IBM RATIONAL APPLICATION DEVELOPER 6.0 – LAB EXERCISE

Application Analysis / Code Review
IBM Rational Application Developer v6.0

What this exercise is about

The objective of this lab is to provide you with an understanding of the new Code Review feature of IBM Rational Application Developer v6.0.

Lab Requirements

List of system and software required for the student to complete the lab.

- IBM Rational Application Developer version 6

What you should be able to do

At the end of this lab you should be able to:

- Configure Code Review in IBM Rational Application Developer v6.0

Introduction

This lab highlights IBM Rational Application Developer v6.0 and the use of Code Review to warn or remind you of certain rules. The goal of Code Review is to make your code more robust, easily. In some cases, a click of a mouse button can update your entire project of a coding mistake. In this lab, you will see all the options in Code Review which is aimed to start you off using Code Review.

Exercise Instructions

Because these instructions are not operating-system specific, the directory locations will be specified in the lab instructions using symbolic references, as follows:

Reference Variable	Windows Location	AIX/UNIX Location
<LAB_FILES>	C:\Labfiles60	/tmp/Labfiles60
<TEMP>	C:\temp	/tmp

Windows users please note: When directory locations are passed as parameters to a Java program such as EJBdeploy or wsadmin, it is necessary to replace the backslashes with forward slashes to follow the Java convention. For example, C:\LabFiles60\ would be replaced by C:/LabFiles60/

Part 1: Lab Setup

____ 1. Start IBM Rational Application Developer v6.0.

__ a. Select **Start** > **Programs** > **IBM Rational** > **IBM Rational Application Developer v6.0** > **Rational Application Developer**.

__ b. For workspace, specify **<LAB_FILES>\IRAD_ApplicationAnalysisCodeReview\workspace**.

__ c. When IBM Rational Application Developer v6.0 opens, close the **Welcome page**.

NOTE: If the Auto Launch Configuration Change Alert window appears click the **Yes** button to change the auto launch eclipse instance to use when opening IBM Rational Software Development Platform in the future.

____ 2. Import the WebSphereBank application into IBM Rational Application Developer.

__ a. Click **File** > **Import...**.

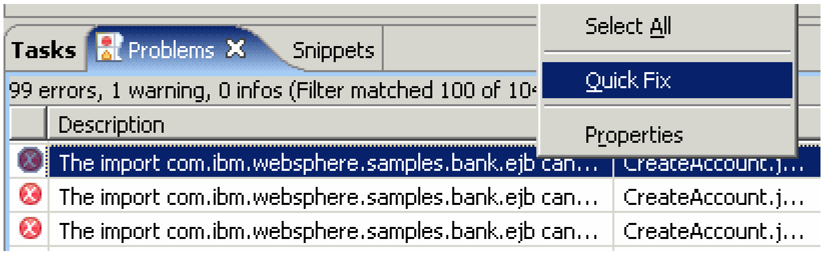
__ b. Click **EAR file** and select **Next.**

__ c. Click **Browse...** and navigate to **<LAB_FILES>\ IRAD_ApplicationAnalysisCodeReview \WebSphereBank.ear** and Click **Open**.

__ d. Click **Finish**.

____ 3. A number of items will be displayed in the Problems view. These errors should be resolved before continuing.

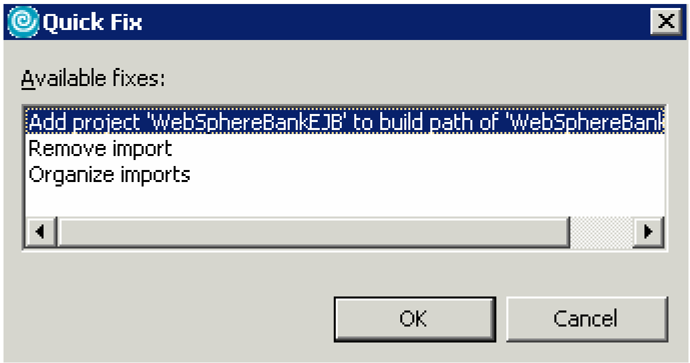
__ a. Right click on an error and select **Quick Fix.**



99 errors, 1 warning, 0 infos (Filter matched 100 of 10

Description	
The import com.ibm.websphere.samples.bank.ejb can...	CreateAccount.j...
The import com.ibm.websphere.samples.bank.ejb can...	CreateAccount.j...
The import com.ibm.websphere.samples.bank.ejb can...	CreateAccount.j...

__ b. A list of available fixes will appear. Select **Add project 'WebSphereBankEJB' to build path of 'WebSphereBankWeb'** and click **OK.** The errors will be resolved.


Quick Fix
Available fixes:
Add project 'WebSphereBankEJB' to build path of 'WebSphereBank
Remove import
Organize imports
OK
Cancel

__ c. Close any Java files which are opened in the process.

Part 2: Open the Code Review view

Now that the workspace is ready with an active project, you can now open the Code Review view. This is a necessary step in order to run Code Review on your code.

Without the Code Review view open, you cannot run Code Review. The Code Review view in IBM Rational Application Developer v6.0 does not come turned on by default and is not part of any IBM Rational Application Developer v6.0 Perspectives (Java, Debug, etc.). The reason for this is due to the additional memory needed to open and run Code Review. However, once you open the Code Review view in your workspace, it will conveniently stay open upon save.

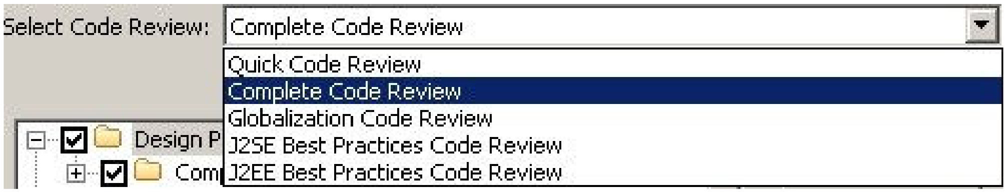
NOTE: To "garbage collect" or free up memory in IBM Rational Application Developer v6.0, minimize the workbench and then maximize again. If you have your Task Manager up in windows, you will see that the memory used by IBM Rational Application Developer v6.0 is reduced greatly just by minimizing .

The assumption here is IBM Rational Application Developer v6.0 is running with the WebSphereBank example imported and correctly set up for the lab.

____ 1. Set Code Review to perform Complete Code Review

__ a. Click on Window > Preferences > Java > Code Review

__ b. In the Select Code Review box, click the drop down button and select **Complete Code Review**. Quick Code Review is selected by default. You will learn more information about these code review types in this lab.



Select Code Review: Complete Code Review

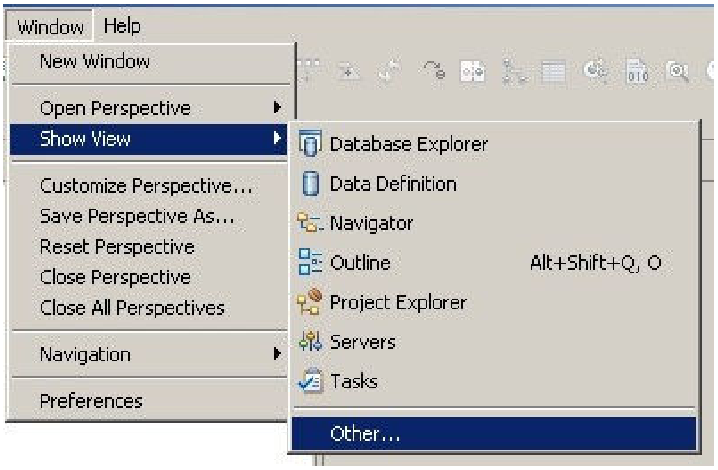
Quick Code Review
Complete Code Review
Globalization Code Review
J2SE Best Practices Code Review
J2EE Best Practices Code Review
Design P
Com

__ c. Click the **OK** button to save this change and return to your workbench.

____ 2. Open the Code Review view.

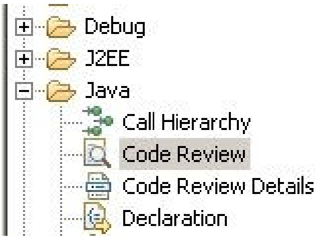
__ a. On the top menu bar in IBM Rational Application Developer v6.0, select **Window > Show View**.

__ b. At the bottom of the Show View menu, click **Other**.



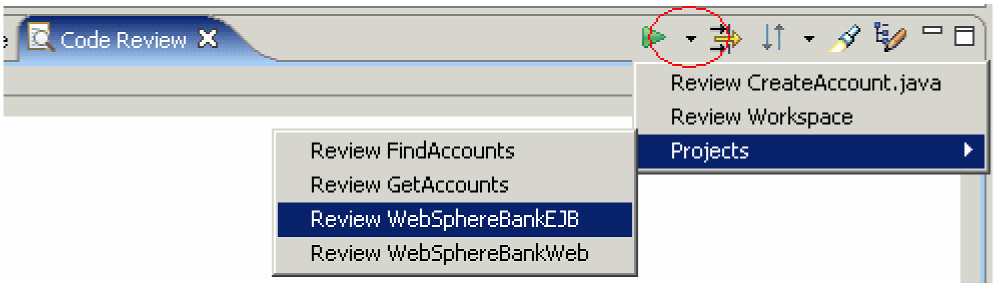
__ c. Once the “Other” Show View menu opens, double click the **Java folder**.

__ d. Click **Code Review** and then the **OK Button** below. The view will take a couple seconds to load.



____ 3. Run Code Review on the WebSphereBankEJB project.

__ a. Click the **drop down arrow** (small black triangle pointing down) next to the Review button (green triangle pointing to the right) in the Code Review view menu.

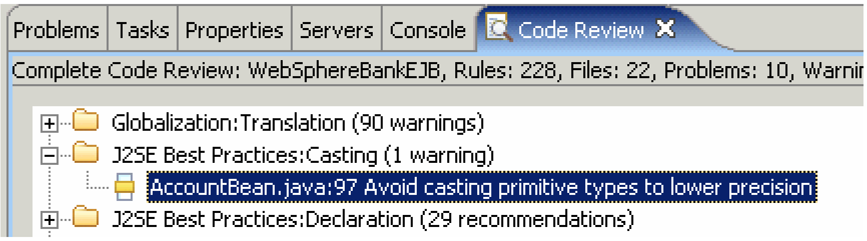


__ b. Roll the mouse over Projects, then to WebSphereBankEJB. Click **WebSphereBankEJB**. Code Review will run. You will see the Code Review view fill up with problems, warnings, and recommendations. This will take a little bit to run through all the rules for all lines of code in the WebSphereBankEJB project.

____ 4. Open a warning from the generated Code Review warnings to see what Code Review is about.

__ a. In the Review results generated in the Code Review view, open **J2SE Best Practices: Casting**.

__ b. Double Click on the first entry, a warning, with the name **AccountBean.java:97 Avoid casting primitive types to lower precision**.

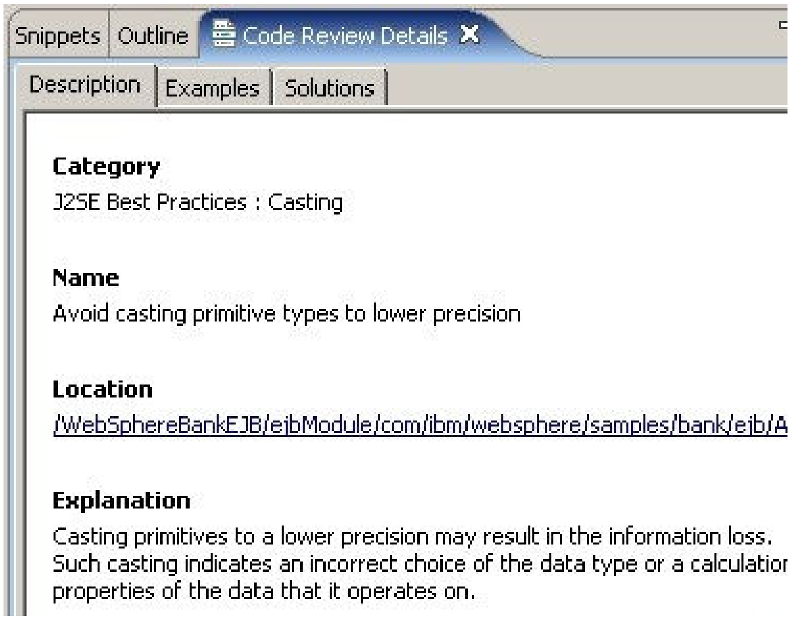


Complete Code Review: WebSphereBankEJB, Rules: 228, Files: 22, Problems: 10, Warnir

Globalization:Translation (90 warnings)
J2SE Best Practices:Casting (1 warning)
AccountBean.java:97 Avoid casting primitive types to lower precision
J2SE Best Practices:Declaration (29 recommendations)

Note: Double Clicking an item is the same as right clicking on the entry and selecting Go To Location, except that using Go To Location does not open the Code Review Details view,

__ c. This action will open AccountBean.java to line number 97 with the warning selected. This action will also open a Code Review Details view in IBM Rational Application Developer v6.0 that allows you to see a description of the warning, code examples of the warning (what not to do), and coding solutions for the warning.



Category
J2SE Best Practices : Casting

Name
Avoid casting primitive types to lower precision

Location
/WebSphereBankEJB/ejbModule/com/ibm/websphere/samples/bank/ejb/A

Explanation
Casting primitives to a lower precision may result in the information loss. Such casting indicates an incorrect choice of the data type or a calculatior properties of the data that it operates on.

__ d. Bring focus to the Code Review Details view. Under the **Description tab** (open by default), you will find Category, Name, Location, and an Explanation of the warning listed there.

Category

J2SE Best Practices : Casting

Name

Avoid casting primitive types to lower precision

Location

/WebSphereBankEJB/ejbModule/com/ibm/websphere/samples/bank/ejb/AccountBean.java:97

Explanation

Casting primitives to a lower precision may result in the information loss.
Such casting indicates an incorrect choice of the data type or a calculation that ignores properties of the data that it operates on.

__ e. Under the **Examples tab**, you will find a code example of the warning.

```
public static void main (String[] args){
double x = 3.1;
int i = (int)x;
System.out.println(i);
}
```

__ f. Under the **Solutions tab**, you will find a couple solutions for this warning.

Solution

Change the underlying data type to be lower precision.

```
public static void main (String[] args){
int i = 3;
System.out.println(i);
}
```

Solution

Update the calculation to operate on the higher precision data type.

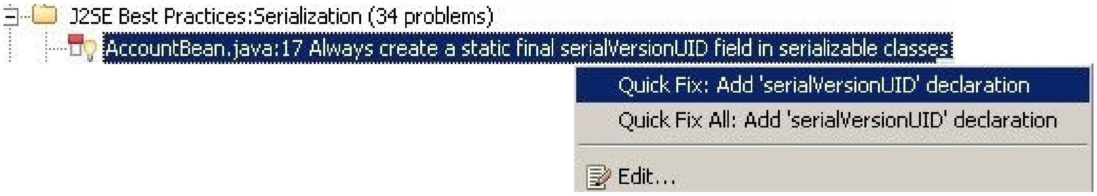
```
public static void main (String[] args){
double x = 3.1;
System.out.println( x );
}
```

_____ 5. Use **Quick Fix** to fix a warning generated by Code Review.

__ a. Go back to the Code Review view and open the **J2SE Best Practices: Serialization** folder.

__ b. Double Click the first entry. It should be **AccountBean.java:17 Always create a static final serialVersionUID field in serializable classes**.

__ c. Go back to the Code Review view, then right click on the entry and select the first QuickFix option. **Quick Fix: Add 'serialVersionUID' declaration**.



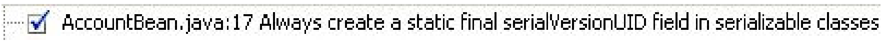
J2SE Best Practices:Serialization (34 problems)

AccountBean.java:17 Always create a static final serialVersionUID field in serializable classes

Quick Fix: Add 'serialVersionUID' declaration
Quick Fix All: Add 'serialVersionUID' declaration
Edit...

Note: The second QuickFix option is Quick Fix All. This will correct the code throughout the scope of Code Review. In this case, it would add 'serialVersionUID' declarations to all of the WebSphereBankEJB project since we set Code Review to review the WebSphereBankEJB project only.

__ d. A quick fix window will pop up to show you what files are going to be changed and highlight what changes are being made to the code. Click **OK button**.
You will now notice that the warning will have a blue check next to it signifying that it has been fixed.



☑ AccountBean.java:17 Always create a static final serialVersionUID field in serializable classes

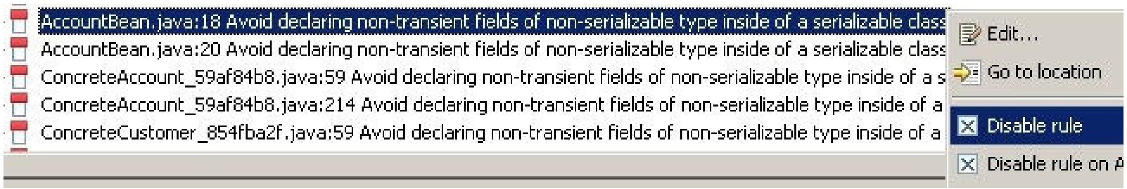
__ e. Right click on the fixed warning and select **Undo: Add 'serialVersionUID' declaration**.

Note: The Undo QuickFix option is available directly after you Quick Fix a file. If the file status changes for any reason, the Undo Quick Fix will not be available.

__ f. If at anytime you do not see the Code Review Details view or if the Code Review Details view is open but grayed out, double click the warning listed in the Code Review view again. Double clicking the warning will enact the warning's Code Review Details.

____ 6. **Disable a Code Review rule.**

__ a. In the Code Review view, under the **J2SE Best Practices: Serialization** folder, right-click the warning **AccountBean.java:18 Avoid declaring non-transient fields of non-serializable type inside of a serializable class**. Under the right-click menu, click **Disable Rule**.



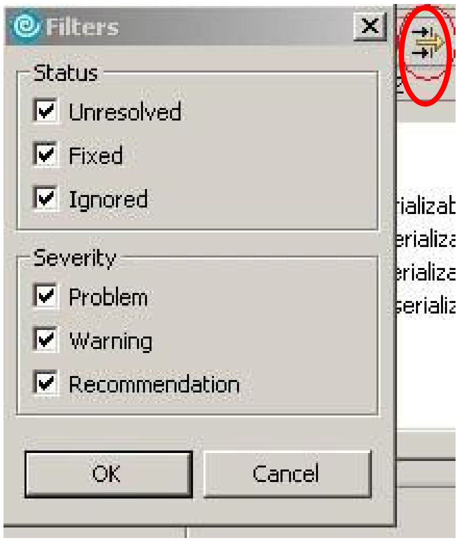
AccountBean.java:18 Avoid declaring non-transient fields of non-serializable type inside of a serializable class
AccountBean.java:20 Avoid declaring non-transient fields of non-serializable type inside of a serializable class
ConcreteAccount_59af84b8.java:59 Avoid declaring non-transient fields of non-serializable type inside of a s
ConcreteAccount_59af84b8.java:214 Avoid declaring non-transient fields of non-serializable type inside of a
ConcreteCustomer_854fba2f.java:59 Avoid declaring non-transient fields of non-serializable type inside of a

__ b. Notice that all the warnings associated with the rule changed icons to Xs.

__ c. To enable that rule again, right-click the warning **AccountBean.java:18 Avoid declaring non-transient fields of non-serializable type inside of a serializable class**. Under the right-click menu, click **Enable Rule**.

____ 7. **Filter generated warnings in the Code Review view.**

__ a. In the Code Review view, click the **Filter icon** from the Code Review view's menu items (three arrows pointing to the right, one bigger yellow arrow in the middle of two smaller black arrows). The Filters window will pop up.



Filters

Status

☑ Unresolved

☑ Fixed

☑ Ignored

Severity

☑ Problem

☑ Warning

☑ Recommendation

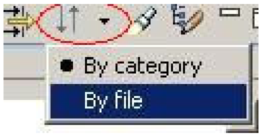
__ b. In the Severity box, uncheck **Warning** and **Recommendation**. Problem should be the only one checked in the Severity box.

__ c. Click **OK** button. Notice all the warnings and recommendations have been deleted from the Code Review view.

__ d. Click the Filter Icon again and recheck **Warning** and **Recommendation**. Click OK button.

____ 8. **Sort the Code Review view.**

__ a. The next icon to the right of the Filter Icon in the Code Review view is the Sort Icon (two arrows pointing up and down with a black drop down triangle). Click the black drop-down triangle and select By file. By default, the Code Review results are sorted By Category.



__ b. Notice the Code Review results will be sorted alphabetically by the file names that have warnings, recommendations, and problems.

__ c. Click the black drop-down triangle and select By category.

____ 9. **Search Code Review results.**

__ a. The next icon to the right of the Sort Icon in the Code Review view is the Search Icon (flashlight icon). Click the Search icon.

__ b. A search text box will appear in the Code Review view. Type **throw** in the Search for: textbox and press the **Enter** (or Return) button on your keyboard.

__ c. Open the results and notice that each problem, warning, or recommendation each have the word "throw" in it. You can now review only those errors with your search selection, instead of all the errors. Ex: "Avoid catching checked exception that is not declared in the **throw**s clause".

__ d. Delete the word throw from the search for: textbox and press the **Enter** (or Return) button on your keyboard. This clears the search and returns the view back to listing all the problems, warnings, and recommendations.

__ e. Click the Search icon again to make the search textbox disappear.

____ 10. Manage Code Review Preferences directly from the Code Review view.

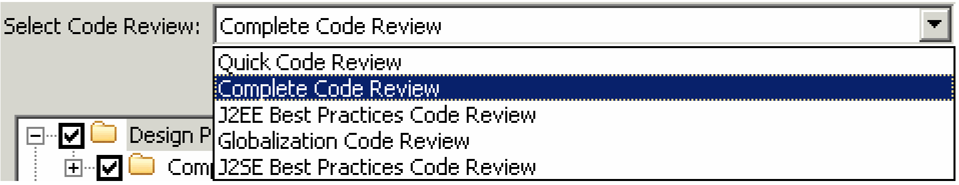
__ a. The last icon to the right of the Search Icon in the Code Review view is the Manage Rules Icon (blue directory with pencil icon). Click the Manage Rules icon.

Note: You can also access the same manage rules window by clicking Window > Preferences > Java > Code Review from the IRAD v6.0 top menu bar.

__ b. You will notice all the rules listed here in the preferences. Now to look inside the preferences.

____ 11. Managing Rules: Select Code Review type

__ a. With the Code Review Preferences open, the first object is a drop down box **Select Code Review:** which organizes the rules for when you run Code Review. Click the drop down arrow on the right side to view all the options. You should see Quick Code Review, Complete Code Review, J2EE Best Practices Code Review, Globalization Code Review, and.J2SE Best Practices Code Review.


Select Code Review: Complete Code Review

Quick Code Review
Complete Code Review
J2EE Best Practices Code Review
Globalization Code Review
J2SE Best Practices Code Review
Design P
Com

Quick Code Review: Chosen when you want to run Code Review on just a small subset of the complete code review, focusing on more important rules taken primarily from J2SE, J2EE, and Performance rule sets.

Complete Code Review: Chosen when you want to run Code Review on all rules in the Code Review rule set. Notice Complete Code Review is the only category that shows and runs User Defined Rules.

J2EE Best Practices Code Review: Chosen when you want to run Code Review on just J2EE Best Practice rules that cover coding topics like Garbage Collection, Resource Management, Performance, and Scalability.

Globalization Code Review: Chosen when you want to run Code Review on just the Globalization rules that help to ensure that global code runs correctly in localized environments.

J2SE Best Practices Code Review: Chosen when you want to run Code Review on just the J2SE Best Practice rules that cover coding topics like Casting, Exceptions, Initialization, and Serialization.

__ b. Flip through the drop down list and select each Code Review type. Notice how the rules listed changes with each selection.

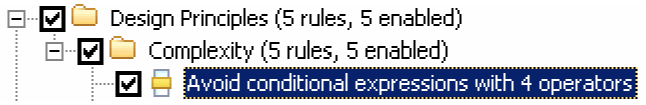
__ c. When finished, go back and select Complete Code Review.

____ 12. View the details of a rule in the Code Review Preferences.

__ a. Expand the first folder under Complete Code Review type, Design Principles, by clicking the [+] sign to the left of the Design Principles folder.

__ b. Expand the Complexity folder in the same way.

__ c. Click on the first rule listed under Complexity, “Avoid conditional expressions with X operators”. Once you click on the rule, you will see the tabbed Code Review properties area for that rule appears at the bottom of the Code Review Preferences window.

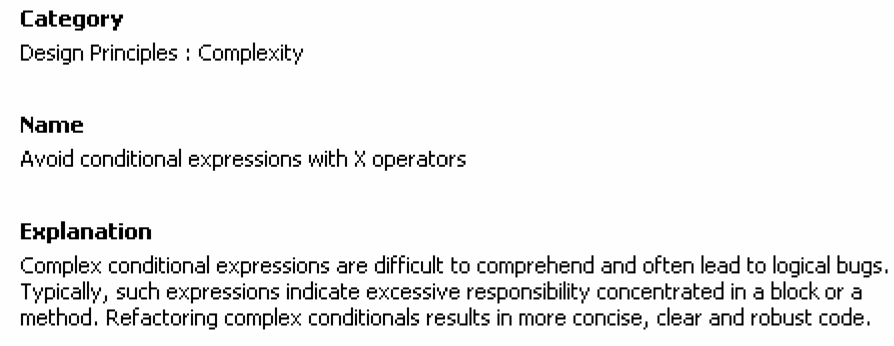


☑ Design Principles (5 rules, 5 enabled)

☑ Complexity (5 rules, 5 enabled)

☑ Avoid conditional expressions with 4 operators

__ d. The first tab **Description** contains the Category, Name, and Explanation.



Category

Design Principles : Complexity

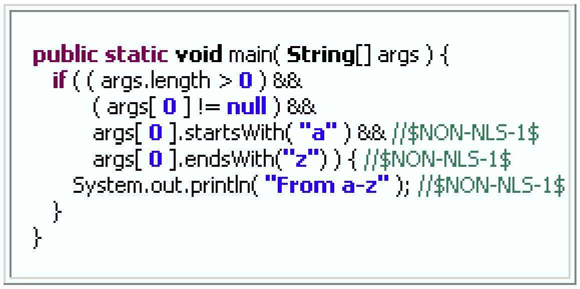
Name

Avoid conditional expressions with X operators

Explanation

Complex conditional expressions are difficult to comprehend and often lead to logical bugs. Typically, such expressions indicate excessive responsibility concentrated in a block or a method. Refactoring complex conditionals results in more concise, clear and robust code.

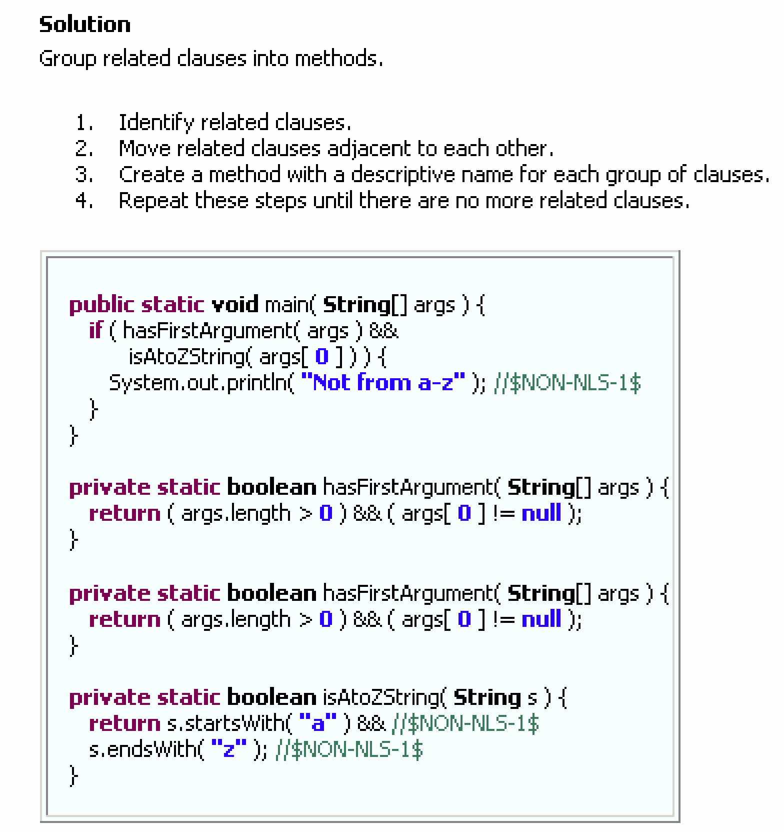
__ e. The second tab **Examples** contains example code for the Code Review rule.



```
public static void main( String[] args ) {
  if ( ( args.length > 0 ) &&
      ( args[ 0 ] != null ) &&
      args[ 0 ].startsWith( "a" ) && //$NON-NLS-1$
      args[ 0 ].endsWith("z") ) { //$NON-NLS-1$
    System.out.println( "From a-z" ); //$NON-NLS-1$
  }
}
```

__ f.

__ g. The third tab **Solutions** contains one or more code solutions for the Code Review rule.



Solution

Group related clauses into methods.

1. Identify related clauses.
2. Move related clauses adjacent to each other.
3. Create a method with a descriptive name for each group of clauses.
4. Repeat these steps until there are no more related clauses.

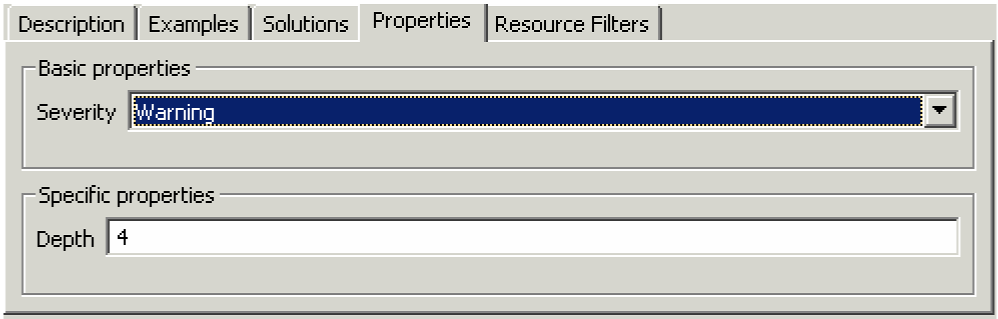
```
public static void main( String[] args ) {
   if ( hasFirstArgument( args ) &&
         isAtoZString( args[ 0 ] ) ) {
      System.out.println( "Not from a-z" ); //$NON-NLS-1$
   }
}

private static boolean hasFirstArgument( String[] args ) {
   return ( args.length > 0 ) && ( args[ 0 ] != null );
}

private static boolean hasFirstArgument( String[] args ) {
   return ( args.length > 0 ) && ( args[ 0 ] != null );
}

private static boolean isAtoZString( String s ) {
   return s.startsWith( "a" ) && //$NON-NLS-1$
   s.endsWith( "z" ); //$NON-NLS-1$
}
```

__ h. The fourth tab **Properties** allows you to set, or change, the properties of the Code Review rule. This is where you can change any rule's severity level between problem, warning, or recommendation. The same Rule Properties can be accessed by right clicking on a rule back in the Code Review view and select **Edit**.



Basic properties

Severity: Warning

Specific properties

Depth: 4

__ i. The fifth tab **Resource Filters** allows you to select which files you want to exclude. The same Resource Filters can be accessed by right clicking on a rule back in the Code Review view and select "Disable rule on <current file>".

Note: Notice the Description, Examples, and Solutions tabs are the same ones that are sent to the Code Review Details view when working with a Code Review object in IBM Rational Application Developer v6.0. Properties and Resource Filters can be accessed by right clicking on the Code Review object.

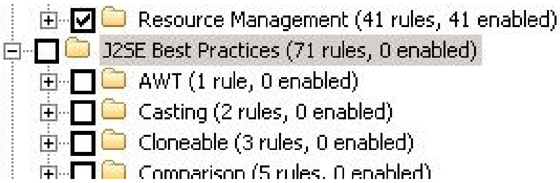
____ 13. **Use Expand All / Collapse All button in Code Review Preferences.**

__ a. Click the **Expand All** button on the right side of the Code Review Preferences window. Notice every rule category has been expanded. Notice the **Expand All** button will change to read **Collapse All**. Click the **Collapse All** button.

____ 14. **Disable a rule/folder/category in the Code Review Preferences.**

__ a. In Code Review Preferences, disabling a rule, folder section, or category is as simple as unchecking the check box next to what you are trying to disable.

__ b. Uncheck the J2SE Best Practices folder. Click **OK** button at the bottom of the Code Review Preferences window.

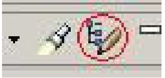


☑ Resource Management (41 rules, 41 enabled)
□ J2SE Best Practices (71 rules, 0 enabled)
□ AWT (1 rule, 0 enabled)
□ Casting (2 rules, 0 enabled)
□ Cloneable (3 rules, 0 enabled)
□ Comparison (5 rules, 0 enabled)

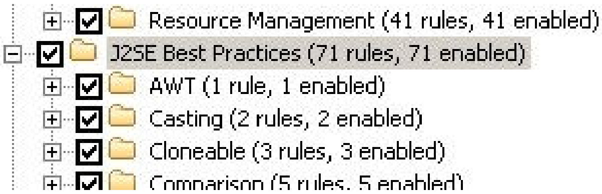
__ c. Now back in IRAD's main view, run Code Review again on WebSphereBankEJB. In the Code Review view's top menu bar, click on the green triangle pointing to the right (from step 2 of this section)

__ d. Notice that J2SE warnings there before are not in this generated code review.

__ e. Click on the Manage Rules button (blue directory with pencil icon) on the Code Review view's top menu bar to go back to the Code Review Preferences.



__ f. Recheck J2SE Best Practices folder.



☑ Resource Management (41 rules, 41 enabled)
☑ J2SE Best Practices (71 rules, 71 enabled)
☑ AWT (1 rule, 1 enabled)
☑ Casting (2 rules, 2 enabled)
☑ Cloneable (3 rules, 3 enabled)
☑ Comparison (5 rules, 5 enabled)

____ 15. **Use Search Box in Code Review Preferences.**

__ a. Just above the **Expand All** button is the **Show Search Box** button. Click the **Show Search Box** button. The **Search for:** textbox will appear above the rules. Notice after you click the **Show Search Box** button, it changes to the **Hide Search Box** button.

__ b. Type the word **throw** in the "Search for:" textbox and press the Enter (or Return) button on your keyboard. Notice there are more rules with the keyword "throw" than the warnings returned when you searched the Code Review ran on WebSphereBankEJB for the keyword "throw".

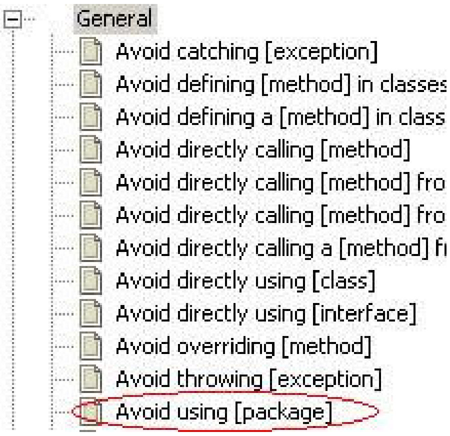
__ c. **Delete** the word throw from the search for: textbox and press the **Enter** (or Return) button on your keyboard. This clears the search and returns the view back to listing all the rules.

__ d. Click the **Hide Search Box** button to make it switch back to the name "Show Search Box".

____ **16. Create a new rule in Code Review Preferences.**

__ a. Above the **Show Search Box** button is the **New Rule** button. Click the **New Rule** Button.

__ b. In the New Code Review Rule window, select **avoid using [package]** under the General category.



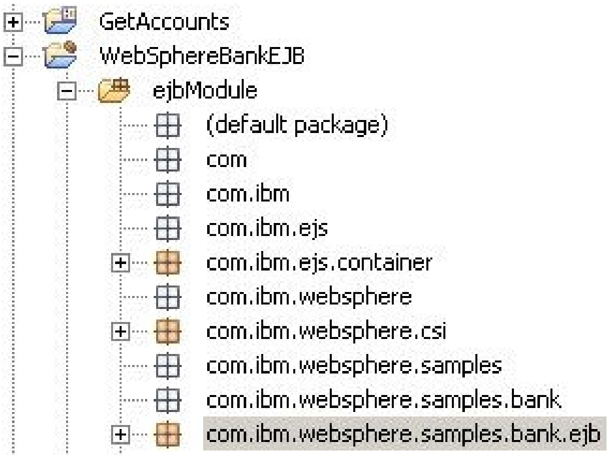
General
- Avoid catching [exception]
- Avoid defining [method] in classes
- Avoid defining a [method] in class
- Avoid directly calling [method]
- Avoid directly calling [method] fro
- Avoid directly calling [method] fro
- Avoid directly calling a [method] fi
- Avoid directly using [class]
- Avoid directly using [interface]
- Avoid overriding [method]
- Avoid throwing [exception]
- Avoid using [package]

__ c. Click the **Next** Button.

__ d. Under Basic Properties, leave this rule in the **User Defined** category. Note that you can add this new rule to the other categories.

__ e. For Severity, select **Warning** from the drop down list.

__ f. For the Package text box, first click the Browse button. Navigate to WebSphereBankEJB/ejbModule/com.ibm.websphere.samples.bank.ejb. Click **OK** button.



GetAccounts
WebSphereBankEJB
- ejbModule
  - (default package)
  - com
  - com.ibm
  - com.ibm.ejs
  - com.ibm.ejs.container
  - com.ibm.websphere
  - com.ibm.websphere.csi
  - com.ibm.websphere.samples
  - com.ibm.websphere.samples.bank
  - com.ibm.websphere.samples.bank.ejb

__ g. click the **Finish** button.

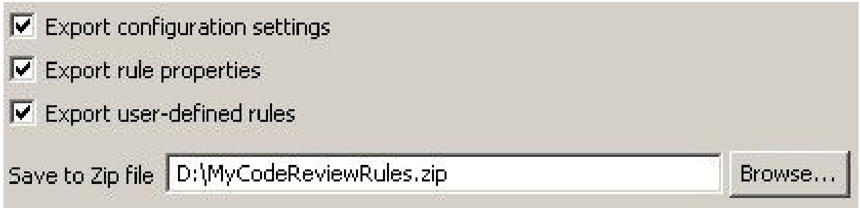
__ h. Notice that a new User Defined rule has been added to the bottom of the Code Review rule list. If you run Code Review again, this new rule will be picked (you will see a warning every time the com.ibm.websphere.samples.bank.ejb package is used).

____ 17. Use Export / Import in Code Review Preferences.

__ a. A team using IBM Rational Application Developer v6.0 might need to exchange their rules. The way to do this is to export and import your rules.

__ b. To start with export, click the Export button in the Code Review Preferences.

__ c. Click the checkboxes for all three options, Export configuration settings, Export rule properties, Export user-defined rules.



☑ Export configuration settings

☑ Export rule properties

☑ Export user-defined rules

Save to Zip file D:\MyCodeReviewRules.zip

__ d. Click the Browse button for any location for the .zip file you will create (rules are exported as .zip files).

__ e. Click **Finish** button. (Optional – go to the location of your exported rules file, unzip, and check out what is inside.)

__ f. For import, click the Import button and select the exported rules (in .zip file format).

Note: Code Review will close when you close IBM Rational Application Developer v6.0. The next time you open the workspace, the Code Review views will be there, however you will have to rerun Code Review to get results when you open IBM Rational Application Developer v6.0,

What you did in this exercise

In this lab we discovered how to use the Code Review view within WebSphere in order to find problem areas that may lead to logical errors.

This page is intentionally left blank.