Programação Avançada

Inspector

Fábio Pinheiro - 69485
Rafael Reia - 69643
Artur Alkaim - 70407

Motivation

Is it easy to inspect a java object? To know everything about their fields at runtime? And changing something is a pain in the ass?

Your pain is over!

Goal

Make an Inspector class that describes the state of an object and support making changes in that state.

Standard Features

Command evaluation

The command evaluation and the call of the functions is made by Reflection.

- No chain of conditional statements.

Modify Command

The problem to solve:

- Parse from String to the correct object.

Solution:

- Again... Reflection.

```
clazz.getConstructor(new Class[] { String.class } ).
newInstance(s);
```

Call Command

The Problem here:

- No Autoboxing

Solution:

- HashMap to return given the primitive type the Boxed Class.

Extra Features

h - show a help message

b - Reinspect the last inspected Object

lm - List the Methods of the inspected Objects

Extra Features

bo [VALUE] - List the last <value> objects inspected and asks the user one to inspect. (<value> is 10 by default.)

so NAME - Save the current inspected with the provided argument as name.

Extra Features

lc VALUE -
List the last <value> commands used and asks the user one to repeat.

r - Repeat the last commands

Extra Features

To call functions with saved objects as arguments, or set a field value we add “@” before.

e.g. > so theG

> m g @theG

The name “ret” is saved to save the return of the last called function.

e.g. > c func @ret

Extra Features

To start searching for a field or a method in the superclass of the inspected object just add the symbol “+” before the name.

e.g. > i +g

This could be used to inspect shadowed superclass fields or call overridden methods. This symbol can be repeated to go up in the class hierarchy.

e.g. > c ++toString

Conclusions

This project is AWESOME!

Questions?