Tutorial 4
More on Refactoring

How to refactoring unstructured code?
How to apply refactoring in Eclipse?

Last lecture… On refactoring

- We explained what is refactoring, what is software refactoring
- How are they related to other restructuring techniques?
- Examples of refactoring
- Refactoring structured source code into goal models
- …

Today…

1. How to refactoring unstructured code into goal models?
2. How to use Eclipse to do refactoring?
3. Discussions
4. Relation to your course project

1. Refactoring an unstructured program

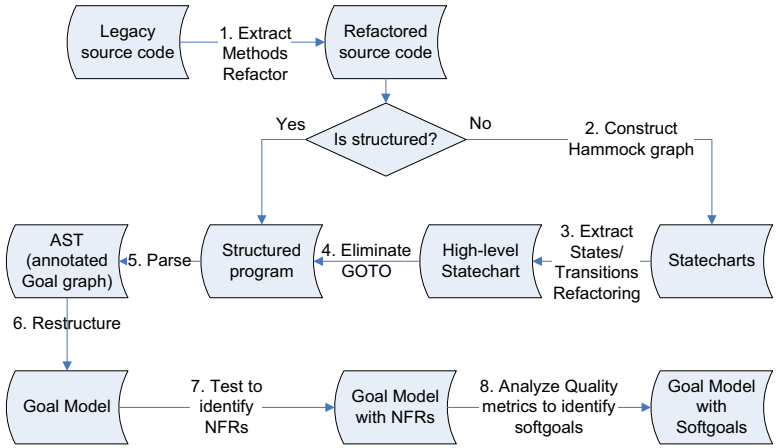
- The subject is called “Squirrel Mail”
- It has 70 KLOC
- Developed in PHP
  Function call
  Foo.php: <?php include(“bar.php”) ?>
- Why it is unstructured?
  Foo.php: <a href=“bar.php”/>
  <a href=“moo.php”/>
  <?php echo “I won super 7!” ?>
  Any idea?

Why a PHP program is unstructured?

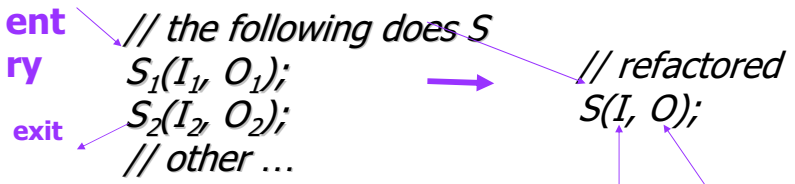
- Every Hyperlink generated from the PHP is an “exit” in the current PHP program
- It may call other PHP routines, other web pages, etc. when user click at them
- Non-deterministic, how could you tell which link will the user click?
- Even “go back” button will change the behaviour of the program
- So …

The process

- Structured program is easier to understand



1.1 Refactoring based on comments



Variables defined before the entry of the block

$$I = (I_1 \cup I_2) \cap \{ v \mid def(v, p) \wedge p \angle p_{entry} \} \neq \phi$$

Variables defined in the block that will be used after the exit

$$O = (O_1 \cup O_2) \cap \{ v \mid use(v, p) \wedge p_{exit} \angle p \} \neq \phi$$

Example

```
/** Path for SquirrelMail required files. */
define('SM_PATH','../');
require_once($SM_PATH . 'functions/strings.php');
require_once($SM_PATH . 'config/config.php');
require_once($SM_PATH . 'functions/i18n.php');
require_once($SM_PATH . 'functions/plugin.php');
require_once($SM_PATH . 'functions/constants.php');
require_once($SM_PATH . 'functions/page_header.php');
require_once($SM_PATH . 'functions/html.php');
require_once($SM_PATH . 'functions/global.php');
require_once($SM_PATH . 'functions/imap_general.php');
```



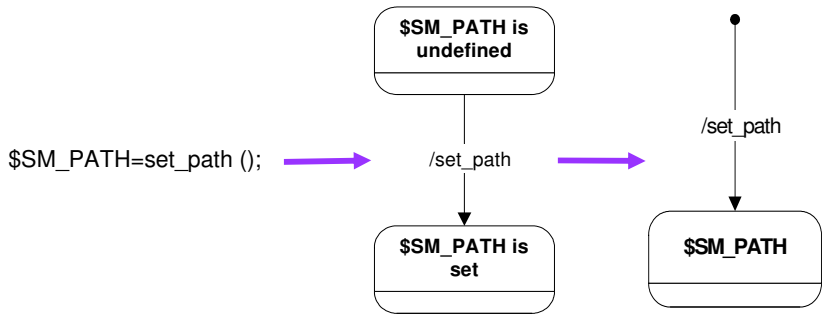
```
$SM_PATH=set_path ();
```

Further …

```
<?php /* login.php */
$SM_PATH=set_path ();
$SM_lang=setup_language();
$base_uri  = findout_base_URI();
$logindisabled = detect_imap_server($base_uri);
if ($logindisabled) {
    explain_situation();
    exit;
}
do_hook('login_cookie');
$header =onload_function("redirect.php");
display_header($header);
load_theme($theme[$theme_default]);
do_hook('login_top');
show_logo();
show_form($loginname, $mailto, $key);
do_hook('login_form');
do_hook('login_bottom');
?>
```

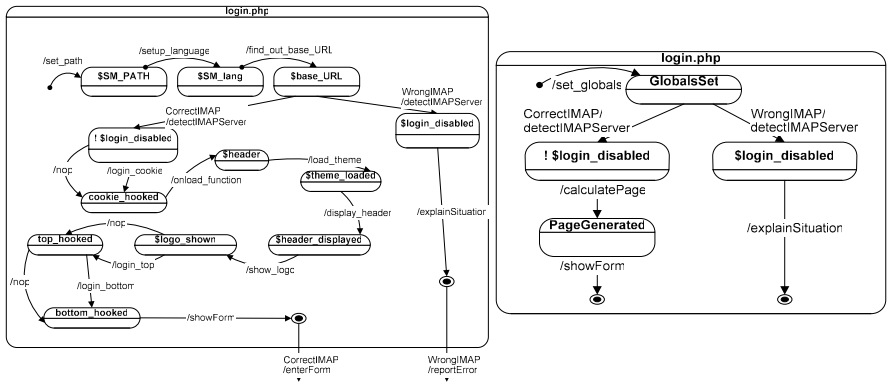

1.2 Convert into statechart

- Statecharts concisely describe behaviour of a system.
- No comments now, but we need to understand its behaviour, therefore …



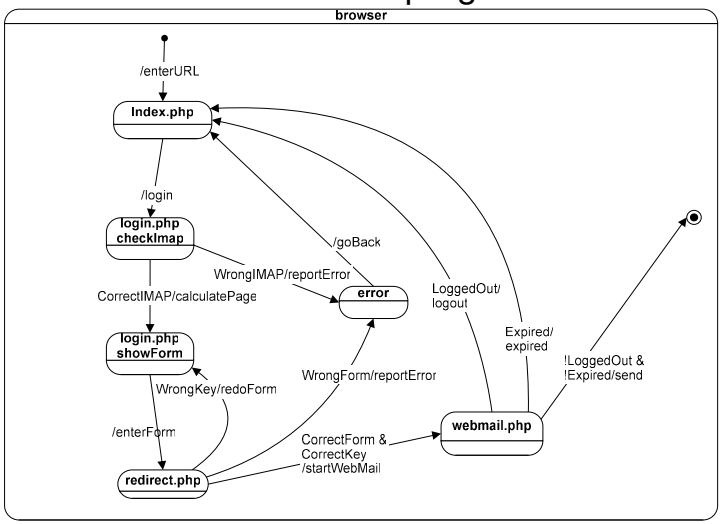
1.3 Statechart refactoring

Extract Method -> Extract States and Transitions based on Hammock graphs



What's new here? You are refactoring behaviour rather than structures!

1.4 Put it together …the high-level statechart of the unstructured program



1.4 Now convert statechart back into a program with GOTO's

- FORTRAN

```
      call EnterURL
10    call Login
      if (wrongIMAP) goto 30
20    call ShowForm
      if (wrongKey) goto 20
      call EnterForm
      if (wrongForm) goto 30
      call StartWebMail
      if (loggedOut) goto 10
      if (expired) goto 10
      call Send
      Stop
30    call ReportError
      call GoBack
      goto 10
      end
```

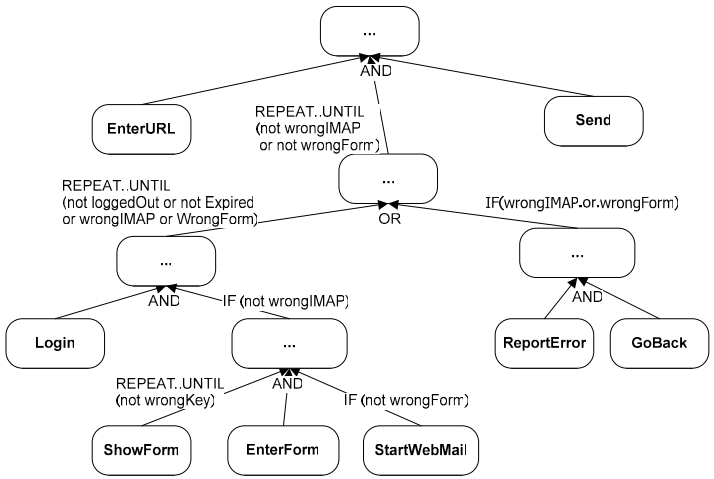
- Rule of thumb: every state is a basic block; adding a label to states with multiple incoming transitions; adding GOTO statements for all outgoing transitions except one; line-up the basic blocks

1.4 Eliminate GOTO's

- FPT (Fortran parallelizing transformer, developed at ELIS, Ghent University, Belgium)
- Result of goto elimination:

```
CALL EnterURL
REPEAT
REPEAT
 CALL Login
 IF (.not.wrongIMAP) THEN
   REPEAT
     CALL ShowForm
   UNTIL (.not.wrongKey)
   CALL EnterForm
   IF(.not.wrongForm)THEN
     CALL StartWebmail
   ENDIF
     ENDIF
UNTIL (.not.loggedOut.or .not.expired.or.wrongIMAP .or.wrongForm)
IF(wrongIMAP.or.wrongForm)
THEN
  CALL ReportError
  CALL GoBack
     ENDIF
UNTIL (.not.wrongIMAP.and.not.wrongForm)
CALL Send
END
```


1.5 Turning structured program into an annotated goal model

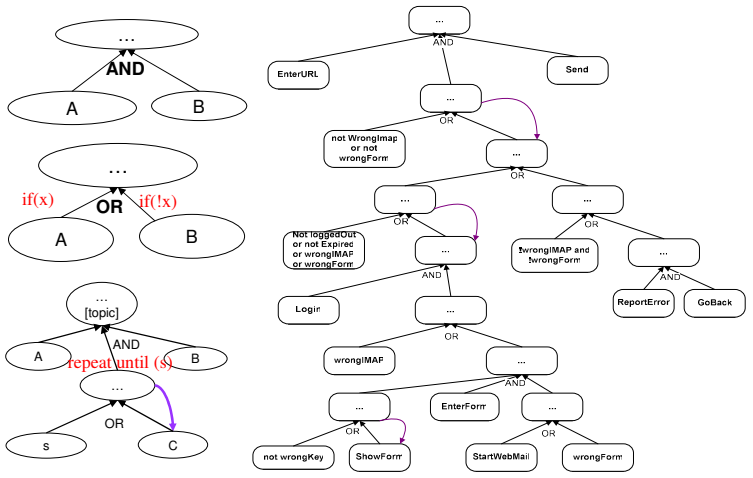


1.6 Turning it into "pure" goal model (AND/OR graph)

•call A
call B

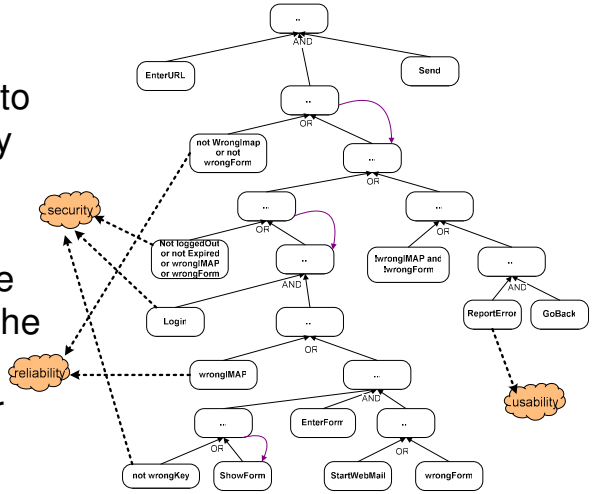
•if (x) then
call A
else
call B
end if

•call A
repeat
call C
until s
call B



1.7 Introducing softgoals

- Identify NFRs
- Add softgoals to categorize why there are the NFRs
- If possible, one can measure the degree of satisfaction for the softgoals



2. How to refactoring in Eclipse

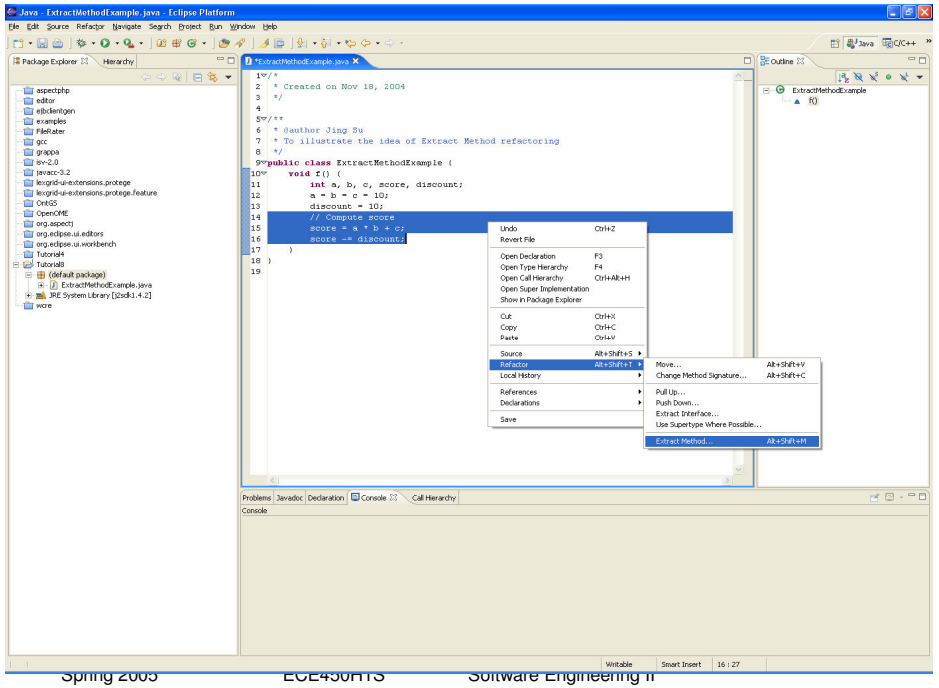
- If you are developing in Java, you are LUCKY!
- The Eclipse IDE, JBuilder IDE are very comprehensive
- Refactoring was developed in Smalltalk, now moved to Java in Eclipse, it has been told in C# for Visual Studio, etc.
- It should not be long to see open-source programming languages to have them supported, such as PHP
- Examples, developed by Jing Su

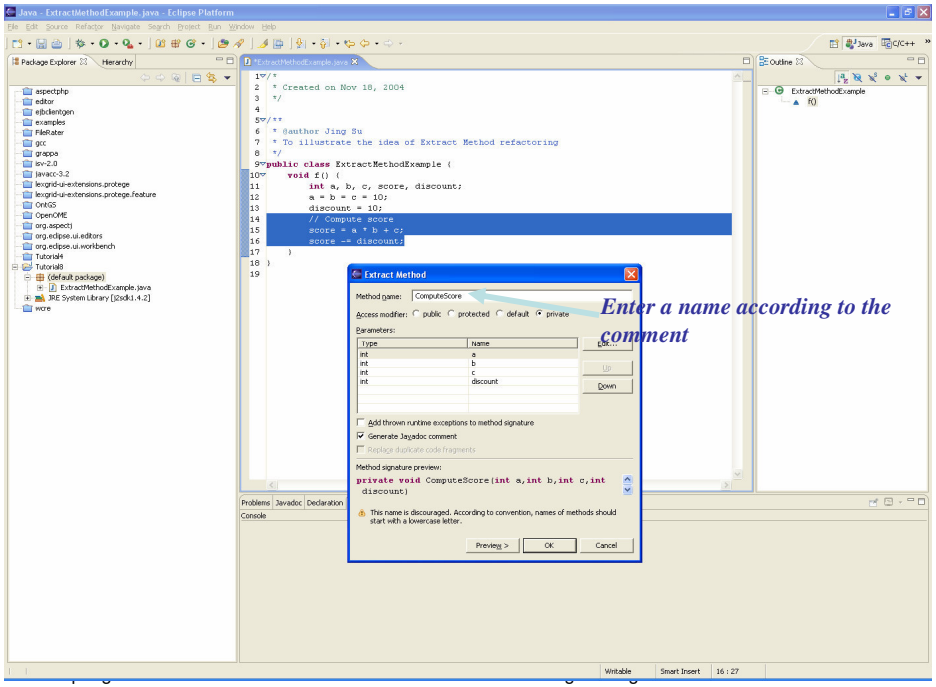
Example 1 – extract method

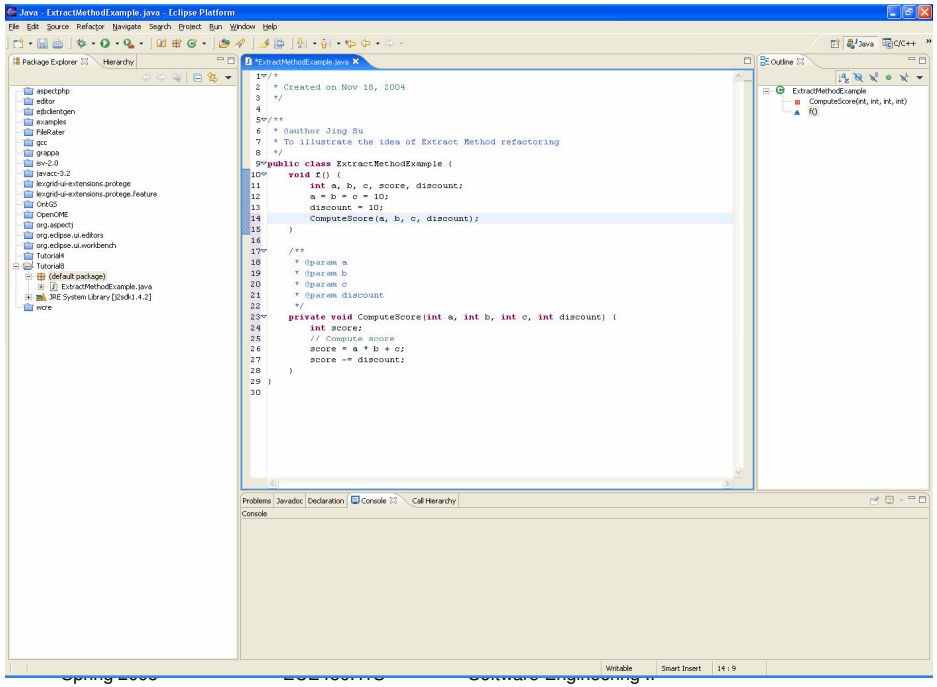
```
void f() {
    ...
    // Compute score
    score = a * b + c;
    score -= discount;
}
```

```
void f() {
    ...
    computeScore();
}

void computeScore() {
    score = a * b + c;
    score -= discount;
}
```

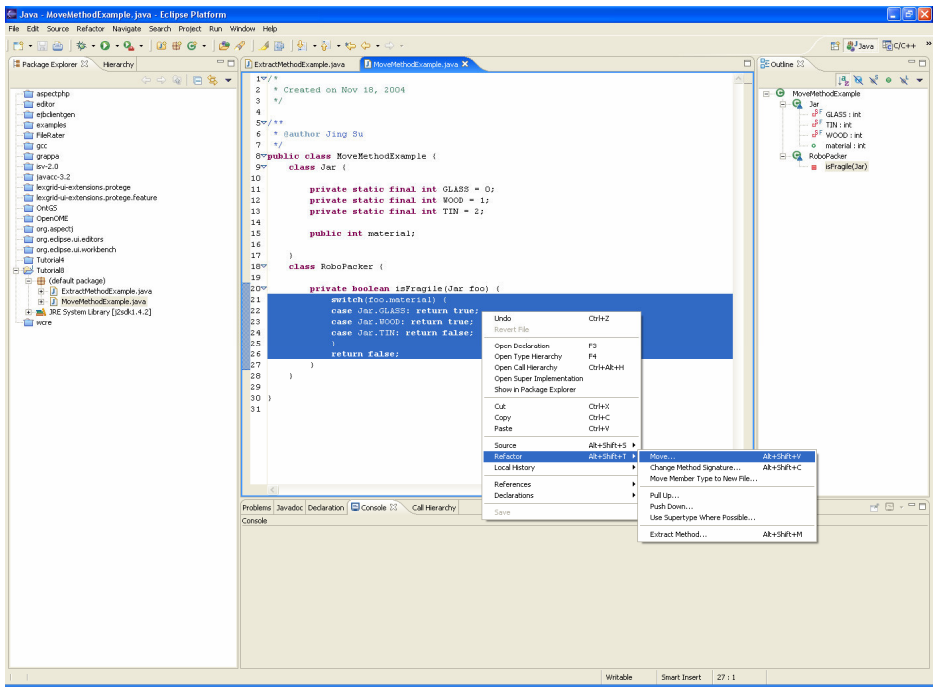


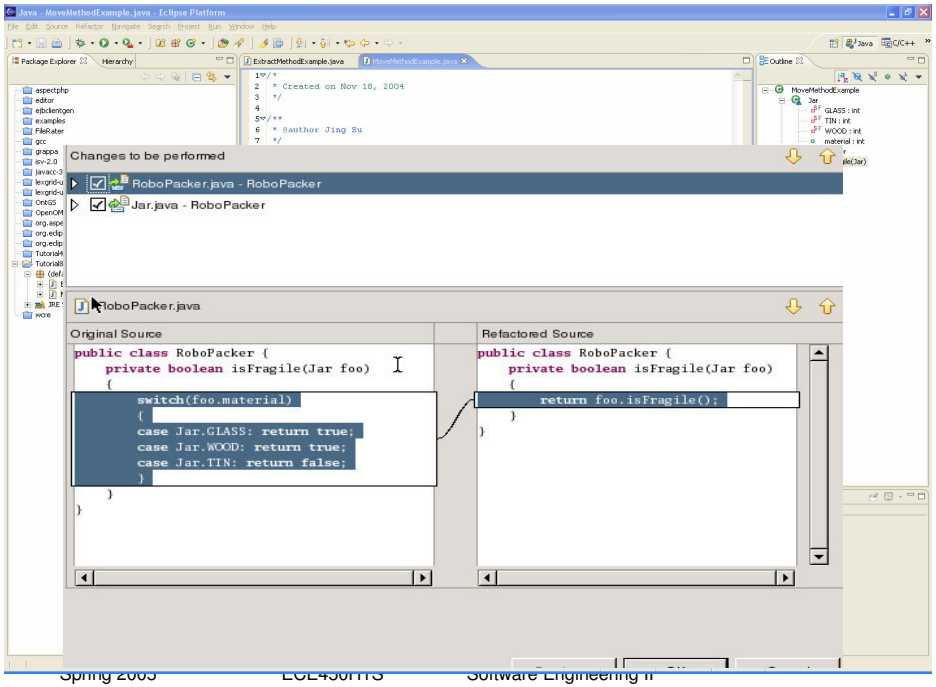


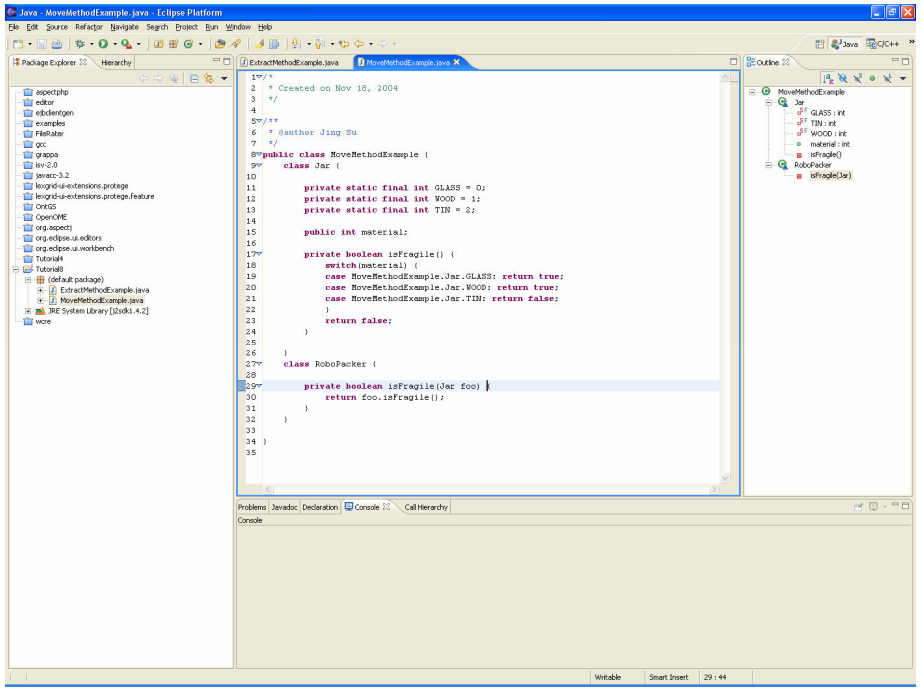
Example 2 – move method

```
class Jar {
    ...
}
class RoboPacker {
    private bool isFragile(Jar foo) {
        switch(foo.material) {
        case GLASS: return true;
        case WOOD: return true;
        case TIN: return false;
        }
    }
}
```

```
class Jar {
    bool isFragile() {
        switch(material) {
        case GLASS: return true;
        case WOOD: return true;
        case TIN: return false;
} } }
class RoboPacker {
    private bool isFragile(Jar foo) {
        return foo.isFragile();
    }
}
```





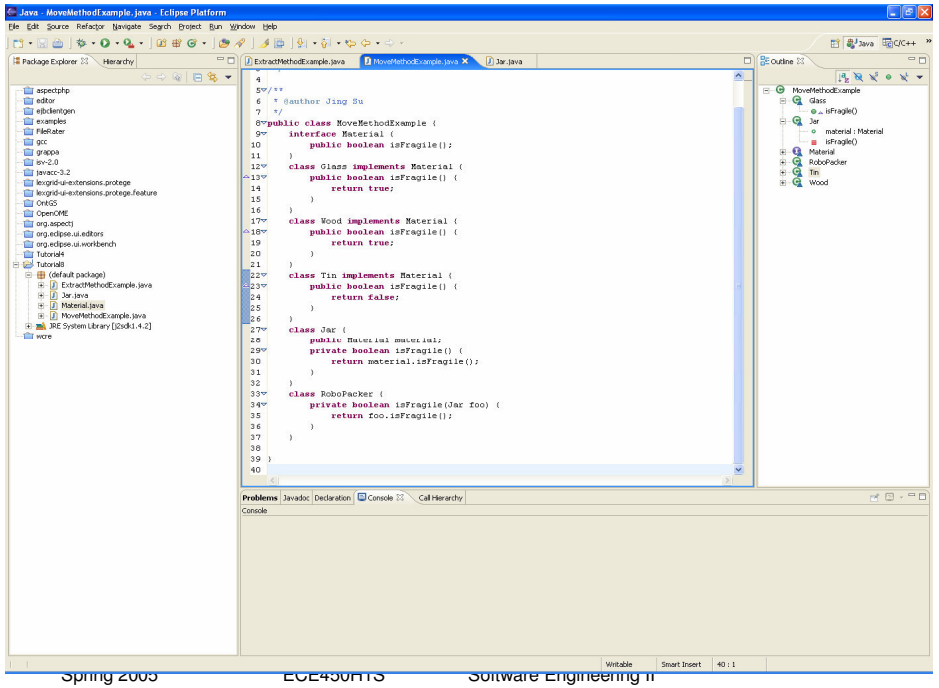


Example 3 – lift method

```
class Jar {
   bool isFragile() {
      switch(material) {
      case GLASS:
         // complex glass calculation
      case WOOD:
         // complex wood calculation
      case TIN:
         // complex tin calculation
} } }
```

```
class Jar {
   bool isFragile() {
      return material.isFragile();
} }

interface Material { ... }
class GlassMaterial:Material { ... }
class WoodMaterial:Material { ... }
class TinMaterial:Material { ... }
```

3. Think about these …

1. How to extend refactoring tool support to other programming languages such as PHP?
2. Can you extend refactoring to documents, such as in various formats: diagrams, textual, xml, etc.?
3. How can know a function is NFR? Can you measure the impact of a NFR on a quality attribute?

4. Relation to your project

- Opportunities:
  - You may add junit test cases to the code base to reveal bugs (publish it to the bug tracking system) and fix them (+5%)
  - *You may apply design patterns, refactoring techniques on this legacy code base, showing as an improved complexity metrics (+2.5%)*
  - You may tune the performance of the system to speed up the display, load/save for scalable graphs (+2.5%)
- Don't forget your major project task (up to 100%!)
  - To study the editor methods in the OpenOME and adapt them to the OmniGraphEditor web service.