How to use Dependency Analyzer

Last updated: November 5, 2014

How to call the tool

To call the tool, use:
```
analyzer = new DependencyAnalyzer();
analyzer.runClassycle();
```

After that, you can call the following methods to get class and package dependencies respectively:
```
analyzer.getAllClassesDependencies();
analyzer.getAllPackagesDependencies();
```

To see how to access the information stored in those vectors, please take a look at the print methods below.
To print out all of the dependencies information, call:
```
analyzer.printClassSummary();
analyzer.printPackageSummary();
```

Back to the vectors

getAllClassesDependencies() returns a vector of ClassDependencyInfo objects. Each object contains:

- class name
- a vector of afferent classes
- a vector of efferent classes

getAllPackagesDependencies() returns a vector of PackageDependencyInfo objects. Each object contains:

- package name
- a vector of afferent packages
- a vector of efferent packages

You can index through each vector to get the objects for each class/package in the codebase. Again, take a look at the `printClassSummary()` and `printPackageSummary()` methods to get an idea on how to parse each vector and retrieve the information from objects.

For example, a code base contains 3 packages and 7 classes. Then, the PackageDependencyInfo Vector will contain 3 objects, and the ClassDependencyInfo Vector will contain 7 objects.

I don’t think one would need to use any of the setters methods from the PackageDependencyInfo and ClassDependencyInfo classes.

Mapping of accessor methods and what they return.

For the ClassDependencyInfo class:

getClassName() – class name

getEfferentVectorSize() – the size of the efferent classes vector. In other words, the number of efferent classes.
getEfferentVectorElemAt(i) – a name of an efferent class at index i

getAfferentVectorSize() - the size of the afferent classes vector. In other words, the number of afferent classes.
getAfferentVectorElemAt(i) – a name of an afferent class at index i.

There are similar accessor methods for the PackageDependencyInfo class.

Note that the names returned by accessor methods are Strings.

How objects are created and appended to the vectors

As the tool analyzes the info, the information on a particular class is stored in an object. Each class in the codebase has its own object. Each object is then appended to the vector, which is later accessible through getAllClassesDependencies().

The tool goes through a similar process of creating an object for each package in the codebase. These objects are then appended to the corresponding vector, which is accessible by getAllPackagesDependencies().