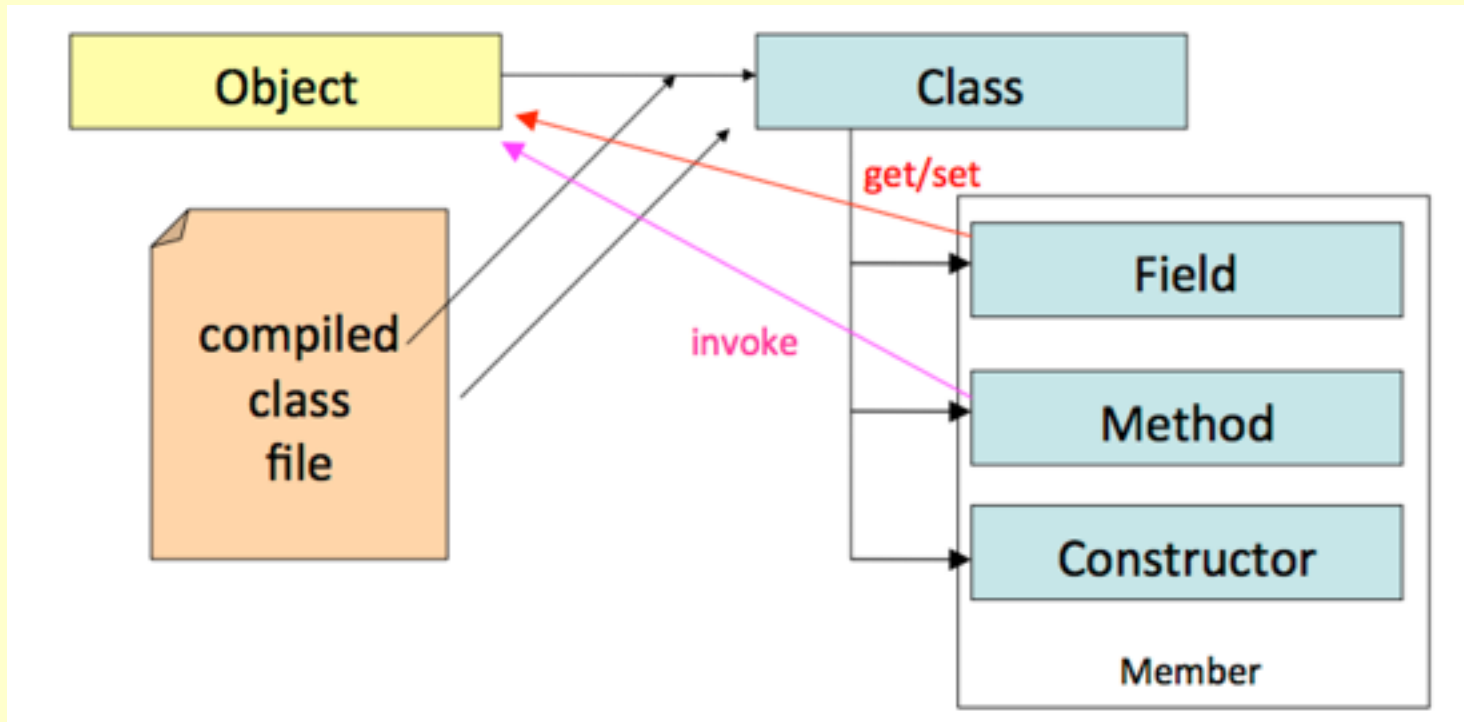
[https://ei.uni-paderborn.de/fileadmin/elektrotechnik/fg/lea/Forschung/Antriebstechnik/Self-Optimization/RTEmagicC\\_SO.png](https://ei.uni-paderborn.de/fileadmin/elektrotechnik/fg/lea/Forschung/Antriebstechnik/Self-Optimization/RTEmagicC_SO.png)

# Reflection In Java- An overview



# Reflection In Java- An overview

An object of type Class represents information about a Java class, and provides access to the Java reflection API





# Reflection In Java- An overview

Three standard ways to get a Class object:

// 1) When you have the object **something**

```
Class <SomeClass> someClass = something.getClass();
```

// 2) When you know the name of the class (**SomeClass**)

```
Class <SomeClass> someClass = SomeClass.class;
```

// 3) When you have the **String** used to represent the class name

```
Class <?> someClass = Class.forName(“models.className”);
```

**NOTE:** Since Java1.5 the class Class is generic : Class <T>

It is essential to understand **generics** to properly use reflection in Java

# Reflection In Java- An overview

Most of the reflection methods can generate exceptions  
All such calls must be wrapped in try/catch or throw

For example, accessing a private field generates an  
IllegalAccessException, but you can still do it :-)

You can use reflection on the exception classes/objects

**Note:** it is important to understand exceptions if you  
wish to properly use reflection in Java

## 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 - Testing non-functional/design/metric requirements

In the following scenario, you are a teacher who asked your students to provide a solution to a given problem (text file “Solution.java”) where the following metrics/code rules had to be respected:

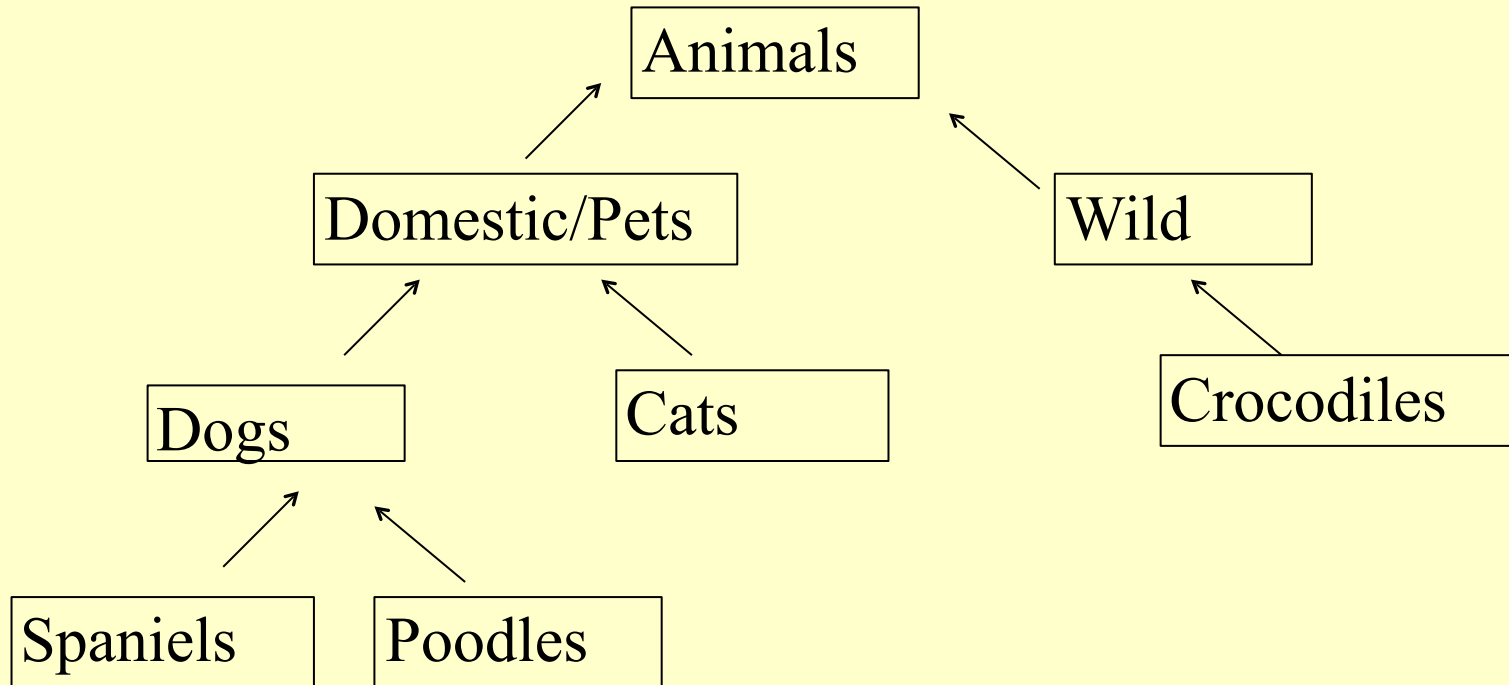
- 1.No method with >5 arguments
- 2.No non-private fields
- 3.There must be a default constructor (with zero arguments)

Write a test class (Test\_Solution\_Metrics.java) that uses reflection in order to test these non-functional requirements. It should output to the screen whether any of these 3 rules are not respected - together with details of which parts of the “Solution.java” code are broken

Write different tests to show that your code is correct

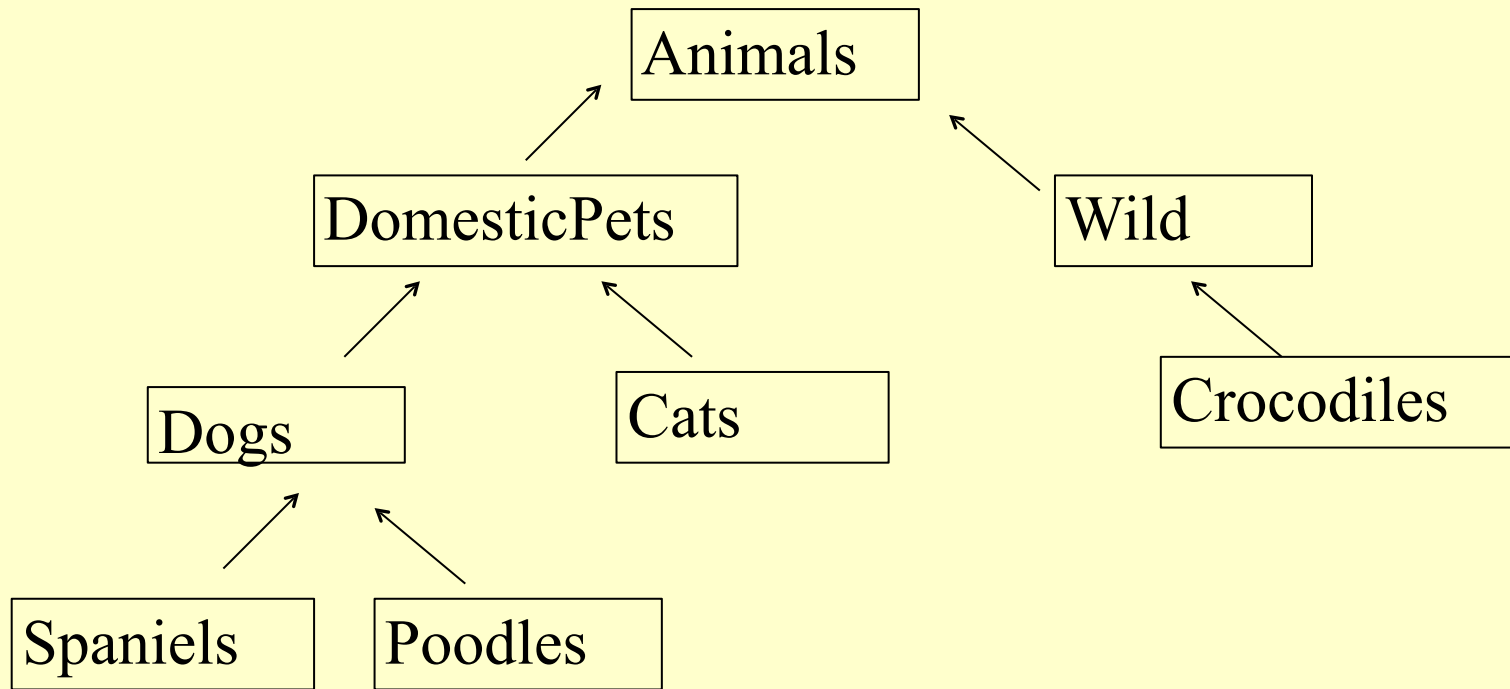
# Reflection PBL – LeastAbstractCommonSuperclass (LACS)

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, poodle2, cat, spaniel} = DomesticPets

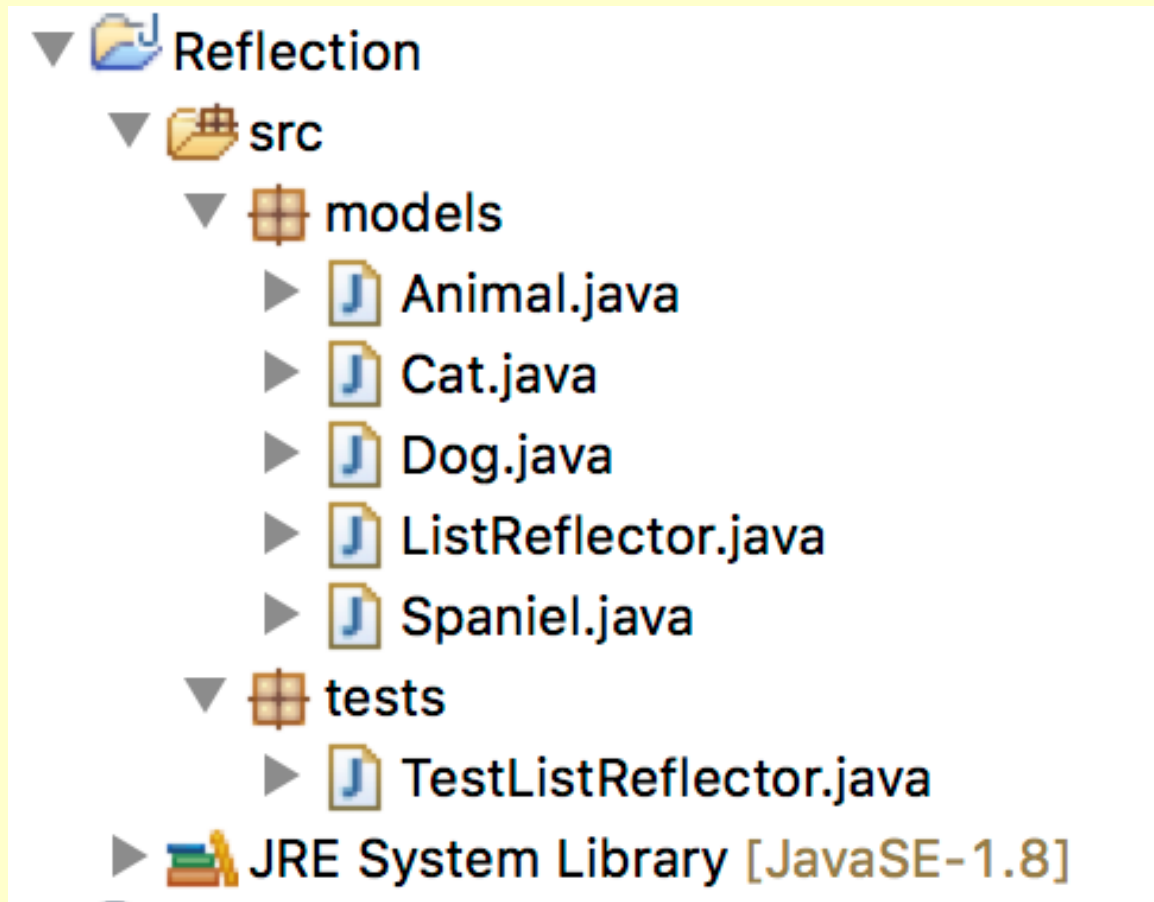
LACS {spaniel, poodle, dog} = Dogs

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 by executing the test**



# Reflection PBL - LeastAbstractCommonSuperclass


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](#).*

TODO



# Reflection PBL - LeastAbstractCommonSuperclass

## TODO

The `TestListReflector` should test 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**