FORTRAN KEYWORD CHECKING

University of Illinois Urbana Champaign

CS 427 Software Engineering

Bhinav Sura
Graduate

Harini Vaidhyanathan
Graduate

Harshit Kharbanda
Graduate

Todd Tomashek
Graduate

Vishnu Nair
Graduate

Adeet Shah
Graduate

Caius Brindescu
TA Advisor

Table of Contents

1. Abstract

Fortran allows the programmer to declare variables, subroutines and modules having the same name as a keyword in Fortran. This is typically not a good programming practice as it makes programs difficult to understand. The plugin marks all these variables, subroutines and modules as warnings and allows the user to use Photran's rename refactoring to rename them. This allows the users to make their Fortran programs more understandable.

2. Project Design and Architecture

i. High Level Overview

The plugin gets the file resource object of the file currently open in the editor. It then creates an Abstract syntax tree (AST) for the Fortran program in the file. All nodes of this AST are traversed to determine the variable names, module names and the subroutine names, which are also Fortran keywords. Following this, a warning marker is generated for these elements of the program. The warning marker is a custom marker created specifically for the plugin. The user is allowed to view these warnings in the Problems menu of eclipse, she can then chose to quick fix these warnings. The quick fix will redirect the user to Photran's rename refactoring which can be used to rename these elements throughout the project.

ii. KeyWordCheckAction.java

This is the main class that contains the `run` function for the plugin. The `run` function first gets the file resource and then calls `deleteMarkers` on the file. This is done so that the new instance of `run` does not mark the same warnings again. Once all the warnings are deleted, `tokensOfIllegalIdentifiers` is called to identify all the tokens in the program, which have the same name as Fortran keywords. This function uses the `getUpdatedAST` function to get the updated AST of the file. `getUpdatedAST` uses an instance of the the lexer and the parser provided by Photran to generate the AST for the Fortran program. The plugin chooses to regenerate the AST everytime instead of using the methods provided by the Photran VPG like `acquireTransientAST` and `acquirePermanentAST` so that the plugin can get an updated AST while the file is being edited. This allows the reconciler to get the current AST of the file while the file is being edited by the user and mark the warnings on this partially edited but unsaved file. The VPG is not updated until the file is saved and hence the AST provided by the VPG is stale. On getting the updated AST, `tokensOfIllegalIdentifiers` traverses the AST and puts all the tokens that violate the KeywordCheck policy in a HashSet. These tokens are then sent to `generateMarkers` to generate a warning icon for the offending tokens. `generateMarkers` creates new markers with helpful warning messages and attaches these markers to the tokens determined by `tokensOfIllegalIdentifiers`.

iii. PhotranKeywords.java

Photran has a class called as KeywordScanner.java but it has list of keywords which is private and not accessible outside the class. This class contains a collection of Fortran 2008 keywords which are basically same as that contained in the KeywordScanner.java class. It also has a method is FortranKeyword which returns if the variable/module name is a Fortran Keyword or not.

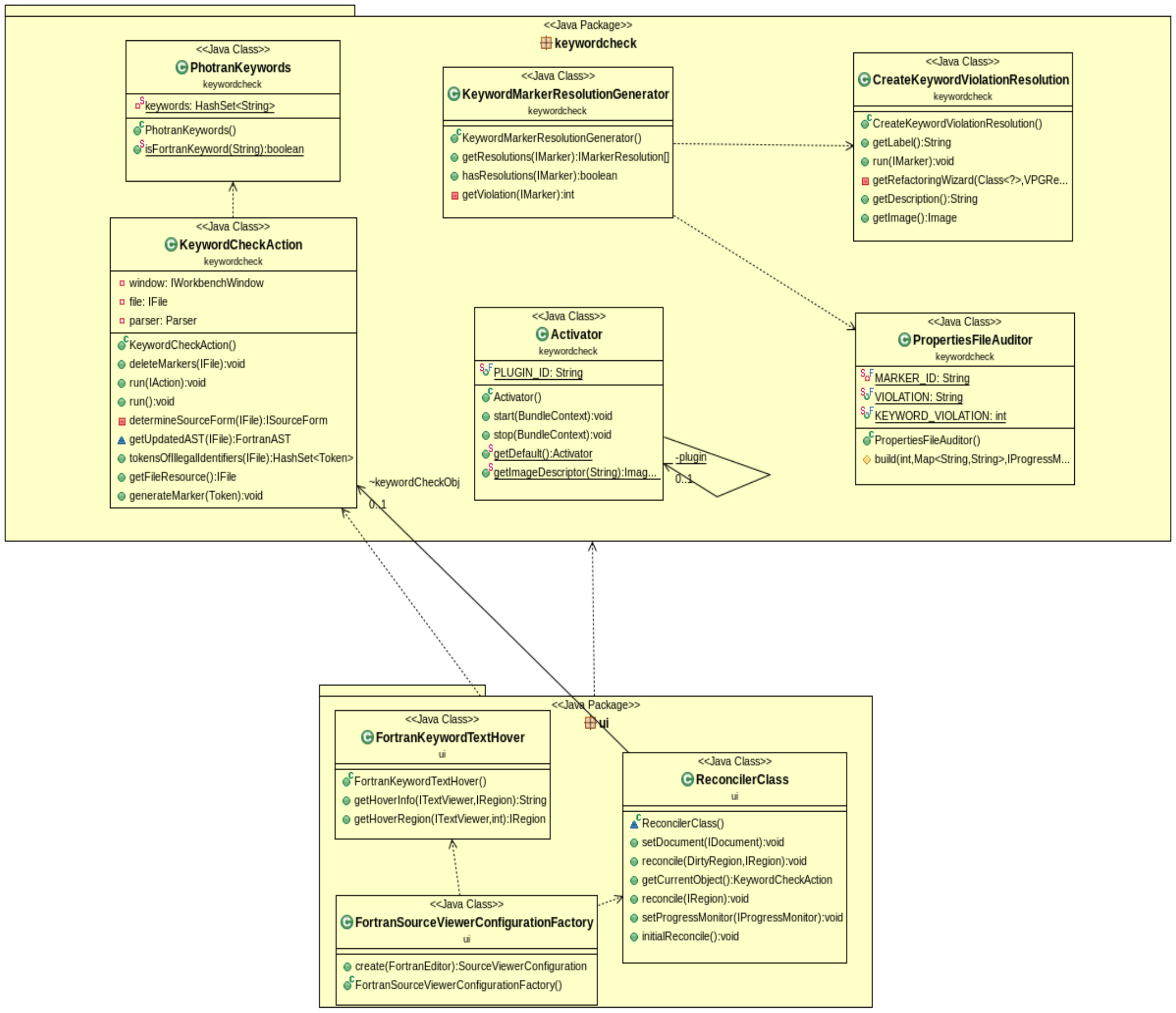
iv. KeywordMarkerResolutionGenerator.java

This class generates a marker resolution for keyword check. The resolution for the marker is created by calling the CreateKeywordViolationResolution.java class.

v. CreateKeywordViolationResolution.java

This class creates a marker resolution by linking the quick fix on warnings from the problems tab to the Rename refactoring in Photran. It enables renaming the variable used as a Fortran keyword by opening the rename refactoring wizard.

vi. Class Diagram


<<Java Package>>
keywordcheck
<<Java Class>>
PhotranKeywords
keywordcheck
keywords: HashSet<String>
PhotranKeywords()
isFortranKeyword(String):boolean
<<Java Class>>
KeywordMarkerResolutionGenerator
keywordcheck
KeywordMarkerResolutionGenerator()
getResolutions(IMarker):IMarkerResolution[]
hasResolutions(IMarker):boolean
getViolation(IMarker):int
<<Java Class>>
CreateKeywordViolationResolution
keywordcheck
CreateKeywordViolationResolution()
getLabel():String
run(IMarker):void
getRefactoringWizard(Class<?>,VPGRe...
getDescription():String
getImage():Image
<<Java Class>>
KeywordCheckAction
keywordcheck
window: IWorkbenchWindow
file: IFile
parser: Parser
KeywordCheckAction()
deleteMarkers(IFile):void
run(IAction):void
run():void
determineSourceForm(IFile):ISourceForm
getUpdatedAST(IFile):FortranAST
tokensOfIllegalIdentifiers(IFile):HashSet<Token>
getFileResource():IFile
generateMarker(Token):void
<<Java Class>>
Activator
keywordcheck
PLUGIN_ID: String
Activator()
start(BundleContext):void
stop(BundleContext):void
getDefault():Activator
getImageDescriptor(String):Imag...
-plugin
0..1
<<Java Class>>
PropertiesFileAuditor
keywordcheck
MARKER_ID: String
VIOLATION: String
KEYWORD_VIOLATION: int
PropertiesFileAuditor()
build(int,Map<String,String>,IProgressM...
~keywordCheckObj
0..1
<<Java Package>>
ui
<<Java Class>>
FortranKeywordTextHover
ui
FortranKeywordTextHover()
getHoverInfo(ITextViewer,IRegion):String
getHoverRegion(ITextViewer,int):IRegion
<<Java Class>>
ReconcilerClass
ui
ReconcilerClass()
setDocument(IDocument):void
reconcile(DirtyRegion,IRegion):void
getCurrentObject():KeywordCheckAction
reconcile(IRegion):void
setProgressMonitor(IProgressMonitor):void
initialReconcile():void
<<Java Class>>
FortranSourceViewerConfigurationFactory
ui
create(FortranEditor):SourceViewerConfiguration
FortranSourceViewerConfigurationFactory()

3. Future Plans

- The plugin will be available in Git-Hub as open source download.
- Alternatively, it is possible to convert this plugin to a general code smell detector for Photran.
- The plugin can give an appropriate (unused) alternate name for renaming.
- The plugin can be an option selectable from the edit/preferences menu.

4. Installation Instructions

To run the plug-in, you need to have the following components installed:

1. Eclipse
2. JavaSE (JDT and/or JRE)
3. Photran and CDT
4. And Finally the Plug-in that we provide

The following presents a step-by-step outline of how each component can be installed. In case you have any of the above components already installed, you can skip the corresponding section.

i. Installing Eclipse and JavaSE

a. Installing Eclipse is relatively easy, but does involve a few steps and software from at least two different sources. Eclipse is a Java-based application and, as such, requires a Java runtime environment (JRE) in order to run.

b. Regardless of the Operating System that you would run Eclipse on, you require a Java Virtual Machine. You can download either Java Development Kit (JDK) or Java Runtime Environment (JRE) using the links provided below. Although you can do away with either of them, it's recommended that you download both of them. (You might JDK in future if you plan on doing Java Development).

   - If you are using Windows, you might probably already have JRE installed.
   - If you are using Linux, then follow the specific instructions given here.
   - ***Note***: GCJ won't work.

   Oracle JDK/JRE

c. After you have installed JavaSE on your machine, it's time to install eclipse. The latest stable version of Eclipse at the time this writing is version 4.2.1 Juno. Download Eclipse from the Eclipse Downloads Page.

d. The whole of eclipse will be downloaded as one compressed (.zip, .tar, .gz) file. You will need to decompress it into a folder of your choice. This will enable you to run eclipse from that folder. Optionally you could also create a shortcut for quicker access.

e. For further questions or doubts you can visit the Eclipse Installation Instructions FAQ page.

ii. Installing Photran

At the time of writing, the below section has been compiled referencing updated content from the Archived Releases section of the Eclipse documentation. So the interested reader can further reference here.

The system requirements to have a working copy of Photran include having the most updated classic version of Eclipse. Also CDT 7.0 is required, the installation details for which are mentioned below.

Also, the version of Eclipse should be running a version of Java that is equal to or higher than 1.5. A variant of make is a prerequisite to build and compile FORTRAN applications and this has to be in your system path. Only instructions for configuration in Windows are specified below because almost Linux/Unix systems include this by default.

To install Photran as an end user follows Part 1, else as a developer follow Part 2.

a. Run eclipse and navigate to Help->Install New Software.

b. Add a new location and reference it to the following and there are no constraints on the name. Click ok.

   http://download.eclipse.org/tools/ptp/releases/helios

c. Expand the Fortran Development Tools (Photran) option and select “Photran End-User Runtime”.

d. On a system running Linux or Macintosh, there might be a default Fortran Compiler (Intel or IBM XL), expand “Fortran Compiler Support” and select the appropriate compiler. Also note that unless you are running a Linux distribution, you cannot install the Intel Fortran compiler.

e. After clicking Next click on Finish after agreeing to the license.

f. The above process will automatically install CDT as well.

If you do not have an internet connection, get one soon. In the meantime you can follow the instructions at the link given above to install without an internet connection and also for other troubleshooting details.

If you are a developer and plan to extend the plugin, you’ll have to compile Photran manually. Since Photran reuses some parts of CDT, even CDT has to be compiled. Hence follow the steps given in the guide here to set up Photran and CDT.

iii. Finally the Plug-in

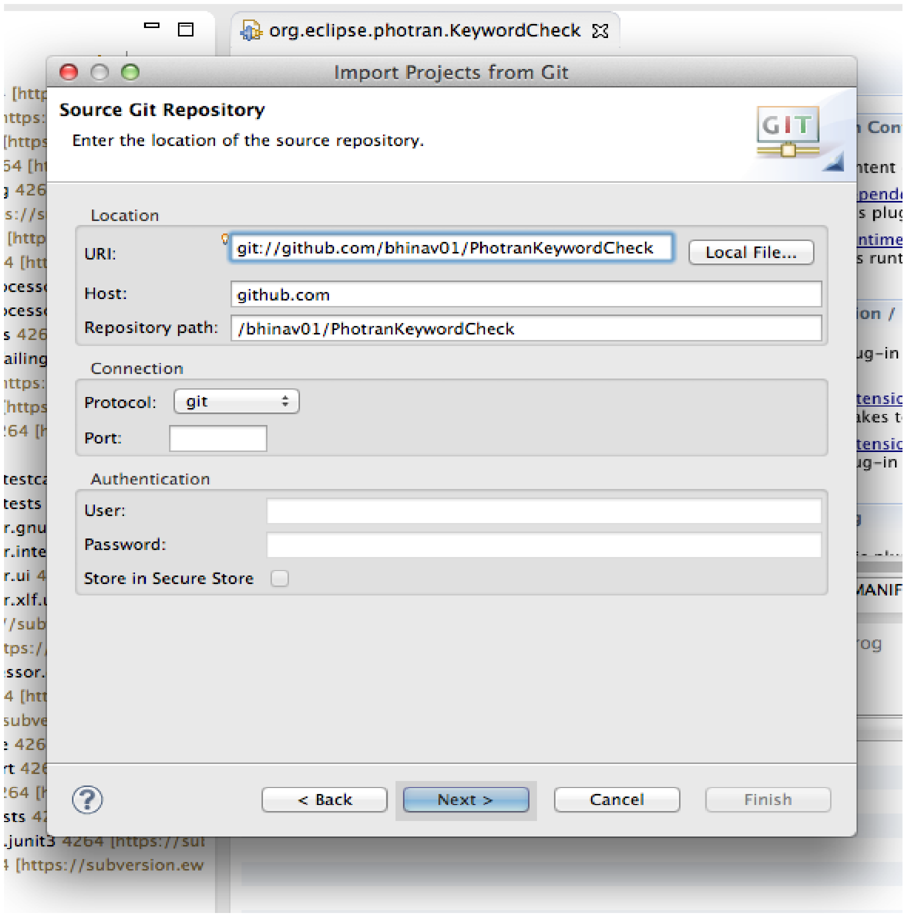
Method I (from Git Repository)

a. Open Eclipse and go to File > Imports…

b. You will find a list of options from where the project can be imported. We have uploaded out project on Github. So you must select the “Git” option from that list. Inside that select the option “Projects from Git” and press *Next*.

c. You will be presented with a list of three options. Select “URI” from that list and press *Next*.

d. The following screen pops-up.



Source Git Repository

Enter the location of the source repository.

Location

URI: git://github.com/bhinav01/PhotranKeywordCheck

Local File...

Host: github.com

Repository path: /bhinav01/PhotranKeywordCheck

Connection

Protocol: git

Port:

Authentication

User:

Password:

Store in Secure Store

e. In the URI field, you need to enter the following link:

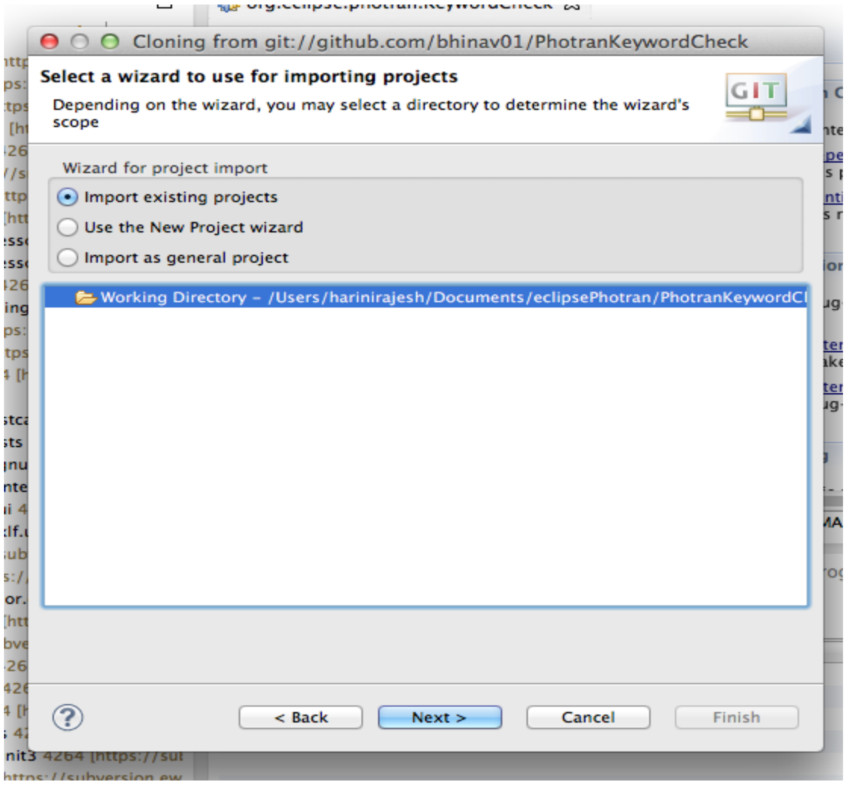
git://github.com/bhinav01/PhotranKeywordCheck

f. Doing this will automatically fill up the remaining blocks. You need not do anything further.

g. A page for Branch Selection will pop-up with master already selected. Keep it as it is and press *Next*.

h. In the next screen you will be asked to select a Local Destination. Enter the name of the directory where you want to import the project into.

i. Now from the list of options select the one which says "Import existing Project" and press *Next*.



Cloning from git://github.com/bhinav01/PhotranKeywordCheck

Select a wizard to use for importing projects

Depending on the wizard, you may select a directory to determine the wizard's scope

Wizard for project import

[✓ Import existing projects]

Use the New Project wizard

Import as general project

Working Directory - /Users/harinirajesh/Documents/eclipsePhotran/PhotranKeywordCl

j. On the next page you select the project you want to import. There is only one project, select that and press *Finish*. And that's it. You are done!!!

Method II (from .zip archive)

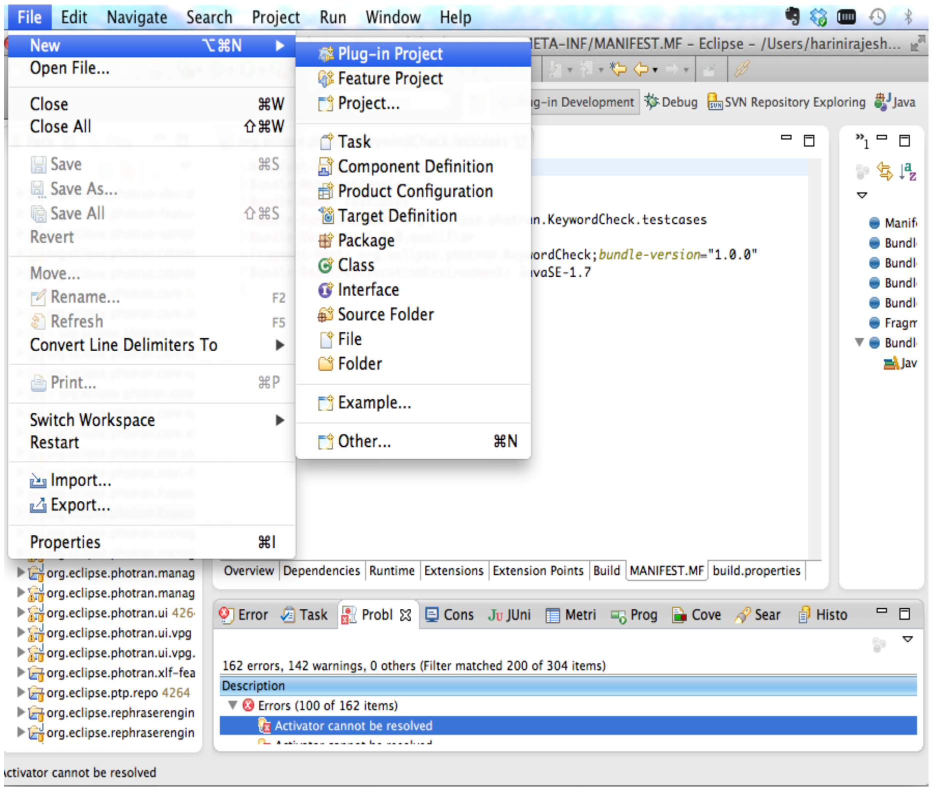
a. Linux/Mac users can use the command the following command to download the zip file from the git repository. This is assuming you have git installed.

"git clone https://github.com/bhinav01/PhotranKeywordCheck"

b. Windows users can navigate to the following link and follow the instructions.

   https://github.com/bhinav01/PhotranKeywordCheck

c. Once the file has been downloaded, unzip it. Open your version of Eclipse and navigate to Eclipse > Plugin Development and select Plugin Project from the expanded list.



```
n.KeywordCheck.testcases
ordCheck;bundle-version="1.0.0"
vaSE-1.7
```

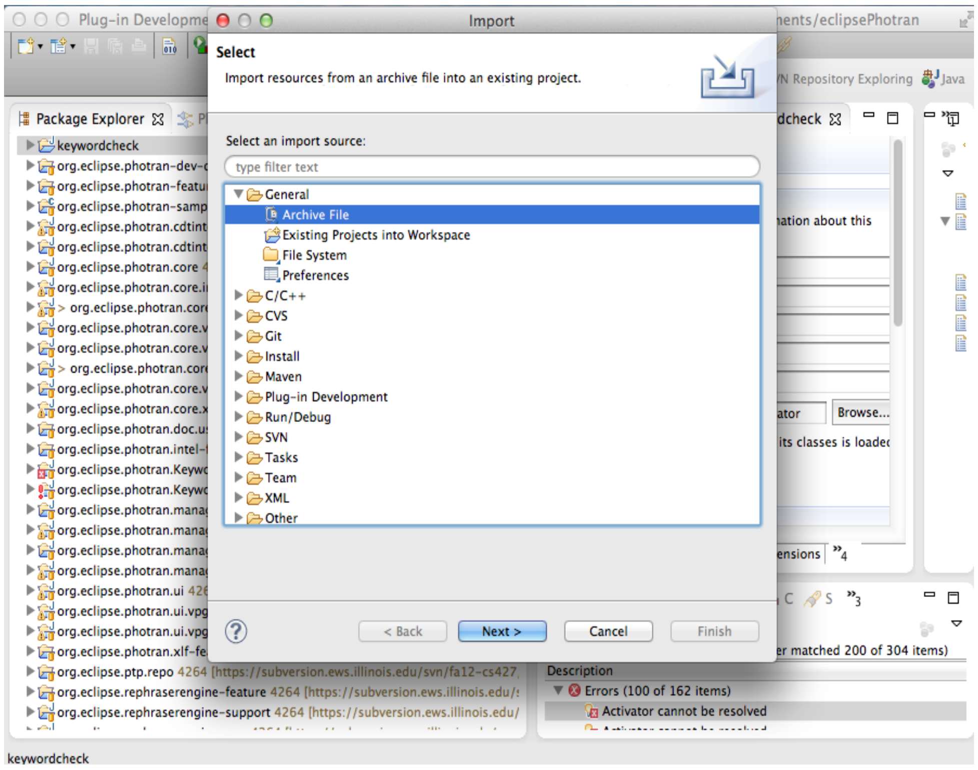
162 errors, 142 warnings, 0 others (Filter matched 200 of 304 items)

Description
Errors (100 of 162 items)
Activator cannot be resolved

ctivator cannot be resolved

d. Name the project as "org.eclipse.photran.KeywordCheck" and all the other options can be the default ones.
e. Click on *Next* and then *Finish*.

f. Now navigate to File->Import and under General select the "File System" option and click *Next*.



Import resources from an archive file into an existing project.

Errors (100 of 162 items)

Activator cannot be resolved

g. In the "From directory" select the folder you just unzipped. The field "Into Folder" has to be populated with the folder name of the Plugin you just created. "...../workspace/org.eclipse.photran.KeywordCheck".

h. Now click on *Finish*.

i. You have a working copy of the plugin now. Build and Run.

Note:

In order to use the plugin, refactoring should be enabled in Photran.

Read this to know more about Photran refactoring: www.eclipse.org/photran/refactoring.php

5. Testing

There are two categories of testing for Keyword Checker plug-in.

- Automated tests
- Manual Tests

i. Automated tests

All the automated tests corresponding to the project are placed in a Fragment Project in

`org.eclipse.photran.KeywordCheck.testcases`

The automated testcases in `org.eclipse.photran.KeywordCheck.tests.illegalkeywords` and `org.eclipse.photran.KeywordCheck.tests.marker` use Markers for running testcases similar to the refactoring testcases in Photran. The fortran programs used for testing are in the folder illegal-keyword-test-code.

For further information about automating testcases for photran refer to Photran developers' guide.

ii. Manual test

The user interface tests are done manually.

If you modify the plugin, perform the following manual tests

Check if the highlighted keywords appear in the Problems view in Photran.

Check if the hover on warning signs shows all the appropriate warnings for the line.

Check if constant editing of files on the editor updates the illegal keywords and warnings in the problems bar.

The illegal keyword should be identified without even saving the file.

6. User Manual

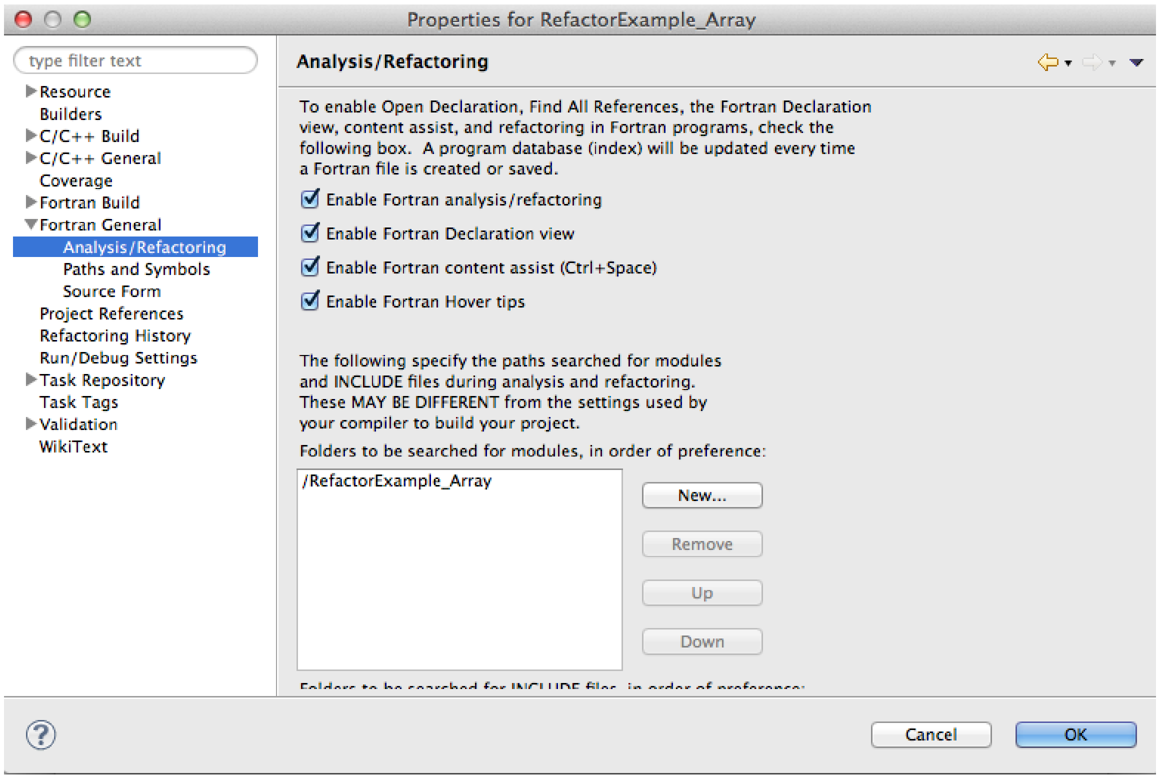
After Eclipse Photran installation, it is necessary to enable refactorings in Photran. To do this,

Step 1: Right-click on your project's folder in the Fortran Project view.

Step 2: Click on Properties

Step 3: Expand Fortran General and select Analysis/Refactoring .

Step 4: Check Enable Fortran analysis/Refactoring. This will add the Refactor menu in the Photran Menu bar.



To enable Open Declaration, Find All References, the Fortran Declaration view, content assist, and refactoring in Fortran programs, check the following box. A program database (index) will be updated every time a Fortran file is created or saved.

- [✓] Enable Fortran analysis/refactoring
- [✓] Enable Fortran Declaration view
- [✓] Enable Fortran content assist (Ctrl+Space)
- [✓] Enable Fortran Hover tips

The following specify the paths searched for modules and INCLUDE files during analysis and refactoring. These MAY BE DIFFERENT from the settings used by your compiler to build your project.

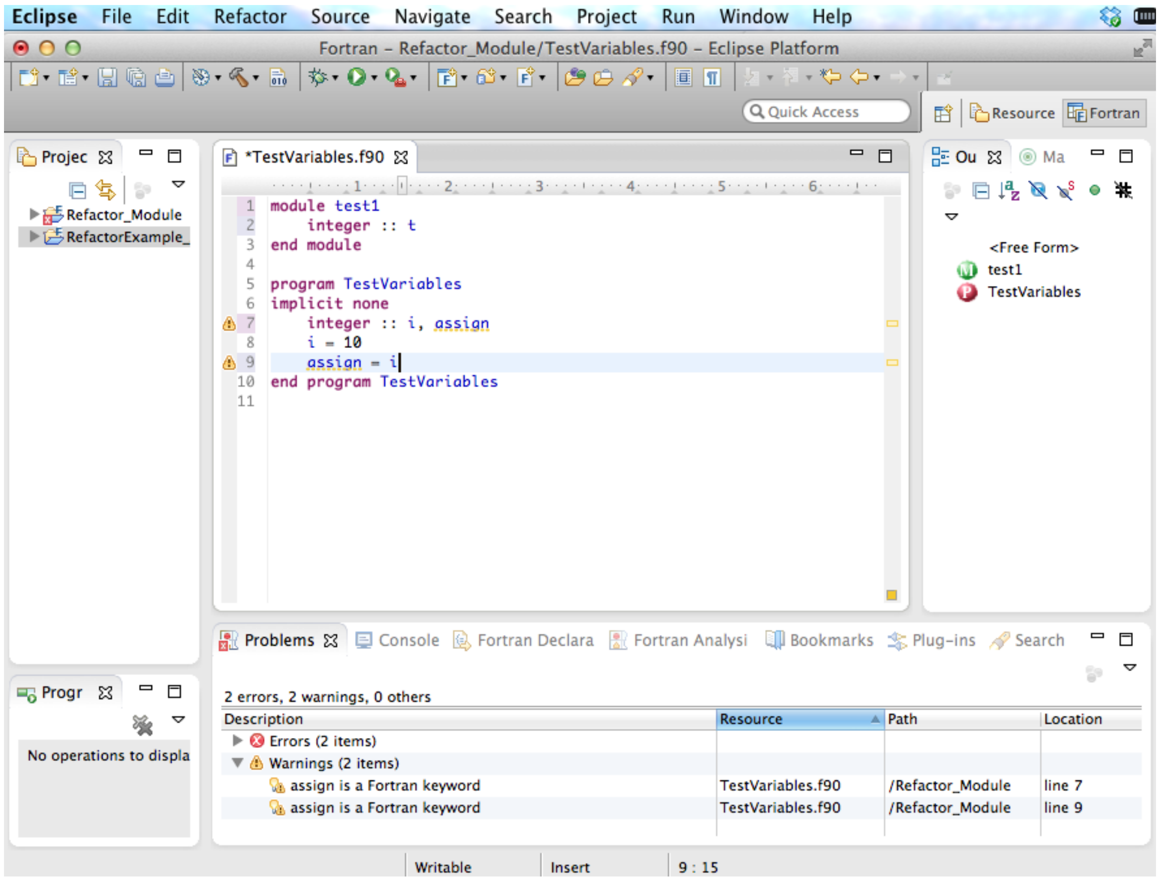
Folders to be searched for modules, in order of preference:

/RefactorExample_Array

This refactoring is to be enabled for each project.

i. How to use Keyword checker

Start typing your Fortran program on the Photran Editor.



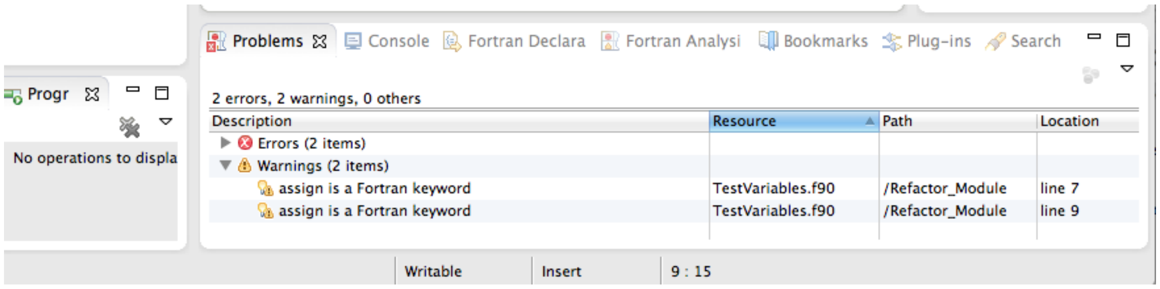
```
module test1
    integer :: t
end module

program TestVariables
implicit none
    integer :: i, assign
    i = 10
    assign = i
end program TestVariables
```

2 errors, 2 warnings, 0 others

Description	Resource	Path	Location
Errors (2 items)			
Warnings (2 items)			
assign is a Fortran keyword	TestVariables.f90	/Refactor_Module	line 7
assign is a Fortran keyword	TestVariables.f90	/Refactor_Module	line 9

The program in the above screen shot uses the keyword “assign” illegally as an identifier.

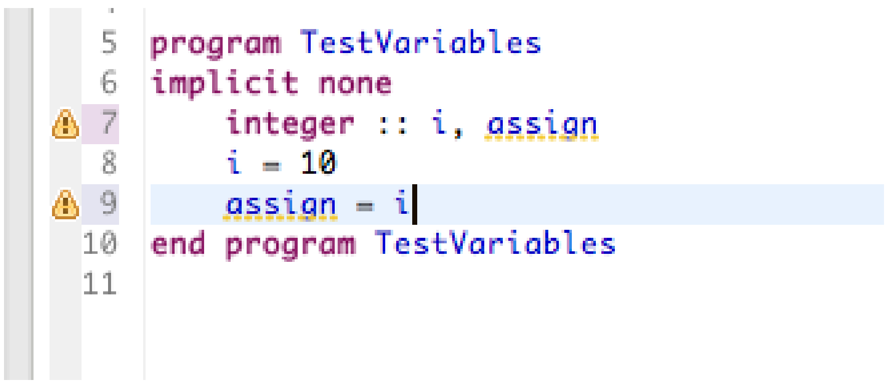
The usage instances of the keyword are highlighted on the editor and are also listed on the warnings pane.



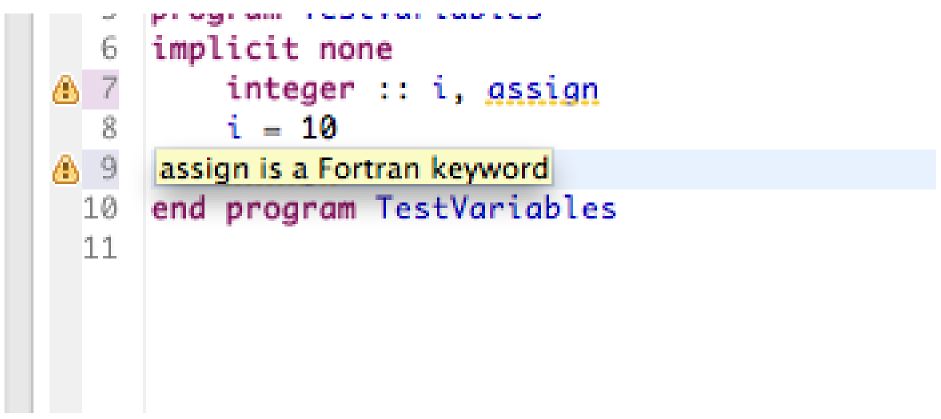
2 errors, 2 warnings, 0 others

Description	Resource	Path	Location
Errors (2 items)			
Warnings (2 items)			
assign is a Fortran keyword	TestVariables.f90	/Refactor_Module	line 7
assign is a Fortran keyword	TestVariables.f90	/Refactor_Module	line 9

Hover over the warning icon on the editor line to see the reason for the warning

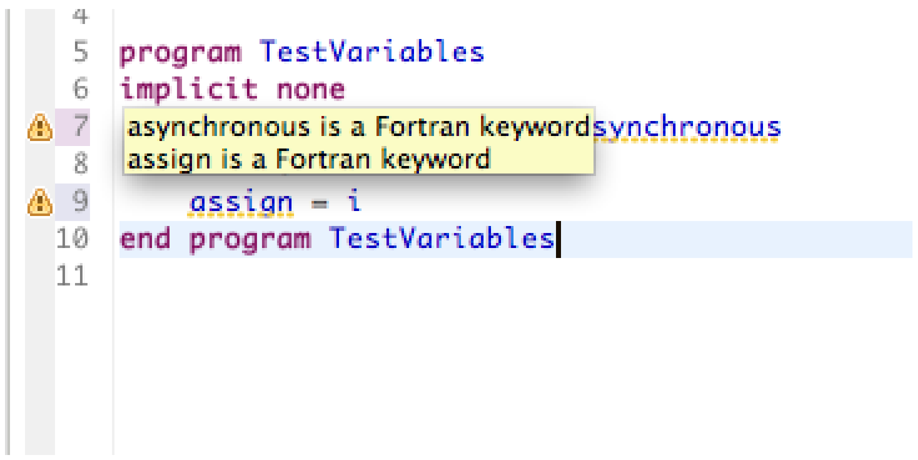


```
program TestVariables
implicit none
    integer :: i, assign
    i = 10
    assign = i
end program TestVariables
```



```
implicit none
    integer :: i, assign
    i = 10
assign is a Fortran keyword
end program TestVariables
```

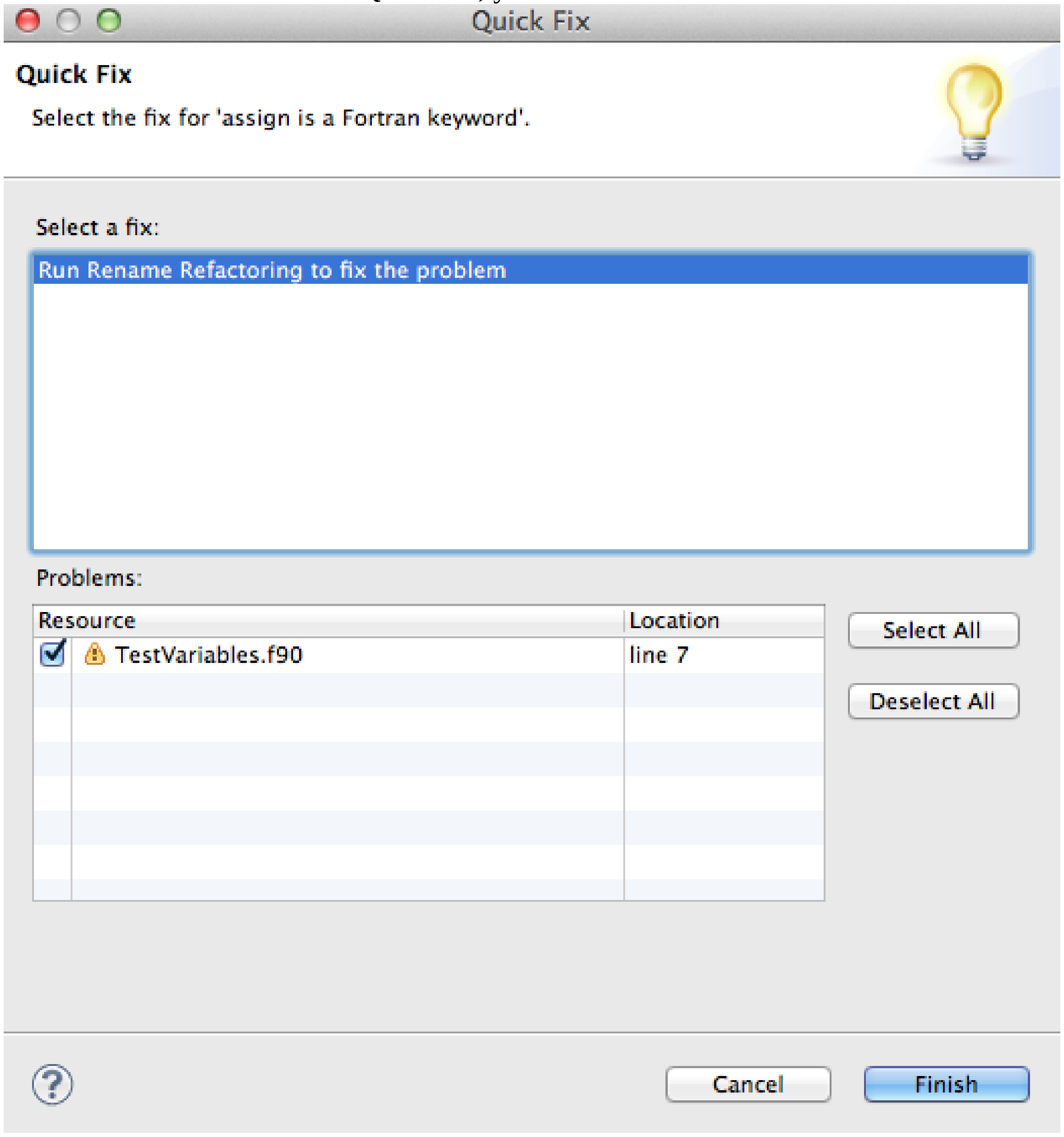
If there are two or more illegal identifiers on the same line you will see all the warnings like -



```
program TestVariables
implicit none
asynchronous is a Fortran keywordsynchronous
assign is a Fortran keyword
    assign = i
end program TestVariables
```

ii. Ways to refactor

1. It is possible to take the strenuous route and manually change all the occurrence of the identifier.
2. Relax! We made it easier for you. Right-click on the variable you need to refactor from the Problems View and select Quick Fix, you will see



Quick Fix

Quick Fix

Select the fix for 'assign is a Fortran keyword'.

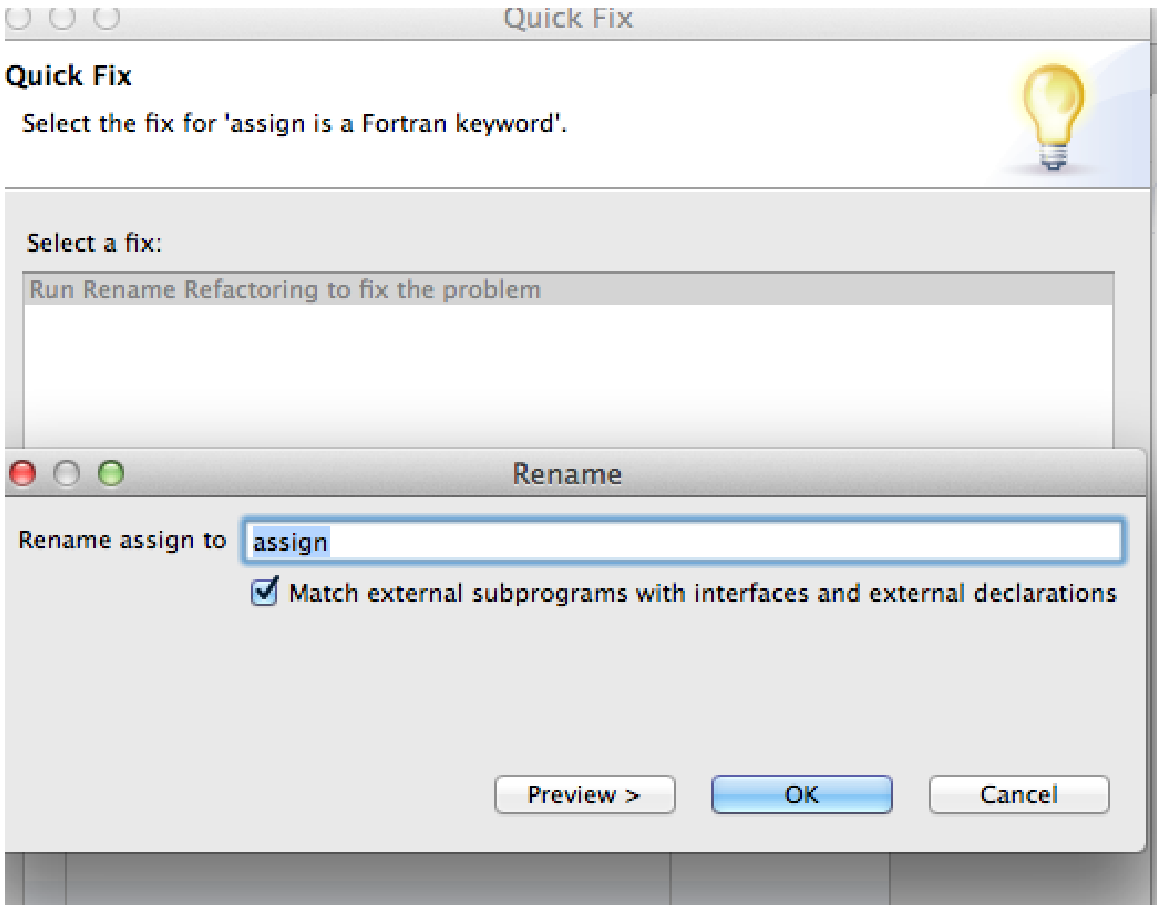
Select a fix:

Run Rename Refactoring to fix the problem

Problems:

Resource	Location
✓ TestVariables.f90	line 7

Click on finish and this will invoke the rename refactor.


Quick Fix

Select the fix for 'assign is a Fortran keyword'.

Select a fix:

Run Rename Refactoring to fix the problem

Rename

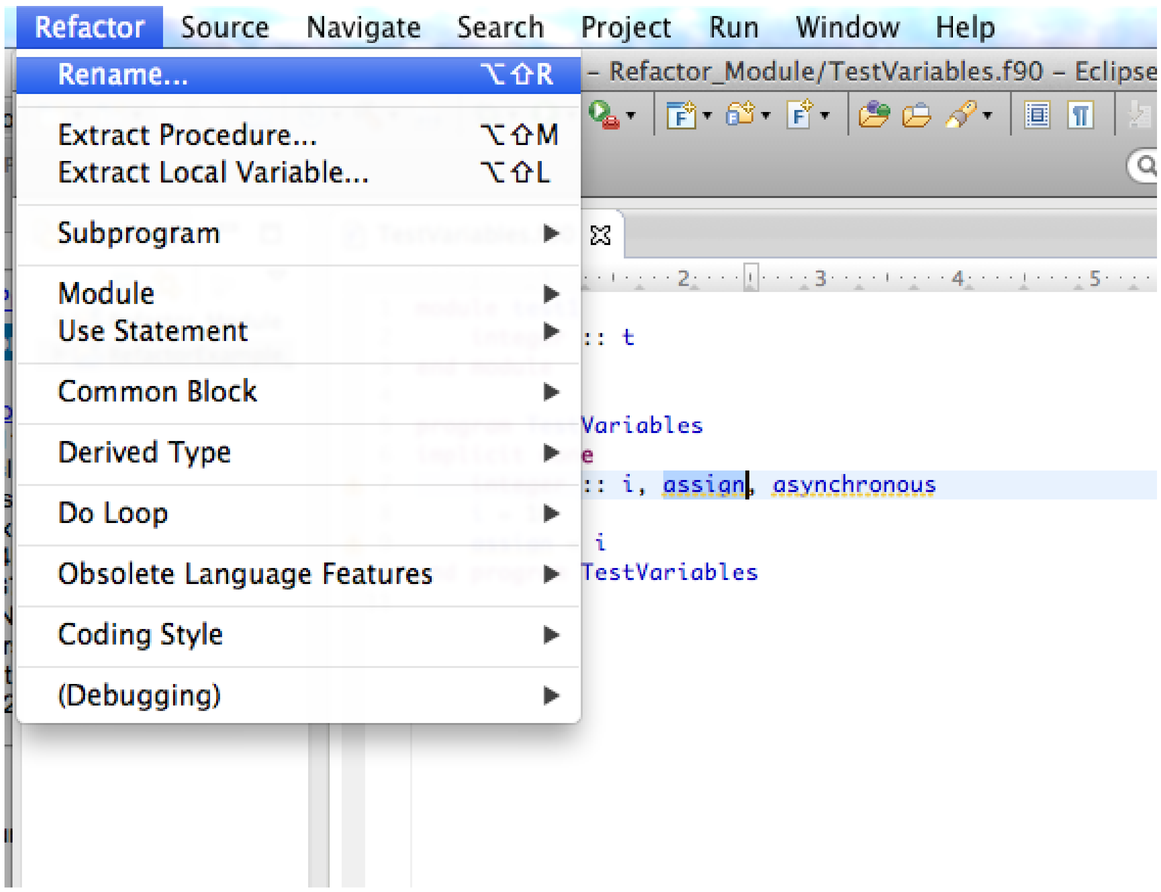
Rename assign to: assign

[✓] Match external subprograms with interfaces and external declarations

Assign a new name to the identifier and click on OK. All occurrences of the illegal variable will now be renamed.

3. Default Refactoring way

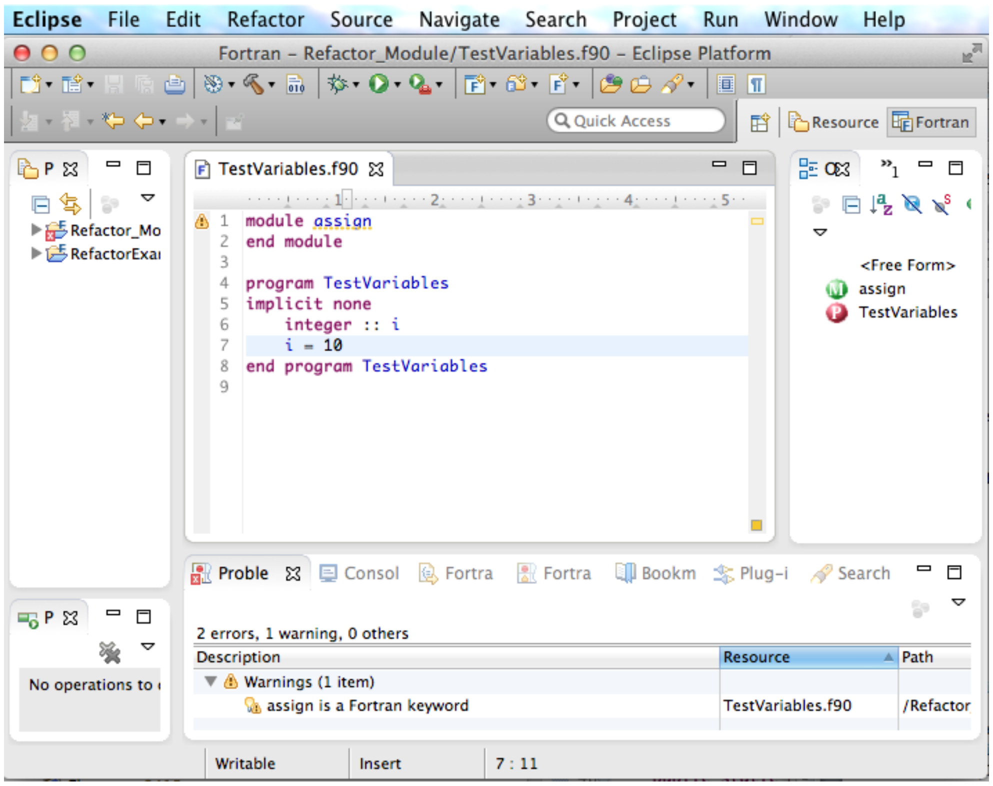
   The third way to refactor using rename refactoring from refactoring menu. Highlight the identifier to be changes and invoke rename refactoring form Refactor menu by clicking "Rename…".



```
:: t

Variables
e
:: i, assign, asynchronous

 i
TestVariables
```

iii. Rename Refactoring Limitations

Rename refactoring, which is used as the quick fix for the illegal use of keyword highlighted by the keyword check plug-in has certain limitations. Currently, as of (Dec 2012), it does not support refactoring variables which are inside user defined data types and the names of modules. Thus, although the keyword check plug-in highlights names of modules or identifiers that use Fortran keywords, you will not be able to refactor them using rename refactoring and manual refactoring is required.



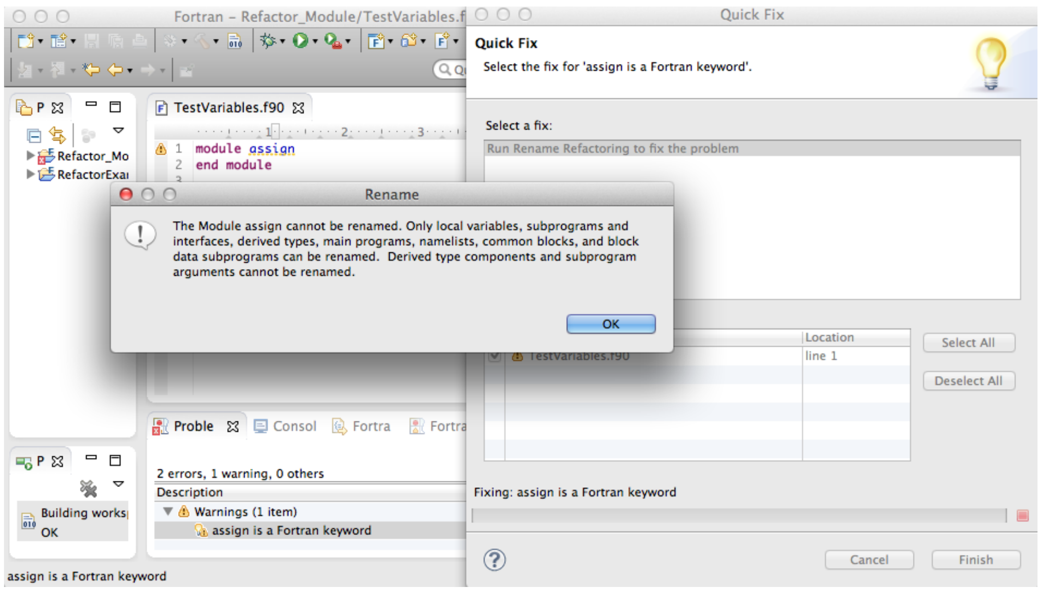
```
module assign
end module

program TestVariables
implicit none
    integer :: i
    i = 10
end program TestVariables
```

2 errors, 1 warning, 0 others

Description	Resource	Path
Warnings (1 item)		
assign is a Fortran keyword	TestVariables.f90	/Refactor

Writable | Insert | 7 : 11

In the Fortran program in the above screenshot, module name “assign” is illegal. If you try refactoring it in the quick fix, you will see the message as mentioned in the screenshot below. IN this case please proceed with a manual refactoring.



```
module assign
end module
```

The Module assign cannot be renamed. Only local variables, subprograms and interfaces, derived types, main programs, namelists, common blocks, and block data subprograms can be renamed. Derived type components and subprogram arguments cannot be renamed.

Quick Fix

Select the fix for 'assign is a Fortran keyword'.

Select a fix:

Run Rename Refactoring to fix the problem

	Location
TestVariables.f90	line 1

Fixing: assign is a Fortran keyword

2 errors, 1 warning, 0 others

Description
Warnings (1 item)
assign is a Fortran keyword

assign is a Fortran keyword

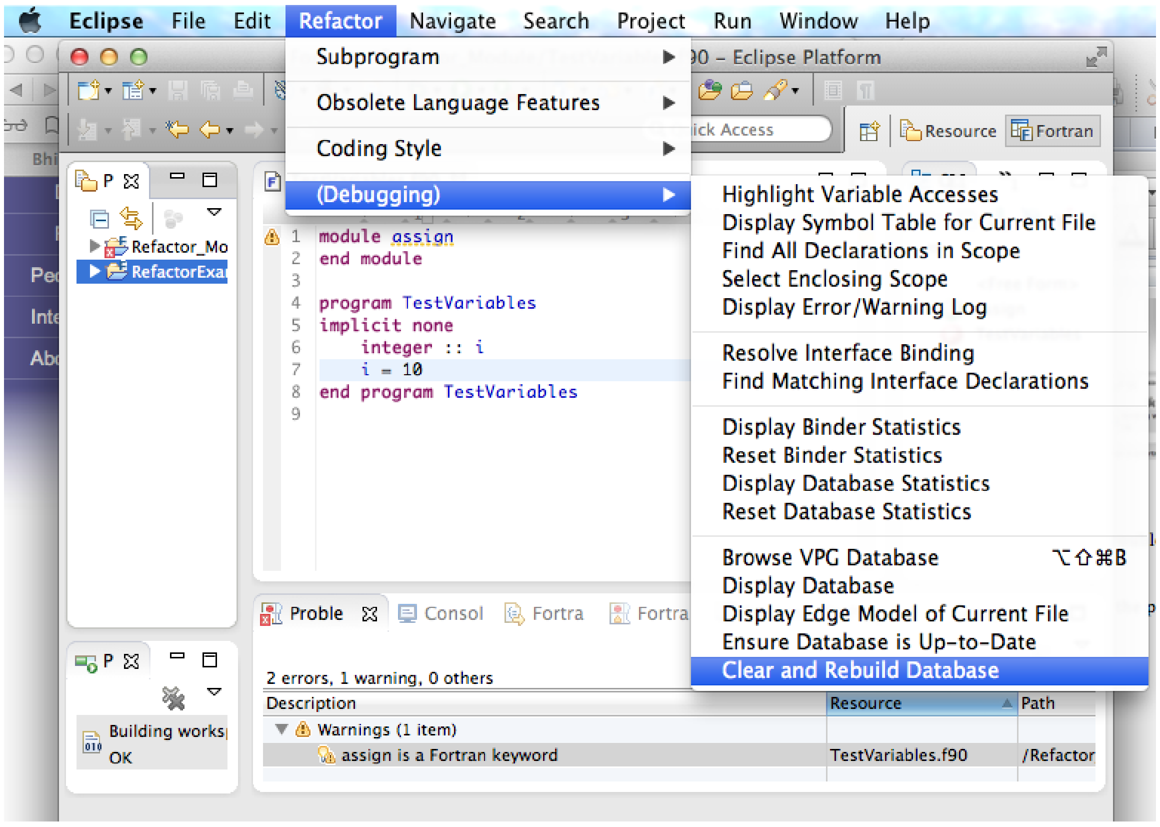
NOTE:

If any new keywords are added in Fortran, the user manually needs to add the keyword in the collection of keywords present in the PhotranKeyword.java class.

7. Troubleshooting

If the plugin throws an error photran indexer array indexer out of bounds do:

This is due to indexer database being corrupted in Photran. Kindly refer to http://dev.eclipse.org/mhonarc/lists/photran/msg01607.html


Eclipse
File
Edit
Refactor
Navigate
Search
Project
Run
Window
Help
Subprogram
Obsolete Language Features
Coding Style
(Debugging)
Highlight Variable Accesses
Display Symbol Table for Current File
Find All Declarations in Scope
Select Enclosing Scope
Display Error/Warning Log
Resolve Interface Binding
Find Matching Interface Declarations
Display Binder Statistics
Reset Binder Statistics
Display Database Statistics
Reset Database Statistics
Browse VPG Database ⌥⇧⌘B
Display Database
Display Edge Model of Current File
Ensure Database is Up-to-Date
Clear and Rebuild Database
90 – Eclipse Platform
Quick Access
Resource
Fortran
Refactor_Mo
RefactorExa
Proble
Consol
Fortra
Fortra
Building works
OK

```
module assign
end module

program TestVariables
implicit none
    integer :: i
    i = 10
end program TestVariables

```

2 errors, 1 warning, 0 others

Description	Resource	Path
Warnings (1 item)		
assign is a Fortran keyword	TestVariables.f90	/Refactor