

Group 2 → Jake Ballinger, Tristan Challener, Adam Wechter, Erich Harkema

Honor Code Signatures:

Deliverable #1

	Inputs	Outputs	Behaviors
:PingEclim	:PingEclim	Lists the current version of both Eclim and Eclipse (this is debugging output)	Pings Eclim to see if it is up and running
:ProjectCreate	:ProjectCreate <folder> [-p <project-name>] -n <nature> ... [-d <project_dependency>.. .] -p: optional argument to specify the project name -n: Required argument which specifies a space separated list of project names (java, php, etc.) to add to the project Use “none” to create a project without natures	Creates a project	Creates a project
:ProjectList	:ProjectList		Echos a lost of available projects
:ProjectInfo	:ProjectInfo		Echo info about the current or supplied project
:JavaDocComment	:JavaDocComment		Will add/update the javadoc comments for the element under the cursor
:JavaFormat	:JavaFormat To format the entire file,		Formats the current visual selection

	use %JavaFormat		
:JavaImport	:JavaImport Place cursor over element to import		If only one matching element is found: - import statement placed in the file If multiple matching elements are found: - Prompts user to choose which to import If element is being imported, is the the Java.lang directory, or is in the same package as the current src file - No changes
:JavaSearch	:JavaSearch -p <pattern> [-t <type> -x <context> -s <scope> -i] -p: Pattern to search for us -t Type of Element -x Context of the search -s Scope of the search -i Ignore case when searching :JavaSearch NPE - camel case searching		Widen a search beyond a single element
CTRL-x CTRL-u	1. Enter insert mode 2. Type Ctrl-x Ctrl-u 3. Ctrl-n procedes to the next option 4. Ctrl-p moves to the previous match :h ins-completion: find out more!		Brings up the code completiong menu whenever you press tab. Shows all options for completing the word at the cursor. “Arr” would bring up a list of options including “ArrayList”, etc.

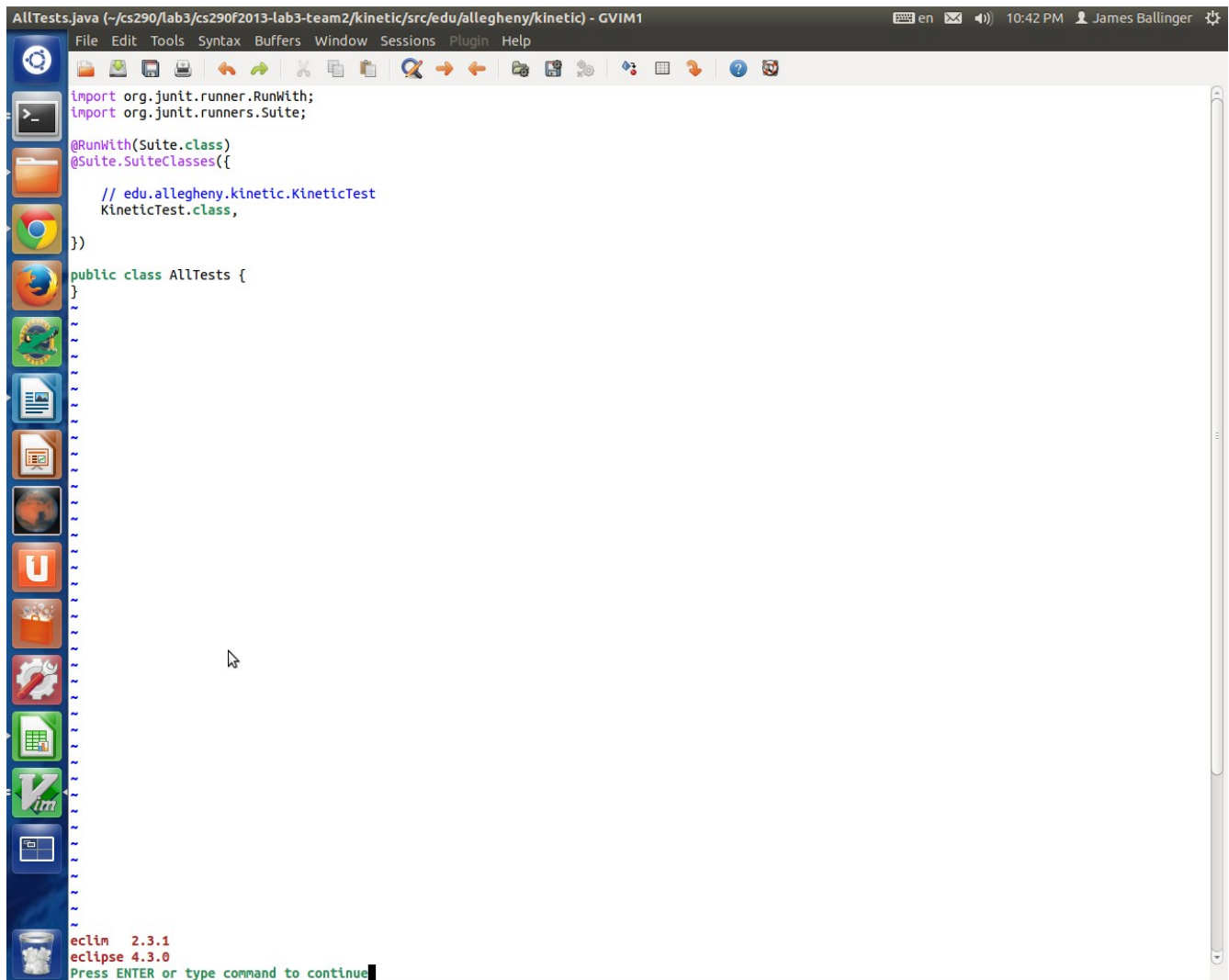
Deliverable #2

```
ballingerj@aldenv137: ~$ eclind
java -d64 -Dosgi.requiredJavaVersion=1.6 -XX:MaxPermSize=256m -Xms40m -Xmx512m -
jar /opt/eclipse/plugins/org.eclipse.equinox.launcher_1.3.0.v20130327-1440.jar -
debug -clean -refresh -application org.eclim.application
Install location:
file:/opt/eclipse/
Configuration file:
file:/opt/eclipse/configuration/config.ini loaded
Configuration location:
file:/home/b/ballingerj/.eclipse/org.eclipse.platform_4.3.0_1473617060_linux
_gtk_x86_64/configuration/
Configuration file:
file:/home/b/ballingerj/.eclipse/org.eclipse.platform_4.3.0_1473617060_linux
_gtk_x86_64/configuration/config.ini loaded
Loading timestamp file from:
file:/home/b/ballingerj/.eclipse/org.eclipse.platform_4.3.0_1473617060_
linux_gtk_x86_64/configuration/.baseConfigIniTimestamp
No timestamp file found
Timestamps found:
config.ini in the base: 1377627936000
remembered -1
Shared configuration location:
file:/opt/eclipse/configuration/
Framework located:
file:/opt/eclipse/plugins/org.eclipse.osgi_3.9.0.v20130529-1710.jar
Framework classpath:
file:/opt/eclipse/plugins/org.eclipse.osgi_3.9.0.v20130529-1710.jar
Debug options:
file:/home/b/ballingerj/.options not found
Time to load bundles: 12
Starting application: 5425
Application started: 8054
2013-09-24 22:31:50,431 INFO [org.eclim.eclipse.EclimDaemon] Workspace: /home/b/ballingerj/workspace
2013-09-24 22:31:50,433 INFO [org.eclim.eclipse.EclimDaemon] Home: /opt/eclipse/plugins/org.eclim_2.3.1/
2013-09-24 22:31:50,433 INFO [org.eclim.eclipse.EclimDaemon] Starting eclim...
2013-09-24 22:31:50,444 INFO [org.eclim.eclipse.EclimDaemon] Loading plugin org.eclim
2013-09-24 22:31:50,446 INFO [org.eclim.eclipse.EclimDaemon] Loading plugin org.eclim.core
2013-09-24 22:31:50,577 INFO [org.eclim.plugin.core.CorePlugin] Loading eclim plugins...
2013-09-24 22:31:50,579 INFO [org.eclim.plugin.core.CorePlugin] Loading plugin org.eclim.jdt
2013-09-24 22:31:50,582 INFO [org.eclim.plugin.core.CorePlugin] Plugins loaded.
2013-09-24 22:31:50,583 INFO [org.eclim.eclipse.EclimDaemon] Loaded plugin org.eclim.core
2013-09-24 22:31:50,584 INFO [org.eclim.eclipse.EclimDaemon] Eclim Server Started on: 127.0.0.1:9091
```

Terminal Window Output from running Eclimd

```
ControlP - (~/.cs290/lab3/cs290f2013-lab3-team2/kinetic/eclim) - GVIM1
File Edit Tools Syntax Buffers Window Sessions Plugin Help
[No Name] [citi(master)] 0,0 ALL
~/.cs290/lab3/cs290f2013-lab3-team2/kinetic/eclim
prt path enrup=< Files >=<buf> <-> /home/b/ballingerj/cs290/lab3/cs290f2013-lab3-team2
>>> build.xml
```

Loading build.xml



```
AllTests.java (~/.cs290/lab3/cs290f2013-lab3-team2/kinetic/src/edu/allegheeny/kinetic) - GVIM1
File Edit Tools Syntax Buffers Window Sessions Plugin Help

import org.junit.runner.RunWith;
import org.junit.runners.Suite;

@RunWith(Suite.class)
@Suite.SuiteClasses({
    // edu.allegheeny.kinetic.KineticTest
    KineticTest.class,
})

public class AllTests {
}

eclim 2.3.1
eclipse 4.3.0
Press ENTER or type command to continue
```

Debugging output from :PingEclim

Deliverable #4

There were multiple defects within the Kinetic.java program. First, the formula for calculating the square of the velocity was incorrect. In the supplied code, it was calculated as:

velocity_squared = 3 * (kinetic / mass)

while the correct formula is:

velocity_squared = 2 * (kinetic / mass)

An additional error was detected in the logic of the program. In its original state, the program accepted negative mass and kinetic values, which is incorrect. Both negative energy and negative mass should not be accepted, since neither are physically possible (arguably possible for negative energy, but only in very specific fringe cases in theoretical physics). This error was corrected by adding additional restraints to the check at the beginning of the computeVelocity() method, as shown below:

if(mass > 0 && kinetic >= 0)

Additional Information

In addition to everything discussed above, there are a few other points worth noting. Firstly, it can be seen that our coverage is in fact not 100% for all classes. Specifically, the KineticTest.java shows a coverage of only 96%. The reason for this is that there are branches of two of our tests which are never visited. However, this is *intended behavior*. The fact the these branches are unvisited is in fact direct evidence the test are being passed; those branches are for handling the case where the test fails, which they never should. As such, this is an acceptable lacking in coverage.

While considering our coverage reports, it is also worth noting that the AllTests and Kinetic classes do show 100% coverage. We achieved this by including a Junit test for the *default constructors* in each of those classes. Although these constructors would never be used in normal use of the classes, we decided to include a simple test (check that they produce an object) so that our coverage would in fact show 100%. All of this information is also reflected in the documentation within those files.