Class Analyzer

technical overview

Contents

1 Introduction

This document provides technical details for the Class Analyzer software suite. The Class Analyzer software suite provides:

- A graphical application to interactively create class hierarchy diagrams.
- A command-line application to generate class hierarchy diagrams.
- A dependency analysis API for compiled Java classes.

1.1 Purpose

Class Analyzer provides software developers with a simple interface to traverse Java class hierarchies.

1.2 Audience

This document is written for a technical audience. Readers should be familiar with the Java programming language, package name spaces, class files, Java archive files, and recursion.

1.3 Software Requirements

The software used by Class Analyzer is listed in Table 1.1.

Software Package	Version	Notes
Java SE Runtime Environment	1.6.0	Runs the Class Analyzer.
BCEL	6.0.0	Java class file analysis.
JOpt Simple	3.1.0	Parses program arguments.
Prefuse	1.0.0	Graphics engine.
Lucene	1.4.3	Search engine.
RegExp	1.2.0	Parses regular expressions.

Table 1.1: Class Analyzer Application Dependencies

Except for Java, these software packages are distributed with Class Analyzer. See Table 4.1 on page 7 for build environment software requirements.

2 System Architecture

This section describes the architecture of the dependency analysis components.

2.1 Component Overview

How the major components interact is illustrated in Figure 2.1.

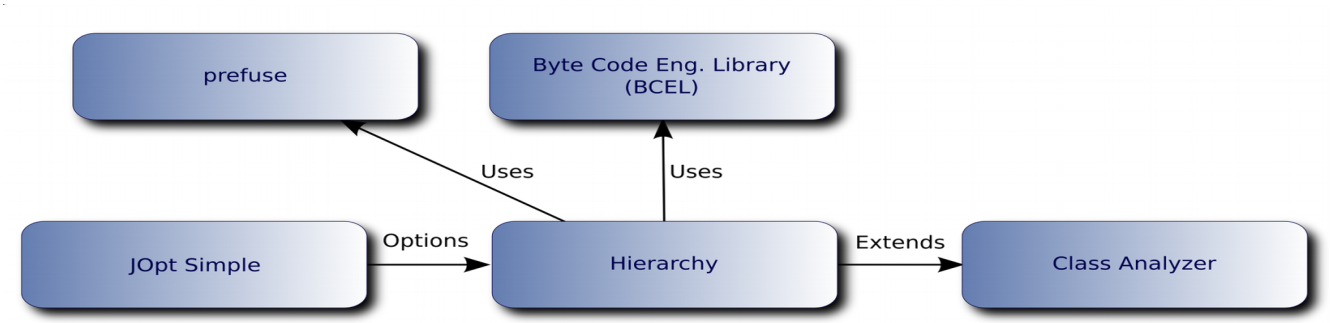

Figure 2.1: System Architecture

When the application is run, JOpt Simple is used to parse the command line arguments. These arguments are used to set various configuration parameters when running Hierarchy (such as the classes to analyze, which directories to search, graphing quality, and so forth). Hierarchy itself inherits from one of the Class Analyzer subclasses. All Class Analyzer subclasses use the Byte Code Engineering Library to parse Java files. The Prefuse Class Analyzer subclass creates an interactive graph.

2.1.1 JOpt Simple

JOpt Simple is used to parse the command line arguments. The major classes used from this library include OptionParser and OptionSet.

2.1.2 Byte Code Engineering Library

The Byte Code Engineering Library (BCEL) is used to determine the relationship of two classes. The major classes used from this library include JavaClass and ClassParser.

2.1.3 Prefuse

Prefuse is the graphing library used to create the interactive hierarchical view of Java classes.

2.1.4 Hierarchy

An application that uses Class Analyzer to display a graphical representation of a Java class hierarchy. This class is the hub for all the other architectural components shown in Figure 2.1. It performs command line argument parsing, creates a graphical user interface, transforms the Java class hierarchy into a tree compatible with the graphing package, and then displays the graph.

2.1.5 Class Analyzer

This component is the abstract superclass for all the different types of hierarchical analysis classes. Section 3 describes this part of the suite in depth.

2.2 Caching

The application heavily employs caching to achieve high speeds for its analysis. These main caches are described in the following sections.

2.2.1 ClassWalker

The ClassWalker is responsible for traversing files and directories in search of Java class files (any file whose name ends with `.class`). The ClassWalker maintains a list of classes that it has found. If that list is not empty, it means that the ClassWalker has already traversed all the classes available, and need not re-traverse the directory (and Java archive) structure again.

This optimization means that any Java class file that has been added after the first time ClassWalker has performed its walk will not be analyzed on subsequent passes. There are a few ways to work around the issue of new classes being added:

- Restart any application that uses ClassWalker.
- Code the application to re-walk the directory structure, rather than re-using ClassWalker's collection of class file names.

2.2.2 ClassAnalyzer

ClassAnalyzer retains a list of file names mapped to JavaClass instances. This prevents the applications from having to invoke BCEL to re-analyze each Java class file.

2.3 Graph

The graph produced by Hierarchy is not a true Manhattan layout (there is no Manhattan distance). The graph uses a NodeLinkTreeLayout, which is an orthogonal tree layout suitable for displaying hierarchical data. The graph is configured, controlled, and drawn using the following classes:

- TreeView
- OrthogonalEdgeRenderer

2.3.1 TreeView

This class configures and controls the user interface. It configures the graphing library, setting up the graph layout, colours, fonts, canvas size, and user interface controls (such as keyboard short-cuts, mouse movements).

This class also controls the graph's animated zooming behaviour.

2.3.2 OrthogonalEdgeRenderer

This class is responsible for creating the lines and arrows between two nodes in the graph. The algorithm it uses is straightforward, but involves a number of steps to avoid overlapping line segments and arrow heads.

The edge rendering algorithm follows:

1. Get the co-ordinates of the source node (sx, sy) and the target node (tx, ty).
2. Calculate the middle point along the y-axis between the source node and target node.
3. If the source node was not previously visited, draw the arrow head and line to the bus.
4. Create a new line segment that represents the line for the bus.
5. Before drawing the new segment, check for previous segment overlaps along the same y-axis.
   a) If the previous line segment is longer than the new segment, then draw nothing.
   b) If the previous line segment is shorter than the new segment, then draw the difference.
   c) If no previous line segment would overlap the new one, then draw the new line.
6. If the source and target nodes are exactly vertically aligned, draw one line between them.

For further implementation details review the source code for OrthogonalEdgeRenderer.

3 Class Analyzer

ClassAnalyzer is an abstract class that implements the `java.lang.Runnable` interface. The class has the following behaviours:

- Modifies the `java.class.path` System property during execution.
- Restores the `java.class.path` System property value upon completion.
- Uses `com.whitemagicsoftware.hierarchy.ClassWalker` to find class files.
- Uses BCEL to parse class files (without trying to instantiate them).

ClassAnalyzer can be used on its own to perform pure hierarchy analysis without dependency on a graphing library. Hierarchy requires a graphing library to produce diagrams, such as Figure 3.1.

3.1 Inheritance Hierarchy

The full ClassAnalyzer inheritance hierarchy is shown in Figure 3.1.

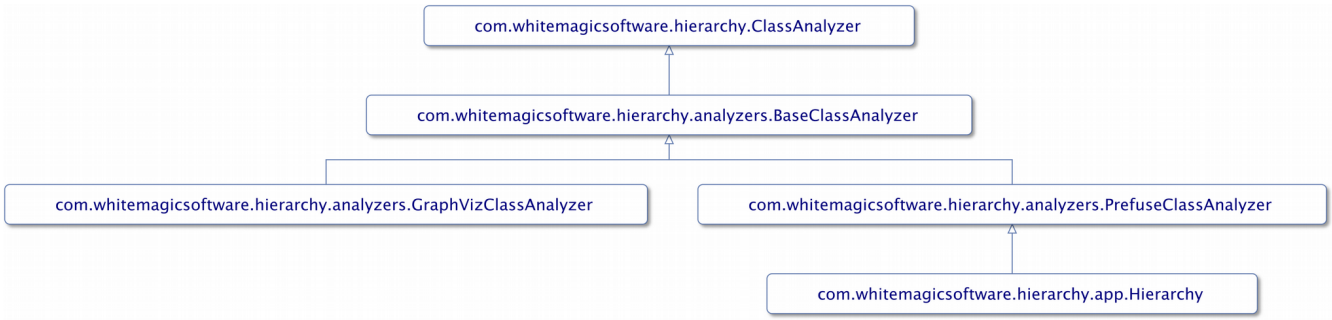

Figure 3.1: ClassAnalyzer Inheritance Hierarchy

The subsequent sections provide an overview of each ClassAnalyzer subclass.

3.1.1 BaseClassAnalyzer

This class abstracts configuring common command line options used by its subclasses. Its purpose is to decouple parsing of options from ClassAnalyzer.

3.1.2 GraphVizClassAnalyzer

This class writes a class hierarchy in a plain text format that is compatible with the GraphViz suite of

tools (to standard output). It shows an example usage of ClassAnalyzer performing a simple task.

This class is executable.

3.1.3 PrefuseClassAnalyzer

This class writes a class hierarchy in an XML format that is compatible with the graphing library's TreeView example (to standard output). It provides a moderately complex example of how to use ClassAnalyzer to perform a non-trivial task. This class also serves as the base class for Hierarchy.

This class is executable.

3.1.4 Hierarchy

This class draws an interactive, graphical representation of a Java class hierarchy. It provides software developers with a fast way of producing high-quality, high-resolution class hierarchies.

This class is executable.

3.2 Extending Class Analyzer

Methods developers must implement, or are of critical importance, are listed in Table 3.1. Full documentation can be found in the Javadocs.

Method	Usage
doAnalysis()	Called when the thread is run; this method must call `analyze()`.
visitSubclass(…)	Called for each unique parent-child subclass relationship found.
visitSuperclass(…)	Called for each unique parent-child superclass relationship found.
addClass(…)	Sets the path to a Java class file to be analyzed.
analyze()	Performs analysis of all Java class files that can be found.
setExcludeRegex(…)	Indicates which class files should be excluded from analysis.
updateClassPath(…)	Appends a new path for scanning onto the class path.

Table 3.1: ClassAnalyzer Methods

4 Building Class Analyzer

The Class Analyzer project can be built using any of the build tools listed in Table 4.1.

Application	Version
Apache Ant	1.7.1
Oracle JDeveloper 11g	11.1.1.1.0

Table 4.1: Build Environments

These tools are described in subsequent sections.

4.1 Apache Ant

Apache's Ant can be used for all the tasks listed in Table 4.2.

Task	Usage	Effect
Build Hierarchy	`ant`	Compiles classes into **classes** directory and creates executable Java archive.
Delete classes and Javadocs	`ant clean`	Deletes **classes**, **doc**, and **dist** directories. Deletes executable Java archive.
Compile Hierarchy	`ant compile`	Compiles classes into **classes** directory and increases the build version.
Create distribution	`ant dist`	Writes zip archive into **dist** directory.
Generate Javadocs	`ant doc`	Writes Javadocs into **doc** directory.
Display version information.	`ant version`	Shows the version of Hierarchy to be created.

Table 4.2: Ant Tasks

4.2 Oracle JDeveloper

To compile Class Analyzer using JDeveloper, complete the following steps:

1. Open the workspace file (e.g., Hierarchy.jws).
2. Click **Build » Rebuild ClassAnalyzer.jpr** or press **Alt-F9**.

5 Directory Structure

This section describes the content of the directories in use by the software.

5.1 classes

Contains compiled classes ready for bundling into a Java archive file.

5.2 dist

Contains the distribution archive for the application.

5.3 doc

Contains supporting documentation.

5.4 javadoc

Created using the Ant tool, this contains HTML documentation.

5.5 lib

Contains Java archives of the third-party libraries, as well as a compiled version of the application.

5.6 licenses

Contains licenses for the third-party libraries.

5.7 src

Contains source code.

6 Copyright

7 License

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal

Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for

informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.