Reflection:

Java reflection is a way to look inside the structure of an object or a class even if we do not have access to the source code.

We can also use reflection to manipulate any object or class.

Object obj = new JFrame();

Class<?> cls = obj.getClass();

cls now stores the class information about the JFrame class. (Note we could have created the JFrame object using just reflection, too!)

We can get a list of all non-private methods of the JFrame class:

cls.getMethods();

or all the methods (including the private ones) declared in the JFrame class:

cls.getDeclaredMethods();

We can get a specific method. For example, the getTitle method. It has no parameters. The declaration is

public Method getMethod(String name, Class<?>... parameters)

The ... is shorthand for an array. We can either pass in an array, or we can pass in the individual elements of the array.

Method m = cls.getMethod("getTitle", new Class[0]);

or

Method m = cls.getMethod("getTitle");

We can call the method to get the title of the JFrame:

m.invoke(obj, new Object[0]);

or

m.invoke(obj)

Remember that obj is the JFrame object. This states that we are calling the method stored in m on the object obj using the given parameters.

Now, let us try the setTitle method. This takes a single String. We can easily get the String class using the same technique we got the JFrame class.

m = cls.getMethod("setTitle", new Class[]{"Hi".getClass()});

or

m = cls.getMethod("setTitle", "Hi".getClass());

or use a field of the String class

m = cls.getMethod("setTitle". String.class);

To call it:

m.invoke(obj, new Object[]{"The Window Title"});

or

m.invoke(obj, "The Window Title");

Now, try the setVisible method. This takes a boolean, but boolean is not a class, it is a primitive! Java has a special field with each wrapper class that gives the

appropriate value to indicate that the parameter is a primitive.

m = cls.getMethod("setVisible", new Class[]{Boolean.TYPE});

or

m = cls.getMethod("setVisible", Boolean.TYPE);

and we can call it:

m.invoke(obj, new Object[]{new Boolean(true)});

or

m.invoke(obj, new Boolean(true));

or

m.invoke(obj, true);

thanks to the automatic wrapping and unwrapping of primitives.

Note that cls.getMethod("setVisible", new Boolean(false).getClass()) would not return a method because there is no setVisible method that takes Boolean as input.

Similarly we can access the fields of the class:

cls.getFields() - returns an array containing all public fields info

cls.getDeclaredFields() - returns an array containing info about all fields declared in this class, public or private

Field[] fields = cls.getDeclaredFields();

Now, given a field, we can access it:

fields[0].get(obj) -> returns the value stored in obj's field

fields[0].set(obj, value) -> changes the value of obj's field to value

Accessing private fields and methods

We can change the accessibility of any field or method in the class using the setAccessible() method.

This is really useful for JUnit testing private methods.

Below is the full transcript of what we did in the interactions pane:

Welcome to DrJava. Working directory is /home/hsc/courses/132

> import javax.swing.JFrame;

> JFrame j = new JFrame()

> Class cls = j.getClass()

> cls

class javax.swing.JFrame

> import java.lang.reflect.\*;

> Method m = cls.getMethod("getHeight", new Class[0])

> m

public int java.awt.Component.getHeight()

> m.invoke(j, new Object[0])

0

> m = cls.getMethod("setVisible", Boolean)

Static Error: Undefined name 'Boolean'

> m = cls.getMethod("setVisible", new Boolean())

Static Error: No constructor in Boolean matches this invocation

Arguments: ()

Candidate signatures:

Boolean(boolean)

Boolean(String)

> m = cls.getMethod("setVisible", new Boolean(true))

Static Error: No method in raw Class with name 'getMethod' matches this invocation

Arguments: (String, Boolean)

Expected return type: Method

Candidate signatures: Method getMethod(String, raw Class...)

> m = cls.getMethod("setTitle", "".getClass())

public void java.awt.Frame.setTitle(java.lang.String)

> m.invoke(j, "My title")

null

> m = cls.getMethod("getTitle")

public java.lang.String java.awt.Frame.getTitle()

> m.invoke(j)

"My title"

> m = cls.getMethod("setVisible", true.getClass())

Static Error: No method in boolean has name 'getClass'

> m = cls.getMethod("setVisible", new Boolean(true).getClass())

java.lang.NoSuchMethodException: javax.swing.JFrame.setVisible(java.lang.Boolean)

at java.lang.Class.getMethod(Class.java:1665)

> m = cls.getMethod("setVisible", Boolean.TYPE)

public void java.awt.Window.setVisible(boolean)

> m.invoke(j, true)

null

> m = cls.getMethod("setSize", Integer.TYPE, Integer.TYPE)

public void java.awt.Window.setSize(int,int)

> m.invoke(400,200)

java.lang.IllegalArgumentException: object is not an instance of declaring class

at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)

at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:57)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)

at java.lang.reflect.Method.invoke(Method.java:606)

> m.invoke(j, 400,200)

null

> Field[] fields = cls.getDeclaredFields()

> fields.length

7

> fields[0]

public static final int javax.swing.JFrame.EXIT\_ON\_CLOSE

> fields[1]

private static final java.lang.Object javax.swing.JFrame.defaultLookAndFeelDecoratedKey

> fields[2]

private int javax.swing.JFrame.defaultCloseOperation

> fields[3]

private javax.swing.TransferHandler javax.swing.JFrame.transferHandler

> fields[4]

protected javax.swing.JRootPane javax.swing.JFrame.rootPane

> fields[5]

protected boolean javax.swing.JFrame.rootPaneCheckingEnabled

> fields[6]

protected javax.accessibility.AccessibleContext javax.swing.JFrame.accessibleContext

> fields[0].get(j)

3

> fields[2].get(j)

java.lang.IllegalAccessException: Class edu.rice.cs.dynamicjava.symbol.JavaClass$JavaMethod can not access a member of class javax.swing.JFrame with modifiers "private"

at sun.reflect.Reflection.ensureMemberAccess(Reflection.java:109)

at java.lang.reflect.AccessibleObject.slowCheckMemberAccess(AccessibleObject.java:261)

at java.lang.reflect.AccessibleObject.checkAccess(AccessibleObject.java:253)

at java.lang.reflect.Field.get(Field.java:376)

> fields[2].setAccessible(true)

> fields[2].get(j)

1

> fields[2].setInt(j, 100)

> fields[2].get(j)

100

> fields[2]

private int javax.swing.JFrame.defaultCloseOperation

> fields[2].setInt(j, 1)

>