JHawk 5 Documentation– Standalone User manual

Virtual Machinery
March 2010
Version 1.0

Overview

Since its release in 1999 JHawk has been a leader in the provision of Software Metrics to Java developers. Originally released as a stand-alone application it has now evolved to include a command line version and an Eclipse plugin. Functionality has also been added over time to allow the export of the metrics gathered by JHawk in CSV, HTML and XML format.

JHawk has proved itself in many different areas and its customers have reflected that – from Fortune 500 companies to Academic institutions, from Banking to Telecommunications companies and across the globe from Norway to Brazil and from the US to China.

A consistent feature of JHawk from the very beginning has been the provision of a simple mechanism to allow users to extend JHawk and to make their own metrics. This feature was one of the main reasons for JHawks success in Academic Institutions. We simplified this in version 4 and in version 5 we have gone a step further, achieving what we thought would be impossible – maintaining the simplicity but increasing the power.

We have also realised that commercial users need a quick overview that gives them an instant summary of the current state of their code. To achieve this we have added a dashboard tab which gives a quick overview of the metrics at System, Package and Class level. The metrics displayed on the dashboard are configurable – whether they are metrics provided by Virtual Machinery or by users themselves.

We have also added a new product to our line – the JHawk Dataviewer – this allows a user to view changes in core metrics over time – for example over a project lifecycle. This is done using the XML files exported by JHawk. This is an extremely powerful tool which is unique in the Java Software metrics area.

This document is designed to satisfy the needs of users of JHawk5 – whether they are users of previous versions of the product or completely new to JHawk. It provides information on the use, design and modification of the JHawk product.

If you have used JHawk before then you should find the product as easy to use as before and the new features should become obvious as you use the product.

We have identified three categories of user for this new version of the product.

1. Users completely new to JHawk.
2. Users upgrading from JHawk4 to JHawk5 who generally use the product ‘as is’
3. Users upgrading from JHawk4 to JHawk5 who have always configured the product to meet their own needs.

For each of these categories we have suggested a subset of this document that you should read to get you up and running as quickly as possible. If you have any difficulties you should consult the FAQs.

1. Users completely new to JHawk.
    - Read the ‘Quick start’ section.
    - Run JHawk on some of your code
    - Read the ‘JHawk5UsingMetrics’ document
    - Read the Section in this document on how to set preferences so that you can tune the settings in JHawk to suit your purpose
    - Read other sections of the document as you need them
2. Users upgrading from JHawk4 to JHawk5 who generally use the product ‘as is’
    - Read the ‘Changes from Version 4’ section to see the new and updated features in JHawk5.
    - Read other sections of the document as you need them
3. Users upgrading from JHawk4 to JHawk5 who have always configured the product to meet their own needs.
    - Read the ‘Changes from Version 4’ section to see the new and updated features in JHawk5.
    - Read the ‘JHawk5CreateMetric’ document to find out how to modify JHawk without access to the source code.
    - Read other sections of the document as you need them

Documentation

The following documents are provided with the distribution (depending on which license you have purchased). They are in PDF format and are located in the ‘docs’ directory of the distribution –

Document	Personal	Professional	Demo
JHawk5CommandLineManual – Documentation for the Command line version of JHawk	No	Yes	Yes
JHawk5CreateMetric – Documentation explaining how to create new metrics that can be added to JHawk	Yes	Yes	No
JHawk5DataViewerManual – Documentation for JHawks DataViewer product	No	Yes	Yes
JHawk5EclipseManual – Documentation for the JHawk Eclipse Plugin	Yes	Yes	Yes
JHawk5Licensing – Licensing details for JHawk products	Yes	Yes	Yes
JHawk5UserManual – Documentation for the JHawk stand alone application	Yes	Yes	Yes
JHawk5UsingMetrics – Documentation outlining how to get the best from JHawk and a list of the metrics implemented by JHawk with details of their calculation.. It also includes an introduction to the area of Java code metrics.	Yes	Yes	No

JHawk – Changes from Version 4

Users familiar with previous versions of JHawk will notice a number of changes in this version –

The dashboard tab – this is a new tab that allows users to select a subset of their data for immediate visual analysis. The dashboard tab allows you to drill down to class and method level. You can find out more about this in the 'Dashboard tab' section below.

Ability to create new metrics – we have provided a mechanism to add new metrics to JHawk using simple Java classes in a framework of classes provided by Virtual Machinery. These classes provide access to the raw data collected by JHawk and metrics already defined in JHawk allowing new and combined metrics to be created. Data outside Jhawk can also be used e.g. bug databases, source code control systems – anything you can code in Java can be used. Metrics created in this way can be reflected all through JHawk including the DataViewer and the Command Line interface. You can find out more in the 'JHawk5CreateMetric' document in the 'docs' directory of the distribution.

Increased range of metrics – we have increased the number of metrics that JHawk provides. Not all of those available are active when you start up JHawk but you can activate them by using the Prefrences tab – see the section 'Preferences Tab' below. A complete list of the metrics available can be found in the 'JHawk5MetricsList' document in the 'docs' directory of the distribution.

Improved CSV output - We have added a 'raw' option which will output packages, classes and methods with out sub-totals at class , package and system level. See the 'CSV-Raw' subsection of the 'Output Formats' section below.

Improved HTML output – We have improved the appearance of the HTML output by using coloured indicators and the dashboard indicators. We've also changed font and table layouts for a neater look. See the 'HTML' subsection of the 'Output Formats' section below.

JHawk Metric Interchange format – The JHawk Metric Interchange Format is an XML-based format for use between the stand alone and Data Viewer products. It means that you can capture the data relating to a particular source code set and re-use it in JHawk or the Data Viewer without having to re-analyse the code. You can find out more in the See the 'XML – JHawk Metric Interchange Format' subsection of the 'Output Formats' section below.

Installing JHawk

In the JHawk Distribution you will find the JHawk.jar. You can locate this jar anywhere as long as you have a jhawk.properties file with it. You can then run the JHawk stand alone application either by double clicking on the jar or starting it from the command line (see section below – 'Starting JHawk from the Command Line').

You can use any jhawk.properties file that you have created and/or modified using either the JHawk stand alone application or the JHawk Eclipse plugin.

If you have created additional metrics that you want to use in JHawk you will need to have these metrics available to JHawk in the jhawk.properties file and in the CustomMetrics.jar. The CustomMetrics.jar must be in the same directory as JHawk.jar.

As part of the distribution you will notice the following properties files –

- jhawkbase.properties
- jhawkfull.properties
- jhawkbasepluscustom.properties

jhawkbase.properties is a version of the properties file with the base level set of metrics that are initially enabled for JHawk. This is the same as the jhawk.properties file that comes with the JHawk distribution.

jhawkfull.properties is a version of the properties file with the full set of metrics available to JHawk enabled.

jhawkbasepluscustom.properties is a version of the properties file with the base level set of metrics that are initially enabled for JHawk plus the sample metric found in the CustomMetrics.jar that comes with the distribution. For more details about this metric see the 'JHawk5CreateMetric' document.

You can use any of these sets of metrics by copying them to the jhawk.properties file and starting as normal or by using the –p flag and the property file name (JhawkCommandLine only).

Properties loaded from these files can be amended in the usual way – see 'Preferences' section in the documentation.

JHawk – A quick Start

How you start the JHawk application will depend on your environment. JHawk has been tested in a large number of different Java environments over the years and should run in any environment that supports Java 1.5 and above.

Startup JHawk

Startup JHawk in a Windows environment

- When you look in the directory that you installed the JHawk application you will find a jar file called JHawk.jar Double clicking on this should start the application. If it does not start then you will have to start the application from the command line – see the section 'Starting JHawk from the command line'.

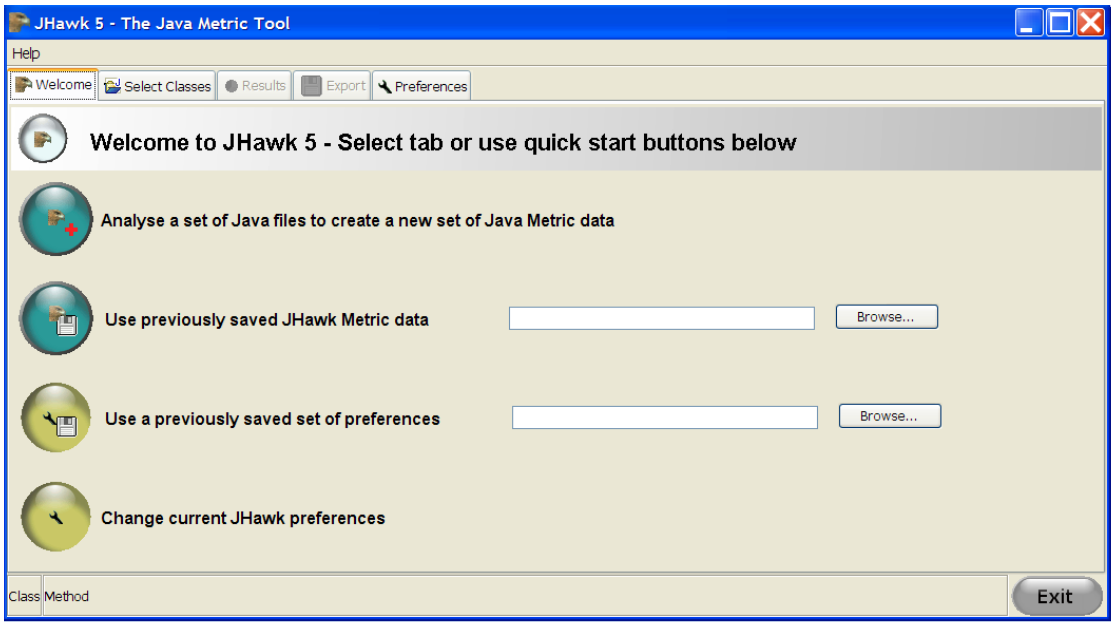
Startup JHawk in a Mac Environment

- When you look in the directory that you installed the JHawk application you will find a jar file called JHawk.jar Double clicking on this should start the application. If it does not start then you will have to start the application from the command line – see the section 'Starting JHawk from the command line'.

Startup JHawk in a Unix Environment

- Startup will depend on the Unix environment. In some cases you may be able to double click on the JHawk jar and in others you may have to start JHawk from the command line. In the latter case you should see the section – 'Starting JHawk from the Command Line'.

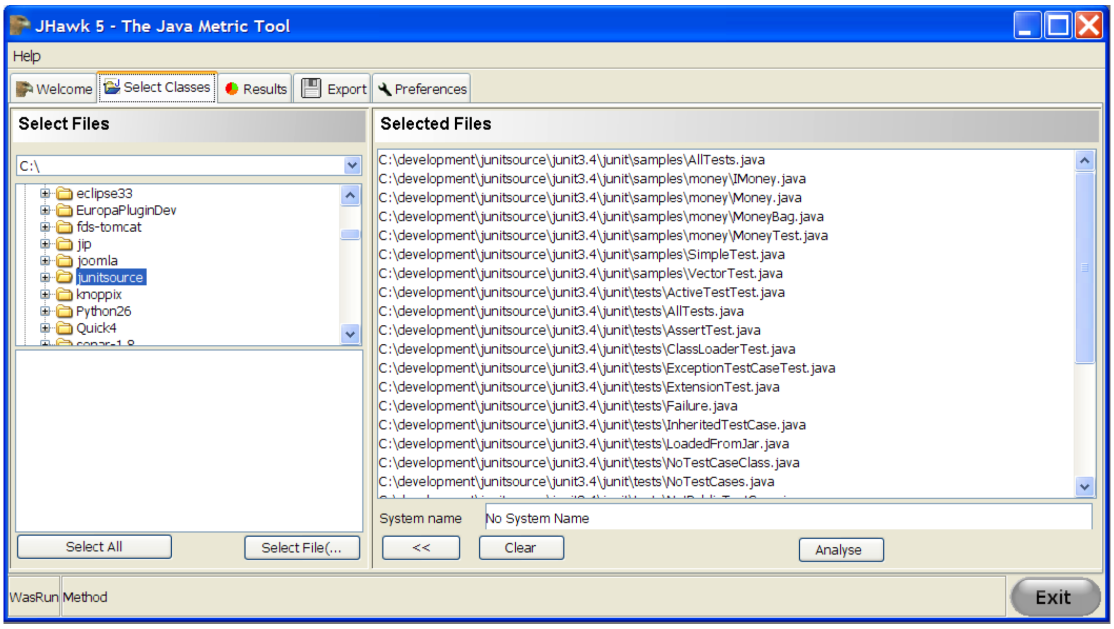
The opening screen


JHawk 5 - The Java Metric Tool
Help
Welcome
Select Classes
Results
Export
Preferences
Welcome to JHawk 5 - Select tab or use quick start buttons below
Analyse a set of Java files to create a new set of Java Metric data
Use previously saved JHawk Metric data
Browse...
Use a previously saved set of preferences
Browse...
Change current JHawk preferences
Class
Method
Exit

The screen opens on the 'Welcome' tab. At initial startup only this tab, the 'Select Classses' tab and the 'Preferences' tab are enabled. The 'Results' and 'Export' tabs are disabled until a set of Java files have been analysed. There are four options on the Welcome tab. Each of them can be selected by pressing on the left hand side icon.

1. **Analyse a set of files to create a new set of Java Metric data.** This is the equivalent of clicking on the ‘Select Classes’ tab and allows you to start the process of analysing a group of Java files using JHawk.
2. **Use previously saved JHawk Metric data.** This allows you to load metric data that you created in a previous JHawk session and saved in the JHawk Metric Interchange format (see details on the Export tab to find out how to do this). You must first select a valid XML file using the ‘Browse’ button before clicking the icon.
3. **Use a previously saved set of preferences.** This allows you to load a set of JHawk preferences that you created in a previous JHawk session and saved (see details on the Preferences tab to find out how to do this). You must first select a valid JHawk properties file using the ‘Browse’ button before clicking the icon. The loaded properties will be applied immediately.
4. **Change current JHawk Preferences.** This is the equivalent of clicking on the ‘Preferences’ tab and allows you to change the current JHawk preferences.

The Select Classes tab



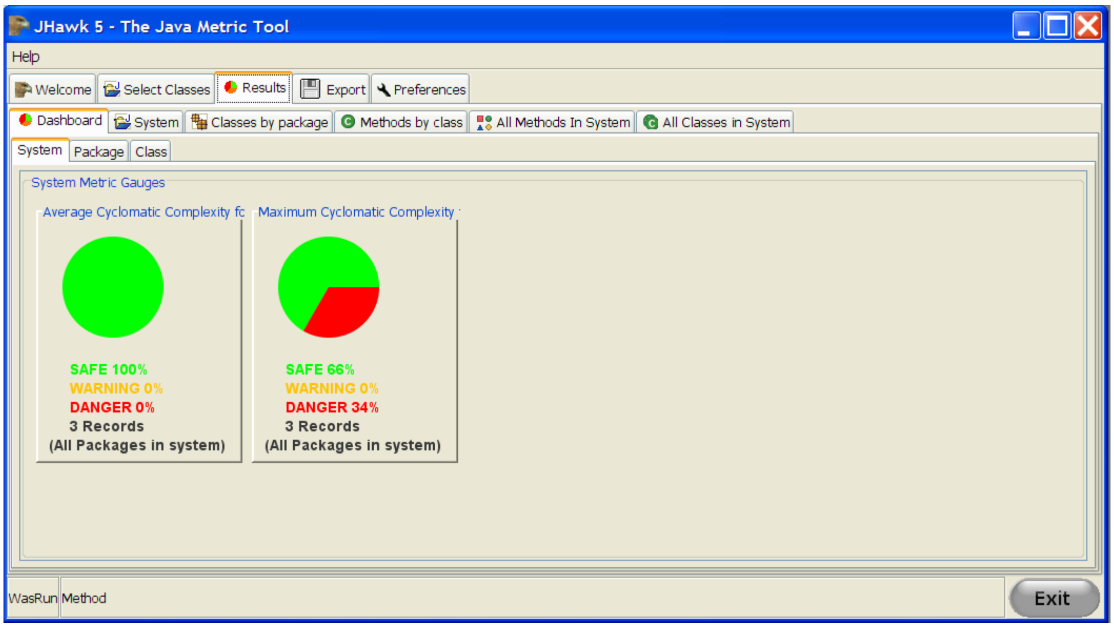
```
C:\development\junitsource\junit3.4\junit\samples\AllTests.java
C:\development\junitsource\junit3.4\junit\samples\money\IMoney.java
C:\development\junitsource\junit3.4\junit\samples\money\Money.java
C:\development\junitsource\junit3.4\junit\samples\money\MoneyBag.java
C:\development\junitsource\junit3.4\junit\samples\money\MoneyTest.java
C:\development\junitsource\junit3.4\junit\samples\SimpleTest.java
C:\development\junitsource\junit3.4\junit\samples\VectorTest.java
C:\development\junitsource\junit3.4\junit\tests\ActiveTestTest.java
C:\development\junitsource\junit3.4\junit\tests\AllTests.java
C:\development\junitsource\junit3.4\junit\tests\AssertTest.java
C:\development\junitsource\junit3.4\junit\tests\ClassLoaderTest.java
C:\development\junitsource\junit3.4\junit\tests\ExceptionTestCaseTest.java
C:\development\junitsource\junit3.4\junit\tests\ExtensionTest.java
C:\development\junitsource\junit3.4\junit\tests\Failure.java
C:\development\junitsource\junit3.4\junit\tests\InheritedTestCase.java
C:\development\junitsource\junit3.4\junit\tests\LoadedFromJar.java
C:\development\junitsource\junit3.4\junit\tests\NoTestCaseClass.java
C:\development\junitsource\junit3.4\junit\tests\NoTestCases.java
```

System name: No System Name

This tab allows you to select the Java files that contain the classes that you wish to analyse. In the simplest scenario you have all your files in a single source directory with the underlying source directory structure reflecting the package structure. To analyse all these files select the root directory of your source code then press the ‘Select All’ button – you should see all the Java file paths listed in the ‘Selected Files’ tab on the right hand side. Note that only the ‘.java’ files in the directories will be selected. To analyse all these files press the ‘Analyse’ button and wait until the other tabs change from the grey ‘disabled’ state to their enabled state.

If you wish to do anything more complex than this you should consult the separate section on the ‘Select Classes’ tab elsewhere in the document.

The Results Tab



SAFE 100%
WARNING 0%
DANGER 0%
3 Records
(All Packages in system)

SAFE 66%
WARNING 0%
DANGER 34%
3 Records
(All Packages in system)

The results tab is a holding tab for those tabs related to the results created after a group of Java files have been analysed. The Dashboard, System, Classes By Package, Methods By Class, All Methods In System and All Classes In System tabs can be found as sub-tabs of this.

Results Tab - Dashboard sub-tab

The dashboard tab provides you with a visual summary of the state of your code at System, Package and Class level. It does this with a series of pie charts that reflect the proportion of elements (packages, classes, methods) whose metrics have crossed a particular warning or danger level. The default settings will be displayed when you first start the product. You can modify these to provide you with the diagnostic information that you need and you can reset the dashboard back to the default values at any time. You can find out how to do this by consulting the separate section on the on the Dashboard sub-tab elsewhere in the document.

Results Tab - System sub-tab

The System tab provides the 'top level' figures for the results of JHawk's analysis of your code. It shows you the main system level metrics relating to the code that you have chosen to analyse and also lists the summary data for each of the packages listed in the system.

Let us say that you noted that the Average Cyclomatic Complexity was quite high and you wanted to find out which code was responsible for this. Obviously packages with a higher level of average cyclomatic complexity will contribute more to this final figure so you need to look at the Packages list. By double clicking on the AVCC (Average Cyclomatic Complexity) column header you will see that the list of packages resorts itself in ascending order based on AVCC. Double clicking again will bring the list into descending order based on AVCC with the highest level at the top.

You can now see which package has the highest Average Cyclomatic Complexity. We now need to see which classes contribute most in this package. We can do this by looking at the 'Classes by Package' tab.

Results Tab - Classes By Package sub-tab.

At the top of this tab we find a list of packages. Clicking on each of these Packages will display a table below which contains a summary line of the metrics for each class contained in the Package. Having found the Package that we identified as having the highest Average Cyclomatic complexity in the previous step, we now perform a similar exercise with the classes as we did with the Packages.

Double clicking on the AVCC header will list the classes in order of AVCC value. As before, double clicking a second time will sort the classes in descending order. It is now easy to see the class with the highest AVCC value.

We can now continue on to the 'Methods By Class' tab to continue our search for the source of our problems,

Results Tab - Methods By Class sub-tab.

At the top of this tab we find a list of Packages on the left hand side and a list of Classes on the right hand side. Clicking on a Package, then a Class in that Package will display a table below which contains a summary line of the metrics for each method contained in the Class. Having found the Class that we identified as having the highest Average Cyclomatic complexity in the previous step, we now perform a similar exercise with the methods as we did with the Classes.

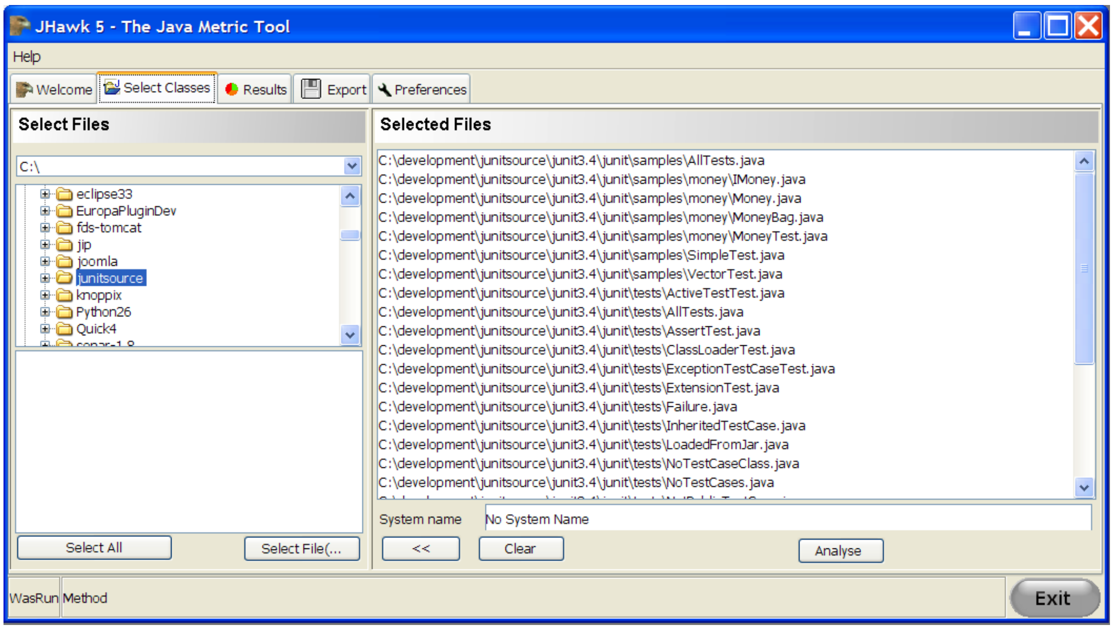
In this case we are interested in the actual Cyclomatic Complexity of each method. Double clicking on the COMP header will list the methods in order of COMP value. As before, double clicking a second time will sort the methods in descending order. It is now easy to see the methods with the highest COMP values.

As you can probably guess you can do this for any metric and using the above approach find where your problem code lies.

If you are looking for issues that are at Method or Class level you can view and sort all the Classes and Methods at once by metric using the 'All Classes in System' and 'All Methods In System' tabs respectively.

This has been a brief introduction to the main screens of the application. To find out more about how to configure JHawk to better suit your purposes you should look at the 'The Preferences Tab' later in this guide. If you want to find out how to export data for use externally then you should read the 'The Export Tab' section.

The Select Classes Tab



```
C:\development\junitsource\junit3.4\junit\samples\AllTests.java
C:\development\junitsource\junit3.4\junit\samples\money\IMoney.java
C:\development\junitsource\junit3.4\junit\samples\money\Money.java
C:\development\junitsource\junit3.4\junit\samples\money\MoneyBag.java
C:\development\junitsource\junit3.4\junit\samples\money\MoneyTest.java
C:\development\junitsource\junit3.4\junit\samples\SimpleTest.java
C:\development\junitsource\junit3.4\junit\samples\VectorTest.java
C:\development\junitsource\junit3.4\junit\tests\ActiveTestTest.java
C:\development\junitsource\junit3.4\junit\tests\AllTests.java
C:\development\junitsource\junit3.4\junit\tests\AssertTest.java
C:\development\junitsource\junit3.4\junit\tests\ClassLoaderTest.java
C:\development\junitsource\junit3.4\junit\tests\ExceptionTestCaseTest.java
C:\development\junitsource\junit3.4\junit\tests\ExtensionTest.java
C:\development\junitsource\junit3.4\junit\tests\Failure.java
C:\development\junitsource\junit3.4\junit\tests\InheritedTestCase.java
C:\development\junitsource\junit3.4\junit\tests\LoadedFromJar.java
C:\development\junitsource\junit3.4\junit\tests\NoTestCaseClass.java
C:\development\junitsource\junit3.4\junit\tests\NoTestCases.java
```

The Select classes tab is the place where you build up the set of files that contains the class that you wish JHawk to analyse together. This set of files is the described as the 'System' level. When this set of files is analysed a set of System level metrics is calculated and collected and is displayed on the System Tab (see section on System Tab). There is no Java equivalent to the JHawk System level – but it might, for example, equate to all the Java files required to build and test a Java application. The next level under the System level is the Packages Level, which exactly equates to Java Packages and includes all the packages that have been defined in the files analysed at the System level.

The screen is divided into three main panels – two on the left and one on the right -

- In the top left panel is a tree view displaying all the directories in the root directory of the computer on which JHawk is being run. You can navigate around these directories as you would in any other application.
- In the bottom left panel is a list of all the files in the directory last selected in the top left tree view.
- The right hand panel lists all the files that have been selected for analysis by the JHawk application. These files constitute the System.

To move a file into the right hand panel you use the buttons under the bottom left panel. You can either select one or more files and move it into the right hand panel using the 'Select File(s) >>' button. Or you can press the 'Select All' button which will move all the .java files in the current directory and its subdirectories into the right hand panel. It is quite likely that you will most frequently use the 'Select All' button as most developers keep their source files in a single directory structure reflecting the package layout of their code.

If you wish to remove files from the right hand panel you can do so by selecting the file(s) that you want to remove and clicking the "<<" button. If you want to remove all the files in the list then press the 'Clear' button.

If you want to give a particular name to your system you can do so by typing a name in the 'System Name' text box. The system name is useful if you are exporting your data in any of the formats supported by JHawk (CSV, HTML, XML) as this will be the name displayed in these files.

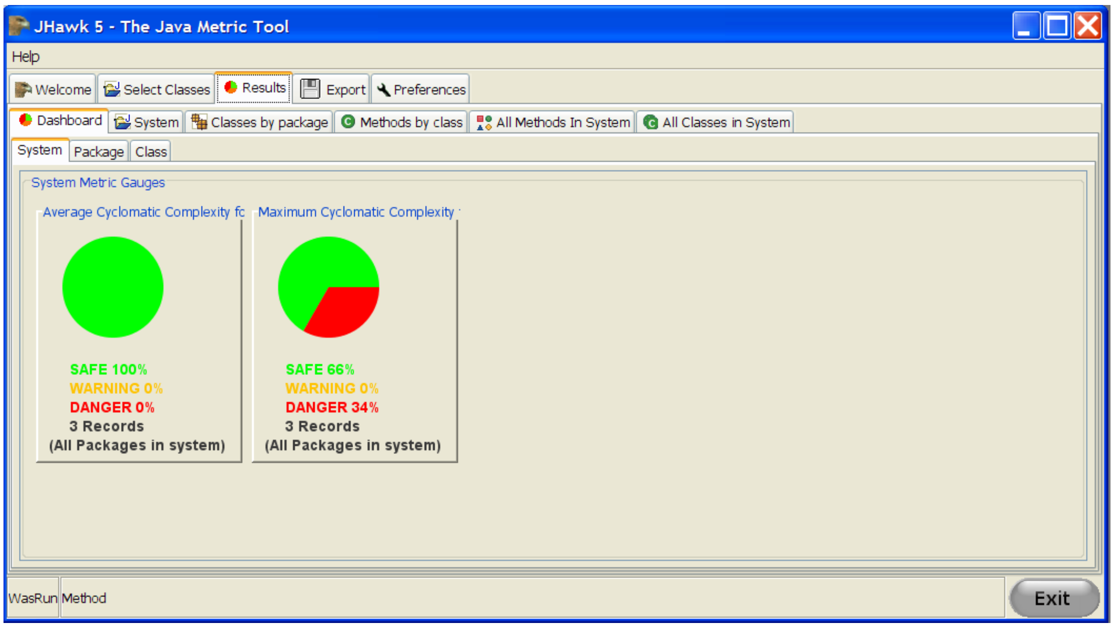
When you have selected all your files and are ready to analyse them press the ‘Analyse’ button. The button will be greyed out and will remain so until all the files have been analysed. As the analysis progresses the names of the classes currently being analysed will be displayed in the small message panes starting in the lower left hand corner of the screen.

When the analyse button is re-enabled you will notice that all the tabs have been enabled and you can now navigate between them. We would suggest going to the results tab first and you can find out more about it in the next section.

The Results Tab

The results tab is a holding tab for those tabs related to the results created after a group of Java files have been analysed. The Dashboard, System, Classes By Package, Methods By Class, All Methods In System and All Classes In System tabs can be found as sub-tabs of this.

Results Tab - Dashboard sub-tab



SAFE 100%
WARNING 0%
DANGER 0%
3 Records
(All Packages in system)

SAFE 66%
WARNING 0%
DANGER 34%
3 Records
(All Packages in system)

For users familiar with previous versions of JHawk you will note that this is a new tab. The dashboard sub-tab allows users to make a quick visual check on the current quality of their code. Although it comes with some predefined metrics the dashboard tab is completely configurable using the Preferences tab (see ‘Preferences Tab – Configuring the Dashboard’ for more details).

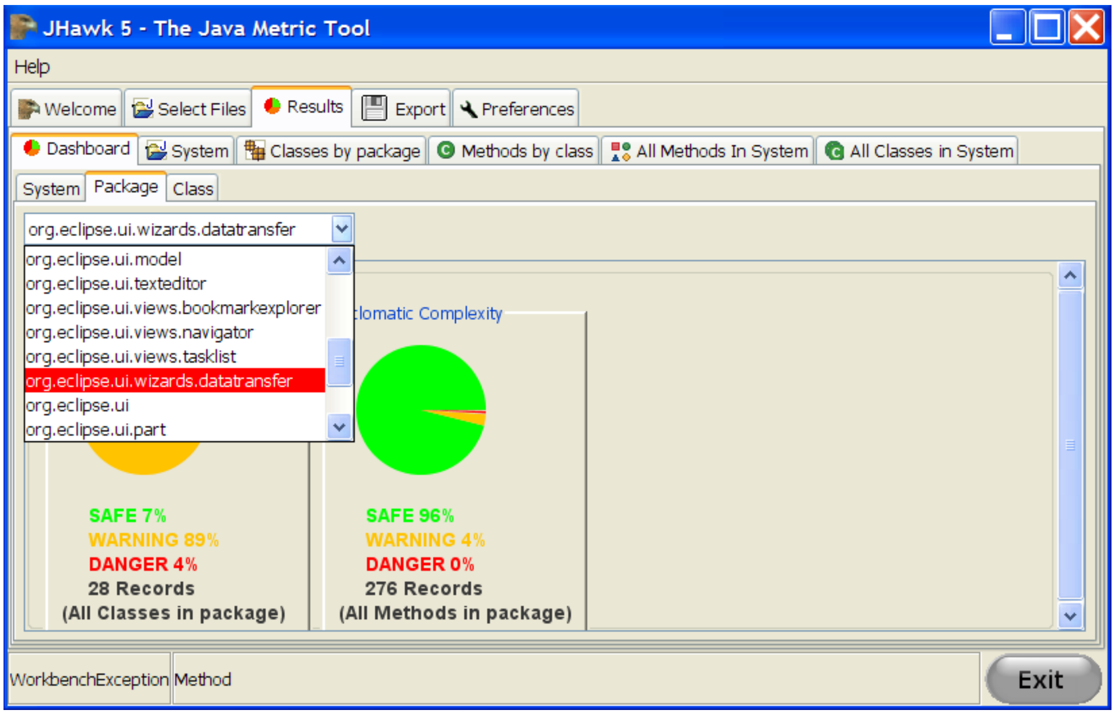
Each metric is displayed as a pie chart with the proportion of values that are OK shown as a green slice, the proportion that exceed the warning level shown as a yellow slice and the proportion exceeding the danger level shown as a red slice. These are the default colours used – if you have changed the colours using the General Preferences tab then these will be different. The name of the metric calculated, the number of entries used in the calculation and the percentage in each category are also displayed.

There are three sub-tabs within the Dashboard Tab –

- The System tab shows all the metrics assigned to the dashboard at package, class and method level. Package level metrics will be displayed for all the packages in the system, class level metrics for all the classes in the system and method level metrics for all the methods in the system.

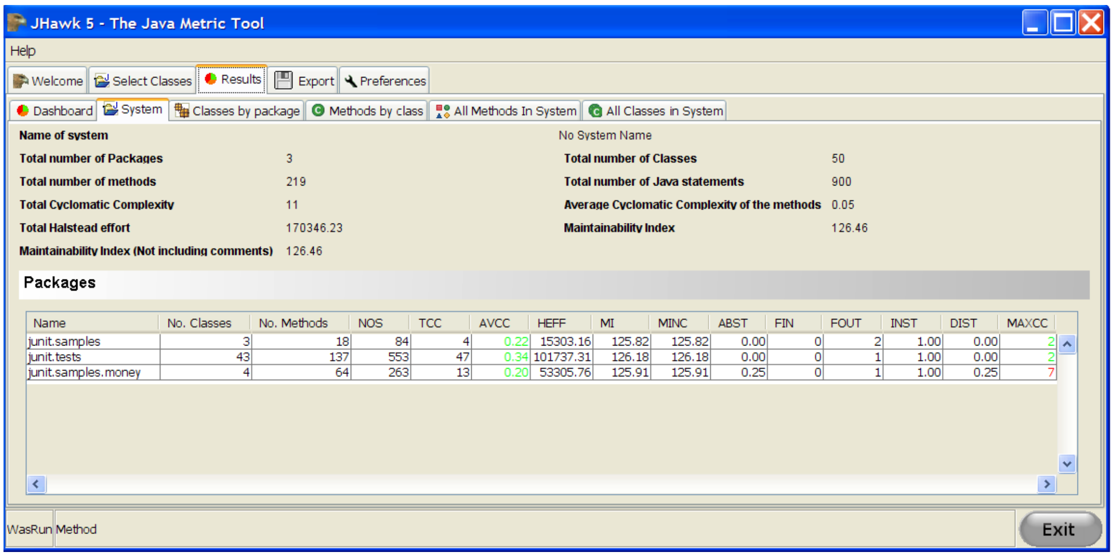
- The Package tab shows all the metrics assigned to the dashboard at class and method level. A dropdown list at the top of the pane allows you to select the package for which you wish the data to be shown.

- The Class tab shows all the metrics assigned to the dashboard at method level. A dropdown list at the top of the pane allows you to select the class for which you wish the data to be shown.



In both the Package and Class sub tabs the drop down lists of sub-elements are ordered according to whether they have any metrics in the danger category, then in the warning category then in the safe category. When you select an individual element in the list will have its background coloured according to the highest level of metric selected for the dashboard. Note that only metrics selected for the dashboard will be considered when ascribing both the order and the background colour of the selected item.

Results Tab - System sub-tab



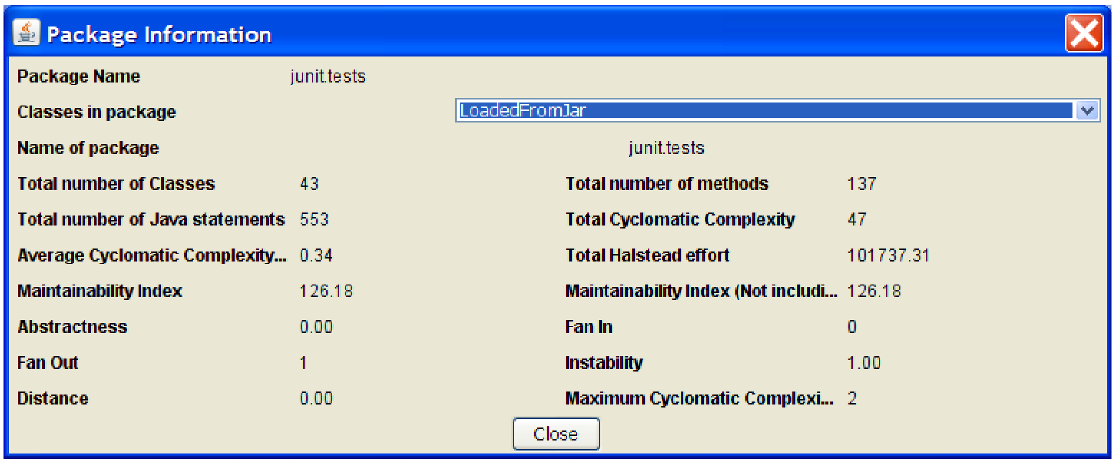
Name of system		No System Name	
Total number of Packages	3	**Total number of Classes**	50
Total number of methods	219	**Total number of Java statements**	900
Total Cyclomatic Complexity	11	**Average Cyclomatic Complexity of the methods**	0.05
Total Halstead effort	170346.23	**Maintainability Index**	126.46
Maintainability Index (Not including comments)	126.46		

Packages

Name	No. Classes	No. Methods	NOS	TCC	AVCC	HEFF	MI	MINC	ABST	FIN	FOUT	INST	DIST	MAXCC
junit.samples	3	18	84	4	0.22	15303.16	125.82	125.82	0.00	0	2	1.00	0.00	2
junit.tests	43	137	553	47	0.34	101737.31	126.18	126.18	0.00	0	1	1.00	0.00	2
junit.samples.money	4	64	263	13	0.20	53305.76	125.91	125.91	0.25	0	1	1.00	0.25	7

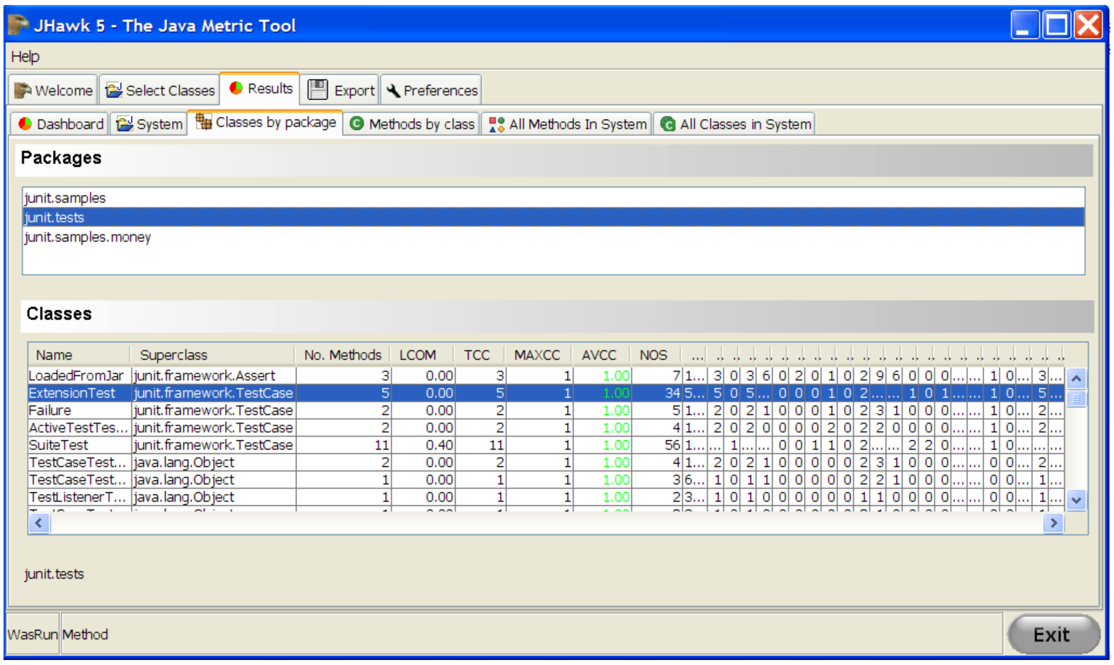
The System sub-tab summarises metrics data at the System level and presents metrics data for each of the packages in tabular form.

The System tab is divided horizontally into two panels – the top panel shows metrics data collected at the System level. The lower panel is a table showing a list of the packages in the system with the metrics collected in each column. Only the active metrics are shown. You can configure the metrics shown in the table by using the Preferences tab (see the section – 'Preferences Tab – Configuring Metrics)'. You can see all the metrics collected for an individual package by double clicking on the listing for that package in the table. The metrics will then be displayed in the format below –



Package Name	junit.tests		
Classes in package		LoadedFromJar	
Name of package		junit.tests	
Total number of Classes	43	**Total number of methods**	137
Total number of Java statements	553	**Total Cyclomatic Complexity**	47
Average Cyclomatic Complexity...	0.34	**Total Halstead effort**	101737.31
Maintainability Index	126.18	**Maintainability Index (Not includi...**	126.18
Abstractness	0.00	**Fan In**	0
Fan Out	1	**Instability**	1.00
Distance	0.00	**Maximum Cyclomatic Complexi...**	2

Results tab -Classes By Package sub-tab



junit.samples
junit.tests
junit.samples.money

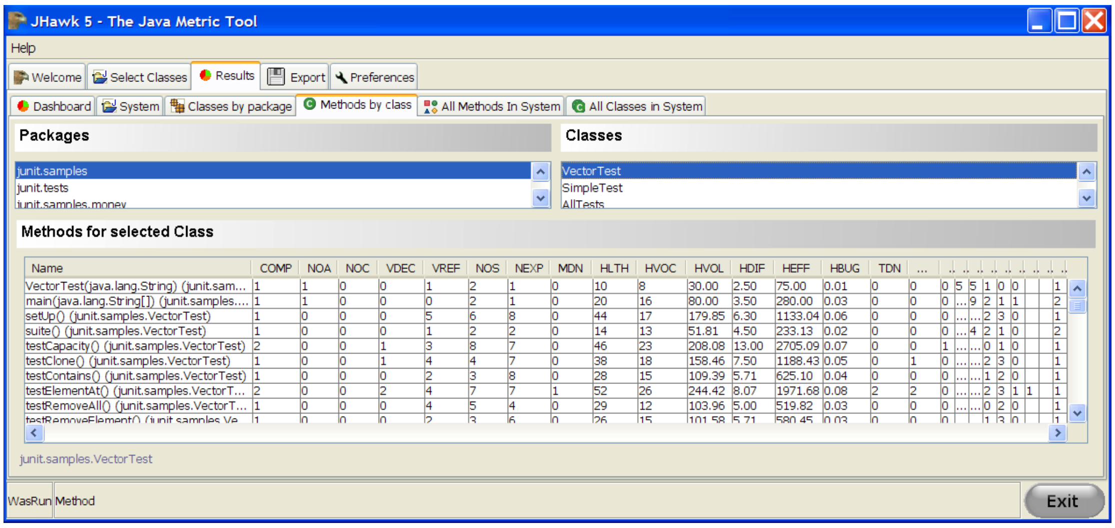
Name	Superclass	No. Methods	LCOM	TCC	MAXCC	AVCC	NOS	...	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..
LoadedFromJar	junit.framework.Assert	3	0.00	3	1	1.00	7	1...	3	0	3	6	0	2	0	1	0	2	9	6	0	0	0	...	...	1	0	...	3	...
ExtensionTest	junit.framework.TestCase	5	0.00	5	1	1.00	34	5...	5	0	5	...	0	0	0	1	0	2	...	...	1	0	1	...	...	1	0	...	5	...
Failure	junit.framework.TestCase	2	0.00	2	1	1.00	5	1...	2	0	2	1	0	0	0	1	0	2	3	1	0	0	0	...	...	1	0	...	2	...
ActiveTestTes...	junit.framework.TestCase	2	0.00	2	1	1.00	4	1...	2	0	2	0	0	0	0	2	0	2	2	0	0	0	0	...	...	1	0	...	2	...
SuiteTest	junit.framework.TestCase	11	0.40	11	1	1.00	56	1...	...	1	...	...	0	0	1	1	0	2	...	...	2	2	0	...	...	1	0	...	...	...
TestCaseTest...	java.lang.Object	2	0.00	2	1	1.00	4	1...	2	0	2	1	0	0	0	0	0	2	3	1	0	0	0	...	...	0	0	...	2	...
TestCaseTest...	java.lang.Object	1	0.00	1	1	1.00	3	6...	1	0	1	1	0	0	0	0	0	2	2	1	0	0	0	...	...	0	0	...	1	...
TestListenerT...	java.lang.Object	1	0.00	1	1	1.00	2	3...	1	0	1	0	0	0	0	0	0	1	1	0	0	0	0	...	...	0	0	...	1	...

junit.tests

The Classes by Package tab allows you to select a Package from the list of packages analysed by JHawk and presents metrics data for each of the classes in the package in tabular form.

The Packages tab is divided horizontally into two panels – the top panel shows the list of Packages in the system. When you select a package in the list the lower panel will display a table showing a list of the classes in the package with the metrics collected in each column. Only the active metrics are shown. You can configure the metrics shown in the table by using the Preferences tab (see the section – 'Preferences Tab – Configuring Metrics)'. You can see all the metrics collected for an individual class by double clicking on the listing for that class in the table.

Results Tab - Methods By Class sub-tab



junit.samples
junit.tests
junit.samples.money

VectorTest
SimpleTest
AllTests

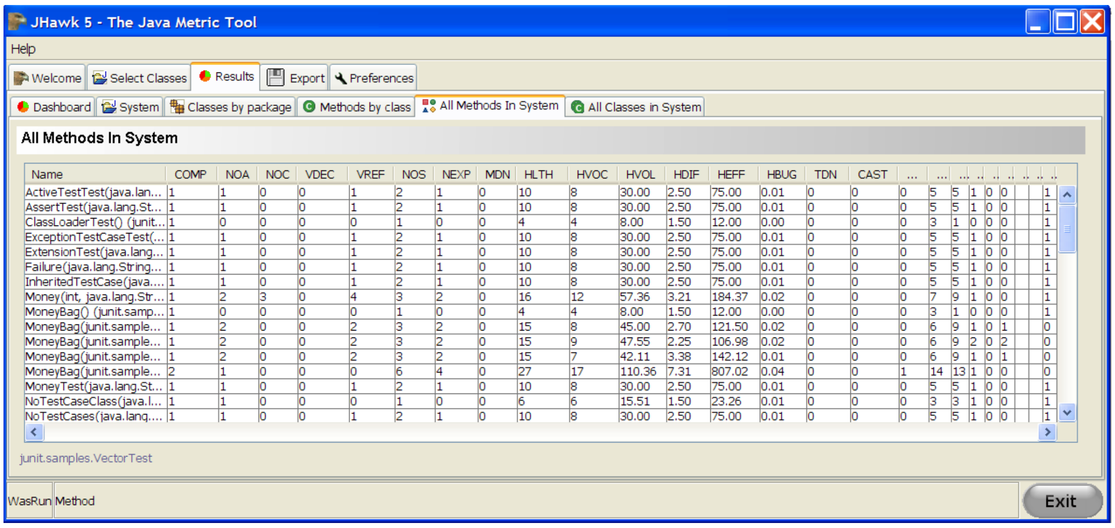
Name	COMP	NOA	NOC	VDEC	VREF	NOS	NEXP	MDN	HLTH	HVOC	HVOL	HDIF	HEFF	HBUG	TDN	...	..	..	..	..	..	..	..	..	..
VectorTest(java.lang.String) (junit.sam...	1	1	0	0	1	2	1	0	10	8	30.00	2.50	75.00	0.01	0	0	0	5	5	1	0	0			1
main(java.lang.String[]) (junit.samples....	1	1	0	0	0	2	1	0	20	16	80.00	3.50	280.00	0.03	0	0	0	...	9	2	1	1			2
setUp() (junit.samples.VectorTest)	1	0	0	0	5	6	8	0	44	17	179.85	6.30	1133.04	0.06	0	0	0	...	...	2	3	0			1
suite() (junit.samples.VectorTest)	1	0	0	0	1	2	2	0	14	13	51.81	4.50	233.13	0.02	0	0	0	...	4	2	1	0			2
testCapacity() (junit.samples.VectorTest)	2	0	0	1	3	8	7	0	46	23	208.08	13.00	2705.09	0.07	0	0	1	...	...	0	1	0			1
testClone() (junit.samples.VectorTest)	1	0	0	1	4	4	7	0	38	18	158.46	7.50	1188.43	0.05	0	1	0	...	...	2	3	0			1
testContains() (junit.samples.VectorTest)	1	0	0	0	2	3	8	0	28	15	109.39	5.71	625.10	0.04	0	0	0	...	...	1	2	0			1
testElementAt() (junit.samples.VectorT...	2	0	0	2	4	7	7	1	52	26	244.42	8.07	1971.68	0.08	2	2	0	...	...	2	3	1	1		1
testRemoveAll() (junit.samples.VectorT...	1	0	0	0	4	5	4	0	29	12	103.96	5.00	519.82	0.03	0	0	0	...	...	0	2	0			1
testRemoveElement() (junit.samples.Ve...	1	0	0	0	2	3	6	0	26	15	101.58	5.71	580.45	0.03	0	0	0			1	3	0			1

junit.samples.VectorTest

The Methods by Class tab allows you to select a Package from the list of packages analysed by JHawk, then a class from the list of classes in that package and presents metrics data for each of the methods in the selected class in tabular form.

The Methods by Class tab is divided horizontally into three panels – the top left panel shows the list of Packages in the system. When you select a package in the list the top right panel will display a list of classes in that package. When you select a class the bottom panel will display a table showing a list of the methods in the class with the metrics collected in each column. Only the active metrics are shown. You can configure the metrics shown in the table by using the Preferences tab (see the section – 'Preferences Tab – Configuring Metrics)'. You can see all the metrics collected for an individual method by double clicking on the listing for that method in the table.

Results Tab - All Methods in System sub-tab



All Methods In System

Name	COMP	NOA	NOC	VDEC	VREF	NOS	NEXP	MDN	HLTH	HVOC	HVOL	HDIF	HEFF	HBUG	TDN	CAST	...	...	...	..	..	..	..	..	..
ActiveTestTest(java.lan...	1	1	0	0	1	2	1	0	10	8	30.00	2.50	75.00	0.01	0	0	0	5	5	1	0	0			1
AssertTest(java.lang.St...	1	1	0	0	1	2	1	0	10	8	30.00	2.50	75.00	0.01	0	0	0	5	5	1	0	0			1
ClassLoaderTest() (junit...	1	0	0	0	0	1	0	0	4	4	8.00	1.50	12.00	0.00	0	0	0	3	1	0	0	0			1
ExceptionTestCaseTest(...	1	1	0	0	1	2	1	0	10	8	30.00	2.50	75.00	0.01	0	0	0	5	5	1	0	0			1
ExtensionTest(java.lang...	1	1	0	0	1	2	1	0	10	8	30.00	2.50	75.00	0.01	0	0	0	5	5	1	0	0			1
Failure(java.lang.String...	1	1	0	0	1	2	1	0	10	8	30.00	2.50	75.00	0.01	0	0	0	5	5	1	0	0			1
InheritedTestCase(java....	1	1	0	0	1	2	1	0	10	8	30.00	2.50	75.00	0.01	0	0	0	5	5	1	0	0			1
Money(int, java.lang.Str...	1	2	3	0	4	3	2	0	16	12	57.36	3.21	184.37	0.02	0	0	0	7	9	1	0	0			1
MoneyBag() (junit.samp...	1	0	0	0	0	1	0	0	4	4	8.00	1.50	12.00	0.00	0	0	0	3	1	0	0	0			1
MoneyBag(junit.sample...	1	2	0	0	2	3	2	0	15	8	45.00	2.70	121.50	0.02	0	0	0	6	9	1	0	1			0
MoneyBag(junit.sample...	1	2	0	0	2	3	2	0	15	9	47.55	2.25	106.98	0.02	0	0	0	6	9	2	0	2			0
MoneyBag(junit.sample...	1	2	0	0	2	3	2	0	15	7	42.11	3.38	142.12	0.01	0	0	0	6	9	1	0	1			0
MoneyBag(junit.sample...	2	1	0	0	0	6	4	0	27	17	110.36	7.31	807.02	0.04	0	0	1	14	13	1	0	0			0
MoneyTest(java.lang.St...	1	1	0	0	1	2	1	0	10	8	30.00	2.50	75.00	0.01	0	0	0	5	5	1	0	0			1
NoTestCaseClass(java.l...	1	1	0	0	0	1	0	0	6	6	15.51	1.50	23.26	0.01	0	0	0	3	3	1	0	0			1
NoTestCases(java.lang....	1	1	0	0	1	2	1	0	10	8	30.00	2.50	75.00	0.01	0	0	0	5	5	1	0	0			1

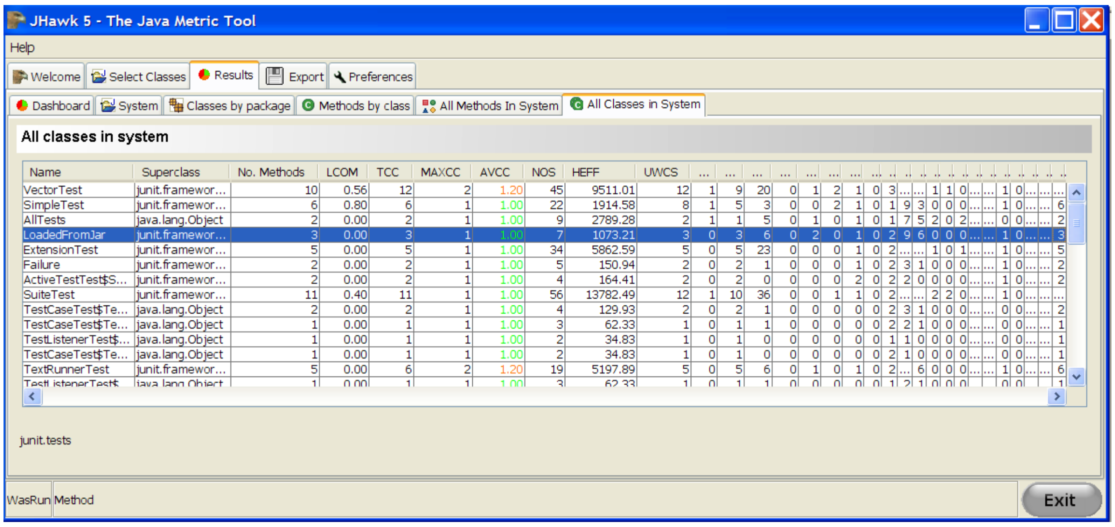
junit.samples.VectorTest

WasRun Method

By default the All Methods in System Tab is not enabled – this is because it expensive to create in terms of memory and processing power and for Systems containing a very large number of methods it could slow down the analysis process. To enable the All Methods in System tab you need to use the Preferences Tab and go the to the 'General' sub-tab. See the section 'Preferences Tab– General Preferences'.

The All Methods tab consists of a table showing a list of all the methods in the System with the metrics collected in each column. Only the active metrics are shown. You can configure the metrics shown in the table by using the Preferences tab (see the section – 'Preferences Tab – Configuring Metrics)'. You can see all the metrics collected for an individual method by double clicking on the listing for that method in the table.

Results Tab - All Classes In System sub-tab



All classes in system

Name	Superclass	No. Methods	LCOM	TCC	MAXCC	AVCC	NOS	HEFF	UWCS	...	...	...	...	...	...	...	...	..	..	..	..	..	..	..	..	..	..	..	..	..
VectorTest	junit.framewor...	10	0.56	12	2	1.20	45	9511.01	12	1	9	20	0	1	2	1	0	3	...	...	1	1	0	...	...	1	0	...	...	...
SimpleTest	junit.framewor...	6	0.80	6	1	1.00	22	1914.58	8	1	5	3	0	0	2	1	0	1	9	3	0	0	0	...	...	1	0	...	...	6
AllTests	java.lang.Object	2	0.00	2	1	1.00	9	2789.28	2	1	1	5	0	1	0	1	0	1	7	5	2	0	2	...	...	0	0	...	...	2
LoadedFromJar	junit.framewor...	3	0.00	3	1	1.00	7	1073.21	3	0	3	6	0	2	0	1	0	2	9	6	0	0	0	...	...	1	0	...	...	3
ExtensionTest	junit.framewor...	5	0.00	5	1	1.00	34	5862.59	5	0	5	23	0	0	0	1	0	2	...	...	1	0	1	...	...	1	0	...	...	5
Failure	junit.framewor...	2	0.00	2	1	1.00	5	150.94	2	0	2	1	0	0	0	1	0	2	3	1	0	0	0	...	...	1	0	...	...	2
ActiveTestTest$S...	junit.framewor...	2	0.00	2	1	1.00	4	164.41	2	0	2	0	0	0	0	2	0	2	2	0	0	0	0	...	...	1	0	...	...	2
SuiteTest	junit.framewor...	11	0.40	11	1	1.00	56	13782.49	12	1	10	36	0	0	1	1	0	2	...	...	2	2	0	...	...	1	0	...	...	...
TestCaseTest$Te...	java.lang.Object	2	0.00	2	1	1.00	4	129.93	2	0	2	1	0	0	0	0	0	2	3	1	0	0	0	...	...	0	0	...	...	2
TestCaseTest$Te...	java.lang.Object	1	0.00	1	1	1.00	3	62.33	1	0	1	1	0	0	0	0	0	2	2	1	0	0	0	...	...	0	0	...	...	1
TestListenerTest$...	java.lang.Object	1	0.00	1	1	1.00	2	34.83	1	0	1	0	0	0	0	0	0	1	1	0	0	0	0	...	...	0	0	...	...	1
TestCaseTest$Te...	java.lang.Object	1	0.00	1	1	1.00	2	34.83	1	0	1	0	0	0	0	0	0	2	1	0	0	0	0	...	...	0	0	...	...	1
TextRunnerTest	junit.framewor...	5	0.00	6	2	1.20	19	5197.89	5	0	5	6	0	1	0	1	0	2	...	6	0	0	0	...	...	1	0	...	...	6
TestListenerTest$	java.lang.Object	1	0.00	1	1	1.00	3	62.33	1	0	1	1	0	0	0	0	0	1	2	1	0	0	0			0	0			1

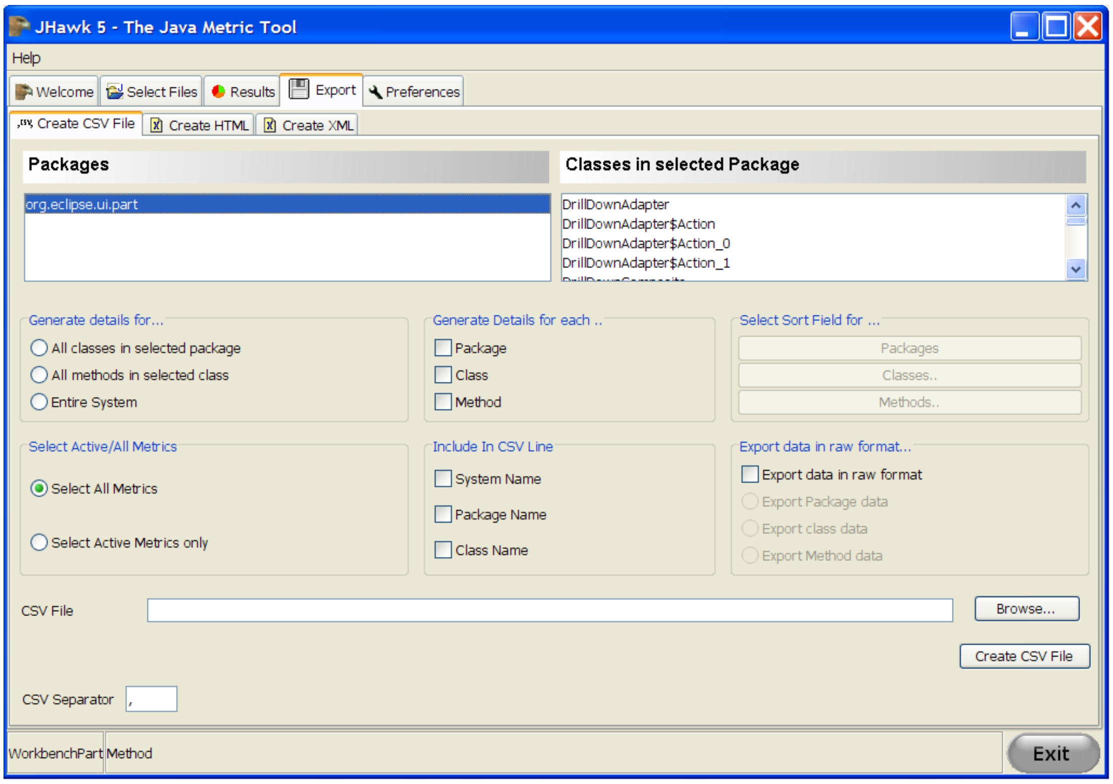
junit.tests

WasRun Method

By default the All Classes in System Tab is not enabled – this is because it expensive to create in terms of memory and processing power and for Systems containing a very large number of classes it could slow down the analysis process. To enable the All Classes in System tab you need to use the Preferences Tab and go the to the ‘General’ sub-tab. See the section ‘Preferences Tab– General Preferences’.

The All Classes tab consists of a table showing a list of all the classes in the System with the metrics collected in each column. Only the active metrics are shown. You can configure the metrics shown in the table by using the Preferences tab (see the section – ‘Preferences Tab – Configuring Metrics)’. You can see all the metrics collected for an individual class by double clicking on the listing for that class in the table.

The Export Tab



org.eclipse.ui.part

DrillDownAdapter
DrillDownAdapter$Action
DrillDownAdapter$Action_0
DrillDownAdapter$Action_1

[✓ Select All Metrics] Select Active Metrics only

CSV Separator: ,

WorkbenchPart Method

The Export tab opens on the ‘Create CSV File’ sub-tab, which is used to export data in CSV format. This is the first of three sub-tabs – the other two are the ‘Create HTML’ tab (which allows you to export data in HTML format) and the ‘Create XML’ tab (which allows you to export data in XML format, including the interchange format that is used in the JHawk Data Viewer).

The Create CSV File tab

This Tab helps you to create a single CSV file containing the results of the analysis of the Java files that you have selected.

At the top of the screen you will see two lists which allow you to select packages and or classes whose data you wish to export. Below this to the left you will see a Box entitled ‘Generate Details For…’ and three radio buttons entitled ‘All Classes in selected Package’, ‘All methods in Selected class’ and ‘Entire System’. These selections are related to the two lists above – obviously if you select ‘Entire System’ the selections in the two lists will be ignored and the details for every method in every class in every package in the system will be exported.

Next we come to a box entitled ‘Generate Details for Each…’ and selections for Package, Class and Method – as each of these is selected the ‘Select Sort Field..’ button for each level will be enabled. This allows you to select the field used to sort the data on when it is exported to the CSV file. Clicking on the button opens a dialog that allows you to select the field and the order (‘Ascending’ or ‘Descending’ that the data is to be selected on.

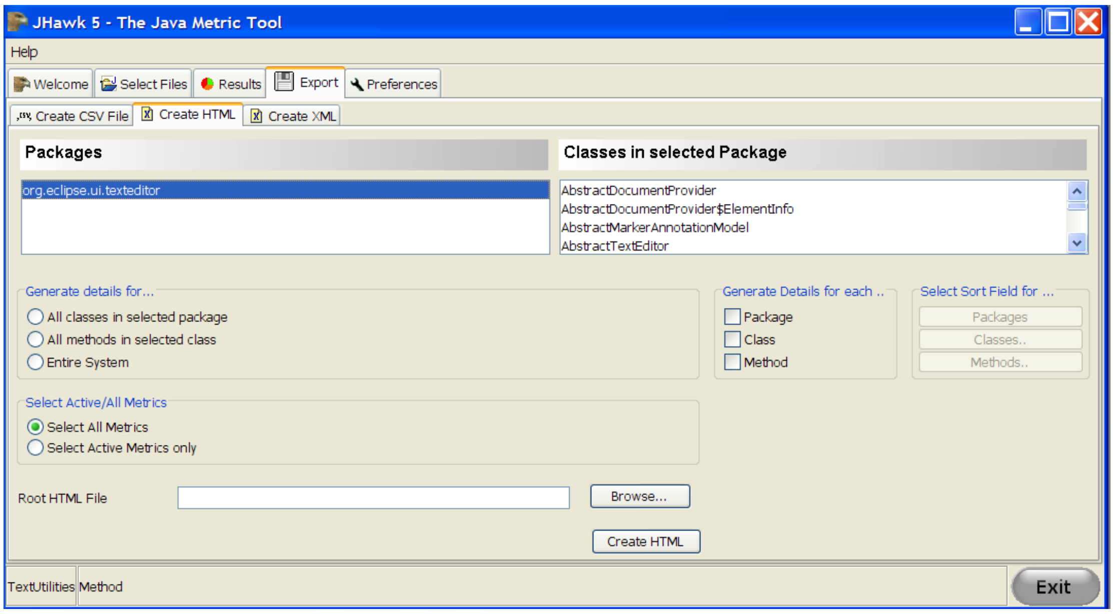
On the next line on the left hand side we can select whether all the metrics are to be exported or only those metrics that are marked as active. To the right of this we can select which name fields will appear on the export line. At the furthest right is an option to allow us to export the data in ‘raw’ format. This simply means that the data is exported on the basis of one line per artefact at the level selected

(Package, Class or Method) . No summary lines are produced with the raw format. Clicking on the raw check box will cause some of the other options to be disabled.

On the next line we can select the file name to which the data is to be exported. You can either enter a file name or use the browse button to select one.

On the bottom line we can select the separator to be used for the exported data (the default is ‘,’).

The Create HTML Tab



org.eclipse.ui.texteditor

AbstractDocumentProvider
AbstractDocumentProvider$ElementInfo
AbstractMarkerAnnotationModel
AbstractTextEditor

( ) All classes in selected package
( ) All methods in selected class
( ) Entire System

☐ Package
☐ Class
☐ Method

(●) Select All Metrics
( ) Select Active Metrics only

Root HTML File

TextUtilities Method

This Tab helps you to create a number of HTML files containing the results of the analysis of the Java files that you have selected. These files are constructed in a logical way from the main HTML file allowing you to ‘drill down’ through the results to find the data that you need.

At the top of the screen you will see two lists which allow to select packages and or classes whose data you wish to export. Below this to the left you will see a Box entitled ‘Generate Details For…’ and three radio buttons entitled ‘All Classes in selected Package’, ‘All methods in Selected class’ and ‘Entire System’. These selections are related to the two lists above – obviously if you select ‘Entire System’ the selections in the two lists will be ignored and the details for every method in every class in every package in the system will be exported.

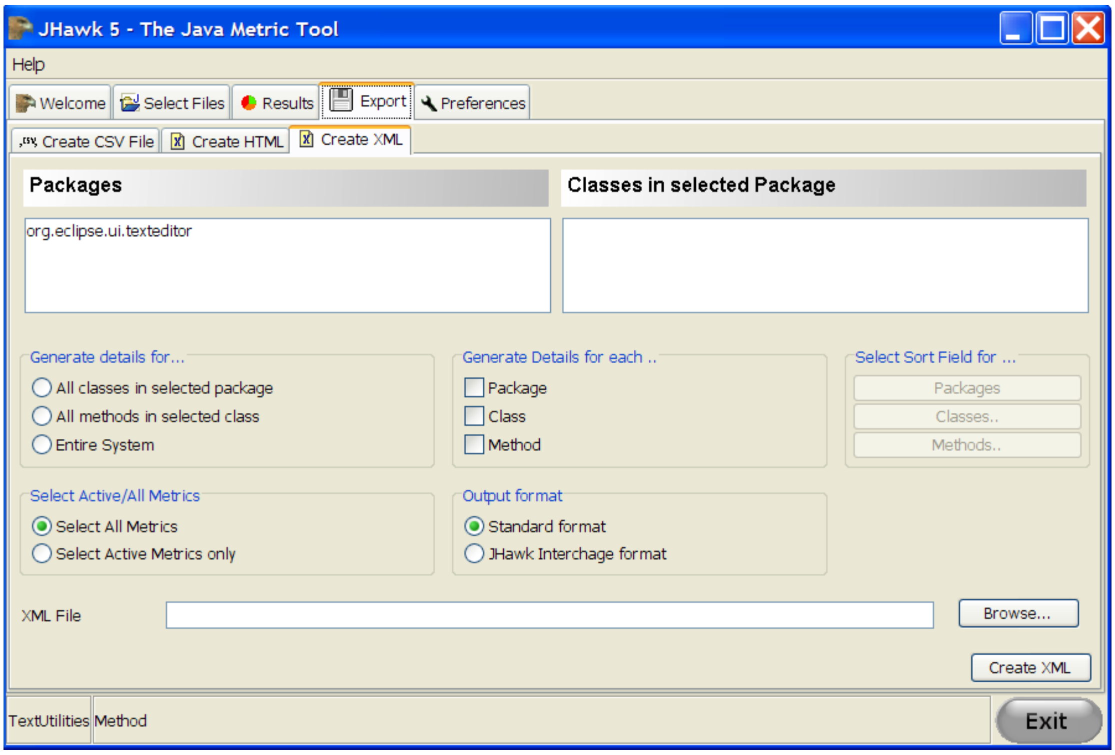
Next we come to a box entitled ‘Generate Details for Each…’ and selections for Package, Class and Method – as each of these is selected the ‘Select Sort Field..’ button for each level will be enabled. This allows you to select the field used to sort the data on when it is exported to the HTML file. Clicking on the button opens a dialog that allows you to select the field and the order (‘Ascending’ or ‘Descending’ that the data is to be selected on.

On the next line we can select whether all the metrics are to be exported or only those metrics that are marked as active.

The final line allows us to select the main HTML file which will act as the index file for all of the other HTML files created. All of the HTML files will be located in the same directory , or sub-directories, of the directory where the root HTML file is located.

The Create XML Tab

This Tab helps you to create a single XML file containing the results of the analysis of the Java files that you have selected.



At the top of the screen you will see two lists which help you select packages and or classes whose data you wish to export. Below this to the left you will see a Box entitled 'Generate Details For…' and three radio buttons entitled 'All Classes in selected Package', 'All methods in Selected class' and 'Entire System'. These selections are related to the two lists above – obviously if you select 'Entire System' the selections in the two lists will be ignored and the details for every method in every class in every package in the system will be exported.

Below this you will see a box entitled 'Generate Details for Each…' and selections for Package, Class and Method – as each of these is selected the 'Select Sort Field..' button for each level will be enabled. This allows you to select the field used to sort the data on when it is exported to the HTML file. Clicking on the button opens a dialog that allows you to select the field and the order ('Ascending' or 'Descending' that the data is to be selected on.

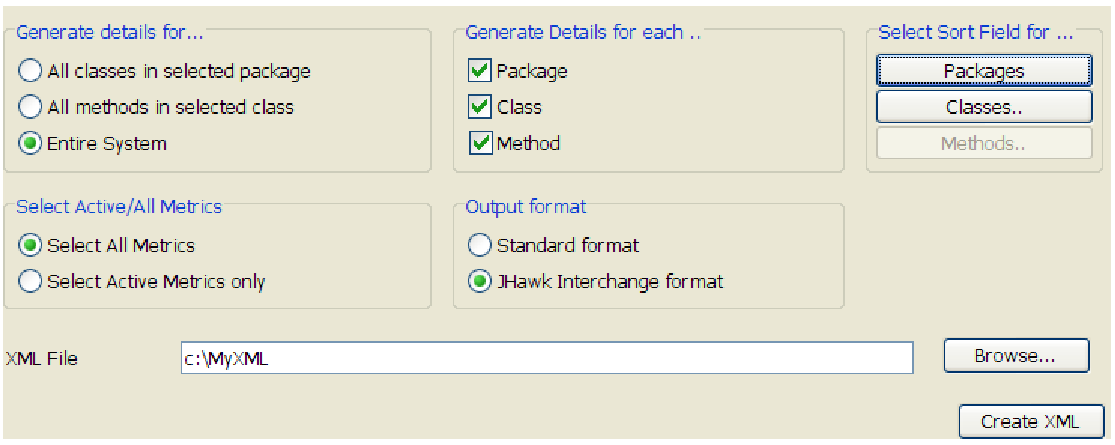
On the next line on the left hand side we can select whether all the metrics are to be exported or only those metrics that are marked as active. On the right side we can select whether the data is exported in standard format (i.e. using the XML tags defined for each metric) or in JHawk Interchange format which can be used to recreate the metric record of a particular analysis set.

On the next line we can select the file name to which the data is to be exported. You can either enter a file name or use the browse button to select one.

Creating files for the Data Viewer

The files used by the Data Viewer must be in the JHawk Metric Interchange Format (JMIF). These are XML files that have been created using the JMIF option when exporting files from JHawk either using the stand alone application, The JHawk Eclipse Plugin or the command line interface.

You must first analyse the Java files, then go to the Export tab and select the 'Create XML' sub tab. On this tab you need to select the 'Entire System' and the 'JHawk Interchange Format' radio buttons. You then need to select the levels that you wish the analysis to occur at – Package, Class or Method. If you want to analyse to a particular level you will need to select the levels above so that the JHawk Data Viewer (which works on a tree basis) can 'drill down' to the lower levels. This means that if you want to see data down to the method level you will need to select the package, class and method levels and if you want to analyse to the class level you will need to select the package and class levels. The screenshot below illustrates the choices that should be made to produce a JHawk Metric Interchange file at the method level -


Generate details for...
( ) All classes in selected package
( ) All methods in selected class
[✓ Entire System]

Generate Details for each ..
✓ Package
✓ Class
✓ Method

Select Active/All Metrics
[✓ Select All Metrics]
( ) Select Active Metrics only

Output format
( ) Standard format
[✓ JHawk Interchange format]

XML File: c:\MyXML

The level to which you wish to carry out analysis will have a bearing on the size of the XML files created. This, in turn, may limit the number of files that JHawk Data Viewer can handle at a given level of memory. As an example the XML file sizes for an analysis of the entire source for Eclipse 3.4 (16,175 source files) were 215Mb at method level, 40Mb at class level and 134k at package level. You can analyse at class or package level but if you wish to analyse metrics that are ultimately based on data collected at method level (and this includes many of the important metrics such as the Halstead Metrics and Cyclomatic Complexity) you will need to analyse right down to the method level.

If you wish to compare files in the tree views (Text Compare and Graph) then all of your systems should have the same name (or all have no name). You can set the system name in the 'Select Files' tab. The default value of the System name is 'No System Name'.

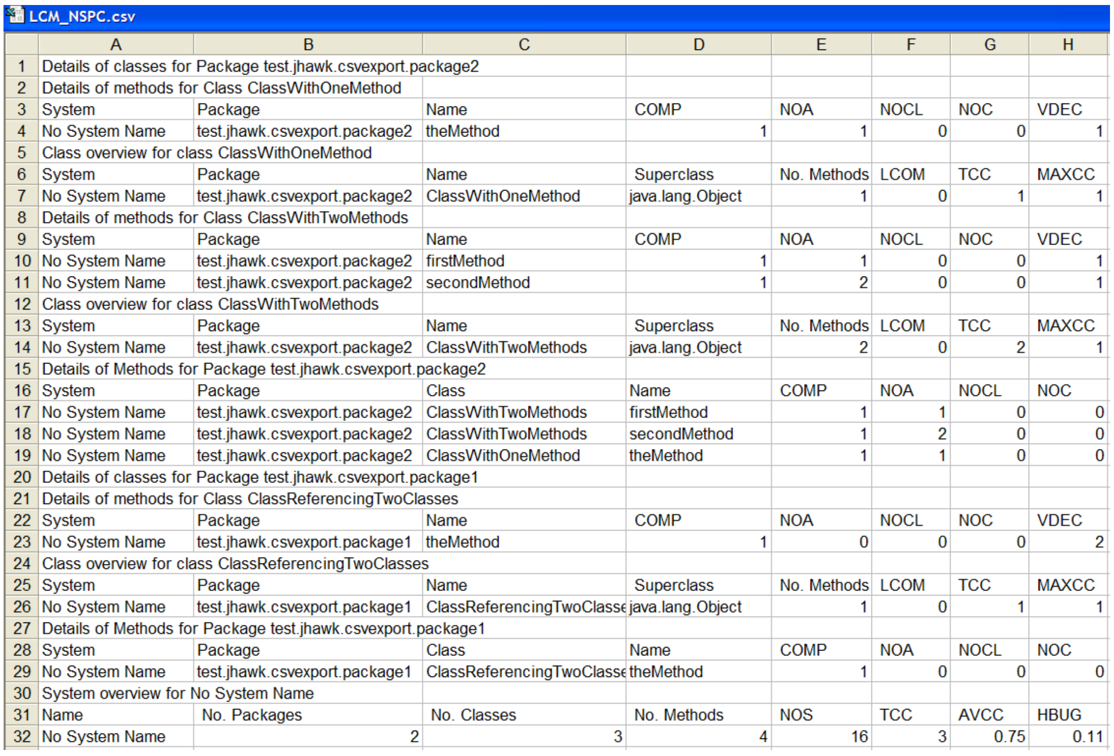
Creating files to load back into JHawk Stand Alone

If you are creating files to view again in the JHawk stand alone version the same criteria apply as for the JHawk Data Viewer

Output Formats

CSV – Standard

In the standard CSV export format a summary record is written at each level in addition to the data at that level.

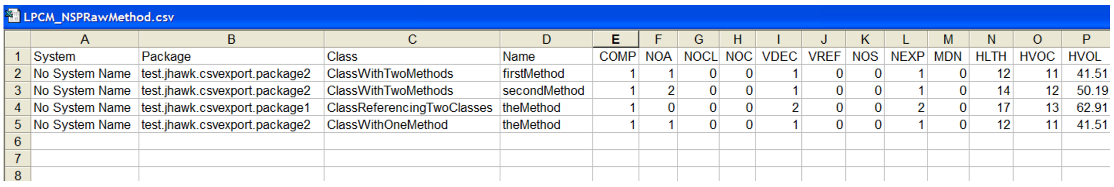


LCM_NSPC.csv

	A	B	C	D	E	F	G	H
1	Details of classes for Package test.jhawk.csvexport.package2							
2	Details of methods for Class ClassWithOneMethod							
3	System	Package	Name	COMP	NOA	NOCL	NOC	VDEC
4	No System Name	test.jhawk.csvexport.package2	theMethod	1	1	0	0	1
5	Class overview for class ClassWithOneMethod							
6	System	Package	Name	Superclass	No. Methods	LCOM	TCC	MAXCC
7	No System Name	test.jhawk.csvexport.package2	ClassWithOneMethod	java.lang.Object	1	0	1	1
8	Details of methods for Class ClassWithTwoMethods							
9	System	Package	Name	COMP	NOA	NOCL	NOC	VDEC
10	No System Name	test.jhawk.csvexport.package2	firstMethod	1	1	0	0	1
11	No System Name	test.jhawk.csvexport.package2	secondMethod	1	2	0	0	1
12	Class overview for class ClassWithTwoMethods							
13	System	Package	Name	Superclass	No. Methods	LCOM	TCC	MAXCC
14	No System Name	test.jhawk.csvexport.package2	ClassWithTwoMethods	java.lang.Object	2	0	2	1
15	Details of Methods for Package test.jhawk.csvexport.package2							
16	System	Package	Class	Name	COMP	NOA	NOCL	NOC
17	No System Name	test.jhawk.csvexport.package2	ClassWithTwoMethods	firstMethod	1	1	0	0
18	No System Name	test.jhawk.csvexport.package2	ClassWithTwoMethods	secondMethod	1	2	0	0
19	No System Name	test.jhawk.csvexport.package2	ClassWithOneMethod	theMethod	1	1	0	0
20	Details of classes for Package test.jhawk.csvexport.package1							
21	Details of methods for Class ClassReferencingTwoClasses							
22	System	Package	Name	COMP	NOA	NOCL	NOC	VDEC
23	No System Name	test.jhawk.csvexport.package1	theMethod	1	0	0	0	2
24	Class overview for class ClassReferencingTwoClasses							
25	System	Package	Name	Superclass	No. Methods	LCOM	TCC	MAXCC
26	No System Name	test.jhawk.csvexport.package1	ClassReferencingTwoClasse	java.lang.Object	1	0	1	1
27	Details of Methods for Package test.jhawk.csvexport.package1							
28	System	Package	Class	Name	COMP	NOA	NOCL	NOC
29	No System Name	test.jhawk.csvexport.package1	ClassReferencingTwoClasse	theMethod	1	0	0	0
30	System overview for No System Name							
31	Name	No. Packages	No. Classes	No. Methods	NOS	TCC	AVCC	HBUG
32	No System Name	2	3	4	16	3	0.75	0.11

CSV – Raw

When data is exported in the raw CSV format a single record for each artefact is written at the requested level. In the example here the data has been exported at the method level. In each case the fully qualified name will be available with the System, package, class and method name written in the columns at the left.

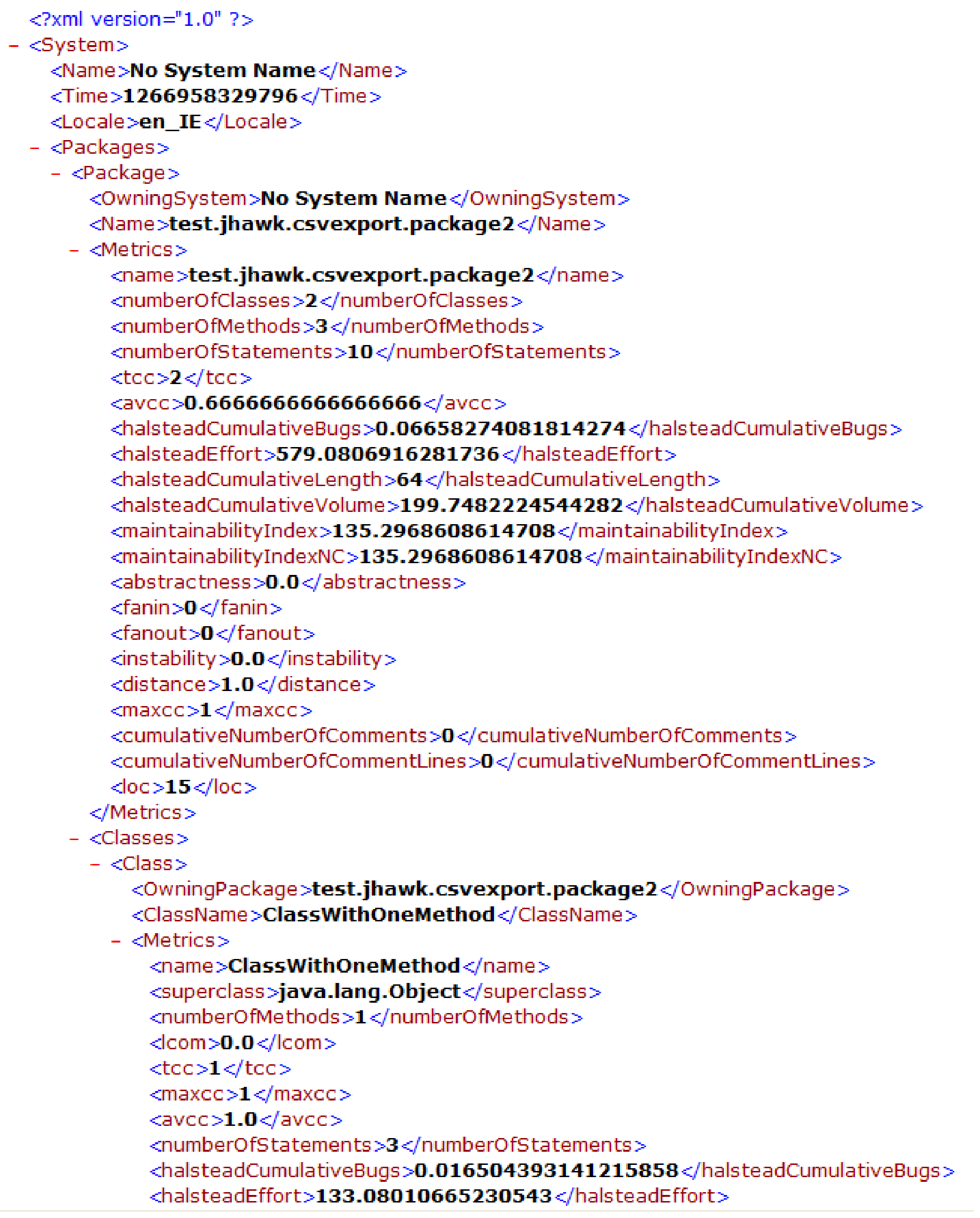


LPCM_NSPRawMethod.csv

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	System	Package	Class	Name	COMP	NOA	NOCL	NOC	VDEC	VREF	NOS	NEXP	MDN	HLTH	HVOC	HVOL
2	No System Name	test.jhawk.csvexport.package2	ClassWithTwoMethods	firstMethod	1	1	0	0	1	0	0	1	0	12	11	41.51
3	No System Name	test.jhawk.csvexport.package2	ClassWithTwoMethods	secondMethod	1	2	0	0	1	0	0	1	0	14	12	50.19
4	No System Name	test.jhawk.csvexport.package1	ClassReferencingTwoClasses	theMethod	1	0	0	0	2	0	0	2	0	17	13	62.91
5	No System Name	test.jhawk.csvexport.package2	ClassWithOneMethod	theMethod	1	1	0	0	1	0	0	1	0	12	11	41.51
6																
7																
8																

XML – Standard

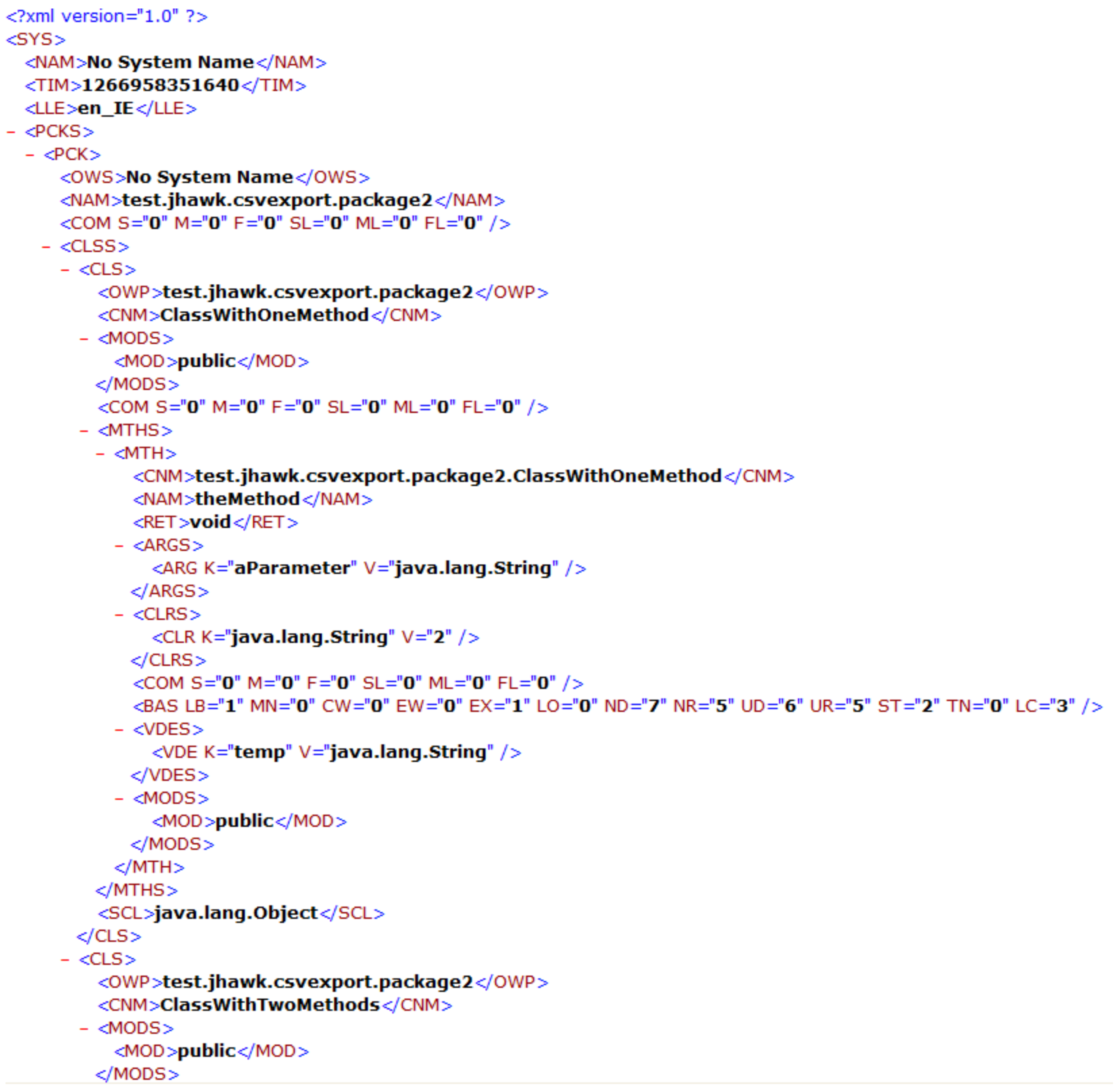
XML output can be written in two forms – Standard or JHawk Interchange . In the standard format xml output is written for the selected metrics (active or all) and at the relevant levels (System, Package, Class, Method etc). Sample output is shown below.



```
  <?xml version="1.0" ?>
- <System>
    <Name>No System Name</Name>
    <Time>1266958329796</Time>
    <Locale>en_IE</Locale>
  - <Packages>
    - <Package>
        <OwningSystem>No System Name</OwningSystem>
        <Name>test.jhawk.csvexport.package2</Name>
      - <Metrics>
          <name>test.jhawk.csvexport.package2</name>
          <numberOfClasses>2</numberOfClasses>
          <numberOfMethods>3</numberOfMethods>
          <numberOfStatements>10</numberOfStatements>
          <tcc>2</tcc>
          <avcc>0.6666666666666666</avcc>
          <halsteadCumulativeBugs>0.06658274081814274</halsteadCumulativeBugs>
          <halsteadEffort>579.0806916281736</halsteadEffort>
          <halsteadCumulativeLength>64</halsteadCumulativeLength>
          <halsteadCumulativeVolume>199.7482224544282</halsteadCumulativeVolume>
          <maintainabilityIndex>135.2968608614708</maintainabilityIndex>
          <maintainabilityIndexNC>135.2968608614708</maintainabilityIndexNC>
          <abstractness>0.0</abstractness>
          <fanin>0</fanin>
          <fanout>0</fanout>
          <instability>0.0</instability>
          <distance>1.0</distance>
          <maxcc>1</maxcc>
          <cumulativeNumberOfComments>0</cumulativeNumberOfComments>
          <cumulativeNumberOfCommentLines>0</cumulativeNumberOfCommentLines>
          <loc>15</loc>
        </Metrics>
      - <Classes>
        - <Class>
            <OwningPackage>test.jhawk.csvexport.package2</OwningPackage>
            <ClassName>ClassWithOneMethod</ClassName>
          - <Metrics>
              <name>ClassWithOneMethod</name>
              <superclass>java.lang.Object</superclass>
              <numberOfMethods>1</numberOfMethods>
              <lcom>0.0</lcom>
              <tcc>1</tcc>
              <maxcc>1</maxcc>
              <avcc>1.0</avcc>
              <numberOfStatements>3</numberOfStatements>
              <halsteadCumulativeBugs>0.016504393141215858</halsteadCumulativeBugs>
              <halsteadEffort>133.08010665230543</halsteadEffort>
```

XML – JHawk Interchange Format

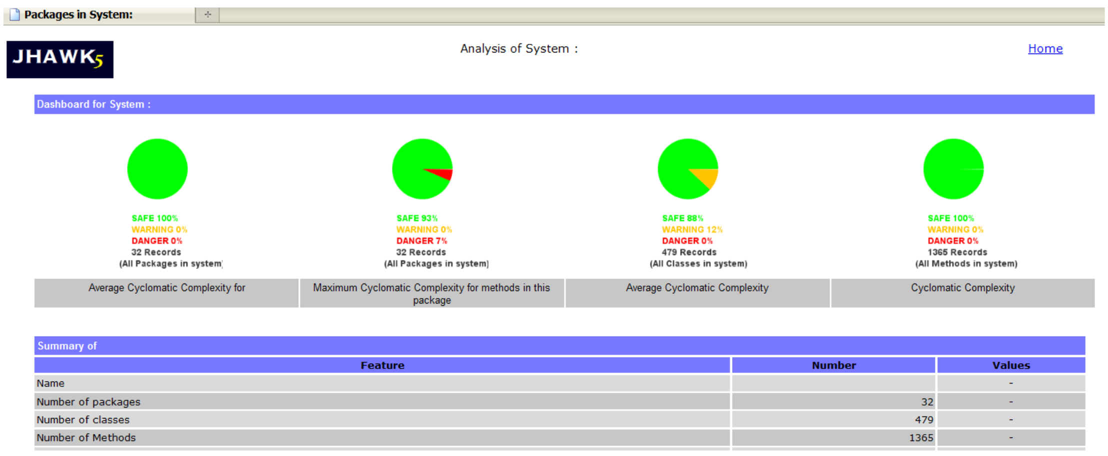
If you select the JHawk Interchange format for your XML output then the data is exported for the entire system at all levels. As this is the base data collected for your system this data can be used to completely rebuild the dataset inside either JHawk or the JHawk DataViewer add on.



```
<?xml version="1.0" ?>
<SYS>
  <NAM>No System Name</NAM>
  <TIM>1266958351640</TIM>
  <LLE>en_IE</LLE>
- <PCKS>
  - <PCK>
      <OWS>No System Name</OWS>
      <NAM>test.jhawk.csvexport.package2</NAM>
      <COM S="0" M="0" F="0" SL="0" ML="0" FL="0" />
    - <CLSS>
      - <CLS>
          <OWP>test.jhawk.csvexport.package2</OWP>
          <CNM>ClassWithOneMethod</CNM>
        - <MODS>
            <MOD>public</MOD>
          </MODS>
          <COM S="0" M="0" F="0" SL="0" ML="0" FL="0" />
        - <MTHS>
          - <MTH>
              <CNM>test.jhawk.csvexport.package2.ClassWithOneMethod</CNM>
              <NAM>theMethod</NAM>
              <RET>void</RET>
            - <ARGS>
                <ARG K="aParameter" V="java.lang.String" />
              </ARGS>
            - <CLRS>
                <CLR K="java.lang.String" V="2" />
              </CLRS>
              <COM S="0" M="0" F="0" SL="0" ML="0" FL="0" />
              <BAS LB="1" MN="0" CW="0" EW="0" EX="1" LO="0" ND="7" NR="5" UD="6" UR="5" ST="2" TN="0" LC="3" />
            - <VDES>
                <VDE K="temp" V="java.lang.String" />
              </VDES>
            - <MODS>
                <MOD>public</MOD>
              </MODS>
            </MTH>
          </MTHS>
          <SCL>java.lang.Object</SCL>
        </CLS>
      - <CLS>
          <OWP>test.jhawk.csvexport.package2</OWP>
          <CNM>ClassWithTwoMethods</CNM>
        - <MODS>
            <MOD>public</MOD>
          </MODS>
```

HTML

HTML data is output at the levels selected and the structure of the pages output will depend on the levels selected. At the very least a root page will be output showing the highest level selected. The data from all the other levels selected can be accessed from this.



Analysis of System :

Dashboard for System :

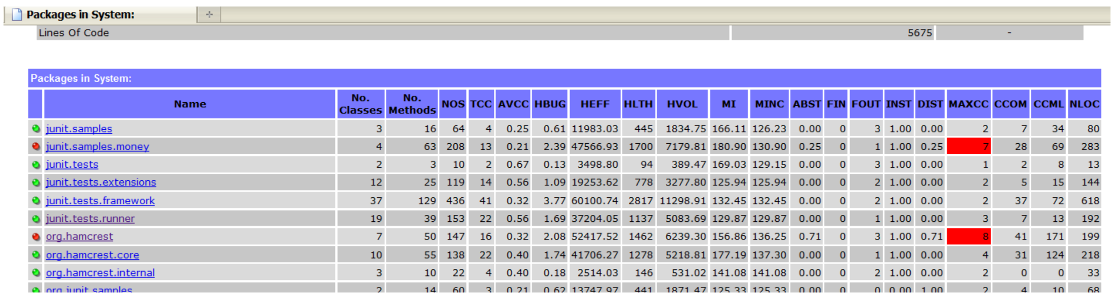
Average Cyclomatic Complexity for	Maximum Cyclomatic Complexity for methods in this package	Average Cyclomatic Complexity	Cyclomatic Complexity
SAFE 100%	SAFE 93%	SAFE 88%	SAFE 100%
WARNING 0%	WARNING 0%	WARNING 12%	WARNING 0%
DANGER 0%	DANGER 7%	DANGER 0%	DANGER 0%
32 Records	32 Records	479 Records	1365 Records
(All Packages in system)	(All Packages in system)	(All Classes in system)	(All Methods in system)

Summary of

Feature	Number	Values
Name		-
Number of packages	32	-
Number of classes	479	-
Number of Methods	1365	-

If you have elected to select all levels then the first page of the HTML output will be as above. All the active dashboard gauges will be displayed and a table setting out the system level metrics will be displayed.

Below this a list of all the packages that constitute the system being analysed will be displayed.



Lines Of Code	5675	-

Packages in System:

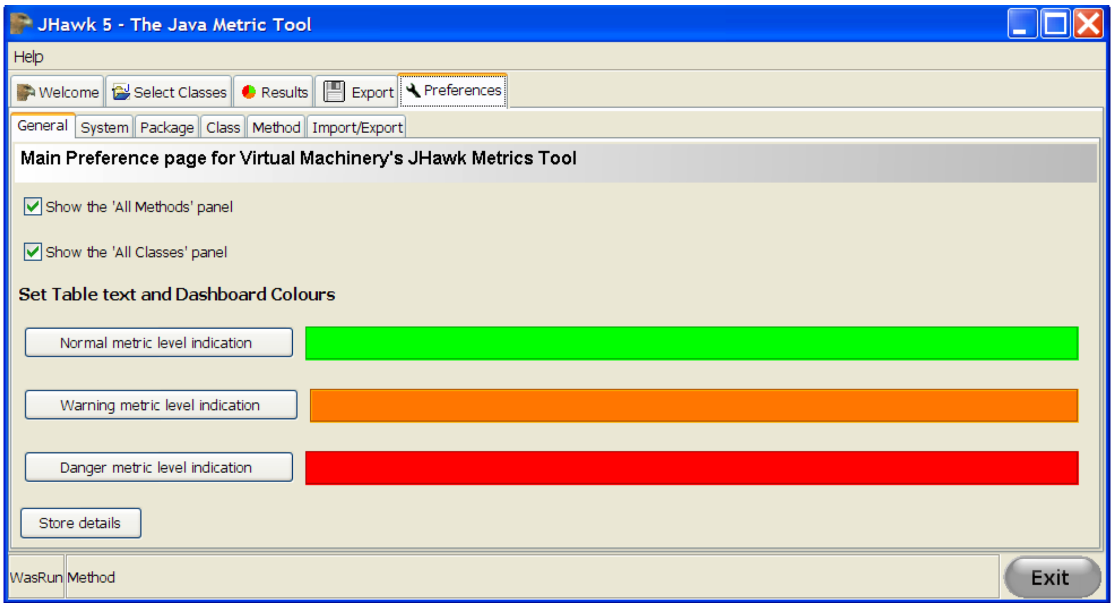
	Name	No. Classes	No. Methods	NOS	TCC	AVCC	HBUG	HEFF	HLTH	HVOL	MI	MINC	ABST	FIN	FOUT	INST	DIST	MAXCC	CCOM	CCML	NLOC
🟢	junit.samples	3	16	64	4	0.25	0.61	11983.03	445	1834.75	166.11	126.23	0.00	0	3	1.00	0.00	2	7	34	80
🔴	junit.samples.money	4	63	208	13	0.21	2.39	47566.93	1700	7179.81	180.90	130.90	0.25	0	1	1.00	0.25	7	28	69	283
🟢	junit.tests	2	3	10	2	0.67	0.13	3498.80	94	389.47	169.03	129.15	0.00	0	3	1.00	0.00	1	2	8	13
🟢	junit.tests.extensions	12	25	119	14	0.56	1.09	19253.62	778	3277.80	125.94	125.94	0.00	0	2	1.00	0.00	2	5	15	144
🟢	junit.tests.framework	37	129	436	41	0.32	3.77	60100.74	2817	11298.91	132.45	132.45	0.00	0	2	1.00	0.00	2	37	72	618
🟢	junit.tests.runner	19	39	153	22	0.56	1.69	37204.05	1137	5083.69	129.87	129.87	0.00	0	1	1.00	0.00	3	7	13	192
🔴	org.hamcrest	7	50	147	16	0.32	2.08	52417.52	1462	6239.30	156.86	136.25	0.71	0	3	1.00	0.71	8	41	171	199
🟢	org.hamcrest.core	10	55	138	22	0.40	1.74	41706.27	1278	5218.81	177.19	137.30	0.00	0	1	1.00	0.00	4	31	124	218
🟢	org.hamcrest.internal	3	10	22	4	0.40	0.18	2514.03	146	531.02	141.08	141.08	0.00	0	2	1.00	0.00	2	0	0	33
🟢	org.junit.samples	2	14	60	3	0.21	0.62	13747.97	441	1871.47	125.33	125.33	0.00	0	0	0.00	1.00	2	4	10	68

This is presented in tabular form with a column for each of the metrics calculated at this level (remember that you can choose whether your HTML output shows all metrics or only those that you have marked as active). There is a row for each package and if any metric in the row has a valid range, and lies outside that range then either a red danger or an orange warning indicator will be displayed to the left of the row. The background of the offending metric(s) will also be set to either red or orange.

You can click on the link in the name column to drill down to the next level. This may or may not be active depending on the levels that you have selected to be displayed in the table.

All pages are basically laid out the same – one or more lines of dashboard gauges (if these have been activated) followed by a list of metrics at that particular level. Below this will be a table of data relating to the level below this with links to that level.

The Preferences Tab



Main Preference page for Virtual Machinery's JHawk Metrics Tool

☑ Show the 'All Methods' panel

☑ Show the 'All Classes' panel

Set Table text and Dashboard Colours

Normal metric level indication
Warning metric level indication
Danger metric level indication
Store details
WasRun
Method
Exit

The Preferences tab opens on the ‘General’ sub-tab which is the first of six sub-tabs on the Preferences tab. The ‘System’, ‘Package’, ‘Class’ and ‘Method’ sub-tabs allow you to configure the metrics available at each of the System, Package, Class and Method levels. The ‘Import/Export’ sub-tab allows you to export your preferences as a file that you can subsequently import in this tab or from the Welcome tab.

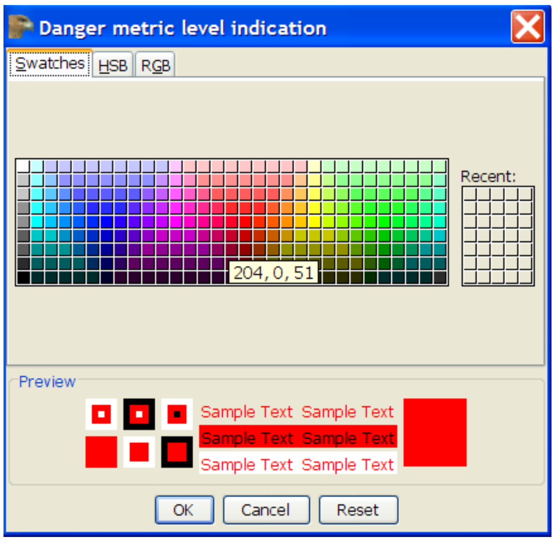
General Preferences

The General Preferences tab allows you to set the following JHawk attributes -

- Whether the ‘All Methods’ panel (a sub-tab of the Results panel) will be populated and displayed. This is left optional as populating this panel can substantially increase the memory and processing burden on JHawk if very large numbers of methods are being analysed.

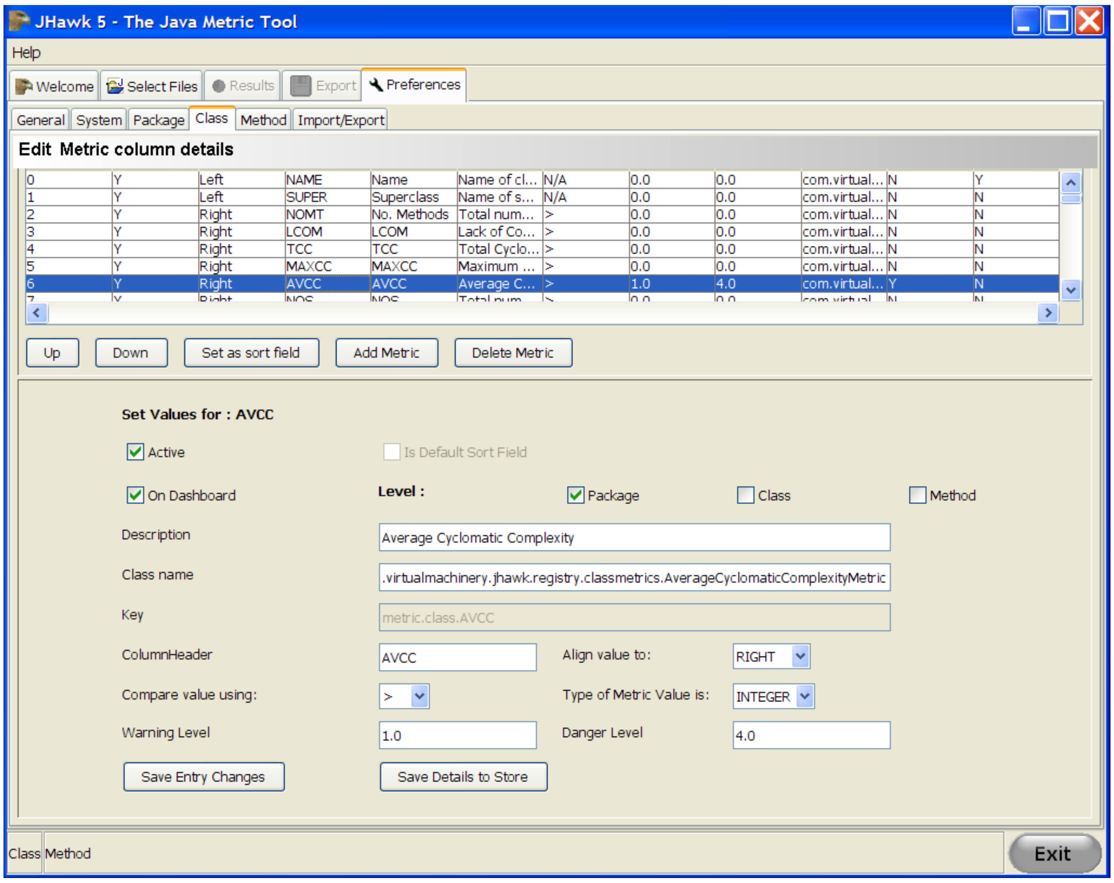
- Whether the ‘All Classes’ panel (a sub-tab of the Results panel) will be populated and displayed. This is left optional as populating this panel can substantially increase the memory and processing burden on JHawk if very large numbers of classes are being analysed.

- Setting the colours used to indicate the Normal, Warning and Danger levels of metrics in the metrics tables and on the dashboard. When the button is pressed the following dialog pops up and the colour to be used can be selected –



If you wish to see these attributes reflected in JHawk then press the ‘Store Details’ button then go back to the ‘Select Files’ tab and run the analysis again – the new settings will then be reflected in JHawk.

Metrics sub-tabs – Edit Metric Column details



Edit Metric column details

0	Y	Left	NAME	Name	Name of cl...	N/A	0.0	0.0	com.virtual...	N	Y
1	Y	Left	SUPER	Superclass	Name of s...	N/A	0.0	0.0	com.virtual...	N	N
2	Y	Right	NOMT	No. Methods	Total num...	>	0.0	0.0	com.virtual...	N	N
3	Y	Right	LCOM	LCOM	Lack of Co...	>	0.0	0.0	com.virtual...	N	N
4	Y	Right	TCC	TCC	Total Cyclo...	>	0.0	0.0	com.virtual...	N	N
5	Y	Right	MAXCC	MAXCC	Maximum ...	>	0.0	0.0	com.virtual...	N	N
6	Y	Right	AVCC	AVCC	Average C...	>	1.0	4.0	com.virtual...	Y	N
7	Y	Right	NOS	NOS	Total num...	>	0.0	0.0	com.virtual...	N	N

Set Values for : AVCC

☑ Active ☐ Is Default Sort Field

☑ On Dashboard **Level :** ☑ Package ☐ Class ☐ Method

Description	Average Cyclomatic Complexity
Class name	.virtualmachinery.jhawk.registry.classmetrics.AverageCyclomaticComplexityMetric
Key	metric.class.AVCC
ColumnHeader	AVCC
Align value to:	RIGHT
Compare value using:	>
Type of Metric Value is:	INTEGER
Warning Level	1.0
Danger Level	4.0

This tab allows you to amend the details of existing metrics and to add and delete metrics. It also allows you to set the order in which metrics will appear in the JHawk tables and which metric will be used as the default sort field. There is a separate tab for each level of metrics – System, Package, Class, Method.

The details relating to the metrics are stored in the jhawk.properties file. All changes to metrics need to be made via the metrics preferences screen. Attempting to edit the jhawk.properties file will probably result in JHawk failing to start successfully. If this happens, or you believe the jhawk.properties file to be corrupt you should replace the file with the original jhawk.properties file that came with your distribution of JHawk.

The Buttons

- The Up button – this moves a metric up the order that it will appear in the metric tables. Moving the metric 'up' the order means that it will display further to the left in the table.
- The Down button – this moves a metric down the order that it will appear in the metric tables. Moving the metric 'up' the order means that it will display further to the right in the table.
- Set as sort field – this will set this metric as the default sort field when the metric table is displayed. This only sets the initial value - you can change the sort field in the table by double clicking on a column header (see the Results Tab Section).
- Add metric – this allows you to add a new metric – see the separate section on this below.
- Delete Metric – this will delete the selected metric from the list of metrics. Unless this is a metric that you have added then you should consider very carefully whether you want to do this. If all you want to do is remove a metric from the tables displayed then you should inactivate it using the 'Active' check box in the metric editing screen – see below.

The Details

It is most likely that you will want to change the following information about a particular metric –

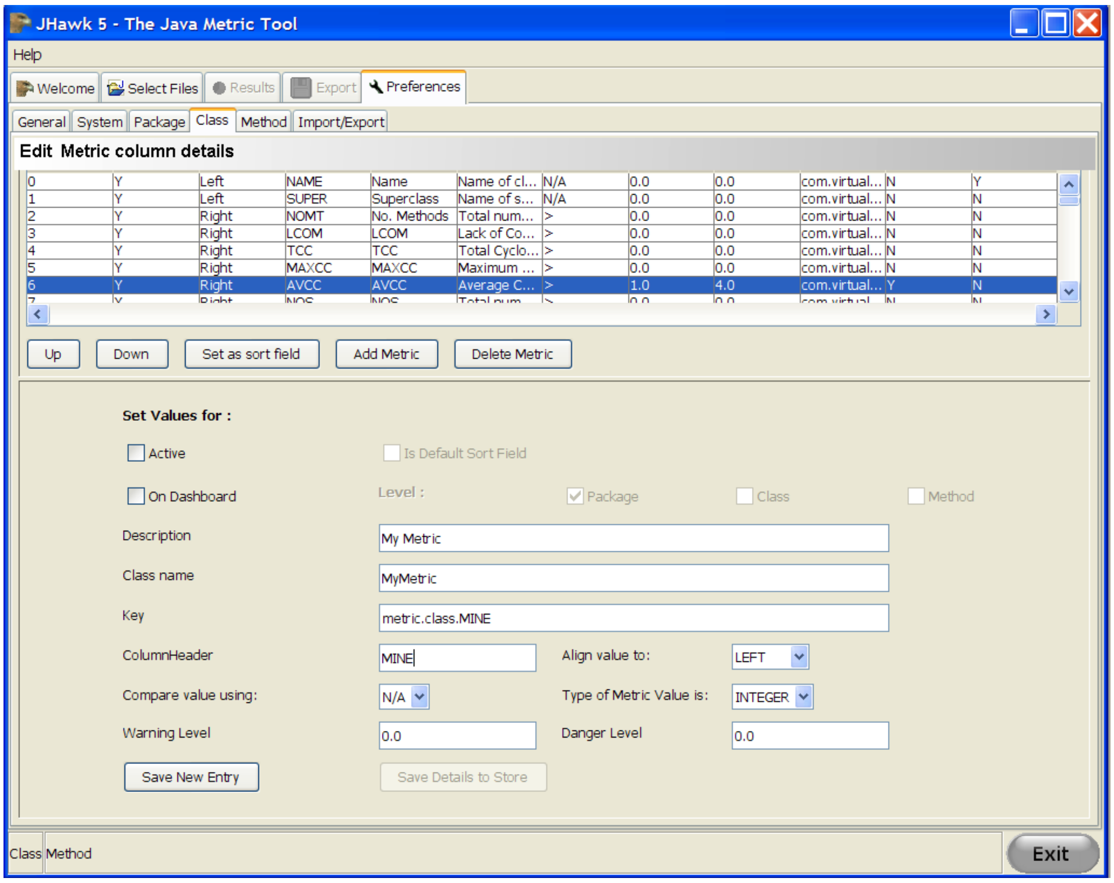
Active – use this to control whether a metric is displayed in any of the tables.
On Dashboard – Enable this to indicate that you want a metric to appear on the JHawk dashboard panel. There are three level indicators which allow you to indicate which level(s) you want the metric to be assessed at. The available levels will depend on the level of the metric selected.
Description – You might choose to describe the metric in your own language or in a different way
Column Header – Again you might change the header to be more descriptive.
Warning Level – You can set the level at which the metric will displayed in the warning colour in the tables and on the dashboard pie charts. This and the danger level will only be available for numerical and list metrics. In the case of list metrics the value will be compared against the count of the number of entries in the list.
Danger Level - You can set the level at which the metric will displayed in the danger colour in the tables and on the dashboard pie charts.

It is unlikely that you will have any need to change the following attributes of a metric except when you are adding a new metric –
Class name – This sets the class name to be used to create the metric. This class must be available when you save the metric (either after adding or editing). Other than the addition of a metric the only reason that you might change the class name of an existing metric would be if you have added a new metric class that re-implements this metric to your particular requirements. You should read the section 'Adding a new metric' before you do this
Align Value – This is a dropdown selection box that allows you to set the alignment of the metric data in a table or display field. The options are LEFT, RIGHT or CENTER
Compare value using: - This value sets the direction of comparison against the Warning and Danger Levels this can either be > (Greater than – the metric will be marked as being in the Warning or Danger zone if the value of the metric is greater than the levels), < (Less than – the metric will be marked as being in the Warning or Danger zone if the value of the metric is greater than the levels) or N/A (Not applicable)
Metric type – This signifies the metric type – the available options are INTEGER, DOUBLE, STRING and LIST. A LIST metric will also provide a count of the values in the list for the purposes of the warning/danger levels

You should only delete metrics that you have created yourself. If all that you want to do is prevent a metric being displayed in the listings then all you have to do is to change the Active flag.

In general changes made will be reflected immediately in JHawk. To do this you will have to press the 'Save Entry changes' for each entry that you have changed then the 'Save details to Store' button when you have completed all your changes to individual entries. If the changes are not reflected immediately then you need to analyse your file set again. You can set your preferences prior to your initial analysis to save you performing the analysis twice.

Adding a new metric



Edit Metric column details

0	Y	Left	NAME	Name	Name of cl...	N/A	0.0	0.0	com.virtual...	N	Y
1	Y	Left	SUPER	Superclass	Name of s...	N/A	0.0	0.0	com.virtual...	N	N
2	Y	Right	NOMT	No. Methods	Total num...	>	0.0	0.0	com.virtual...	N	N
3	Y	Right	LCOM	LCOM	Lack of Co...	>	0.0	0.0	com.virtual...	N	N
4	Y	Right	TCC	TCC	Total Cyclo...	>	0.0	0.0	com.virtual...	N	N
5	Y	Right	MAXCC	MAXCC	Maximum ...	>	0.0	0.0	com.virtual...	N	N
6	Y	Right	AVCC	AVCC	Average C...	>	1.0	4.0	com.virtual...	Y	N
7	Y	Right	NOS	NOS	Total num...	>	0.0	0.0	com.virtual...	N	N

Set Values for :

☐ Active ☐ Is Default Sort Field

☐ On Dashboard Level : ☑ Package ☐ Class ☐ Method

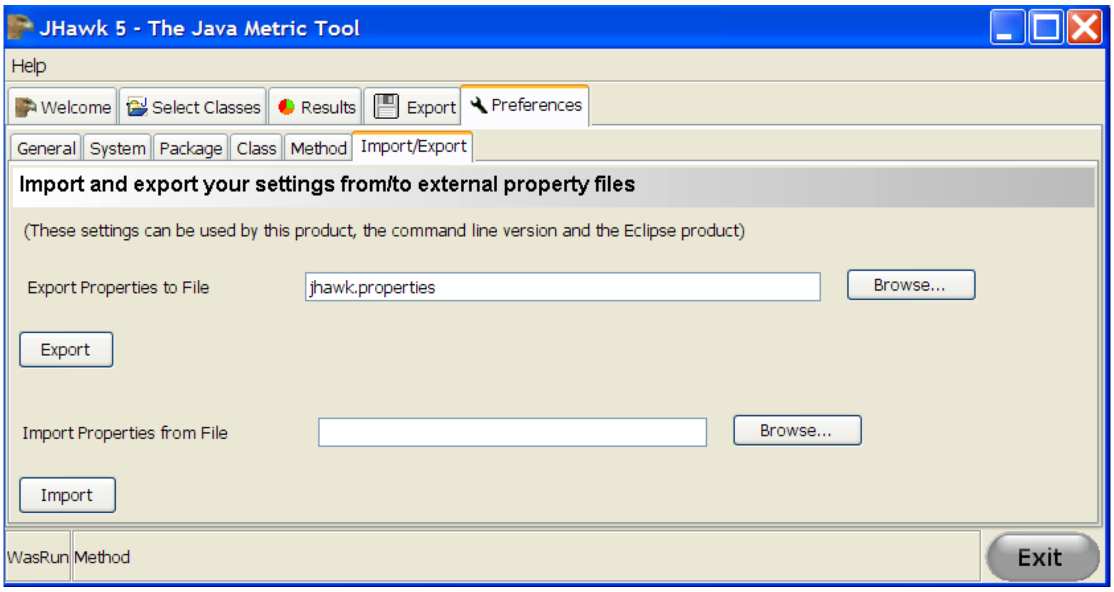
Description	My Metric
Class name	MyMetric
Key	metric.class.MINE
ColumnHeader	MINE
Align value to:	LEFT
Compare value using:	N/A
Type of Metric Value is:	INTEGER
Warning Level	0.0
Danger Level	0.0

As you can see there a number of new fields available to you when you come to add a metric. You can now set the class name and key fields. You need to be careful setting these as they are vital to the working of JHawk.

In the case of the class name you will need to provide the full name (i.e. including package name) of the class. The class will also have to be visible to JHawk (you can find out how to do this in the section on creating a new metric). Before allowing you to save a new Metric entry to the preferences store, JHawk will check that the class can be instantiated and will display an error if this is unsuccessful.

The metric key must start with "metric." and then be followed by either "system.", "package.", "class." or "method." - this value being dependant on the level at which the metric is being added (as shown in the title of the tab that the metric is being added on .The key at the end must be unique to that particular level . Before allowing you to save a new Metric entry to the preferences store, JHawk will check that the key is valid and will display an error if this is not the case.

Importing and exporting settings



Import and export your settings from/to external property files

(These settings can be used by this product, the command line version and the Eclipse product)

Export Properties to File: jhawk.properties

Import Properties from File:

WasRun Method

You may have created a set of preferences that you wish to keep for later use (for example you may want to use different preferences for different Java code sets). You do this using this screen. Having created your metrics using the preceding three tabs and stored them using the ‘Store’ button on each screen you can export them to a file. Then if you wish to re-use these metrics you can load them up from the file in one of four ways –

1. Using this tab select the file by using the ‘Browse’ button beside the ‘Import Properties from file’ text entry. Then press the ‘Import’ button – this will load up the metrics and apply them to your currently selected file set. This may take some time if you have a very large file set.
2. On the ‘Welcome’ tab select the ‘Use a previously saved set of preferences’ ‘Browse’ button and select the file that you wish to load – then press the graphic button the left hand side of the line - this will load up the metrics and apply them to any file set that you have already selected. This may take some time if you have a very large file set. If you have not yet selected a file set then the preferences will be applied to the files that you select
3. When you call JHawk using the command line interface using the –p option e.g. –p mynewproperties.properties
4. Rename the file to jhawk.properties – by default JHawk will always start up with the jhawk.properties file.

If your properties file is corrupt for any reason JHawk may not start. In this case you should use the last known good properties file. If you cannot find a non-corrupt file then you can go back to the file from the original distribution.

Starting JHawk from the Command Line

If you need to start the JHawk application from the command line then you should make sure that you are in a directory that has access to the Java JRE (or JDK). Then type the following –

```
java –jar JHawk.jar
```

This will start JHawk with the standard java startup parameters - a heap size of 64Mb and a stack size of 0.5Mb. These should be adequate for most small systems but in larger systems more memory may be required and these defaults may need to be overridden. In particular you may get stack overflow errors as the recursive nature of the parser exercises the stack. You can override these parameters by using the following –

```
java –Xss2m -Xms64m –Xmx1100m -jar JHawkStandard.jar
```

In this case we have set the stack size to 2Mb using the –Xss parameter, the minimum heap size to 64Mb using the –Xms, and the maximum heap size to 1100Mb using the –Xmx parameter. It is unlikely that you will need to increase the stack size above 2Mb. Note that in many Windows systems even if you have more than 1200Mb of RAM the heap size cannot be set greater than a figure somewhere between 1100 and 1200Mb. This restriction does not apply on most Unix systems. 1100Mb is big enough for all but the very largest systems.