CMPSC 290
Principles of Software Development
Fall 2013

Laboratory Assignment Three: Using Vim to Implement and Test Java Programs

Introduction

While it is possible to implement and test Java programs in Vim without the use of any additional plugins, it is often useful to extend Vim with a plugin called Eclim. As described at `http://eclim.org/`, Eclim is a plugin that makes it possible to access the features of Eclipse through Vim. This means that you can program in Vim and use Eclipse features that support the completion, searching, validation, and compilation of source code. In this laboratory assignment, you will work with your team members to learn how to use the basic features associated with Eclim. Then, leveraging your knowledge of Git, Vim, and Eclim, you will implement a test suite and user interface for a Java program. In next week's session, you will publicly demonstrate your test suite and program.

Learning the Basics of Eclim

Eclim provides a convenient and easy-to-use interface to all of the features provided by Eclipse. Since we will be using Eclim's Java-based features in Vim, you should study `http://eclim.org/vim/java/index.html` to learn how to use this plugin. In particular, make sure you can use:

1. `:PingEclim`
2. `:ProjectCreate`
3. `:ProjectList`
4. `:ProjectInfo`
5. `:JavaDocComment`
6. `:JavaFormat`
7. `:JavaImport`
8. `:JavaSearch`
9. Code completion with CTRL-x, CTRL-u

Implementing and Testing a Java Program

Once you have finished learning the basics of Eclim, you are ready to access a Bitbucket-hosted Git repository that contains the source code for the system that you will finish implementing and testing. First, the members of each team should ask the instructor to share the Bitbucket repository with them. Once the repository is in your Bitbucket dashboard, you should use a terminal to execute this command in an appropriate directory: `git clone git@bitbucket.org:gkapfham/kinetic.git`.

Once you have finishing cloning the repository, please take some time to study the directory structure. What files did you find? What did you notice about the directories? Now, you should examine the .eclimrc_k file that is responsible for configuring Eclim to find the Kinetic project that is this assignment's focus. Please use Vim to change the `sgi.instance.area.default` variable so that it points to the directory containing the Git repository.

Now, you are ready to run the Eclim daemon called `eclimd`. Using the `which` command in your terminal, find out where this command was previously installed on your workstation. Once you have verified that `eclimd` is installed on your machine, run `eclimd` with the `-f` flag to specify that the .eclimrc_k file should be used as the runtime configuration file for Eclim. After running this program, what output do you see in the terminal window?

If `eclimd` is running correctly, then you can type `gvim` in a terminal window from the `kinetic` directory. Using CTRL-P, load the build.xml file and study the targets that it provides. Next, use CTRL-P to load the Kinetic.java, KineticTest.java, and AllTests.java files. Again, please carefully review these Java classes to make sure that you understand the provided methods. Once you understand both the build file and the Java classes, you should use the `:PingEclim` command to make sure that GVim can connect to the Eclim daemon. If you see debugging output showing the current version of both Eclim and Eclipse, then you can issue the `:ProjectCreate` command in GVim to create an Eclipse project for Kinetic. Once the project was successfully created, you can run `:ProjectList` and `:ProjectInfo` to ensure that you properly initialized Eclim.

Since Vim is now configured to operate as an integrated development environment, it contains features that support the compilation and testing of your Java source code. For instance, we can use the Apache Ant build system, described at `http://ant.apache.org/`, to perform automated compilation and testing of our program. If you type `:Ant compile` you can run the Java compiler for the three Java classes that are part of the project. Next, call `:Ant test` to run the test suite. Finally, do not forget to run the `:Ant clean` command. What output do you see when you use these commands? Please see the instructor if these commands do not work correctly.

Performing Test Coverage Monitoring

Practicing software engineers commonly use test suites to identify defects in and establish a confidence in the correctness of their programs. You can run the tests for the Kinetic system by executing the `:Ant test` command. What output does the test suite produce? Do the tests find a bug in the Kinetic program? Of course, it is very important to evaluate the quality of the test suite that you implement. Under the assumption that it is not possible to find bugs in code that has not been executed, software engineers commonly use code coverage to evaluate test quality.

Once again, we can use Vim and the `:Ant coverage` command to run the JaCoCo coverage monitoring tool. After learning more about JaCoCo by visiting `http://www.eclemma.org/jacoco/`, you should load the coverage report in the `bin/site/jacoco/AllTests/edu.allegheny.kinetic/` directory to determine if Kinetic has a good test suite. How much of the code does `KineticTest` cover? Is this an acceptable score? As you continue to enhance Kinetic and add new features and test cases, you should regularly use JaCoCo to calculate the coverage of your test suite. If you notice that you have code or conditional logic branches that are not covered, you should write additional test cases to cover this code. Please ensure that you submit both the coverage report that you produced before modifying Kinetic and the one that you created for the final version.

Fixing and Extending the Program

If you carefully studied the Kinetic.java file, you will have noticed that it contains a defect! In light of the fact that this program has a test suite with tests that always pass, it is alarming to note the existence of this bug. Can you write a test case that will reveal this defect? Can you fix the bug? Finally, you should use Vim and Eclim to complete the following implementation tasks:

1. Add a simple command-line interface that will allow a user to perform velocity computations
2. Add test cases for any source code associated with the new command-line user interface
3. Create a build.xml target that will allow Kinetic to be called with the `:Ant kinetic` command
4. Add comments to all of the Java classes and methods in Kinetic, KineticTest, and AllTests

Summary of the Required Deliverables

This assignment invites your team to submit one printed version of the following files:

1. A description of the input(s), output(s), and behavior of the basic Eclim commands
2. A description of the output(s) and behavior of Vim's implementation and testing commands
3. The first and final coverage reports produced by the JaCoCo coverage analysis tool
4. A discussion of the defect that you found in the Kinetic Java class
5. The final version of the Kinetic, KineticTest, and AllTests Java classes
6. Screenshots demonstrating that your program and test suite run correctly

You must also ensure that the instructor has read access to your Bitbucket repository that is named according to the convention `cs290F2013-lab3-team`*k*, with *k* representing the number of your assigned team. Please ensure that your team does not modify the source code in the Kinetic repository — instead, it is your responsibility to make a new repository that will contain your team's version of Kinetic. Each team should submit the appropriately documented and tested source code for Kinetic — you must turn in all of the Java source code that you created or modified for this assignment. Please see the instructor if you have questions about these deliverables.