



# JHawk 6.1 Documentation- Standalone User manual

Virtual Machinery  
September 2015  
Version 1.3

<b>OVERVIEW .....</b>	<b>4</b>
<b>DOCUMENTATION .....</b>	<b>5</b>
<b>INSTALLING JHAWK .....</b>	<b>6</b>
<b>JHAWK – A QUICK START .....</b>	<b>7</b>
<b>Startup JHawk.....</b>	<b>7</b>
Startup JHawk in a Windows environment .....	7
Startup JHawk in a Mac Environment .....	7
Startup JHawk in a Unix Environment .....	7
<b>The opening screen .....</b>	<b>7</b>
<b>The Select Classes tab .....</b>	<b>8</b>
<b>The Results Tab .....</b>	<b>9</b>
<b>Results Tab - Dashboard sub-tab.....</b>	<b>9</b>
<b>Results Tab - System sub-tab .....</b>	<b>9</b>
<b>Results Tab - Classes by Package sub-tab.....</b>	<b>9</b>
<b>Results Tab - Methods By Class sub-tab. ....</b>	<b>10</b>
<b>Quick start - Filters .....</b>	<b>10</b>
<b>THE SELECT FILES TAB .....</b>	<b>14</b>
<b>THE RESULTS TAB.....</b>	<b>16</b>
<b>Results Tab - Dashboard sub-tab - Snapshot.....</b>	<b>16</b>
<b>Results Tab - Dashboard sub-tab – Gauge panels .....</b>	<b>17</b>
<b>Results Tab - Dashboard sub-tab – table panel .....</b>	<b>19</b>
<b>Results Tab - System sub-tab .....</b>	<b>20</b>
<b><u>Results tab -Classes By Package sub-tab.....</u></b>	<b>21</b>
<b>Results Tab - Methods By Class sub-tab .....</b>	<b>21</b>
<b>Results Tab - All Methods in System sub-tab .....</b>	<b>22</b>
<b>THE EXPORT TAB.....</b>	<b>24</b>
<b>Filtering Exported Data .....</b>	<b>24</b>
<b>The Create CSV File tab .....</b>	<b>24</b>

The Create HTML Tab.....	25
The Create XML Tab.....	26
Creating files for the Data Viewer .....	27
Creating files to load back into JHawk Stand Alone.....	27
<b>OUTPUT FORMATS .....</b>	<b>28</b>
CSV – Standard .....	28
CSV – Raw .....	28
XML – Standard.....	29
XML – JHawk Interchange Format .....	30
HTML.....	31
<b>FILTERS.....</b>	<b>32</b>
Storing a filter record group set in a file .....	33
Installing filter record groups from a file.....	33
Managing individual Filter Record Groups.....	34
Viewing the Filter Records in a Filter Record Group .....	35
Filters – Exporting Data .....	35
Filter Preferences.....	35
Standard Filter Files.....	35
<b>THE PREFERENCES TAB .....</b>	<b>36</b>
General Preferences .....	36
The Filters sub-tab .....	37
Dashboard Table sub-tab.....	39
The Dashboard table records list .....	39
The Add/edit dashboard table panel .....	39
Metrics sub-tabs – Edit Metric Column details .....	41
The Buttons .....	41
The Details .....	42
Adding a new metric .....	43
Importing and exporting settings.....	44
<b>STARTING JHAWK FROM THE COMMAND LINE.....</b>	<b>45</b>

# 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 the success of JHawk in Academic Institutions.

This document is designed to satisfy the needs of users of JHawk – 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 are already using JHawk you will find details of changes made to JHawk since the last version (JHawk 5.1) in the document JHawk6Changes.pdf.

This issue refers to JHawk Version 6.1. The only significant change since JHawk 6.0 is the addition of the Filters functionality. These changes are covered in the Filters section of this document and in the individual sections where filters are relevant

# 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	Starter	Demo
JHawk6CommandLineManual – Documentation for the Command line version of JHawk	No	Yes	No	Yes
JHawk6CreateMetric – Documentation explaining how to create new metrics that can be added to JHawk	Yes	Yes	No	No
JHawk6DataViewerManual – Documentation for the JHawk DataViewer product	No	Yes	No	Yes
JHawk6EclipseManual – Documentation for the JHawk Eclipse Plugin	Yes	Yes	No	Yes
JHawk6Licensing – Licensing details for JHawk products	Yes	Yes	Yes	Yes
JHawk6StarterManual – Documentation for the JHawk Starter edition	No	No	Yes	No
JHawk6UserManual – Documentation for the JHawk standalone application	Yes	Yes	No	Yes
JHawk6UsingMetrics – 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	Yes	No

# Installing JHawk

In the JHawk Distribution you will find the file `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 standalone 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 a jar located on the classpath (e.g. the `CustomMetrics.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’.

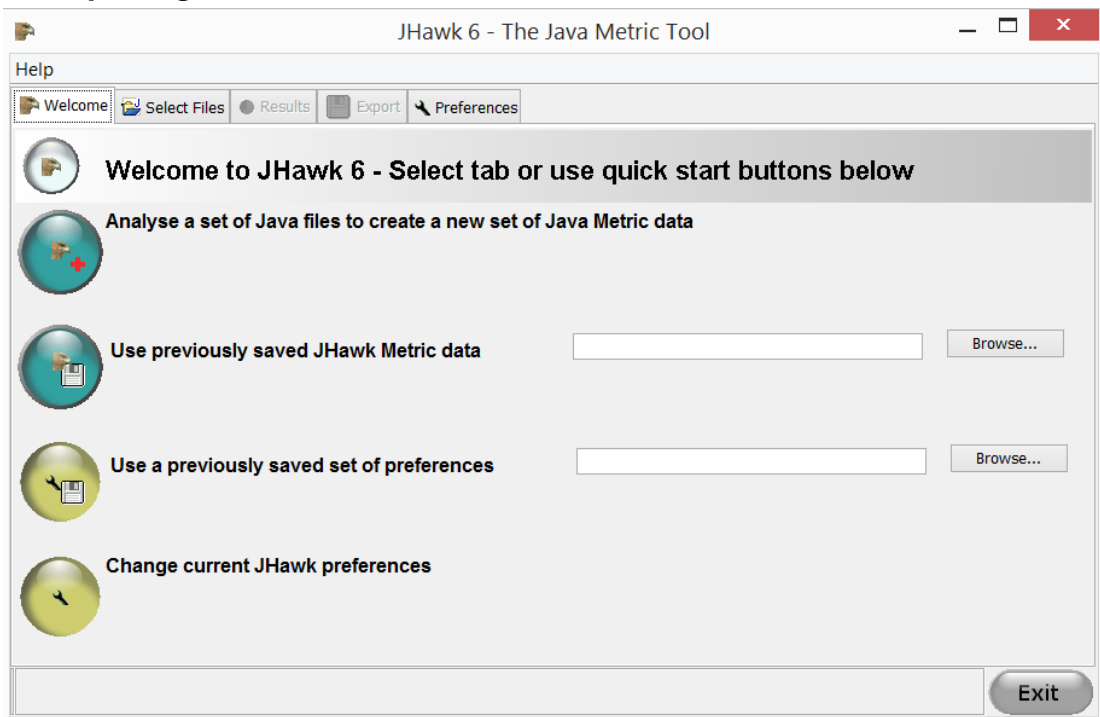
### Startup JHawk in a Mac Environment

- 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’.

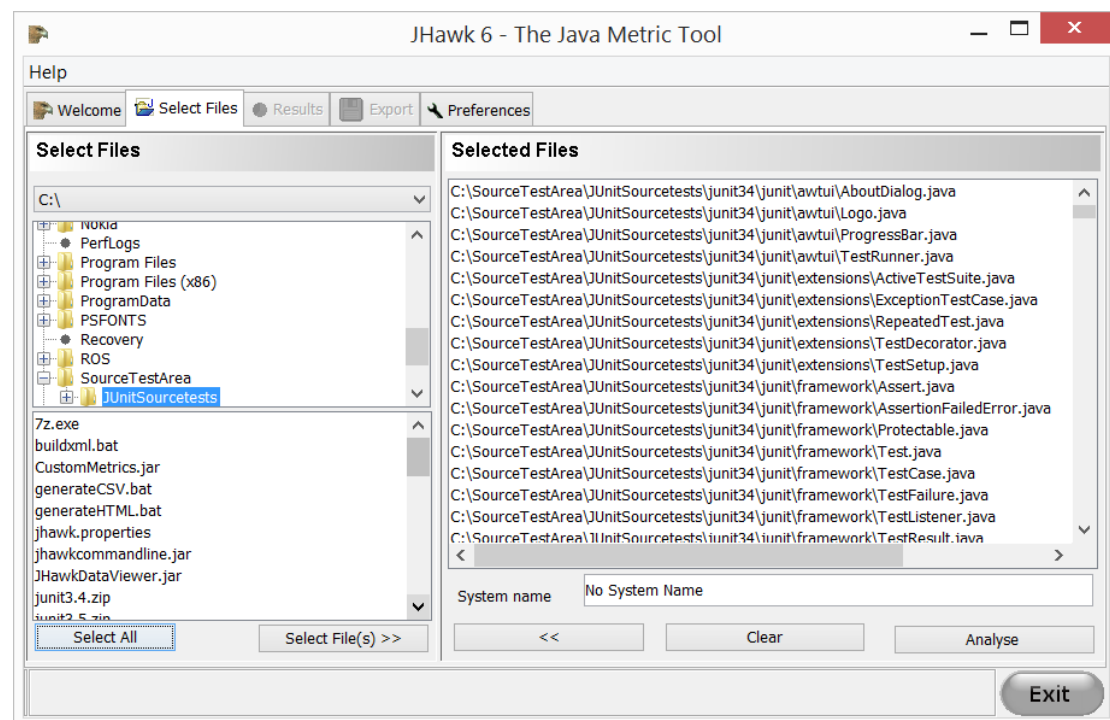
## **The opening screen**



The screen opens on the ‘Welcome’ tab. At initial startup only this tab, the ‘Select Classes’ 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



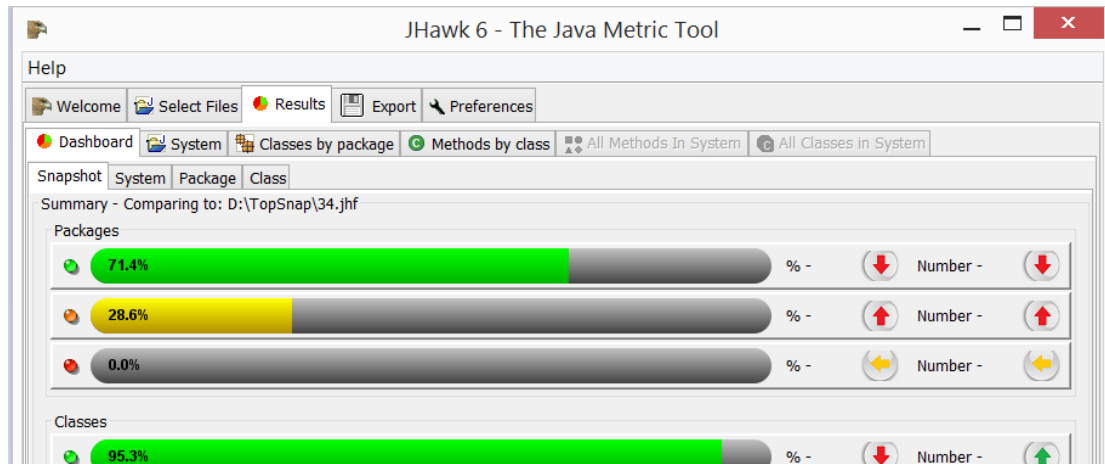
This tab allows you to select the Java files that contain the classes that you wish to analyse. In the simplest scenario you have all your files in a single source directory with the underlying source directory structure reflecting the package structure. To analyse all these files select the root directory 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.

The progress of the analysis will be indicated in the status bar at the bottom of the screen (on the same line as the 'Exit' button). When the analysis is complete the status bar will display the message 'Your results are now available' and the 'Results' and 'Export' tabs will be enabled.

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



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 tabs – firstly the Snapshot tab which allows you to compare the current state of the code with that of a previous version. The System, Package and Class sub-tabs of the ‘Dashboard’ contain two elements –

1. Pie charts that reflect the proportion of elements (packages, classes, methods) whose metrics have crossed a particular warning or danger level.
2. Ordered tables that are based on selected metrics at package, class and method level.

The default settings for the pie charts and tables 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.

### ***Quick start - Filters***

JHawk produces a large amount of data as a result of its analysis. Filters allow you to separate out and display only that data that is of interest to you. They can operate at Package, Class or Method Level and can be applied to tables in the GUI, export data (in CSV HTML and XML format) and can be applied in the user interface or at the command line.

They use the warning and danger levels for metrics that are set through the JHawk preferences screens and are contained in the JHawk properties file. If you are unfamiliar with these you could find out more about them later in the documentation – The Preferences Tab --> Metrics sub-tabs - Edit Metric column details.

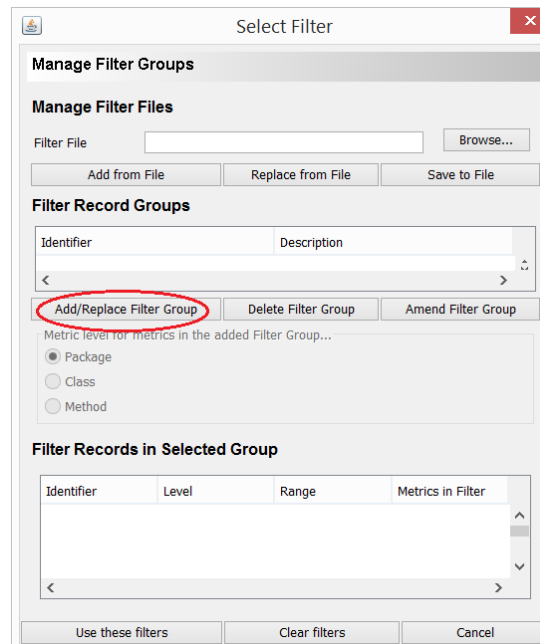
Filters can be created for temporary use ('on the fly') or they can be stored in files for subsequent use either to be loaded automatically as default filters in JHawk or for use at the command line. You can find out more in the main Filters section later in this document.

Here we'll look at applying filters to a table 'on the fly' i.e. we will create and use a filter on a table after we have created it.

Here is a table created after the analysis of a group of code. This table is on the System sub-tab of the results tab:

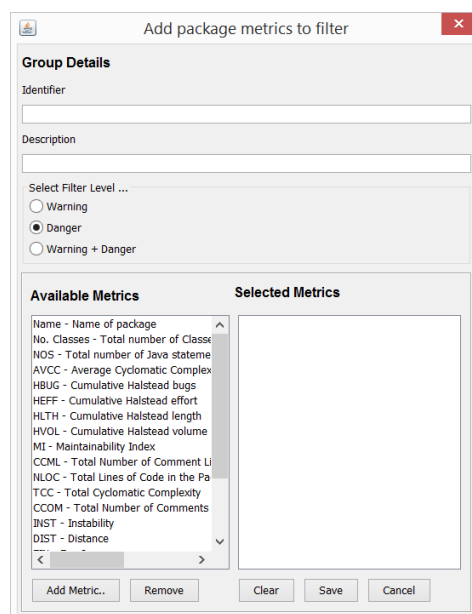
Packages											
Name	No. Classes	NOS	AVCC	HBUG	HEFF	HLTH	HVOL	MI	CCML	NLOC	
org.eclipse.jdt.core.search	7	150	1.32	2.55	68727.83	1543	7637.35	136.30	525	229	
org.eclipse.jdt.core.jdom	12	176	1.08	14.25	322736.52	8256	42758.06	101.02	771	212	
org.eclipse.jdt.core.eval	3	41	1.00	4.53	100517.62	2448	13600.51	139.58	325	53	
org.eclipse.jdt.core	41	1378	1.55	60.72	1802381.14	33682	182147.20	81.43	3945	1740	

Press the 'Filter...' button on the top right of the table. The following dialog appears:



The 'Select Filter' dialog box contains several sections: 'Manage Filter Groups' at the top, followed by 'Manage Filter Files' with a 'Filter File' input and a 'Browse...' button. Below this are 'Add from File', 'Replace from File', and 'Save to File' buttons. The 'Filter Record Groups' section has an 'Identifier' and 'Description' list. A red circle highlights the 'Add/Replace Filter Group' button. Below this is a 'Metric level for metrics in the added Filter Group...' section with radio buttons for 'Package' (selected), 'Class', and 'Method'. The 'Filter Records in Selected Group' section has a table with columns 'Identifier', 'Level', 'Range', and 'Metrics in Filter'. At the bottom are 'Use these filters', 'Clear filters', and 'Cancel' buttons.

This is the dialog that allows us to create and use filters. As you can see there are no filters currently defined for this table – so we want to add one. To do this we press the Add/Replace Filter Group button. Note that the metric selection is limited to Package metrics and the screen to select the metric level is disabled. The following dialog is displayed:



The 'Add package metrics to filter' dialog box has a 'Group Details' section with 'Identifier' and 'Description' fields. Below is a 'Select Filter Level ...' section with radio buttons for 'Warning', 'Danger' (selected), and 'Warning + Danger'. The main section is divided into 'Available Metrics' and 'Selected Metrics'. The 'Available Metrics' list includes: Name - Name of package, No. Classes - Total number of Classes, NOS - Total number of Java statements, AVCC - Average Cyclomatic Complexity, HBUG - Cumulative Halstead bugs, HEFF - Cumulative Halstead effort, HLTH - Cumulative Halstead length, HVOL - Cumulative Halstead volume, MI - Maintainability Index, CCML - Total Number of Comment Lines, NLOC - Total Lines of Code in the Package, TCC - Total Cyclomatic Complexity, CCOM - Total Number of Comments, INST - Instability, and DIST - Distance. At the bottom are 'Add Metric...', 'Remove', 'Clear', 'Save', and 'Cancel' buttons.

Using this dialog you can select the metrics to be used and the level of filter to be applied. You can select more than one metric at a time. We also have to enter a filter identifier and (optionally) a description:

In this case we have called the record group 'test', used the 'Warning and Danger' level and selected the 'MI - Maintainability Index' metric. We then press the 'Save' button to save the metrics.

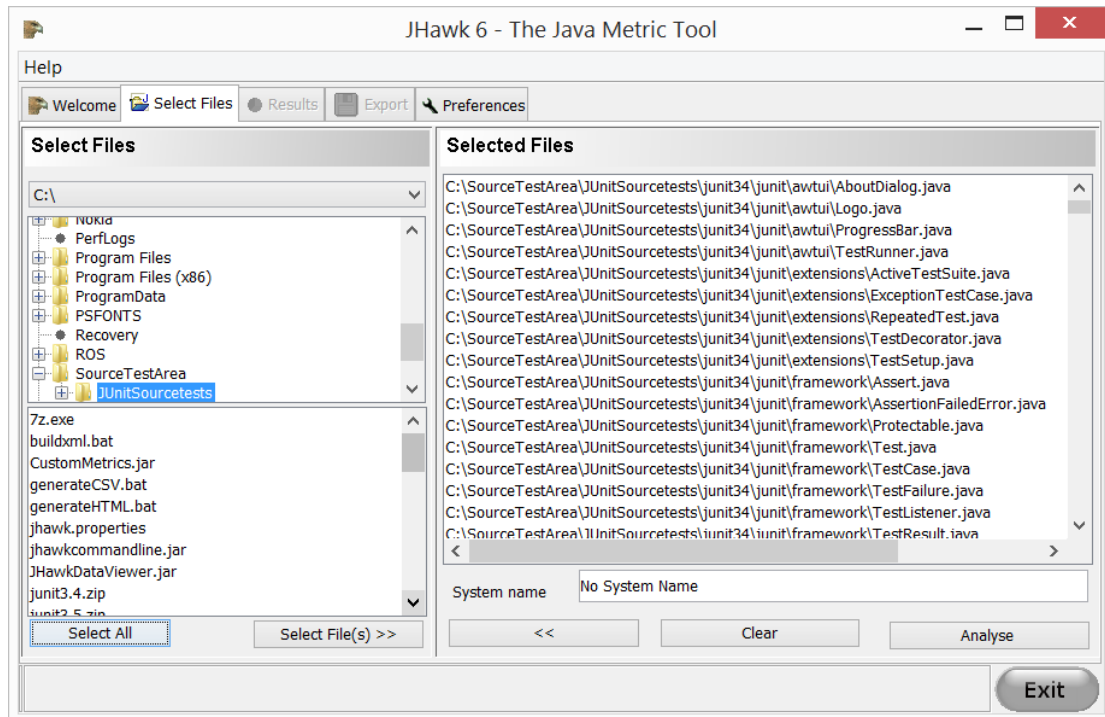
Identifier	Level	Range	Metrics in Filter
Test	Package	Warning+Danger	MI

After saving the filter we are returned to the 'Select Filter' dialog and can use the filter to filter the packages table. As you can see the only entry left in the table is the entry for the package org.eclipse.jdt.core which is the only package whose value for the Maintainability Index is in either the Warning or Danger categories.

Packages											Filter..
Name	No. Classes	NOS	AVCC	HBUG	HEFF	HLTH	HVOL	MI	CCML	NLOC	
org.eclipse.jdt.core	41	1378	1.55	60.72	1802381.14	33682	182147.20	81.43	3945	1740	

To remove the filters all you need to do is to press the 'Filters' button again and when the dialog reopens select the 'Clear Filters' button. This will close the dialog and remove the filters from the packages table causing it to be restored to its original condition.

# The Select Files Tab



The Select files 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 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.

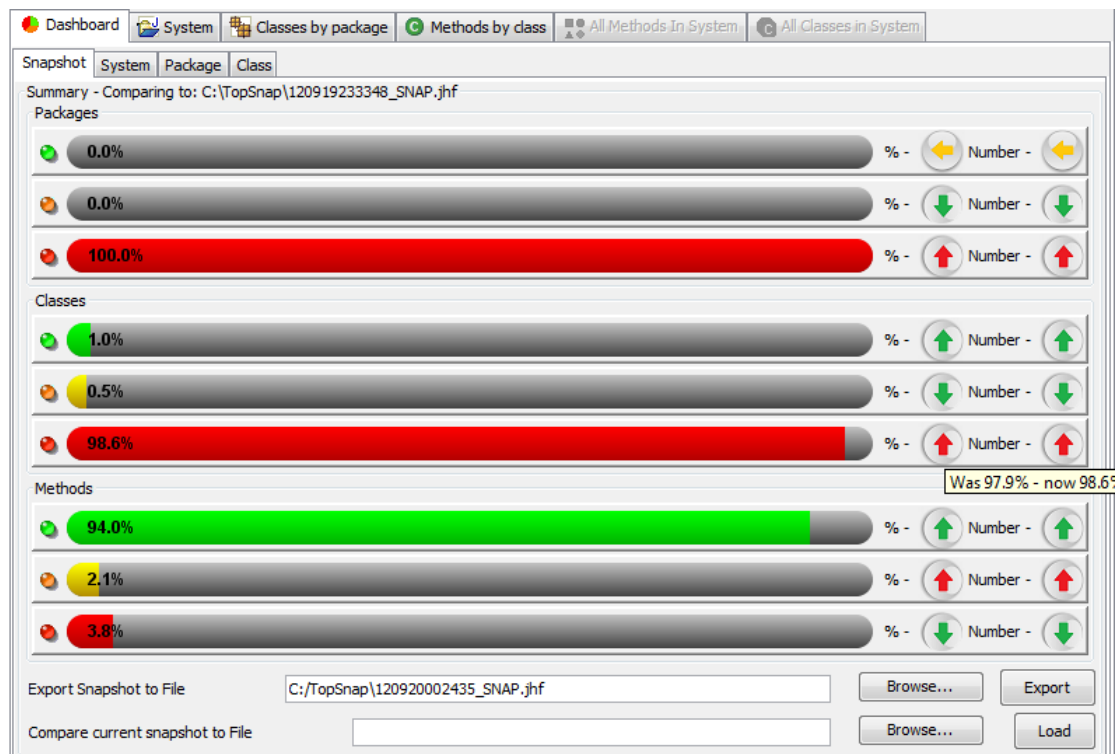
When the analyse button is re-enabled you will notice that all the tabs have been enabled and you can now navigate between them. We would suggest going to the results tab first and you can find out more about it in the next section.

Note that if you have chosen the option to use previously saved JHawk Metric data (available on the opening screen) then you will not need to select files and the files originally used to create the previously saved data will not be shown.

# 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 - Snapshot***



The Snapshot panel is the first sub tab of the Dashboard tab. It compares the results of your current analysis with those of the last snapshot that you created with the Snapshot panel. This gives you a quick overview of any changes since the last JHawk analysis that you were interested in.

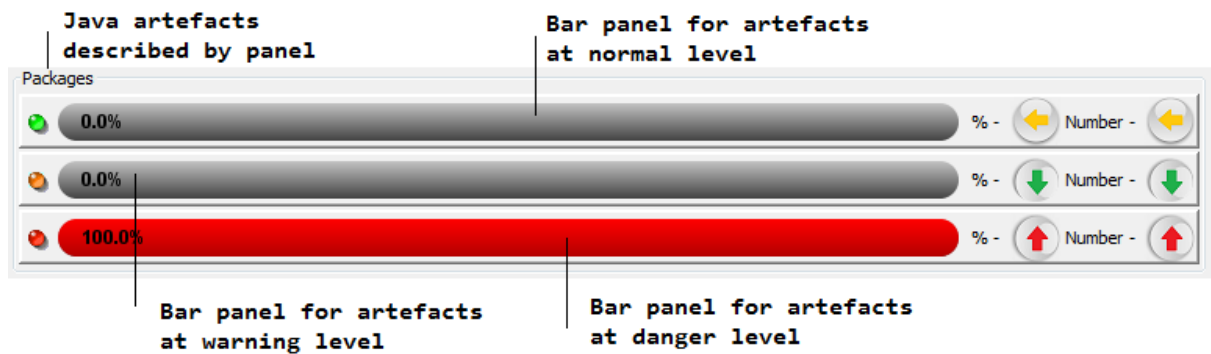
The tab is broken into five panels. Reading from the top –

1. The first panel shows the last snapshot file that the current analysis is being compared to.
2. The next panel (with three ‘thermometer’ bars) compares the current analysis at package level to the last snapshot
3. The next panel compares the current analysis at class level to the last snapshot
4. The next panel compares the current analysis at method level to the last snapshot
5. The bottom panel allows you to export the current analysis to a named file and/or to change the snapshot file that the current analysis is being compared to.

For panels 1 and 5 the default directory is a directory defined using the JHawk Preferences. The snapshot is not shown in either the Eclipse or DataViewer versions of the product. The display is too large to fit comfortably in the Eclipse environment and the Data Viewer provides more easily defined comparative views of JHawk analyses.

Panels 2 to 4 provide the same comparative information at three of the levels of artefact analysed by JHawk – Packages, Classes and methods. Each panel is composed as follows –





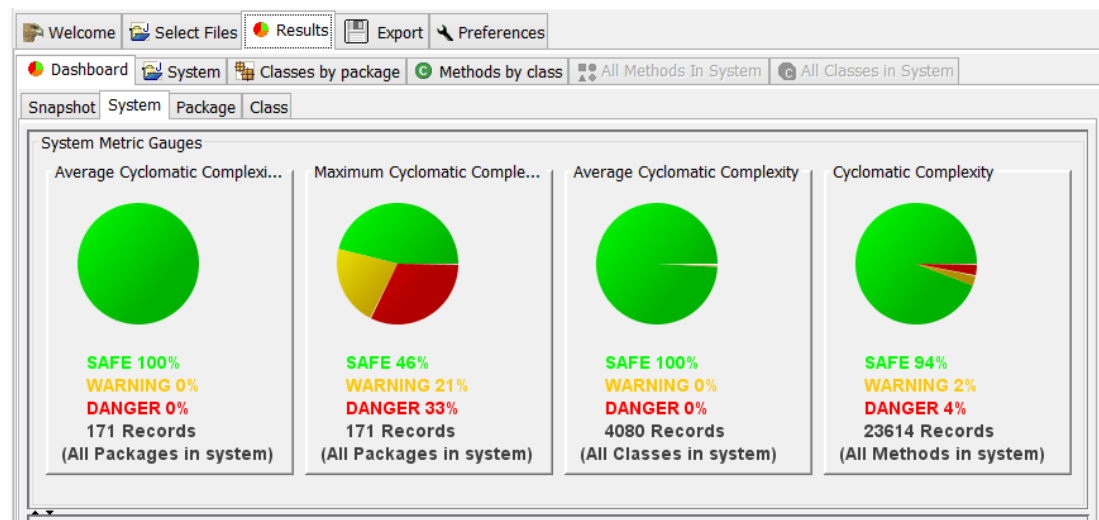
An artefact level is viewed as being at warning or danger level in any of the active metrics for any of the artefacts have crossed the warning or danger levels defined for them. In the case above 100% of the packages analysed have at least one active metric at danger level.

There is also data associated with each individual bar panel –



## Results Tab - Dashboard sub-tab – Gauge panels

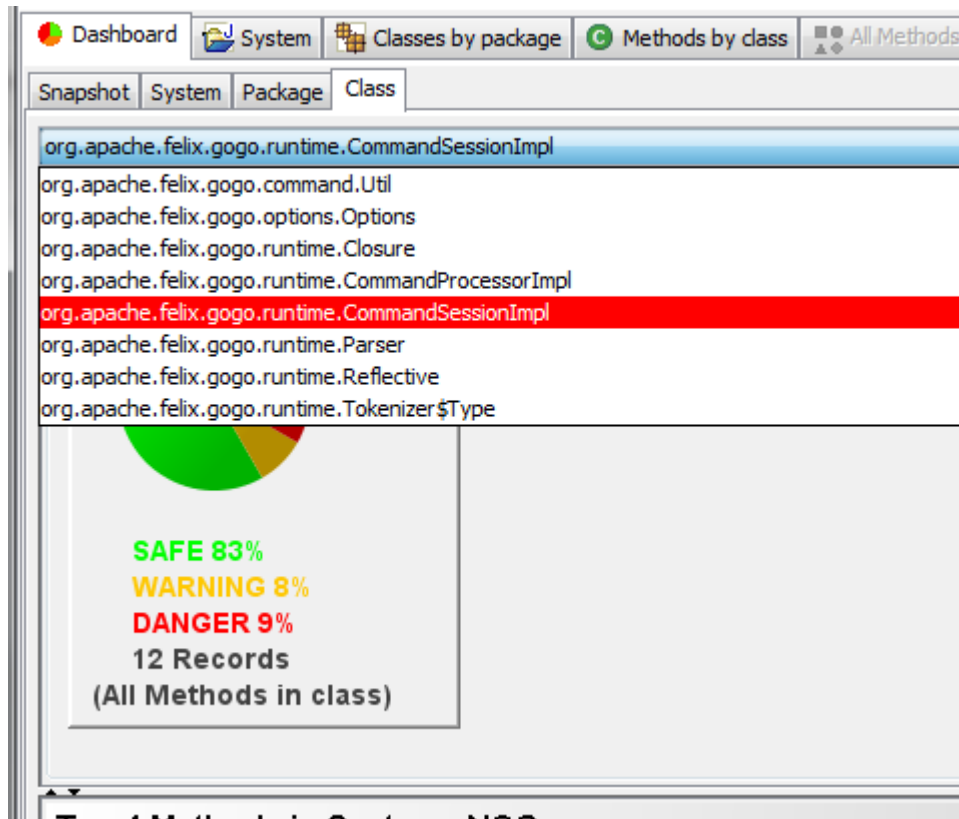
The dashboard sub-tab allows users to make a quick visual check on the current quality of their code. Although it comes with some predefined metrics the dashboard tab is completely configurable using the Preferences tab (see 'Preferences Tab – Configuring the Dashboard' for more details).



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 - Dashboard sub-tab – table panel**

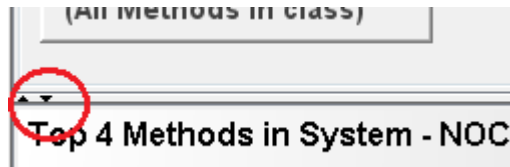
Each dashboard tab can have dashboard tables which allow you to select specific metrics to be analysed at a particular level (Package, Class, Method) and displayed on the lower part of the dashboard panel – below the dashboard gauges. For each of these tables you can select the metrics displayed, a metric to base the selection and the table ordering on and the number of rows to be displayed. You can have as many of these tables as you wish and they will be displayed sequentially in the panel. Package related tables will be displayed on the System tab, Class related tables on the Package tab and Method related tables on the Class tab.

12 Records (All Methods in class)				
Top 4 Methods in System - NOC				
Name of method	Number of arguments	Number of comments	Number of Variable references	
foldToASCII	2	1245	887	
CheckMethodAdapter	1	212	1	
setSignature	1	205	2	
configure	1	163	870	

In the screenshot shown above the 4 methods in the system with the greatest number of Comments (NOC) are shown.

These tables are the same as the normal tables available at each artefact level – they can be re-ordered by double-clicking on the header of a column and double clicking on entry will bring up more detail on that entry.

You can choose to display only one of the panels by using the One-Touch icon at the left hand side of the dividing bar between the panels –



You can configure the dashboard tables by using the Dashboard tables sub tab of the Preferences Tab – see the later section – ‘The Preferences Tab’ for more details on this.

## Results Tab - System sub-tab

**Name of system** No System Name

Total number of Packages	34	Total number of Java statements	32507
Average Cyclomatic Complexity of the meth...	1.79	Cumulative Halstead bugs	339.74
Cumulative Halstead effort	10458373.52	Maintainability Index	163.21
Total Number of Comment Lines in the Syst...	6943	Total Lines of Code in the System	37237
Total number of Classes	802	Total Number of Comments in the system	842
Cumulative Halstead length	204946	Total number of methods	4233
Total Cyclomatic Complexity	7573	Maintainability Index (Not including commen...	113.21
Cumulative Halstead volume	1019215.44		

**Packages** Filter..

Name	No. Classes	NOS	AVCC	HBUG	HEFF	HLTH	HVOL	MI	CCML	NLOC
org.eclipse.pde.internal.component	28	1178	2.01	11.29	308254.20	6596	33864.60	104.59	86	2652
org.eclipse.pde.internal	11	866	2.20	9.06	287938.81	5097	27186.39	157.62	150	2002
org.eclipse.pde.internal.schema	16	929	1.76	7.86	267213.89	5087	23577.98	116.24	18	2250
org.eclipse.pde.internal.editor.jars	28	746	1.55	6.28	141448.28	3838	18832.75	112.15	66	1694
org.eclipse.pde.internal.base	6	135	1.44	1.30	32651.18	814	3902.59	76.17	226	260
org.eclipse.pde.internal.model.build	7	342	1.56	2.27	53775.26	1537	6806.34	118.13	2	768
org.eclipse.pde.internal.launcher	4	304	2.97	3.58	144289.76	2004	10742.62	104.34	0	770
org.eclipse.pde.internal.wizards	18	736	1.90	6.73	214154.62	4032	20182.40	160.30	86	1684
org.eclipse.pde.internal.wizards.extensi...	18	862	2.06	7.51	231960.46	4366	22524.82	102.81	24	1842

Analysis complete Your results are now available Exit

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** org.eclipse.pde.internal.launcher

**Classes in package** WorkbenchLauncher

**Name of package** org.eclipse.pde.internal.launcher

Total number of Classes	4	Total number of Java statements	304
Average Cyclomatic Complexity for	2.97	Cumulative Halstead bugs	3.58
Cumulative Halstead effort	144289.76	Cumulative Halstead length	2004
Cumulative Halstead volume	10742.62	Maintainability Index	104.34
Total Number of Comment Lines in th...	0	Total Lines of Code in the Package	770
Total Cyclomatic Complexity	86	Total Number of Comments in the pac...	3
Instability	0.75	Distance	0.25
Fan In	1	Total number of methods	29
Maintainability Index (Not including co...	104.34	Abstractness	0.00
Maximum Cyclomatic Complexity for ...	9	Fan Out	3

Close

You can filter the contents of the table by using the ‘Filters...’ button just above the ‘Packages’ table on the right hand side– see the filters section later in this document or the ‘Quick Start – Filters’ section above.

## **Results tab -Classes By Package sub-tab**

Name	No. Methods	LCOM	AVCC	NOS	HBUG	HEFF	UWCS	INST	PACK	RFC	CBO	MI	CCML	NLOC	CUST
ArraySorter	1	1.00	1.00	4	0.04	666.61	2	1	0	2	4	126.54	4	6	999
BaseProject	8	1.00	2.00	57	0.64	29530.04	9	1	3	8	3	111.23	5	73	999
CoreUtility	3	0.00	2.00	22	0.34	10254.12	3	0	8	3	6	111.49	0	40	999
ExternalModelManager	20	0.06	2.70	156	1.88	59582.02	25	5	22	21	18	111.89	4	201	999
OverlayIcon	8	0.09	3.38	74	0.83	35207.60	13	5	4	8	2	106.43	3	90	999
PDEPerspective	2	0.00	1.00	15	0.24	4610.78	2	0	3	2	0	163.00	2	36	999
PDEPlugin	31	0.03	1.97	169	1.86	51832.18	54	23	19	31	195	119.39	10	219	999
PDEPluginImages	7	0.01	1.86	166	2.82	68868.47	123	116	8	12	59	88.85	28	194	999

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.

You can filter the contents of the table by using the ‘Filters...’ button just above the ‘Classes’ table on the right hand side– see the filters section later in this document or the ‘Quick Start – Filters’ section above.

## **Results Tab - Methods By Class sub-tab**

Name	COMP	NOCL	NOS	HLTH	HVOC	HEFF	HBUG	CREP	XMET	LMET	NLOC
ExternalModelManager() (org.eclipse.pde.intern...	1	0	1	4	4	12.00	0.00	0	0	0	2
addModelProviderListener(org.eclipse.pde.intern...	1	0	2	13	11	131.17	0.01	1	1	0	3
clear() (org.eclipse.pde.internal.ExternalModelM...	1	0	2	9	9	71.32	0.01	0	0	0	3
createEclipseRelativeHome(java.lang.String) (or...	1	0	5	37	20	1012.77	0.05	3	3	1	6
findExtensionPoint(java.lang.String) (org.eclipse...	6	1	16	124	40	12758.44	0.22	3	6	1	15
findPlugin(java.lang.String) (org.eclipse.pde.inte...	5	0	11	75	30	4048.18	0.12	3	5	1	13
fireModelProviderEvent(org.eclipse.pde.intern...	2	0	7	39	20	816.84	0.06	3	4	0	6
getEclipseHome() (org.eclipse.pde.internal.Exter...	4	0	11	92	37	5451.69	0.16	8	8	0	17
getModels() (org.eclipse.pde.internal.ExternalM...	3	0	7	47	21	2477.27	0.07	1	3	1	9

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.

You can filter the contents of the table by using the ‘Filters...’ button just above the ‘Methods’ table on the right hand side– see the filters section later in this document or the ‘Quick Start – Filters’ section above.

## Results Tab - All Methods in System sub-tab

Name	COMP	NOCL	NOS	HLTH	HVOC	HEFF	HBUG	CREF	XMET	LMET	NLOC
AbstractComponentModel() (org.eclipse.pde.internal....	1	0	2	7	6	36.19	0.01	0	0	0	3
AbstractGroupMarker() (org.eclipse.jface.action.Abstr...	1	3	1	4	4	12.00	0.00	0	0	0	2
AbstractGroupMarker(java.lang.String) (org.eclipse.jfa...	1	7	3	25	17	545.00	0.03	2	2	0	4
AbstractModel() (org.eclipse.pde.internal.model.Abstr...	1	0	2	7	6	36.19	0.01	0	0	0	3
AbstractPluginModelBase() (org.eclipse.pde.internal.m...	1	0	2	7	6	36.19	0.01	0	0	0	3
AbstractSchemaDescriptor() (org.eclipse.pde.internal....	1	0	2	7	6	36.19	0.01	0	0	0	3
Action() (org.eclipse.jface.action.Action)	1	6	1	4	4	12.00	0.00	0	0	0	2
Action(java.lang.String) (org.eclipse.jface.action.Action)	1	8	3	13	9	98.90	0.01	1	0	2	4
Action(java.lang.String, org.eclipse.jface.resource.Ima...	1	9	4	19	12	170.29	0.02	2	0	3	5
ActionContributionItem(org.eclipse.jface.action.Action...	1	6	3	19	12	306.51	0.02	1	1	0	4
AlertSection(org.eclipse.pde.internal.editor.manifest.M...	1	0	7	51	26	1015.67	0.08	4	6	1	8
AttributeClassCodeGenerator(IJavaProject, org.eclipse...	1	0	5	39	21	942.15	0.06	5	2	0	10
AttributePropertySource(org.eclipse.pde.internal.base....	1	0	2	22	15	236.37	0.03	1	0	0	3
BaseExtensionPointMainPage(org.eclipse.core.resourc...	1	0	3	16	12	229.44	0.02	1	0	0	4
BaseProject() (org.eclipse.pde.internal.BaseProject)	1	0	2	7	6	36.19	0.01	0	0	0	3
BaseWizardSelectionPage(java.lang.String, java.lang....	1	0	3	18	12	322.65	0.02	1	0	0	4
Build() (org.eclipse.pde.internal.model.build.Build)	1	0	1	4	4	12.00	0.00	0	0	0	2
BuildComponentAction() (org.eclipse.pde.internal.edito...	1	0	3	18	14	239.86	0.02	2	2	1	4
BuildConsoleViewer(org.eclipse.swt.widgets.Composit...	1	1	6	57	28	1479.70	0.09	5	6	2	7
BuildEditorContributor() (org.eclipse.pde.internal.edito...	1	0	2	8	7	44.92	0.01	0	0	0	3

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 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.

You can filter the contents of the table by using the ‘Filters...’ button just above the ‘All Methods in System’ table on the right hand side– see the filters section later in this document or the ‘Quick Start – Filters’ section above.

## Results Tab - All Classes In System sub-tab

Welcome

Select Files

Results

Export

Preferences

Dashboard

System

Classes by package

Methods by class

All Methods In System

All Classes in System

All classes in system

Filter..

Name	No. Methods	LCOM	AVCC	NOS	HBUG	HEFF	UWCS	INST	PACK	RFC	CBO	MI	CCML	NLOC	CUST
PDEMultiPageEditor	54	0.01	1.80	247	2.86	76585.97	71	17	26	58	32	123.59	6	325 999	
IPDEEditorPage	7	0.00	1.00	8	0.49	5241.99	7	0	6	7	25	148.36	0	9 999	
PDEMultiPagePropertySheet	15	0.08	1.80	74	0.64	14599.32	21	6	10	16	3	124.87	2	100 999	
PDEEditorContributor\$GlobalAction	2	1.00	1.00	6	0.03	300.21	3	1	9	2	1	136.06	0	9 999	
SystemFileDocumentProvider	9	0.38	1.56	43	0.45	14154.33	11	2	8	9	4	123.27	0	59 999	
PDEMultiPageEditor\$WorkspaceModi...	1	0.00	1.00	3	0.03	427.75	1	0	26	1	0	132.48	0	8 999	
PDEChildFormPage	10	0.33	1.00	23	0.14	1517.26	11	1	4	10	4	142.60	0	33 999	
XMLConfiguration	7	0.00	1.43	36	0.38	9309.46	12	5	5	7	12	120.25	4	60 999	
PDEEditorContributor\$SaveAction	3	0.00	1.67	8	0.06	805.06	3	0	9	3	2	140.53	0	14 999	
PDEEditorContributor	14	0.05	1.14	61	0.68	24483.85	20	6	9	15	9	122.98	1	95 999	
ModifiedTextCellEditor\$Listener	1	0.00	1.00	4	0.02	203.00	1	0	4	1	0	150.99	3	6 999	
SubActionBars	17	0.08	2.00	73	0.62	12082.54	26	9	10	18	8	125.93	2	107 999	
PDEMultiPageXMLEditor\$UTF8FileDo...	3	0.00	4.67	47	0.74	33078.01	3	0	12	3	0	94.94	1	72 999	
PDESourcePage\$DocumentListener	2	0.00	1.50	5	0.03	215.10	2	0	20	2	0	143.68	0	9 999	
ModifiedTextCellEditor	3	0.00	1.00	10	0.10	1389.82	3	0	4	3	7	132.21	0	13 999	
PropertiesAction	2	0.00	1.50	17	0.15	2127.41	4	2	5	2	8	113.38	0	22 999	
PDEMultiPageEditor\$IFormSelection...	1	0.00	3.00	7	0.07	1775.20	1	0	26	1	3	113.64	0	11 999	
PDEMultiPageEditor\$IModelChanged...	1	0.00	2.00	3	0.03	342.66	1	0	26	1	1	133.08	0	6 999	
PDEMultiPageEditor\$IMenuListener	1	0.00	1.00	3	0.01	83.69	1	0	26	1	1	138.06	0	5 999	
FormOutlinePane\$BasicLabelProvider	2	0.00	1.50	8	0.05	793.90	2	0	14	2	1	127.98	0	12 999	

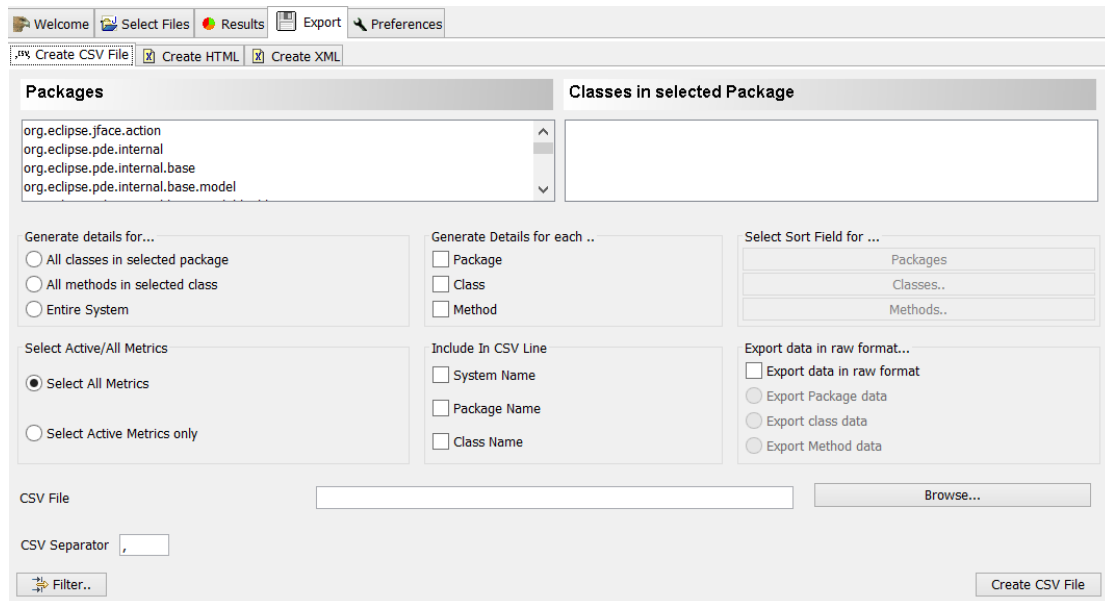
Package name

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.

You can filter the contents of the table by using the ‘Filters...’ button just above the ‘All Classes in System’ table on the right hand side– see the filters section later in this document or the ‘Quick Start – Filters’ section above.

# The Export Tab



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).

## Filtering Exported Data

You can apply filters when you are exporting data. This means that only the data that you select with the filter will be exported. The filters can be activated using the 'Filter...' button on each of the export tabs – CSV, HTML and XML. The 'Filters...' button opens a Filter Dialog that is identical to those opened on the tables.

It is important to match the filter(s) applied to the level of detail that is being generated. For example if you are exporting the Package, Method and Class details for a system and you apply a Package and a Class filter, the Packages that pass the Package Filter will be exported but only the classes in those packages that pass the Package filter will be passed on to the class filter. For more detail on this see the main section on filters in this document.

## 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



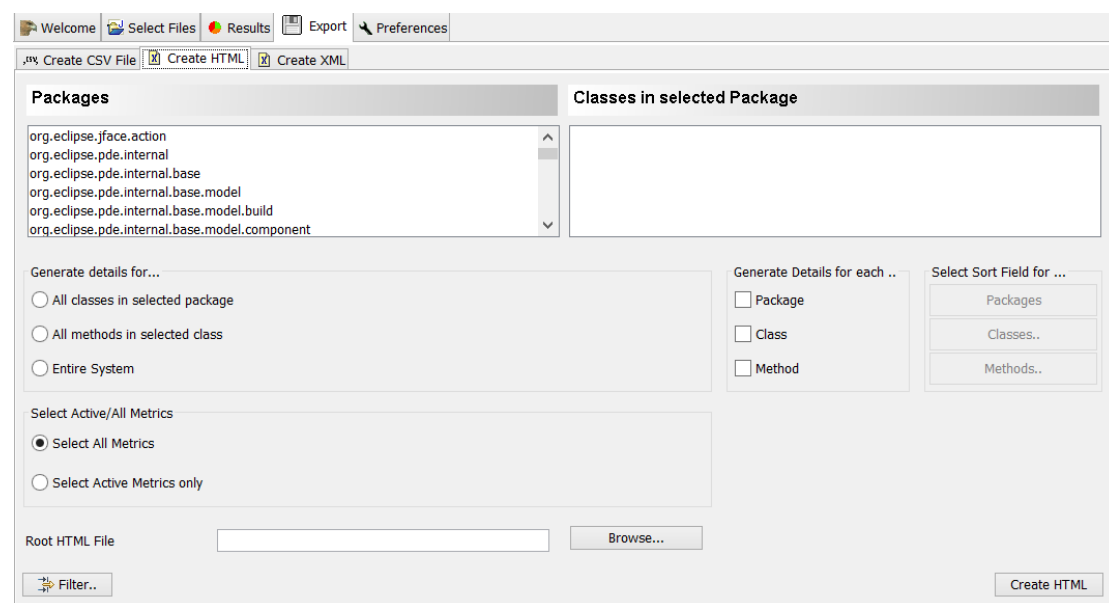
the button opens a dialog that allows you to select the field and the order ('Ascending' or 'Descending' that the data is to be selected on.

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



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 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.

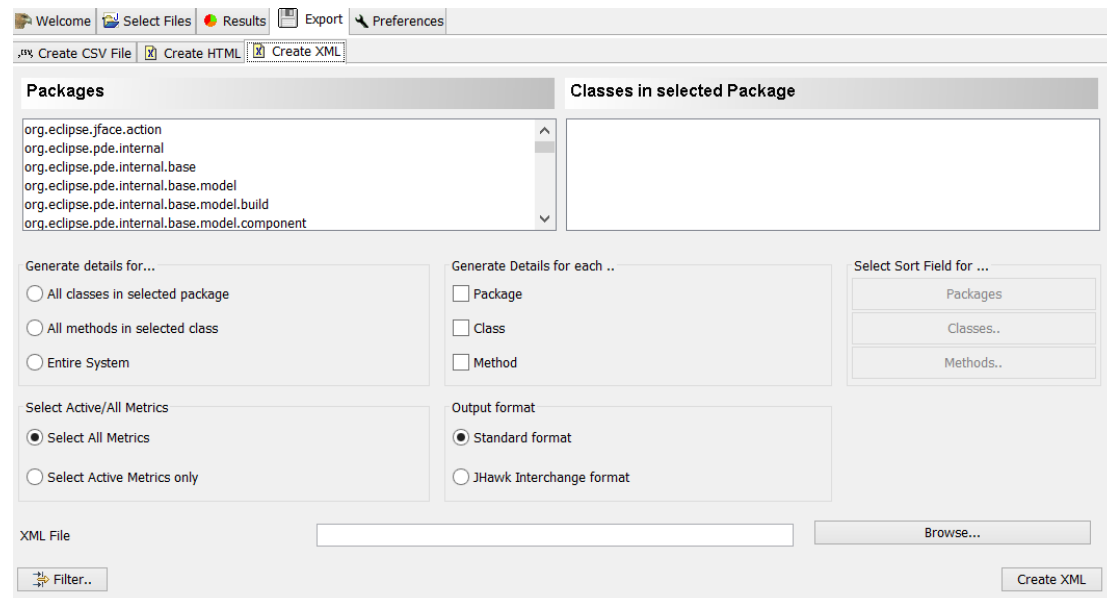
Next we come to a box entitled 'Generate Details for Each...' and selections for Package, Class and Method – as each of these is selected the 'Select Sort Field..' button for each level will be enabled. This allows you to select the field used to sort the data on when it is exported to the HTML file. 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 Sort Field for ...

Packages

Classes..

Methods..

Select Active/All Metrics

☒ Select All Metrics

☐ Select Active Metrics only

Output format

☐ Standard format

☒ JHawk Interchange format

XML File

C:\MyXML

Browse...

Filter..

Create XML

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 232Mb 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 you may have to analyse right down to the method level. JHawk keeps summaries of lower level data but this is sometimes not sufficient to cover all metrics.

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.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Details of classes for Package junit.framework														
2	System	Package	Name	No. Methods	LCOM	AVCC	NOS	HBUG	HEFF	UWCS	INST	PACK	RFC	CBO	MI
3	No System Name	junit.framework	Assert	39	0	1.36	92	1.15	24438.84	39	0	0	39	4	158.29
4	No System Name	junit.framework	AssertionFailedError	2	1	1	5	0.03	344	3	1	0	2	8	192.17
5	No System Name	junit.framework	ComparisonCompactor	8	0.3	2.12	49	0.62	19152.76	16	8	0	8	2	115.19
6	No System Name	junit.framework	ComparisonFailure	4	0.25	1	15	0.12	1631.71	8	4	0	4	3	100.42
7	No System Name	junit.framework	JUnit4TestAdapter	12	0	1.42	37	0.36	6863.56	15	3	11	12	12	131.41
8	No System Name	junit.framework	JUnit4TestAdapterCache	5	0.5	2.2	27	0.32	8983.28	7	2	8	6	10	119.12
9	No System Name	junit.framework	JUnit4TestAdapterCache\$Rur	3	0	1	7	0.09	1197.97	3	0	8	3	2	139.29
10	No System Name	junit.framework	JUnit4TestCaseFacade	5	0.75	1	12	0.07	762.86	6	1	2	5	5	142.58
11	No System Name	junit.framework	Protectable	1	0	1	2	0.01	72	1	0	0	1	2	97.04
12	No System Name	junit.framework	Test	2	0	1	3	0.02	91.02	2	0	0	2	19	140.07
13	No System Name	junit.framework	TestCase	13	0.67	1.62	48	0.42	11605.44	14	1	3	13	8	79.86
14	No System Name	junit.framework	TestFailure	7	0.08	1	24	0.19	2870.95	9	2	2	7	3	172.51
15	No System Name	junit.framework	TestListener	4	0	1	5	0.03	156.54	4	0	0	4	7	179.07
16	No System Name	junit.framework	TestResult	18	0	1.39	68	0.59	10410.7	23	5	4	18	17	151.99
17	No System Name	junit.framework	TestResult\$Protectable	1	0	1	3	0.02	136.54	1	0	4	1	0	136.39
18	No System Name	junit.framework	TestSuite	24	0.35	2.29	102	1.36	56645.36	26	2	10	25	9	122.03
19	No System Name	junit.framework	TestSuite\$TestCase	1	0	1	3	0.02	137	1	0	10	1	0	136.38
20	Package overview of package junit.framework														
21	System	Name	No. Classes	NOS	AVCC	HBUG	HEFF	HLTH	HVOL	MI	CCML	NLOC	TCC	CCOM	INST
22	No System Name	junit.framework	17	557	1.54	6.16	167133	4266	18474.07	149.7	523	792	229	125	0.33
23	Details of classes for Package org.junit.experimental.theories.internal														
24	System	Package	Name	No. Methods	LCOM	AVCC	NOS	HBUG	HEFF	UWCS	INST	PACK	RFC	CBO	MI
25	No System Name	org.junit.experimen	AllMembersSupplier	7	1	2.86	40	0.6	18084.58	8	1	12	7	6	167.81

## 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.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	System	Package	Class	Name	COMP	NOCL	NOS	HLTH	HVOC	HEFF	HBUG	CREF	XMET	LMET	NLOC	NOC	NOA	MOD
2	JUnit	junit.extensions	ActiveTestSuite	ActiveTestSuite	1	0	1	4	4	12	0	0	0	0	2	0	0	1
3	JUnit	junit.extensions	ActiveTestSuite	ActiveTestSuite	2	0	2	15	13	266.43	0.02	1	0	0	3	0	1	1
4	JUnit	junit.extensions	ActiveTestSuite	ActiveTestSuite	2	0	2	18	15	361.67	0.02	2	0	0	4	0	2	1
5	JUnit	junit.extensions	ActiveTestSuite	ActiveTestSuite	1	0	2	10	8	75	0.01	1	0	0	3	0	1	1
6	JUnit	org.junit.internal.builders	AllDefaultPossib	AllDefaultPossibilitiesBu	1	0	2	10	9	99.06	0.01	0	0	0	4	0	1	1
7	JUnit	org.junit.experimental.theories.i	AllMembersSupr	AllMembersSupplier	1	3	2	10	9	99.06	0.01	1	0	0	4	1	1	1
8	JUnit	org.junit.runners	AllTests	AllTests	2	3	2	15	13	266.43	0.02	2	0	0	3	1	1	1
9	JUnit	org.junit.internal.builders	AnnotatedBuilde	AnnotatedBuilder	1	0	2	10	9	99.06	0.01	1	0	0	4	0	1	1
10	JUnit	org.junit.internal	ArrayComparator	ArrayComparisonFailure	1	7	4	22	15	279.34	0.03	2	0	1	5	1	3	1
11	JUnit	junit.framework	Assert	Assert	1	3	1	4	4	12	0	0	0	0	2	1	0	1
12	JUnit	org.junit	Assert	Assert	1	3	1	4	4	12	0	0	0	0	2	1	0	1
13	JUnit	junit.framework	AssertionFailedE	AssertionFailedError	1	0	1	4	4	12	0	0	0	0	2	0	0	1
14	JUnit	junit.framework	AssertionFailedE	AssertionFailedError	1	0	2	10	8	75	0.01	1	0	0	3	0	1	1
15	JUnit	org.junit.experimental.theories.i	Assignments	Assignments	1	0	4	28	18	557.25	0.04	2	0	0	6	0	3	1
16	JUnit	org.junit.internal	AssumptionViol	AssumptionViolatedExce	3	0	4	31	20	1004.85	0.04	2	0	0	5	0	2	1
17	JUnit	org.junit.internal	AssumptionViol	AssumptionViolatedExce	1	0	2	11	9	83.69	0.01	1	0	0	4	0	1	1
18	JUnit	org.junit.runners	BlockJUnit4Class	BlockJUnit4ClassRunner	2	6	2	15	13	266.43	0.02	2	0	0	4	1	1	1

## 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>1428342139825</Time>
  <Locale>en_IE</Locale>
  <LOC CO="0" CI="90868" BL="41076" TL="385023"/>
- <Packages>
  - <Package>
    <OwningSystem>No System Name</OwningSystem>
    <Name>org.eclipse.pde.internal.core.text.plugin</Name>
    - <Metrics>
      <name>org.eclipse.pde.internal.core.text.plugin</name>
      <numberOfClasses>18</numberOfClasses>
      <numberOfMethods>251</numberOfMethods>
      <numberOfStatements>1524</numberOfStatements>
      <tcc>494</tcc>
      <avcc>1.9681274900398407</avcc>
      <halsteadCumulativeBugs>16.450559354962618</halsteadCumulativeBugs>
      <halsteadEffort>703825.7273457007</halsteadEffort>
      <halsteadCumulativeLength>10223</halsteadCumulativeLength>
      <halsteadCumulativeVolume>49351.67806488785</halsteadCumulativeVolume>
      <maintainabilityIndex>135.28195483445472</maintainabilityIndex>
      <maintainabilityIndexNC>114.66695833864173</maintainabilityIndexNC>
      <abstractness>0.0</abstractness>
      <fanin>19</fanin>
      <fanout>10</fanout>
      <instability>0.3448275862068966</instability>
      <distance>0.6551724137931034</distance>
      <maxcc>11</maxcc>
      <cumulativeNumberOfComments>411</cumulativeNumberOfComments>
      <cumulativeNumberOfCommentLines>872</cumulativeNumberOfCommentLines>
      <loc>1960</loc>
    </Metrics>
  </Package>
- <Package>
  <OwningSystem>No System Name</OwningSystem>
  <Name>org.eclipse.pde.api.tools.ui.internal.preferences</Name>
  - <Metrics>
```

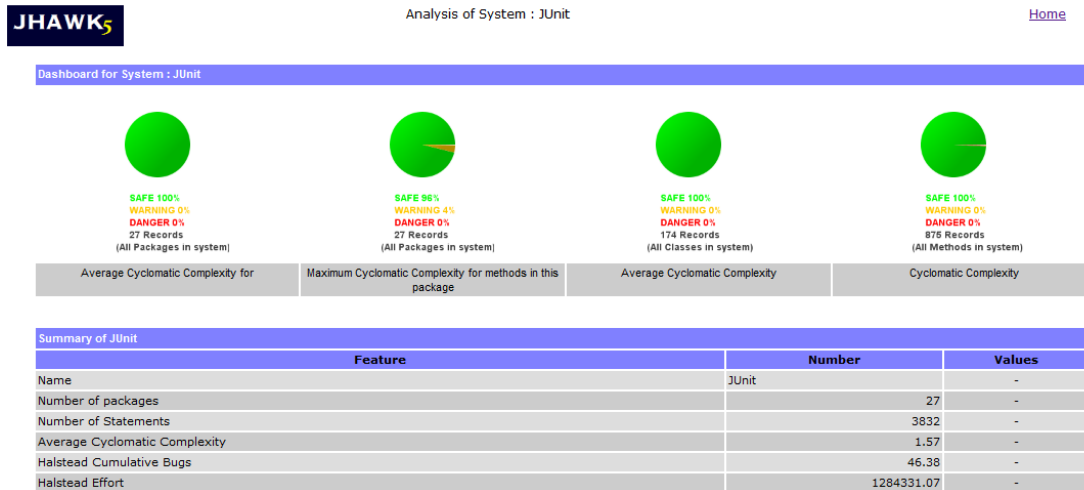
## XML – JHawk Interchange Format

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>1428342367044</TIM>
  <LLE>en_IE</LLE>
  <LOC CO="0" CI="90868" BL="41076" TL="385023"/>
- <CUS>
  - <CU>
    <NAM>D:\EclipseTest\Eclipse421\src\org\ eclipse\pde\internal\ui\nls\GetNonExternalizedStringsOperation.java</NAM>
    <OWS>No System Name</OWS>
    <LOC CO="0" CI="23" BL="28" TL="229"/>
  </CU>
  - <CU>
    <NAM>D:\EclipseTest\Eclipse421\src\org\ eclipse\pde\internal\ui\parts\ILinkLabelProvider.java</NAM>
    <OWS>No System Name</OWS>
    <LOC CO="0" CI="10" BL="3" TL="19"/>
  </CU>
  <!-- <MORE_COMPILATION_UNITS> ..... </MORE_COMPILATION_UNITS> -->
</CUS>
- <PKCS>
  - <PCK>
    <OWS>No System Name</OWS>
    <NAM>org.eclipse.pde.internal.core.text.plugin</NAM>
    <COM PL="0" FL="180" ML="0" SL="0" P="0" F="18" M="0" S="0"/>
    <HALS HV="49351.68" HE="703825.73" TA="4747" TO="5476"/>
    <TOTP MC="11" TC="494" NS="1524" NIP="19" NSP="10" NPR="10" NC="18" NA="0" NI="0" LOC="1960" NM="251"/>
    <LOC CO="184" CI="688" BL="329" TL="3161"/>
  - <CLSS>
    - <CLS>
      <SCL>org.eclipse.pde.internal.core.text.DocumentElementNode</SCL>
      <OWP>org.eclipse.pde.internal.core.text.plugin</OWP>
      <CNM>DocumentGenericNode</CNM>
    - <IMPS>
      <IMP>org.eclipse.pde.internal.core.text.DocumentElementNode</IMP>
    </IMPS>
    - <IVRS>
      <IVR V="long" K="serialVersionUID"/>
    </IVRS>
    <COM PL="0" FL="4" ML="0" SL="0" P="0" F="1" M="0" S="0"/>
    <LOC CO="4" CI="5" BL="3" TL="18"/>
  - <MTHS>
    - <MTH>
      <CNM>org.eclipse.pde.internal.core.text.plugin.DocumentGenericNode</CNM>
      <NAM>DocumentGenericNode</NAM>
      <RET>void</RET>
    - <ARGS>
      <ARG V="java.lang.String" K="name"/>
    </ARGS>
    - <CLRS>
      <CLR V="1" K="java.lang.String"/>
    </CLRS>
    - <LMCS>
      <LMC V="1" K="setXMLTagName"/>
    </LMCS>
    <COM PL="0" FL="3" ML="0" SL="2" P="0" F="1" M="0" S="2"/>
    <LOC CO="2" CI="2" BL="0" TL="7"/>
  </MTH>
    </MTHS>
  </CLSS>
</PCK>
</PKCS>
```

## 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.



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.

Packages in System: JUnit																					
	Name	No. Classes	NOS	AVCC	HBUG	HEFF	MI	CCML	NLOC	TCC	CCOM	HLTH	INST	DIST	FIN	No. Methods	MINC	ABST	HVOL	MAXCC	FOUT
🔗	junit.extensions	6	90	1.29	0.66	11767.43	181.96	37	131	31	11	540	0.50	0.50	2	24	131.96	0.00	1989.50	3	2
🔗	junit.framework	17	557	1.54	6.16	167133.46	149.70	523	792	229	125	4266	0.33	0.43	8	149	129.09	0.24	18474.07	9	4
🔗	junit.runner	3	184	1.76	2.07	69255.09	172.73	63	258	67	24	1341	0.40	0.07	3	38	122.73	0.67	6205.11	11	2
🔗	junit.textui	2	156	1.57	1.91	57310.68	163.01	76	217	55	20	1247	0.75	0.25	1	35	123.13	0.00	5720.95	9	3
🔗	org.junit	5	251	1.39	3.26	98323.41	84.06	801	368	93	68	2248	0.38	0.62	5	67	128.19	0.00	9789.64	5	3
🔗	org.junit.experimental	3	47	1.89	0.57	11140.98	120.43	0	65	17	0	395	1.00	0.00	0	9	120.43	0.00	1698.43	3	3
🔗	org.junit.experimental.categories	0	3	0.00	0.03	266.43	171.00	35	3	0	1	27	0.00	1.00	0	0	171.00	0.00	99.91	0	0
🔗	org.junit.experimental.max	7	158	1.62	2.06	46964.34	163.38	83	227	52	25	1350	0.88	0.12	1	32	123.49	0.00	6168.63	6	7
🔗	org.junit.experimental.results	6	71	1.22	0.74	12283.09	170.72	42	96	22	10	541	0.83	0.17	1	18	130.84	0.00	2221.51	3	5
🔗	org.junit.experimental.runners	1	6	2.00	0.07	960.06	151.71	20	8	2	2	57	1.00	0.00	0	1	118.33	0.00	205.76	2	1
🔗	org.junit.experimental.theories	10	172	1.62	2.28	58181.68	123.69	11	274	68	5	1533	0.67	0.13	3	42	123.69	0.20	6825.59	5	6
🔗	org.junit.experimental.theories.internal	4	160	1.83	2.16	59805.58	118.69	25	239	53	9	1391	0.60	0.40	2	29	118.69	0.00	6469.97	6	3
🔗	org.junit.experimental.theories.suppliers	1	20	2.00	0.29	7983.50	91.58	0	22	2	0	186	1.00	0.00	0	1	91.58	0.00	868.45	2	1
🔗	org.junit.internal	8	163	1.65	1.89	59839.62	168.94	41	227	51	7	1262	0.33	0.42	8	31	118.94	0.25	5681.58	7	4
🔗	org.junit.internal.builders	8	104	2.10	1.31	41184.19	168.66	24	163	42	8	892	0.70	0.30	3	20	118.66	0.00	3931.83	5	7
🔗	org.junit.internal.matchers	7	120	1.67	1.59	46632.08	123.29	21	174	50	6	1097	0.17	0.55	5	30	123.29	0.29	4766.24	6	1
🔗	org.junit.internal.requests	3	49	1.43	0.56	11200.98	154.05	17	66	10	4	391	0.67	0.33	2	7	114.16	0.00	1671.28	2	4
🔗	org.junit.internal.runners	9	291	1.75	3.99	151453.48	111.17	26	370	70	7	2350	0.53	0.47	7	40	111.17	0.00	11962.96	7	8
🔗	org.junit.internal.runners.model	3	48	1.45	0.46	8421.08	125.75	10	65	16	3	343	0.44	0.22	5	11	125.75	0.33	1384.64	3	4

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.

# Filters

Filters are available on all tables. As well as allowing the anonymous approach described in the ‘Quick start’ sections above filters can be stored to file and restored as required using the same interface.

Filters can be managed at three levels:

- Filter Record Group Sets
- Filter Record Groups
- Filter Records

The Filter Record is the lowest level. It contains a record of the level of the metrics being filtered (Package, Class or Method) , the filter applied relating to the calculated value of the metric (Danger, Warning, Warning and Danger) and the Metric(s) that are to be calculated for the filter.

A Filter Record group contains one or more filter records, it has an identifier and a description. If a number of Filter Record Groups are contained in a Filter Record Group Set then the Filter Record Group identifier must be unique within that Filter Record Group set.

A Filter Record Group set is a logical grouping of Filter Record Groups. When the Manage Filter Groups dialog is used then all the Filter Record Groups being managed in the dialog are treated as a single Filter Record Group Set. When Filter Record groups are stored in a file they are stored as a Filter Record Group Set. If two Filter Record Group Sets are sequentially loaded from two different files then any Filter Record Group in the second file that has the same identifier as one in the first file will overwrite the first record.

Note that you must use the same JHawk properties file to run the filters as you use to create them. You can create sets of filters that include filters for different levels of artefact (package, method and class) and for different levels of metric (Warning level only, Danger level only, warning and danger level). All of the filters will be applied in turn and as soon as one of the filter criteria is met any artefact that meets the criterion will be added to the filtered set. Note that if you are analysing a hierarchy of artefacts, only the artefacts that pass the filter at the highest level will be used in the subsequent analysis. As an example let’s say that you have the following artefacts that pass the filters that you have set:

- Package1
- Package1.Class1
- Package1.Class1.Method1
- Package2.Class2
- Package3.Class3.Method3

Any artefact that is not mentioned does not pass any of the filters so if we analyse at package level only the following artefacts will pass the filters:

- Package1
- Package1.Class1
- Package1.Class1.Method1

Because Package2 and Package3 do not pass the filter no further analysis will be carried out on them. If we analyse at class level the following artefacts will pass the filters:

- Package1.Class1
- Package1.Class1.Method1
- Package2.Class2

Because Package3.Class3 does not pass the filter no further analysis will be carried out on it. Finally if we analyse at method level the following artefacts will pass the filters:

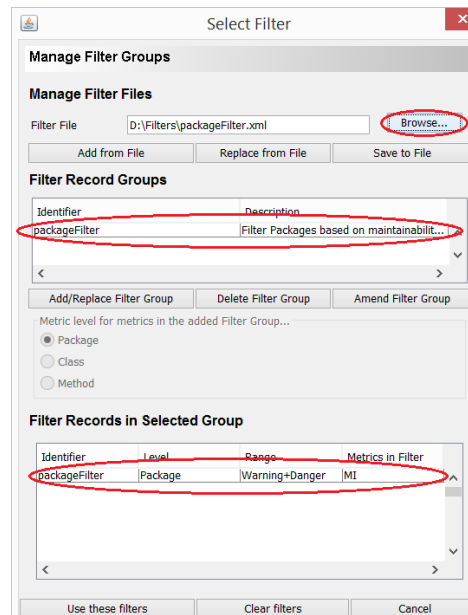


- Package1.Class1.Method1
- Package3.Class3.Method3

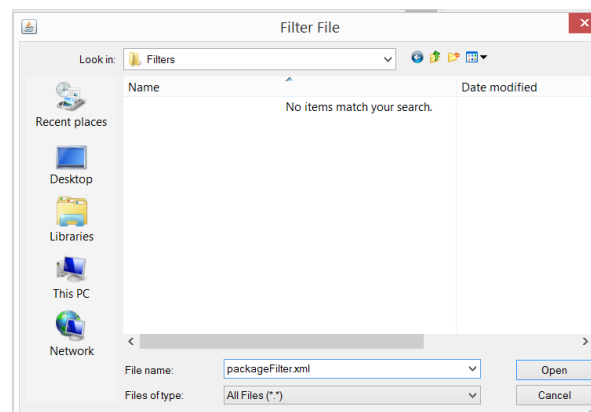
The following sections cover the creation, management and use of filters in more detail.

### ***Storing a filter record group set in a file***

In this case we want to store the anonymous filter that we created earlier. To do this we need to populate the identifier and description fields. We then go to the ‘Manage Filter Files’ section:



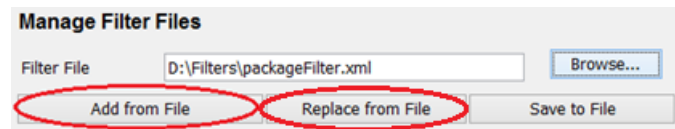
When we press the ‘Browse’ button we see the following dialog that allows us to enter the name of the file that we want to store the filter in:



### ***Installing filter record groups from a file***

In this case we use the ‘Browse’ button to select the file that we want to restore the filter(s) from. We have two options at this point:

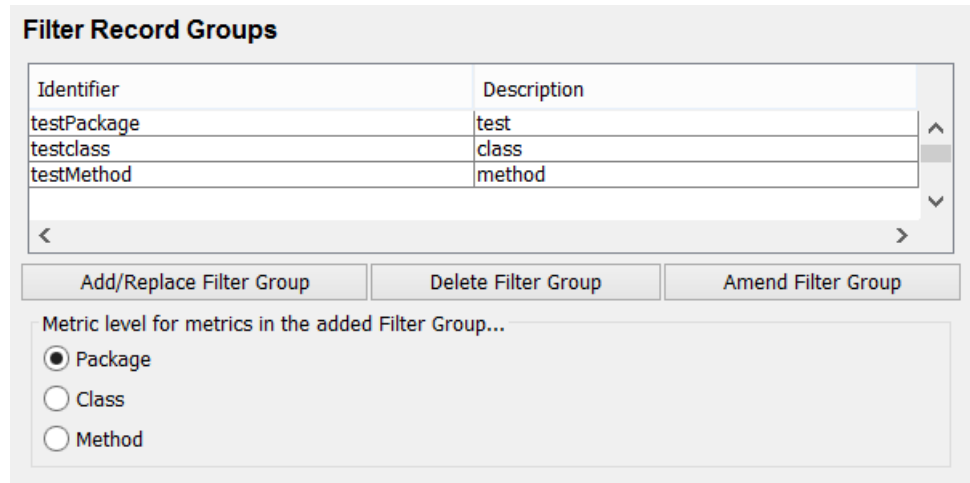
1. We can add the filter groups in this file to the filter groups that we currently have defined (i.e. all the filter groups that you can see listed in the ‘Filter Record Groups’ table on the ‘Manage Filter Groups’ dialog). To do this we press the ‘Add from File’ button.
2. We can replace all the filter groups that we currently have defined with the filter groups in the file. To do this we press the ‘Replace from File’ button.



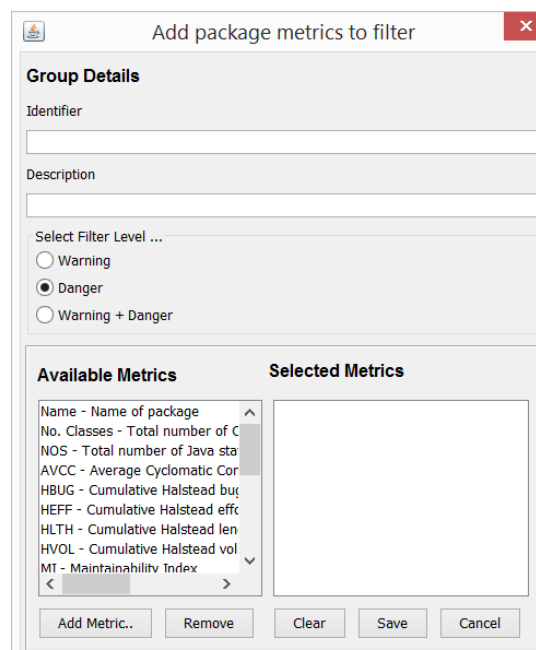
In each case the Filter Record Groups table will be updated with the new filters.

### ***Managing individual Filter Record Groups***

Filter Record Groups are managed using the 'Filter Record Groups' section of the dialog. This includes a table of the current record groups, buttons to Add/Replace, Delete and Amend record groups and buttons to set the level of metrics to include in the group.



The level of metrics that you can use in this group will be preselected according to the context from which you started the dialog. When you press the 'Add/Replace Filter Group' button you will then see the following dialog:



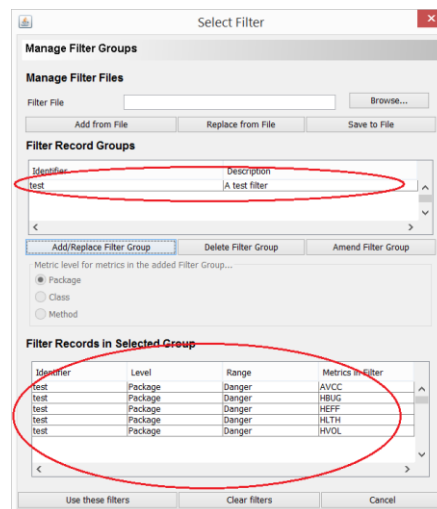
You must at least provide an identifier and select some metrics before saving the record group. All of the metrics in a Filter Record group must be at the same filter level (Danger, Warning and Danger or Warning level only). As soon as you have selected some metrics the Filter Level panel will be disabled and only metrics at the currently selected level can be chosen.

When you select metrics to add to the group they will be moved from the ‘Available Metrics’ list to the ‘Selected Metrics’ list. You can move selected metrics back from the ‘Selected Metrics’ to the ‘Available Metrics’ list by selecting them in the ‘Selected Metrics’ list and pressing the ‘Remove’ button. You can move all of the metrics back from the ‘Selected Metrics’ to the ‘Available Metrics’ list by pressing the ‘Clear’ button.

If you elect to save the newly added or amended group then you will be returned to the main dialog and if you select the group that you were working on you will see the individual records displayed in the ‘Filter Records in Selected Group’ table.

## Viewing the Filter Records in a Filter Record Group

All of the filters in the currently selected Filter Record Group are shown in the ‘Filter Records in Selected Group’ table. When you select a Filter Record Group in the ‘Filter Record Groups’ table the constituent records will be displayed in the ‘Filter Records in Selected Group’ table:



## Filters – Exporting Data

You can apply filters when you are exporting data. This means that only the data that you select with the filter will be exported. The filters can be activated using the ‘Filter...’ button on each of the export tabs – CSV, HTML and XML. For more details see the section ‘Filtering Export Data’ above.

## Filter Preferences

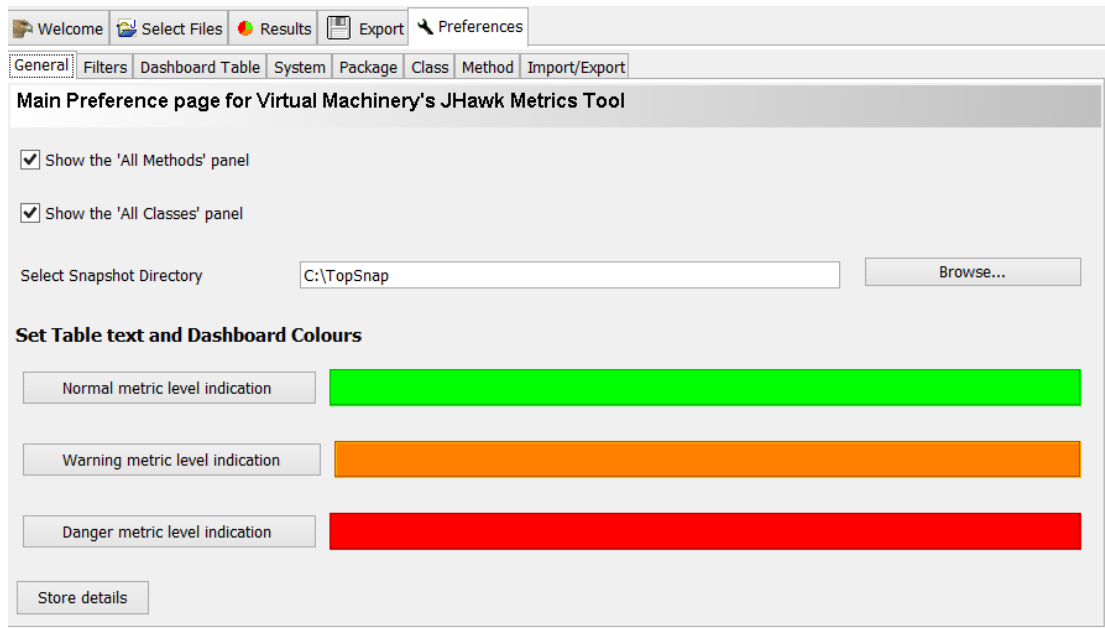
Filter preferences are set using the ‘Filters’ sub-tab of the ‘Preferences’ tab. See the ‘Preferences’ section further down in this document.

## Standard Filter Files

A number of standard filter files come with JHawk. These files can be found in the filters directory of the distribution. These filters reflect the standard Warning and Danger levels that are predefined for a number of different metrics and levels in the standard JHawk properties files. These can be used as they are or as a basis for building your own metrics filters.

File Name	Artefact	Metrics	Level
StandardPackageD.xml	Package	AVCC, MI, MAXCC	Danger
StandardPackageWD.xml	Package	AVCC, MI, MAXCC	Warning + Danger
StandardClassD.xml	Class	AVCC, MI, MAXCC	Danger
StandardClassWD.xml	Class	AVCC, MI, MAXCC	Warning + Danger
StandardMethodD.xml	Method	COMP	Danger
StandardMethodWD.xml	Method	COMP	Warning + Danger

# The Preferences Tab

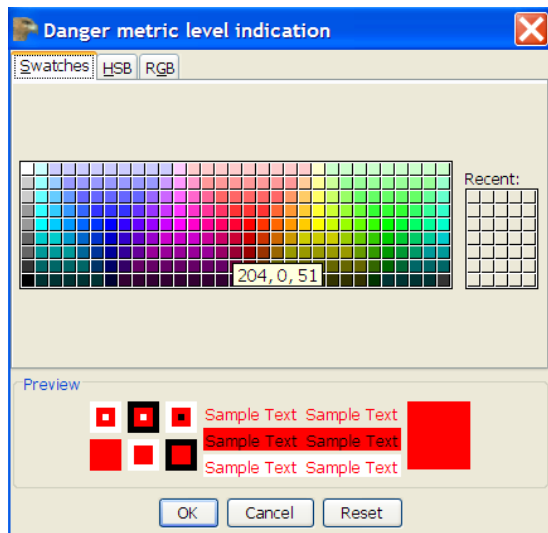


The Preferences tab opens on the 'General' sub-tab which is the first of seven sub-tabs on the Preferences tab. The Dashboard Table tab allows you to specify the tables that will appear on the Dashboard sub tabs. The 'System', 'Package', 'Class' and 'Method' sub-tabs allow you to configure the metrics 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.
- The default directory to be used for storing the Snapshot files used by the Snapshot sub-tab of the Dashboard sub-tab.
- 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 –

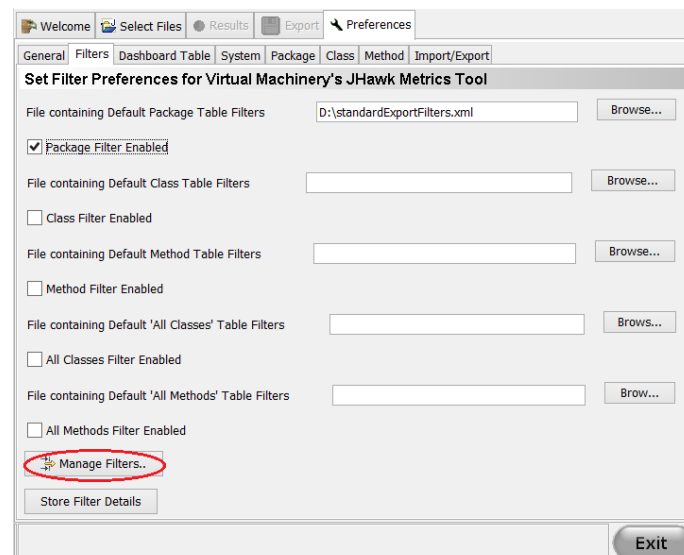


If you wish to see these attributes reflected in JHawk then press the 'Store Details' button then go back to the 'Select Files' tab and run the analysis again – the new settings will then be reflected in JHawk.

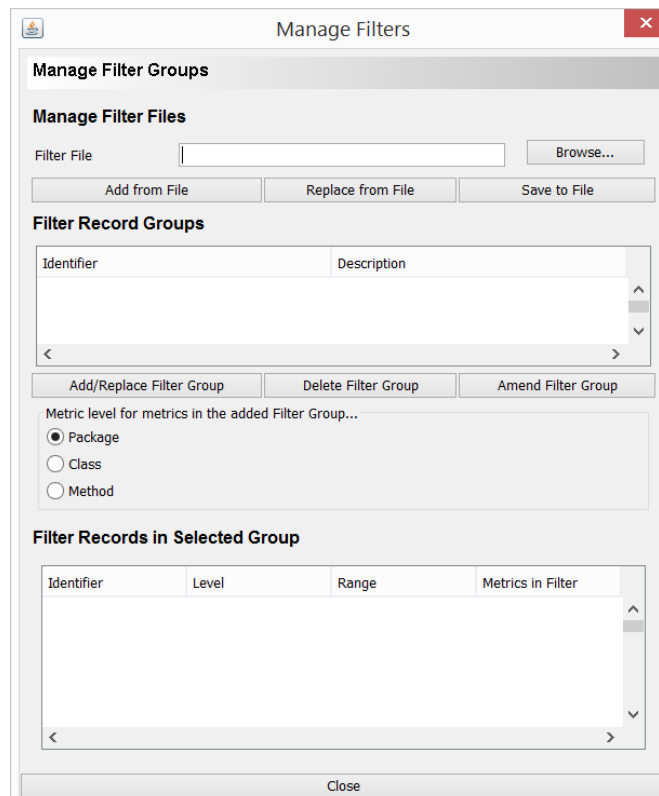
### ***The Filters sub-tab***

Using this sub-tab you can set the file containing the filters to be used when filtering tables at the package, class and method level as well as on the 'All classes in system' and 'All methods in system' tables. You do this by selecting the file that contains the filters that you wish to use at each level.

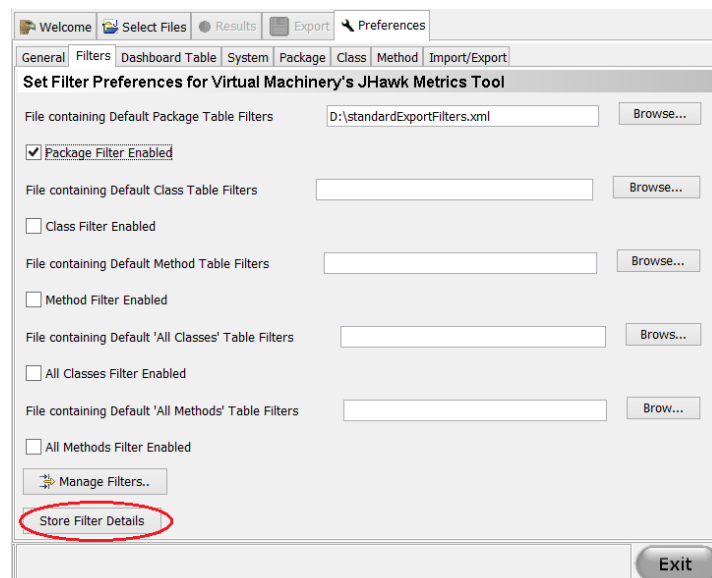
As well as selecting the filters you can also set whether they are enabled or not. The default filters will override any other filters that you have already set on the tables but you can add additional filters on the tables by using the 'Filters...' button on the table.



You can open the 'Manage Filters' dialog by pressing the 'Manage Filters' button. This will open a dialog similar to that available from the 'Filters....' button on the tables.



Using this dialog you can create and edit filters and load them from files. You can also save filters to files. You can use this dialog to create filter files which can then be assigned as the default filters for particular tables. To close the dialog use the 'Close' button at the bottom of the dialog.



When you have finished setting the preferences you can press the 'Store Filter Details' button. The filters will then be stored in the current preferences and the tables will be updated with the default values. If you save the current preferences to a properties file then the default preferences will also be stored to the file and will be loaded when that properties file is used to start JHawk.

## Dashboard Table sub-tab

You can use this to configure the dashboard tables. This tab is divided in two – the top half contains a table showing the records defined and provides facilities to manipulate and save this list; the lower half allows you to modify or add a particular record.

#	Level	# Shown	Order Metric	Metrics Shown
0	Method	4	NOC	Name,NOA,NOC,VREF,

Up Down Add Record Delete Record Save Details to Store

**Select Metrics for Dashboard table**

Select level for metric to add...

☐ Package ☐ Class ☒ Method

COMP - Cyclomatic Complexity  
NOCL - Number of comment lines

Name - Name of method  
NOA - Number of arguments  
NOC - Number of comments

Add Metric.. Clear Selected Remove Order On : NOC Up Down

Number of rows 4 Save Entry Changes

## The Dashboard table records list

The list has five columns – a sequence number representing the order in which the tables will be displayed (#), a level (Package, Class or Method), the number of entries shown (# Shown), the metric used to order the list (Order metric) and the other metrics displayed for each entry (Metrics Shown). The entry shown in the screen shot shows that the 4 methods with the greatest number of comments (NOC) will be shown. The columns shown will be the Name, the number of arguments, the Number of Comments and the number of variable references. The list will be ordered on the number of comments.

The display order can be changed by using the 'Up' and 'Down' buttons to move the selected dashboard table display record.

Use the 'Add' button to create a new record – this will clear the current entry in the Add/Edit dashboard table panel.

Use the 'Delete' record to remove a record from the list.

Use the 'Save details to store' button to save the current list of entries to the store. Remember that this will not update the properties file used to start JHawk – to modify this file you will need to use the 'Import/Export' sub tab of the 'Preferences' tab to export the properties to an external properties file.

## The Add/edit dashboard table panel

You can use this either to edit an existing dashboard table entry or to create a new one. If you wish to create a new dashboard table you need to press the 'Add Record' button on the dashboard table list panel.

A set of three radio buttons at the top of the panel allows you to select the level of metric to be used - Package, Class or Method. Changing this button will select the list of metrics that can be selected at this level.

The two list panels below allow you to select the metrics to be considered when creating the dashboard table. Metrics can be selected in the left hand panel and moved to the right hand panel using the 'Add metric' button. A metric in the right hand (selected) panel can be removed by using the 'Remove' button – it will then reappear in the left hand panel and be available for selection. The 'Clear Selected' button will move all the metrics selected in the right hand pane back to the left hand pane.

To choose the metric to be used to select and order the list select one of the metrics in the right hand panel and press the 'Order On' button. The key of the metric will then be displayed in the label to the right of the 'Order On' button.

Note that you should always select a 'name' metric to your list of selected metrics so that you can identify the entries in the table.

The left to right order of the columns in the dashboard table is represented by the top to bottom order of the metric list in the right hand panel. Metrics can be moved up and down this list by using the 'Up' and 'Down' buttons.

On the bottom line you can enter the number of rows to be displayed in the table using the 'Number of rows' entry panel. Pressing the 'Save Entry Changes' button will add the entered details to (or alter the existing details in) the dashboard table list in the upper panel.



## Metrics sub-tabs – Edit Metric Column details

The screenshot shows the 'Edit Class Metric column details' window. At the top, there's a menu bar with 'Welcome', 'Select Files', 'Results', 'Export', and 'Preferences'. Below it, a sub-menu bar includes 'General', 'Filters', 'Dashboard Table', 'System', 'Package', 'Class', 'Method', and 'Import/Export'. The main title is 'Edit Class Metric column details'.

#	Active	Align	Code	Column ...	Descript...	Compare	Warning	Danger	Class N...	On Dash...	Sort Field
2	Y	Right	LCOM	LCOM	Lack of C...	>	0.0	0.0	com.virtu...	N	N
3	Y	Right	AVCC	AVCC	Average ...	>	10.0	15.0	com.virtu...	Y	N
4	Y	Right	NOS	NOS	Total nu...	>	0.0	0.0	com.virtu...	N	N

Below the table are buttons: 'Up', 'Down', 'Set as sort field', 'Add Metric', and 'Delete Metric'.

The 'Set Values for : AVCC' section contains the following fields:

- ☒ Active
- ☐ Is Default Sort Field
- ☒ On Dashboard
- Level :** ☒ System ☒ Package ☐ Class
- Description: Average Cyclomatic Complexity
- Class name: virtualmachinery.jhawk.registry.basemetrics.AverageCyclomaticComplexityMetric
- Key: metric.class.AVCC
- Column Header: AVCC
- Align value to: Right
- Compare value using: >
- Type of Metric Value is: Double
- Warning Level: 10.0
- Danger Level: 15.0

At the bottom are buttons: 'Save Entry Changes' and 'Save Details to Store'.

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	MOC	MOC	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:

Class name:

Key:

ColumnHeader: 
 Align value to:

Compare value using: 
 Type of Metric Value is:

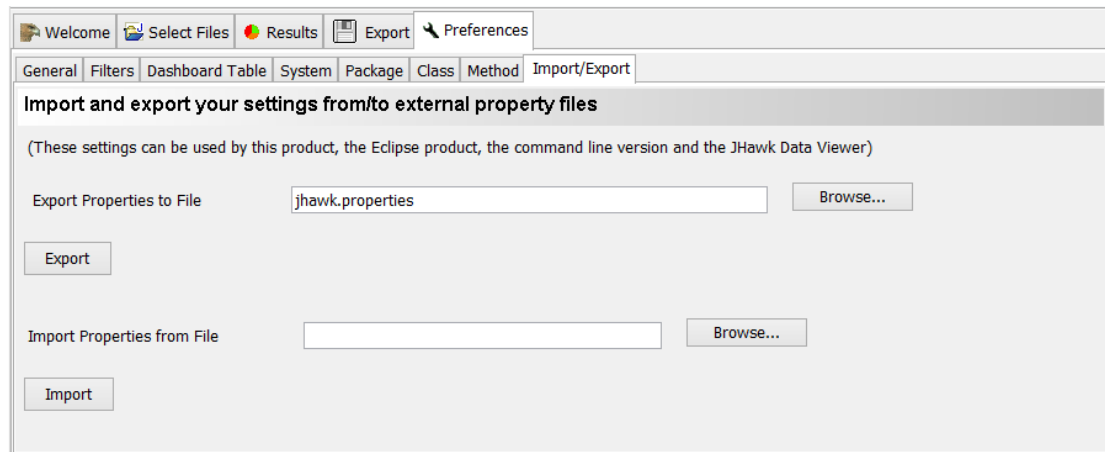
Warning Level: 
 Danger Level:

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



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 JHawk.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.