

Find All References in Photran - README

1. Installation Instructions

Originally the project was intended to be a separate package that would make the additional functionality very modular. However, the nature of our project made it difficult to do so. Dependencies were one problem, in that we had circular references when our code in core.vpg depended on code in ui.vpg. Another problem with modularity was the need to add lines to the plugin.xml files. Ultimately we decided that integrating our code into existing packages was the best design decision. Unfortunately this also makes installation somewhat difficult.

Environment

The following describes the environment in which our code has been built and tested. While java is supposed to be cross platform and other environment settings may work, they cannot be guaranteed.

Eclipse For RCP/Plugin Development.

- Version: 3.3.1
- Build id: M20070921-1145

CDT for Eclipse

- Tag Name: CDT_4_0_1
- Available from dev.eclipse.org/cvsroot/tools/
- More instructions are available from the Photran Install Guide. See <http://www.eclipse.org/photran/newcontributors.php> for the link.

Photran for Eclipse

- Build as of December 3, 2007 at 12PM CST.
- Available from dev.eclipse.org/cvsroot/technology
- More instructions are available from the Photran Install Guide. See <http://www.eclipse.org/photran/newcontributors.php> for the link.

JDK and JRE 1.6.0_02

- The code was tested with JRE 1.6.0_02 on a linux machine (Fedora Core 4.) We have tested that the functionality works correctly in Windows or Mac environments when a Photran is correctly configured or installed. However, while java itself is cross platform, we know that there have been issues with Photran working in Windows.

In particular, the tests have been problematic due to line-break incompatibility. Our two automated tests (described in the Testing section below) are susceptible to this problem because windows uses two characters for line-breaks while linux uses only one. This means that the offset indicators used in our tests will be inaccurate if the development system is switched. This problem could be resolved if PhotranTokenRef could use return line numbers instead of a straight byte offset.

Adding Sources to the Workspace

Because the code is so well integrated into the existing Photran source, overriding the existing projects with those in our repository is the easiest method for installation. If you have the environment described above, this shouldn't be a problem.

1. Remove the two projects org.eclipse.photran.core.vpg.tests and org.eclipse.photran.ui.vpg from the Workspace.

2. Checkout the two projects by the same name from <http://csil-projects.cs.uiuc.edu/svn/fa07/cs427/mixed/Fig-Guava/Project/trunk/>
3. Doing "Clean All" and "Build All" is strongly recommended at this point to ensure all dependences are resolved correctly during the build, although the code may compile correctly without doing "Clean All".

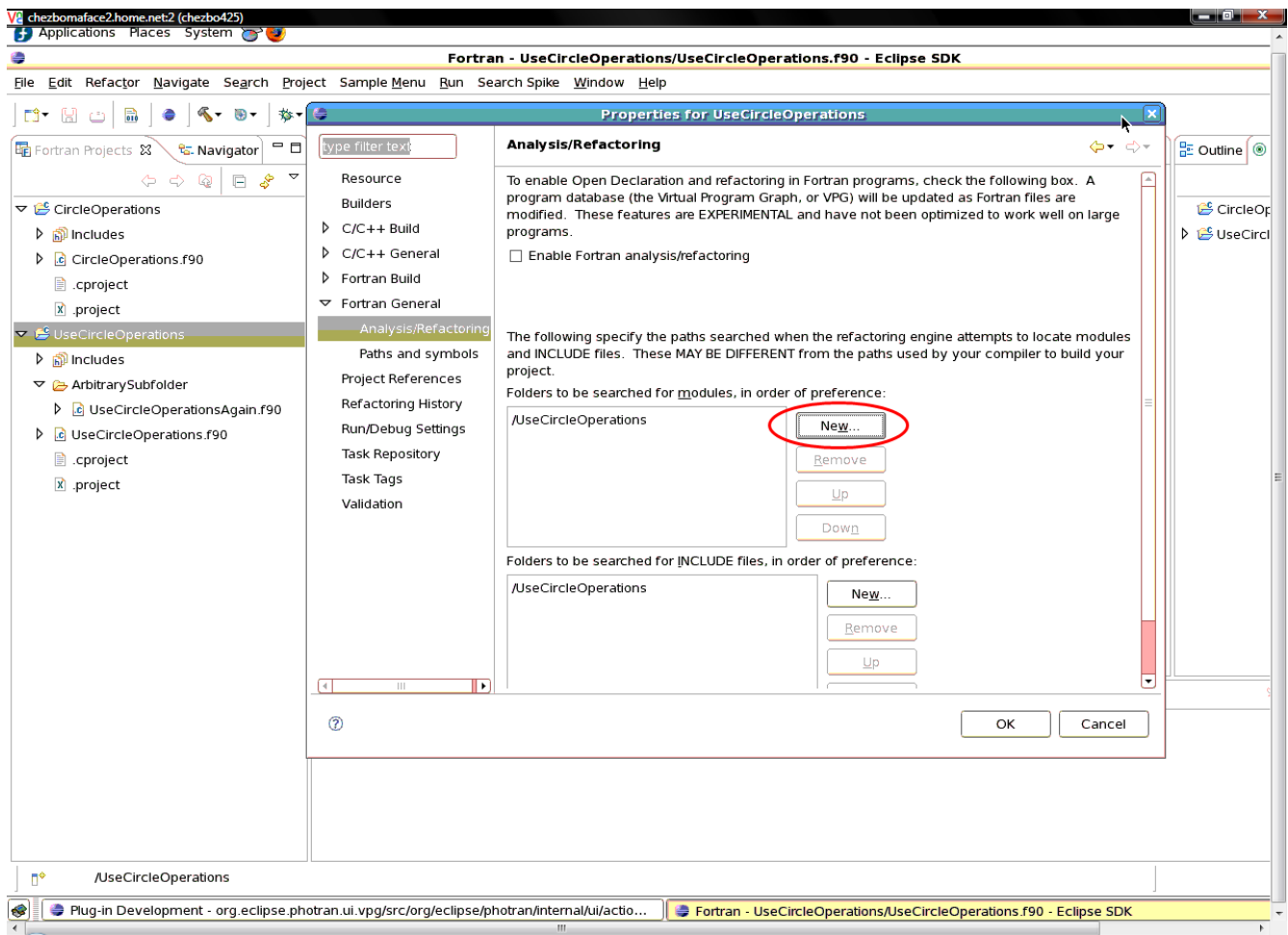
2. Using the Project

Some features of "Find All References" include the following:

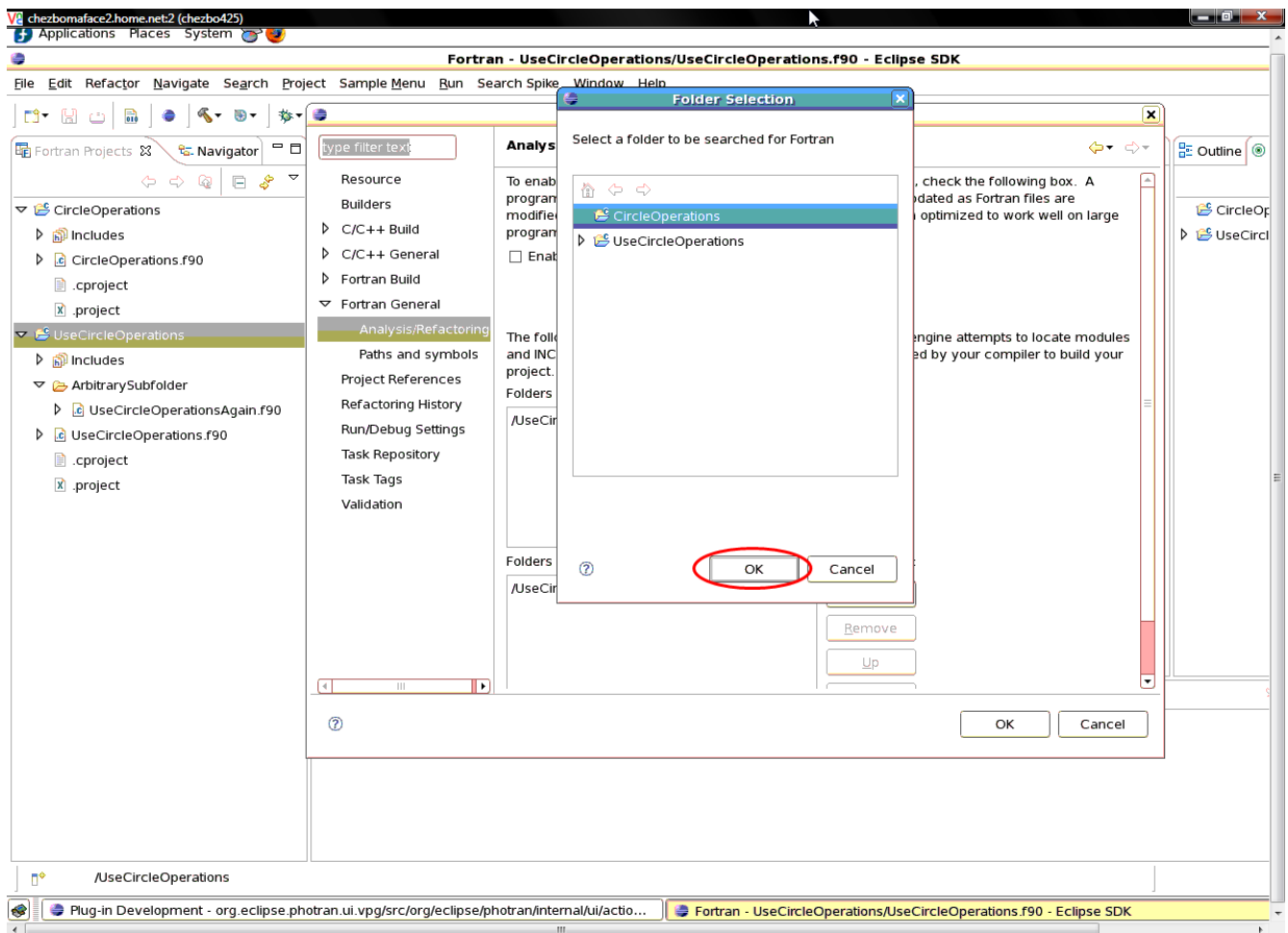
- The options for searching are available by choosing "Search", then "References" and also through an editor context menu.
- All search scopes are available, which include searching for an entity in a file, project and workspace.
- Clickable search results are displayed in a search view.
- The search results can be presented in a tree or list view.
- The search results are displayed grouped by the appropriate project.
- The search results are highlighted in the code.
- The search results can be iterated through using the arrows on the search view.
- Double-clicking on a search result takes you to the related reference in the code.

1. Setting up Searching

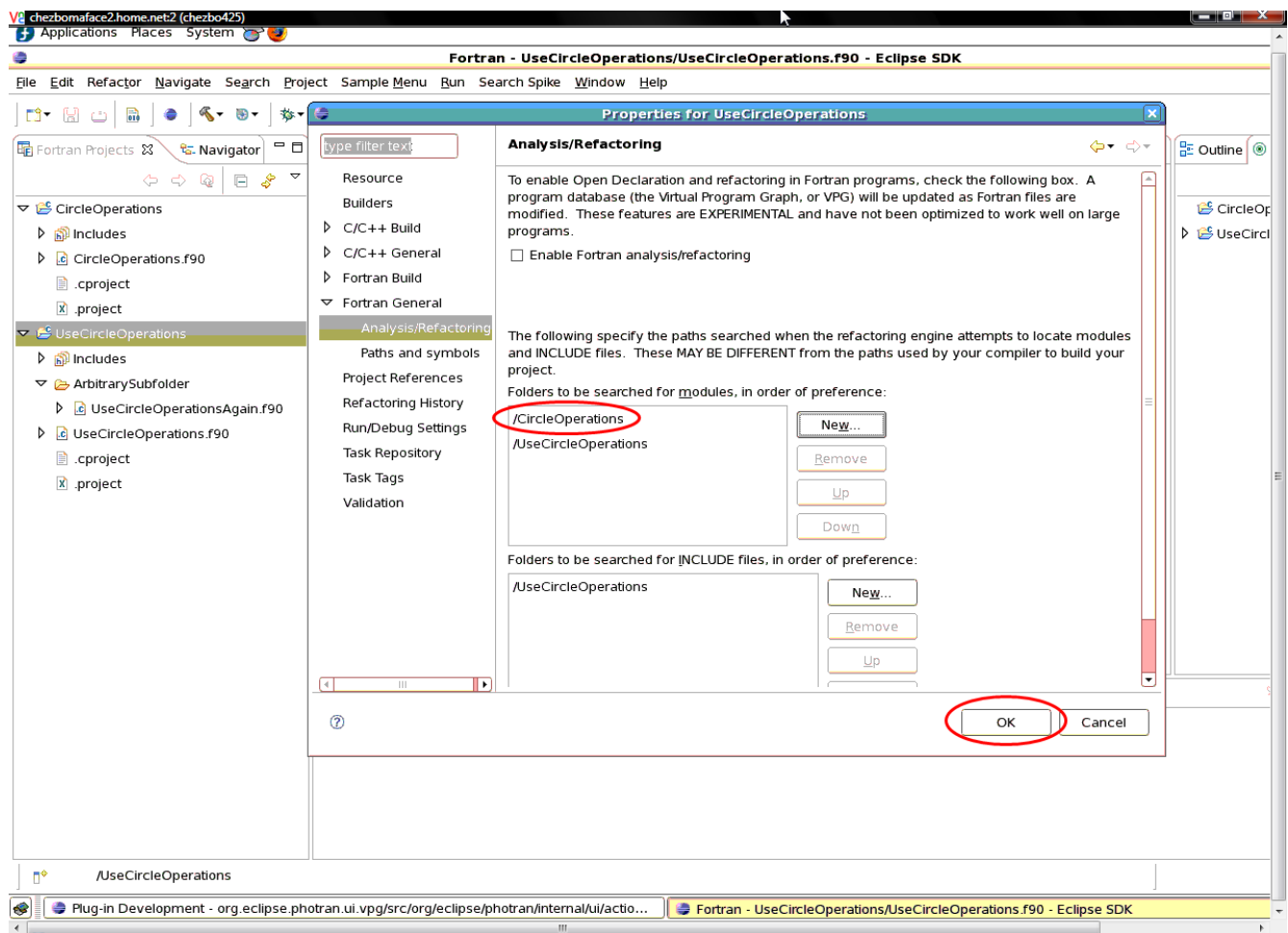
1. If your search will span multiple projects, you must set up which files to search. Start by going to Properties on your project, choose Fortran General and then Analysis/Refactoring.
2. Click on the "New" button to add in the additional folders to be searched for modules.



3. Once this dialog opens, choose the additional resource to be searched. In this example, we are adding in the "CircleOperations" module. Then click ok.



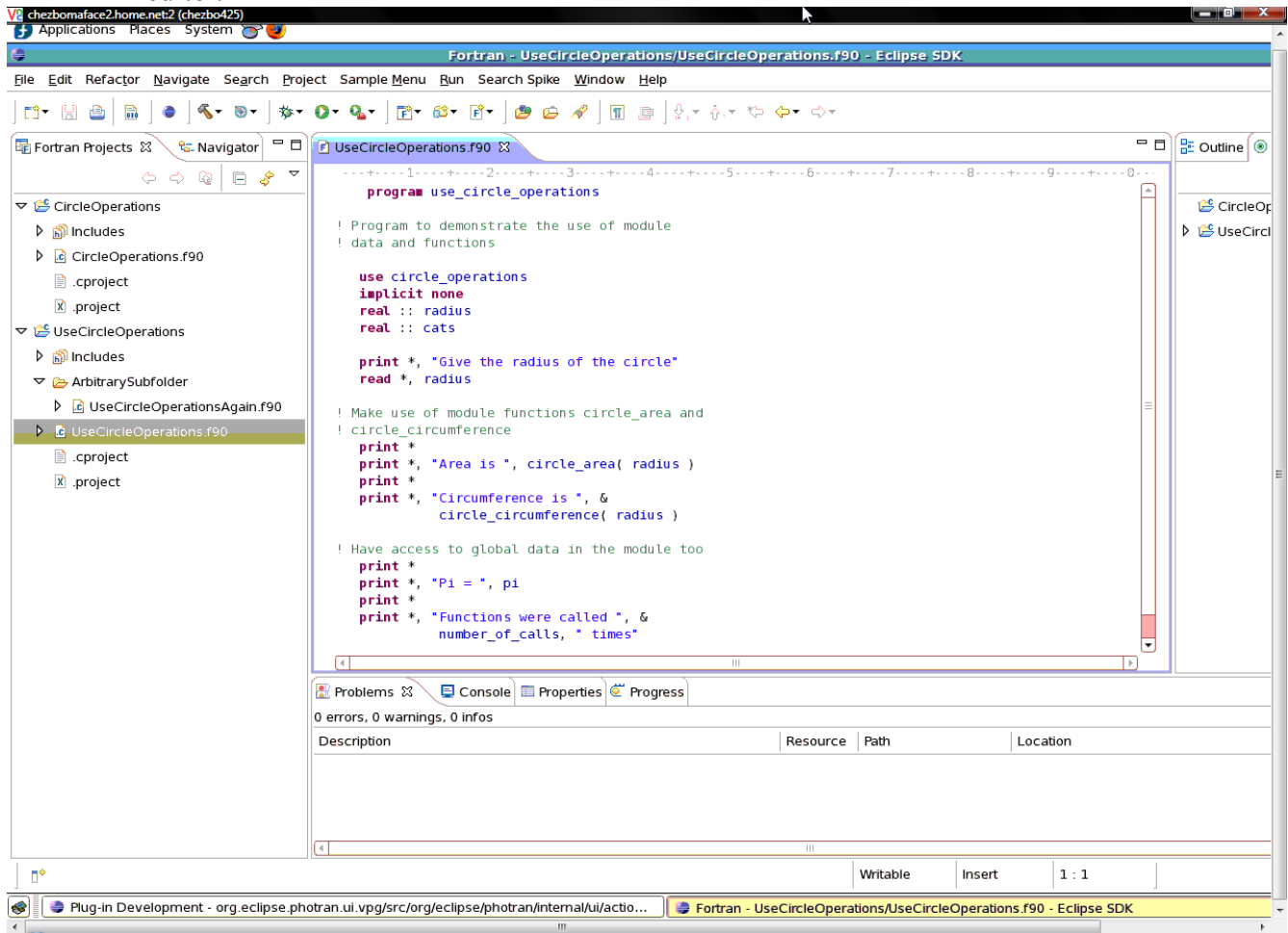
4. You should see it added in the "Folders to be searched for" list.



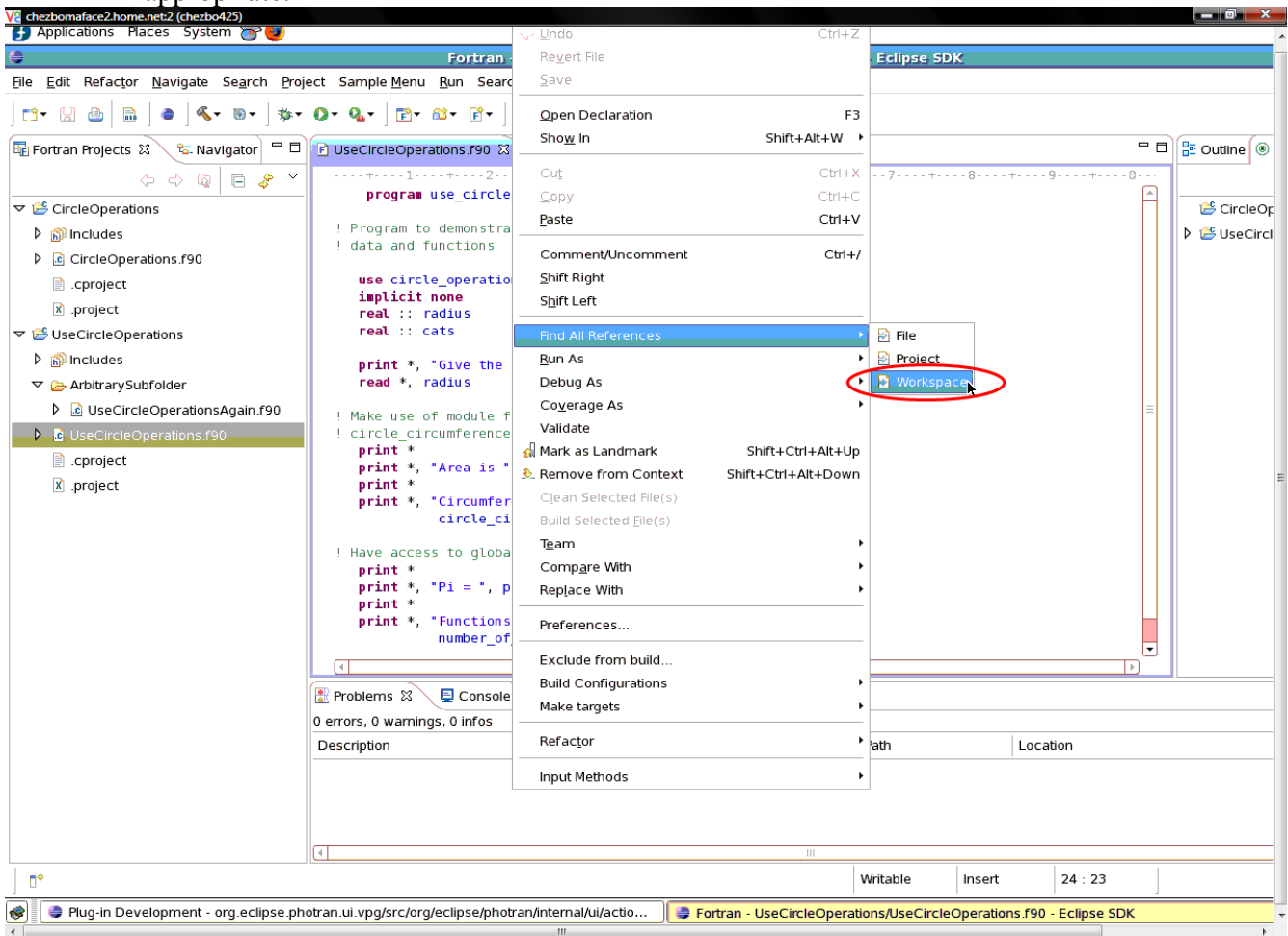
5. Click ok to exit the dialog.

2. Using the Find All References feature in Photran

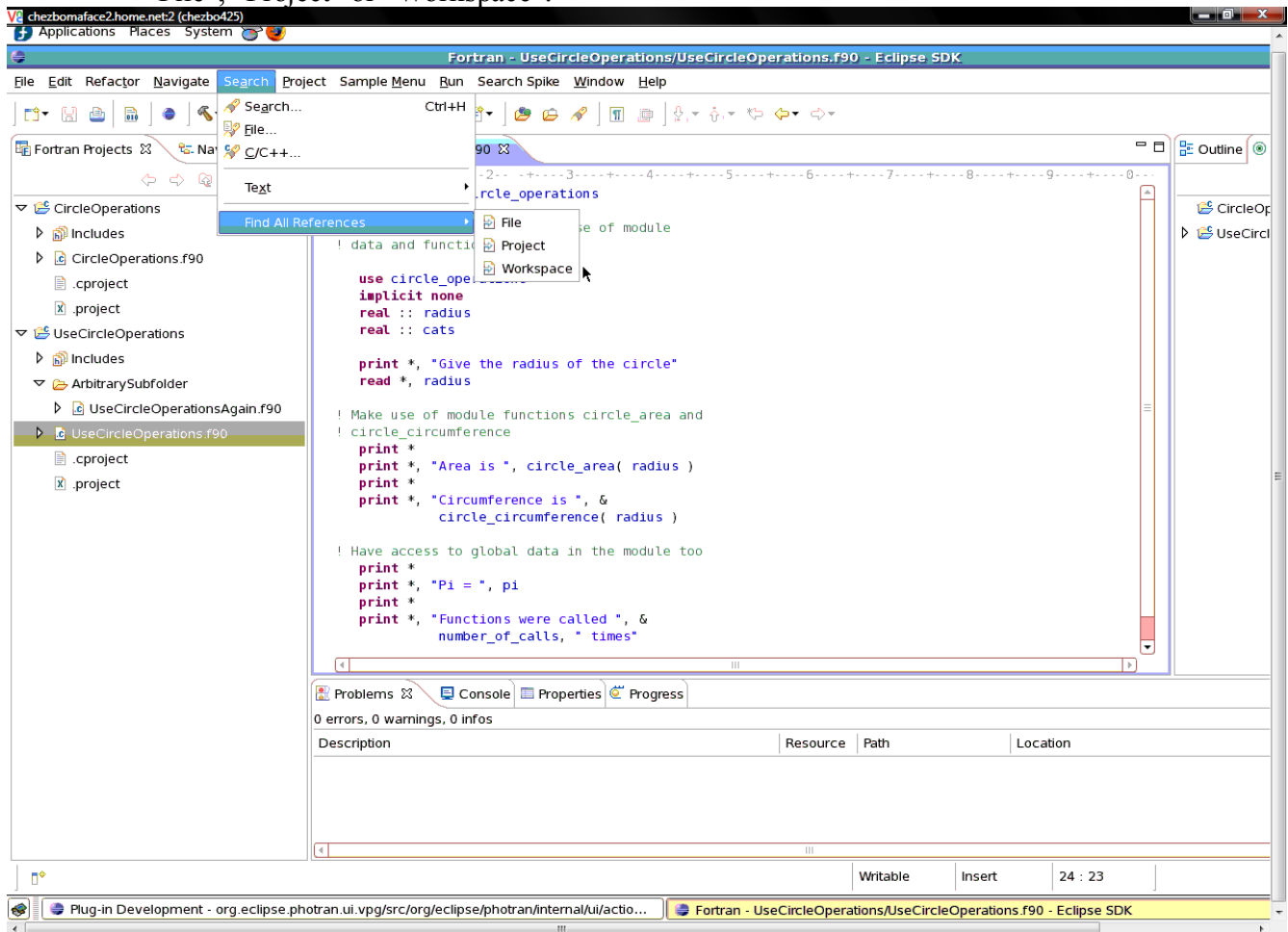
1. Open your Fortran source file that contains the entity you want to search for in the Photran editor.



2. Place your mouse in the text of the declaration, and right-click. From this menu you will choose "Find all References" and then choose "File", "Project" or "Workspace" as appropriate.

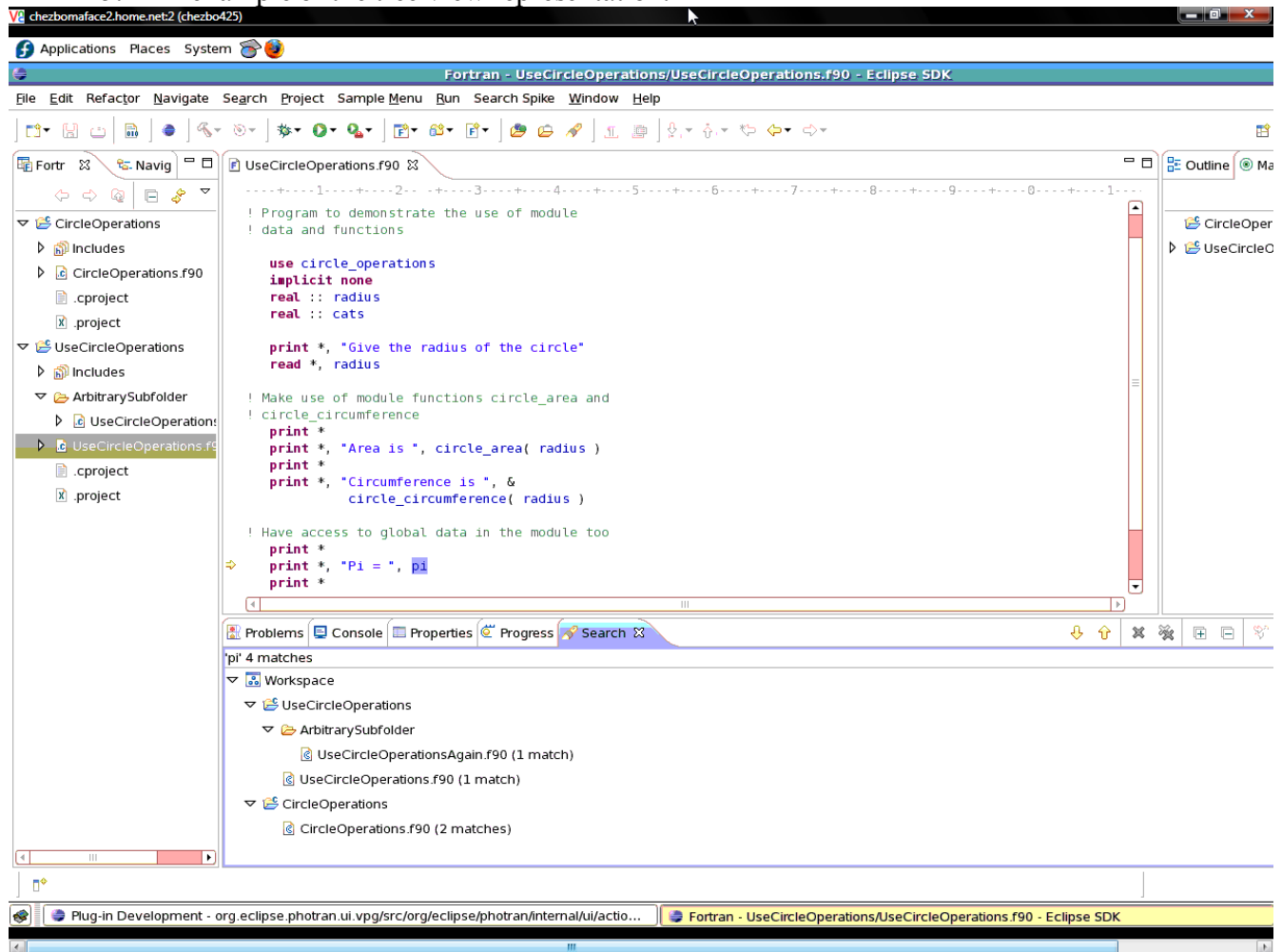


3. You can also access the search menu by choosing the "Search" menu item and then choosing "File", "Project" or "Workspace".

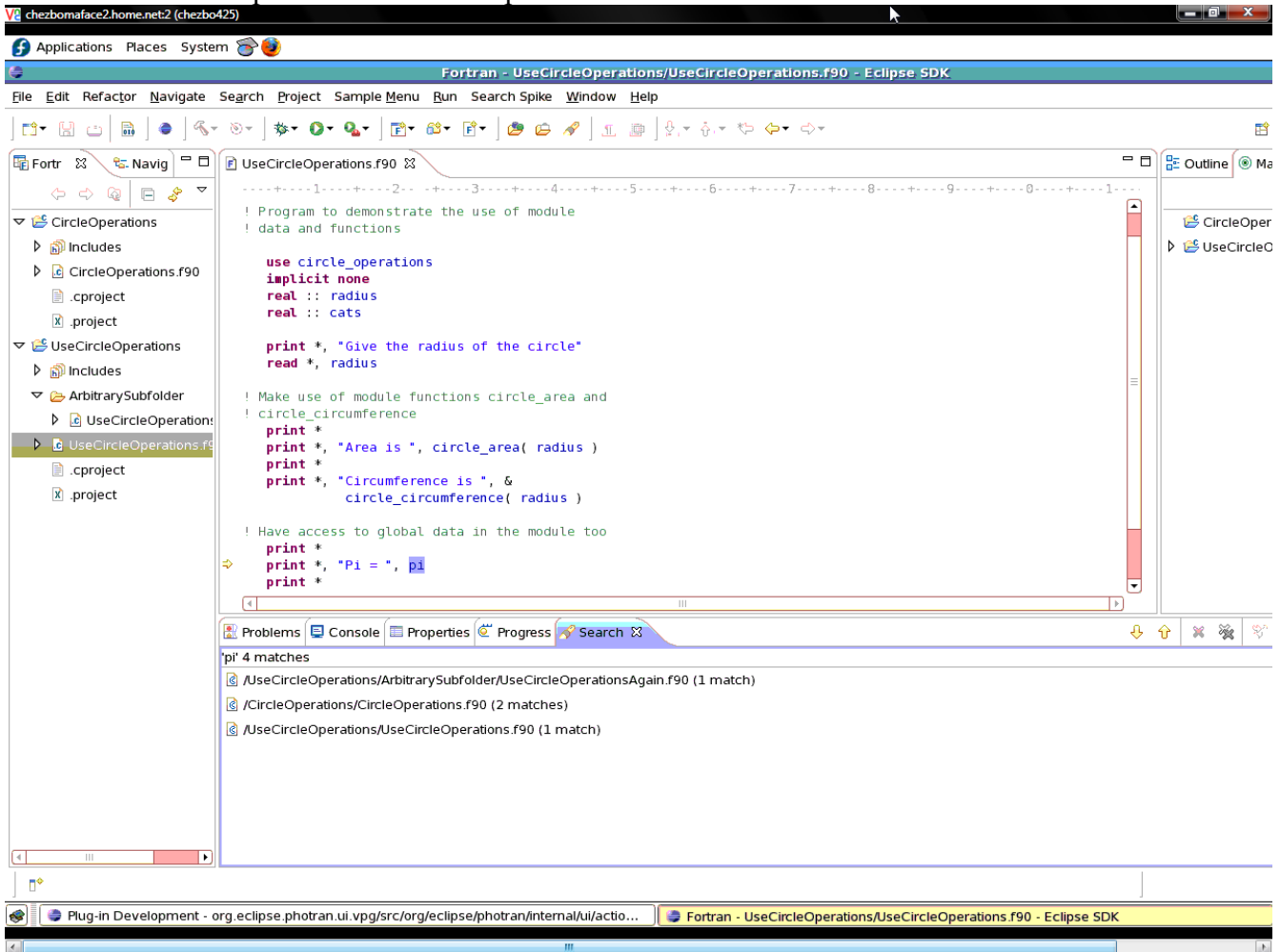


4. Once the search action has completed, you will see a search view shown, where your results will be displayed in a list or tree view. The search view will show how many references were found and display them in groups of the resources they were found in.

5. An example of the tree view representation.



6. An example of the list view representation.



7. You will also notice that the matches are highlighted, and by using the arrows provided in the search view, you can scroll through the list of matches that were returned. In addition, double-clicking on a reference will take you to the related code.

4. Testing Instructions

The testing of the "Find All References" functionality contains automated and manual tests. Due to line spacing differences, the tests will only run successfully on the Linux platform.

Automated tests were created prior to development to determine or design the necessary functionality for finding all references of a specific parameter. The automated tests for the Find All References functionality supplement the pre-existing automated tests for the Photran editor in the `org.eclipse.photran.core.vpg.tests` project. `FileReadingBaseTestFramework` was refactored to be able to load files easier. This code is not a test but allowed code reuse.

Automated Unit Tests

Location: <https://csil-projects.cs.uiuc.edu/svn/fa07/cs427/mixed/Fig-Guava/Project/trunk/org.eclipse.photran.core.vpg.tests/src/org/eclipse/photran/internal/core/tests>

ReferenceSearchTest.java

An automated test to search for a specific token type (ie. variable or subroutine) in the 3 scope options of file, project, and workspace. The test verifies all expected locations of the variable or subroutine calls against the search results.

VPGInforrmationTest.java

This automated test verifies the ability to locate references to a variable within a Fortran file. Manual testing helps discover defects related to the usability testing and GUI testing area. While performing manual tests the software application can be validated whether it meets the various standards defined for effective and efficient usage and accessibility.

Manual Tests

Location: <https://csil-projects.cs.uiuc.edu/svn/fa07/cs427/mixed/Fig-Guava/Projecttrunk/Docs>

ContextMenuManualTest.txt

This test demonstrates how to use the popup menu option in the Photran editor for initiating the Find All References functionality. The range of the test covers the file, project, and workspace search scopes.

MenuBarManualTest.txt

This demonstrates how to use the menubar in the Photran editor to initiate the Find All References functionality. The range of the test covers the file, project, and workspace search scopes.

3. TODO Items

1. Merge the i18n providers
2. Remove unnecessary i18n keys