DoodleDebug

October 9, 2012

Abstract

Introduction

Development

Planning

Programming

Communication between Java Virtual Machines

Because a user program is running in a different Virtual Machine (VM) than Eclipse itself, we had to find a way to somehow communicate between those two in order to display the DoodleDebug rendering in an Eclipse tab. As a first attempt, we tried to use Java's built-in Remote Method Invocation (RMI) what resulted in frustration as some Java Security Manager always put obstacles in our way. Looking for alternatives, we found SIMON (Simple Invocation of Methods Over Network), a more comfortable alternative, which allows to create a registry on a specified port of localhost and add a Server object to it. The Server class implements an Interface he client, knowing the server's Interface, can then search for this server name and send messages to it.

User Interface

Output

Semantic Zoom

In order to save space and keep information available to users, DoodleDebug uses the concept of "Semantic Zoom"[link/reference]. Object visualizations are divided in levels of nesting, where level 0 represents outermost objects, referenced in `Doo.dle(object)`, level 1 objects are (semantically) nested ones inside level 0 etcetera. Saving space is achieved by only completely rendering level 0 and 1 objects, and use a smaller representation for a objects of level 2. Level 1 objects are clickable, which will cause them to be repainted as new virtual level 0 objects, so previous level 2 objects will move to level 1. This pattern allows to arbitrarily explore an objects nesting tree, similar to a debugger.

References

1. SIMON: `http://dev.root1.de/projects/simon`