J-Hawk is a valuable tool for analyzing the performance of Java code developed by Virtual Machinery. J-Hawk’s primary focus is on performance statistics and provides many additional details about code complexity and relationships. The application is a professional development tool with commercial licenses available from $30 for a starter license up to $900 for a corporate license. There is a free demo is available for testing which is sufficient for analyzing a small project and an academic license available for research. Licenses are available to be purchased from the Virtual Machinery website.

As you know the end result of any project can be achieved with numerous different approaches which vary in efficiency and complexity. J-Hawk’s statistical analysis helps you to understand which sections of code might be problematic so you can focus on refactoring these areas to produce an end product that is better handled by all systems. As a result, the more efficient code achieved with J-Hawk will result in less frustration for the end-user.

Testing with J-Hawk can be done in seconds by simply importing the java files to be tested or with Eclipse integration. This fast execution time ensures that testing will be completed more often with immediate feedback. J-Hawk provides the following information:

* Cyclomatic Complexity
* Halstead Complexity Statistics
* Predicted Number of Bugs
* External Classes/Methods Referenced
* Lines of Code

While different levels of complexity are expected with each class/method, checking that these values are consistent with expectation helps reduce unnecessary operations and to see where the biggest time-saving efforts can potentially be made. One especially useful feature is the prediction of which class/methods could contain bugs. Having these problematic class/methods flagged helps you to focus efforts while searching for bugs. Image 1 demonstrates this by highlighting the most problematic method of a particular class.

Unlike many development tools that need to be integrated into a project, J-Hawk analyzes code statically as a whole. This means that any Java project can be easily analyze at any time. Since J-Hawk is extremely simple and easy to use, you could analyze your code upon completion of any class/method to ensure that you are producing a high-quality product that is not excessively complex or problematic. If code is too complex it can be refactored and quickly reanalyzed until it is satisfactory. J-Hawk can provide useful feedback during the entire development process, however its ability to analyze a project as a whole is especially useful later in development process to identify bottlenecks which need attention.

J-Hawk comes in two easy to use options. The first option is a stand-alone app packaged as a jar file and only requires JRE to be installed. The second option is Eclipse integration which requires the user to copy a jar file and properties file into the eclipse installation folder. Both options come preconfigured.

While J-Hawk is an incredibly powerful tool, its holistic nature is also its biggest drawback. The feedback given by J-Hawk is rather abstract and only points the developers towards areas that need attention. It doesn’t specifically highlight where the problems occur, nor does it provide any specific insight into how to correct problem therefore, it’s necessary to use J-Hawk in conjunction with other testing tools. Even though J-Hawk has some drawbacks the level of general coding heuristics provided as immediate feedback can provide exceptionally useful.

Image 1

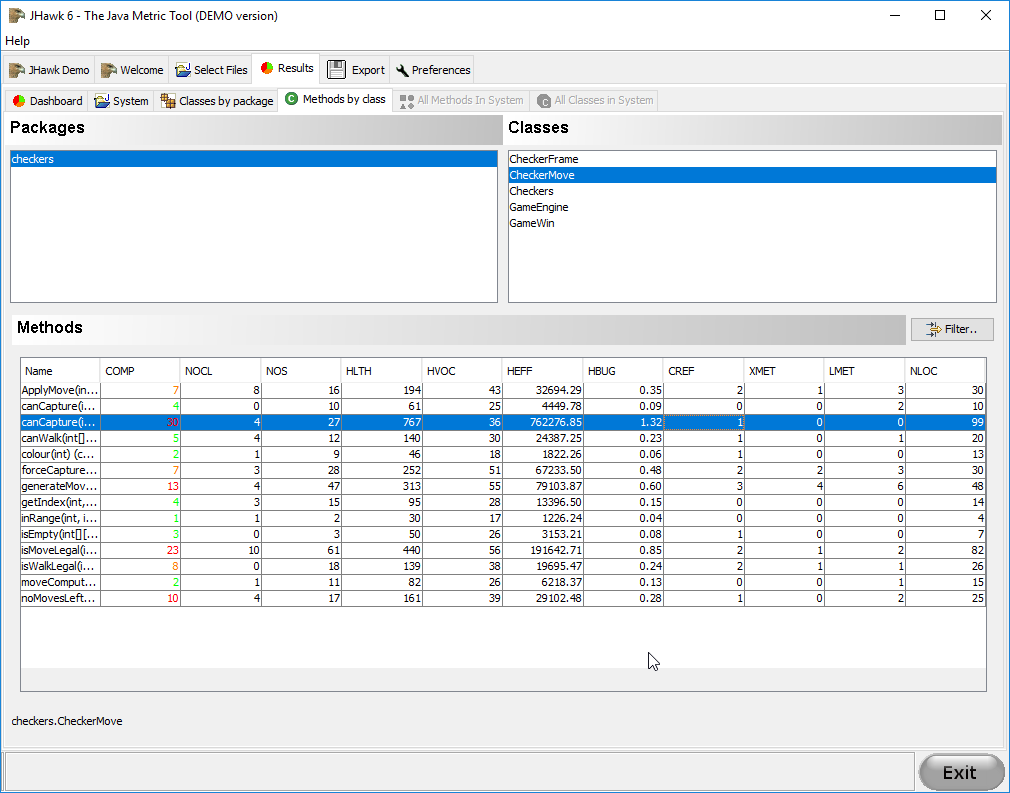


Image 2 – Importing java files to be analyzed.

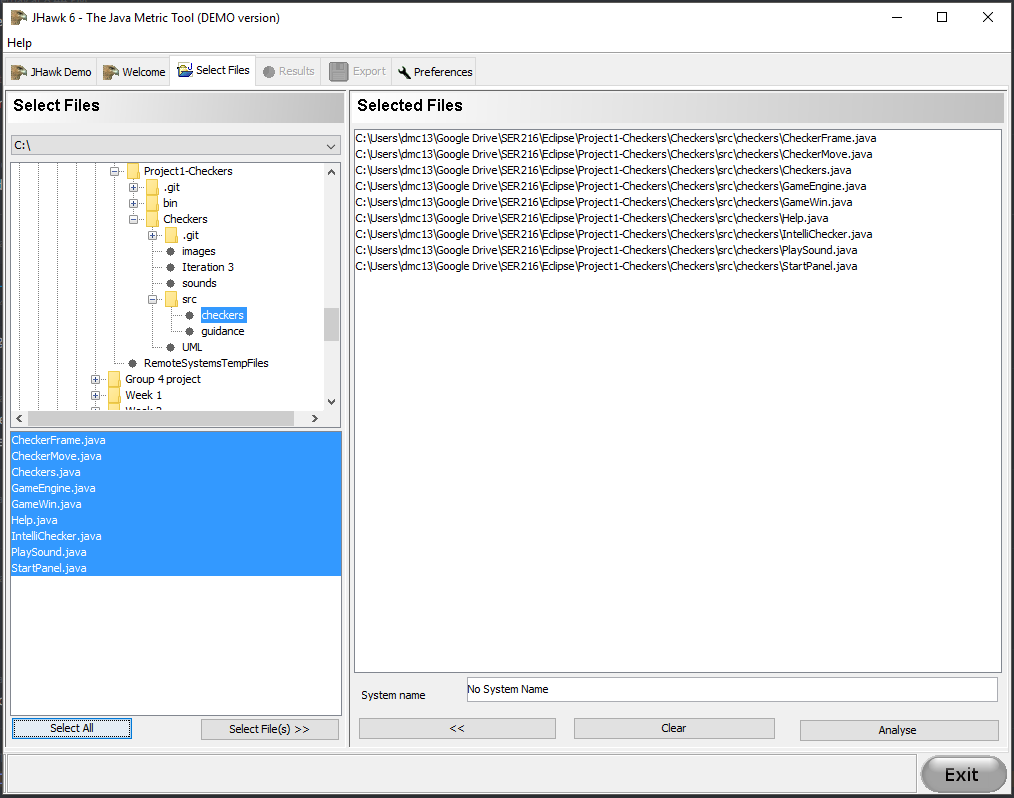


Image 3 – the dashboard view of the tool showing breakup by package, class and methods.

