Run a code review

This tutorial shows you some of the code review features. It is written for software developers.

Time Required

To simply read through this tutorial you will need approximately **15 minutes**. To do the exercise using the supplied sample project, you will need approximately **30 minutes**.

Prerequisites

In order to complete this tutorial, you should be familiar with developing Java software applications. It will also help if you understand how to use the perspectives and views in the IBM Rational Software Development Platform.

Learning Objectives

This tutorial is divided into sections that you should take in sequence. You will read about the benefits of automated code reviews and learn how to perform the following tasks:

- Run a code review
- Apply a supplied quick fix to resolve a problem in the code

When you are ready, begin “Overview of code review.”

Overview of code review

Purpose

Code review is a set of rules that automates the process for a software developer to review code. While the manual code review process can consist of time-consuming and subjective discussions, the automated code review is effective, quick, and consistent. The automated code review supplements the manual code review. It does not replace it.

Benefits

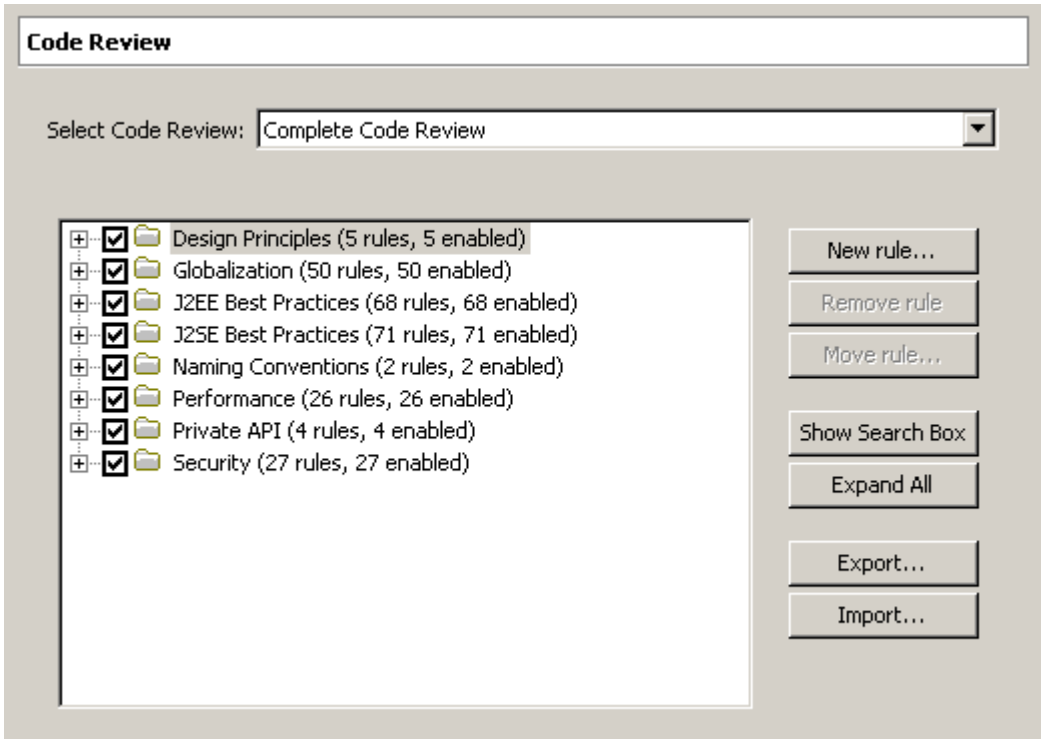
The automated code review tool adds value to the software development process in several ways because it performs the following tasks:

- Finds bugs in the code
- Checks for adherence to best practices
- Explains each finding and provides solutions for it
- Provides an automated fix for some typical findings
- Allows you to create rules to ensure that you follow the application design and standards as you write code

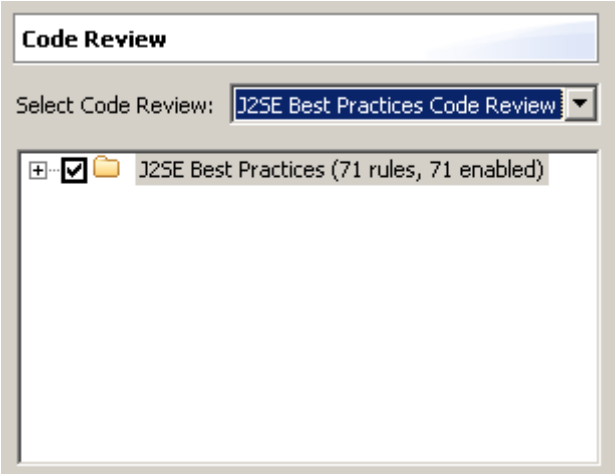
Because the automated process is quick, you can run code reviews often. The code review findings let you catch and correct problems early, when it is easiest and cheapest to make changes.

Supplied code reviews

Several code reviews are supplied. Each code review applies a different set of rules, which are organized in folders. Depending on which stage of the development process you are in and what your review objective is, you can choose the code review to suit your needs. The broadest review is the Complete Code Review that applies rules from all categories, as shown in the following screen capture:



Some categories also have a code review associated with them. For example, you could select the J2SE Best Practices Code Review when you want to apply rules for that category only, as shown in the following screen capture below. This lets you run code reviews that focus on a particular aspect of the code.



User-defined code reviews

You can create rules from a supplied wizard. The wizard lets you choose from two types of rules: general and J2EE best practices. These rules allow software developers to extend the capabilities of code review by creating rules to ensure the integrity of the code.

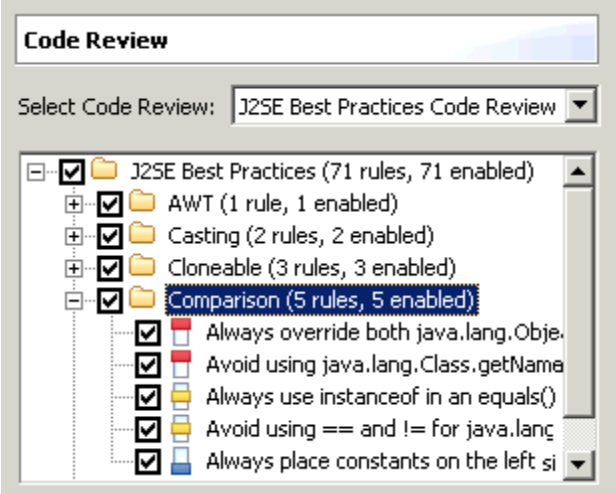
Severity levels for rules

Each rule has a severity level. For a supplied rule, you can modify the severity level assigned to it. When you create a rule from the wizard, you specify a severity level. The three severity levels are indicated by the following icons:

- Problem ( ): This finding must be addressed.
- Warning ( ): This finding is likely a problem that needs to be addressed.
- Recommendation ( ): This finding is not serious yet, but it is highly recommended that you address it now.

Although "recommendation" is the lowest severity level, do not discount how important it is to address these findings. They reflect a set of best practices and industry standards that engineering teams should adhere to. Even if these findings are not immediate problems, they could lead to problems in the future.

The following screen capture shows rules in the Comparison folder of the J2SE Best Practices Code Review. Rules in the folder have all three severity levels.



Automated fixes for some problems

Some common findings come with a quick fix, which is a supplied automated solution. If a finding in the code review has a quick fix, it is indicated by one of the icons in the following illustration:

Summary

In the software development life cycle, code review automates the peer review process on a body of code. The supplied code reviews allow you to run the following types of reviews:

- Broad, complete code reviews that apply a wide range of rules from all categories to a code base
- Narrow, focused code reviews that apply rules from one or more specific categories, such as globalization, design principles or both

You can also use a supplied wizard to create your own rules that are specific to ensuring the integrity of your application's design structure.

Because automated code reviews are quick to run, you can detect problems and inconsistencies in a code base early. Consequently, you can fix these problems early before they affect your application's maintenance, scalability, and performance.

Now you are ready to begin Exercise 1.1: Importing the required resources.

Exercise 1.1: Importing the required resources

This exercise tells you how to import the sample project, CodeReview_Examples. You use the sample project to do Exercise 1.2: Running a code review and applying a quick fix.

Unzipping the sample project

The sample project for this tutorial is included in a ZIP file. The following steps lead you through extracting files from that ZIP file into your Workspace folder.

1. Navigate to <*installdir*>\rad\eclipse\plugins\com.ibm.r2a.rad.tutorial.doc_6.0.1.0\resources where the ZIP file, CodeReview_Examples, is located.
2. Extract CodeReview_Examples to <*installdir*>\updater\eclipse\workspace. The sample project files are extracted in your Workspace folder so you can import them.

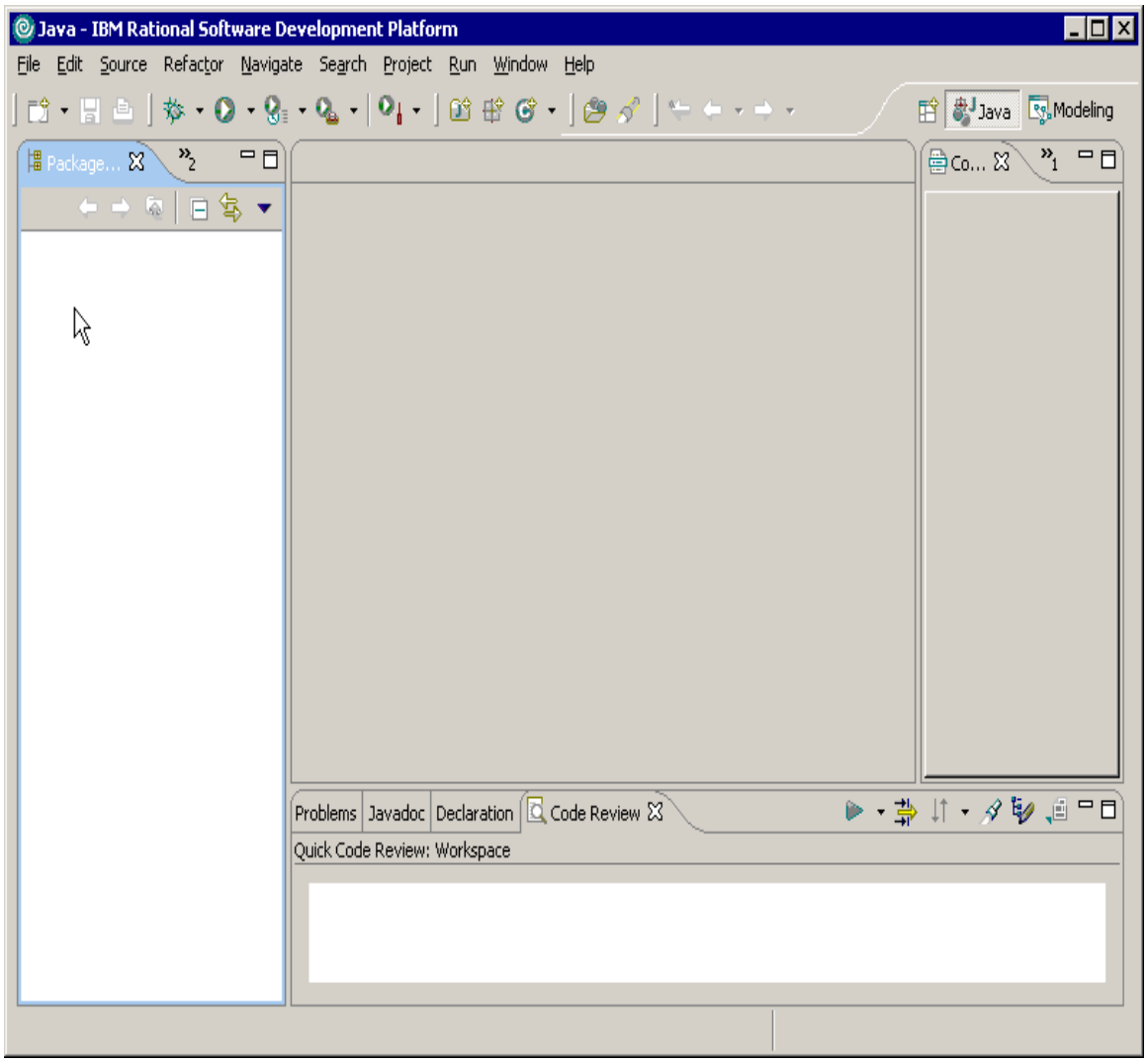
Opening the Code Review view

To open a perspective showing the Code Review view:

1. Start IBM Rational Software Development Platform.

2. Click **Window > Preferences**.
3. In the left pane expand **Workbench** and click **Capabilities**.
4. In the **Capabilities** list click **Java Developer**. Then click **OK**.
5. Click **Window > Open Perspective > Java**.
6. Click **Window > Show View > Other > Java > Code Review**.
7. Click **Window > Show View > Other > Java > Package Explorer**.

After you open the Java perspective and show the Code Review and Package Explorer views, the perspective shows the views in the following screen capture. Your layout might differ. That is, the perspective might show the views in different locations. The tutorial uses the layout in the screen capture.



Importing the sample project

To import the sample project to the workspace:

1. Right-click in the Package Explorer view to open the pop-up menu. Then click **Import** to open the Import wizard.
2. In the **Select** list click **Existing Project into Workspace**. Then click **Next**.
3. Next to the **Project contents** text box click **Browse** and select <*installdir*>\updater\eclipse\workspace\CodeReview_Examples.

4. Click **Finish** The sample project and all its associated files are imported to Package Explorer.

Beginning the exercise

To begin go to Exercise 1.2: Running a code review and applying a quick fix.

Exercise 1.2: Running a code review and applying a quick fix

This exercise assumes you have completed Exercise 1.1: Importing the required resources. In the exercise you read a user scenario first. Then you assume the role of the software developer described in the user scenario.

User scenario

A large group of geographically dispersed developers is coding a new software application. It is important that the developers routinely run code reviews to check for problems in their code.

One of the developers wants to run a code review to see how the code is doing in general. To review newly written code to assess adherence to best practices in several areas, the developer runs an automated quick code review. This review applies several categories of supplied rules to code. Each category of rules checks the quality of the code in a specific area, such as performance.

When the code review finishes, you see a list of findings. Each finding represents a string of code that does not adhere strictly to an applied rule. One of the findings has a quick fix available for it, so the developer applies the automated solution and corrects the problem right away.

In the first part of the exercise, you perform the following tasks to run a code review:

1. Select a code review to run.
2. View the rules applied in the code review.
3. Choose what code to run the review on.
4. Run the code review.
5. View the findings of the code review.
6. Select a finding to see the following information for it:
    - Source code
    - Description, examples, and solutions

Next, to apply a quick fix to one finding in the code review you perform the following tasks:

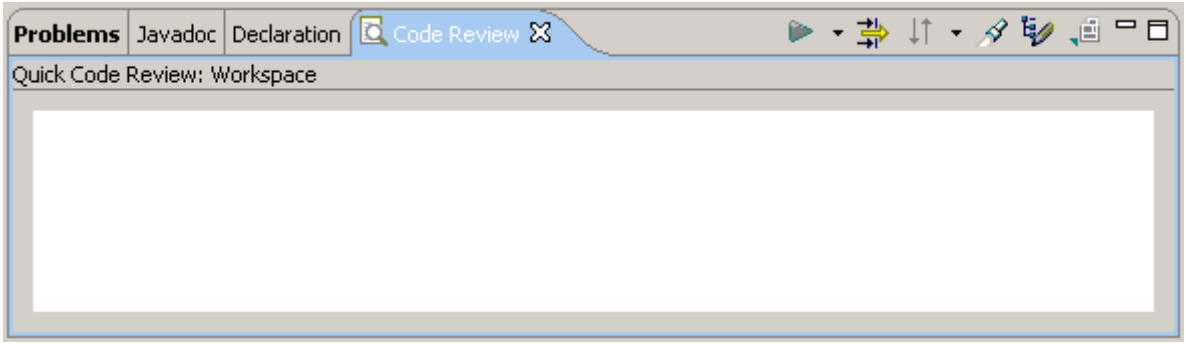
1. Recognize when a quick fix is available for a finding.
2. See a list of changes that the quick fix will make to the code.
3. Preview the original and refactored code before you apply the quick fix.
4. Apply the quick fix to refactor the code.
5. Get a confirmation after the quick fix has been applied.

Exercise

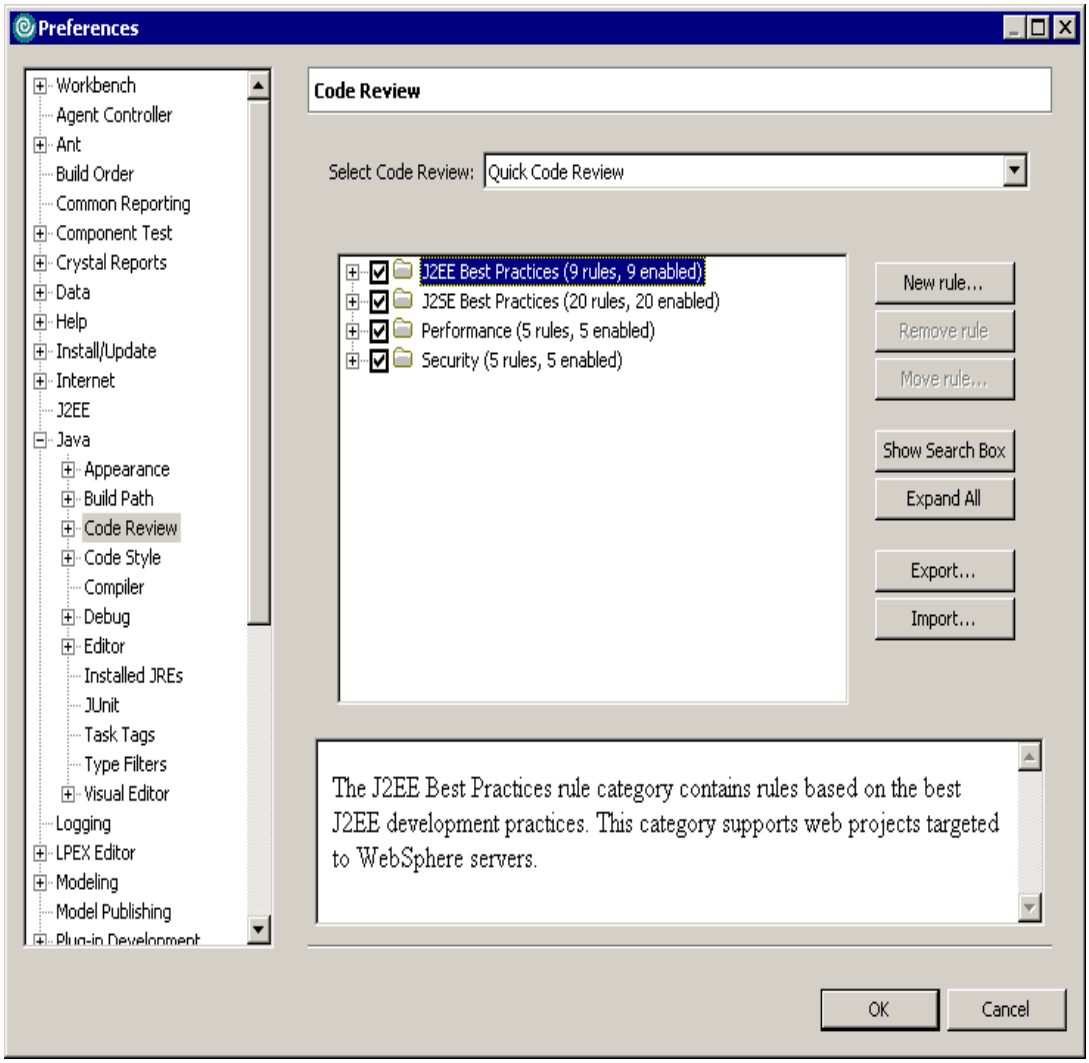
Selecting a code review

To select a quick code review:

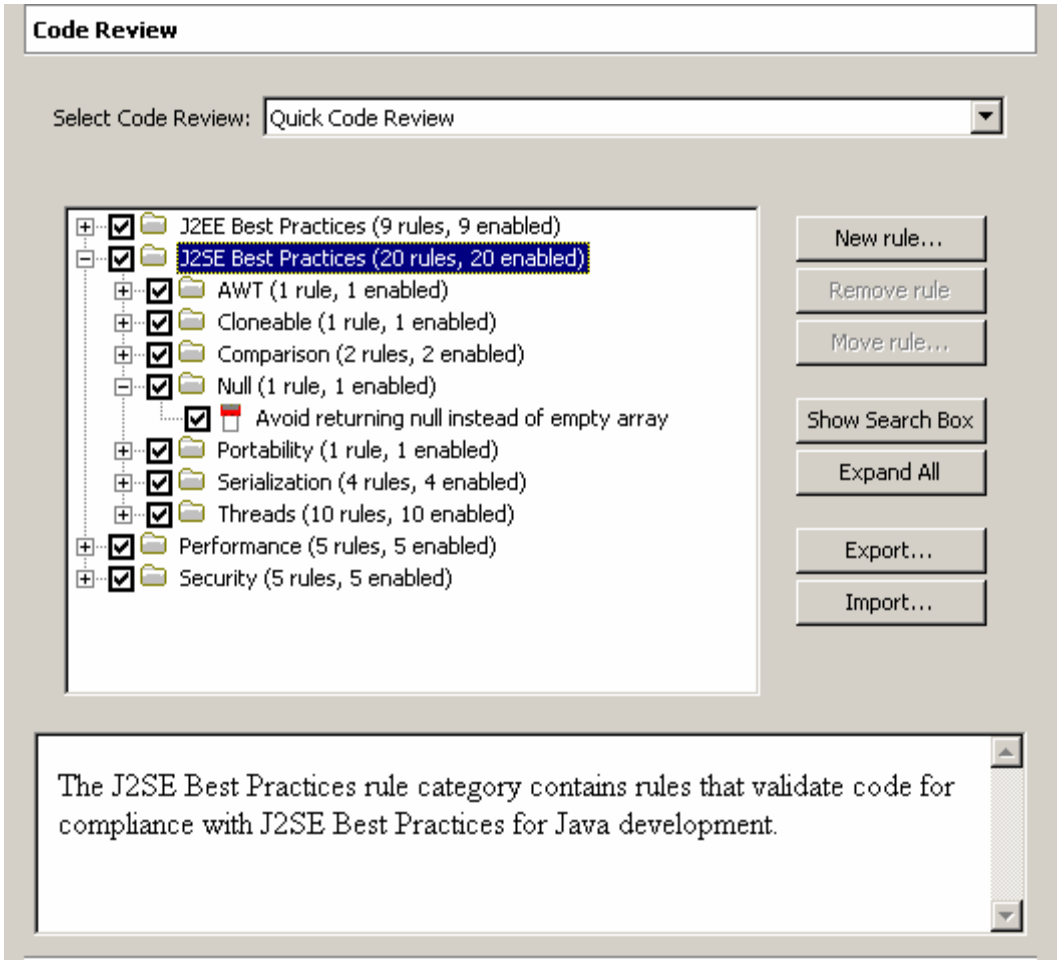
1. On the toolbar in the Code Review view click the **Manage Rules** icon,  .



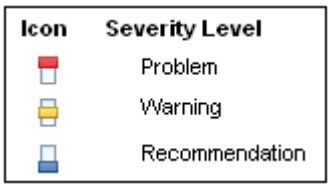
2. In the **Select Code Review** list, click **Quick Code Review**. The folders of rules for the code review you selected are displayed, as shown in the following screen capture:



3. To see one of the rules that will be applied in the code review, expand the **J2SE Best Practices** folder and then the **Null** subfolder. The Null folder shows one rule with a problem severity level, as shown in the following screen capture:



As a review, the severity level icons are shown in the following illustration:



4. Click **OK** to choose the Quick Code Review.

Selecting a code base to review

To select the project as the code base to review:

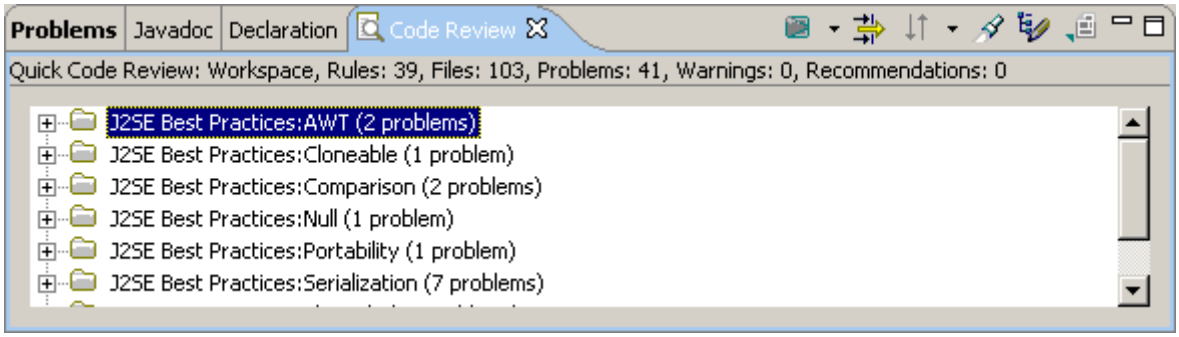
- On the toolbar in the Code Review view click the **Review** icon ( ) > **Projects** > **Review CodeReview_Examples**.

Running the code review

Once you select the code base to review, the code review runs. You can track its status by checking the progress bar in the lower-right corner of the view.

Viewing the code review findings

When the code review is finished, the findings are shown in the Code Review view,as shown in the following screen capture:



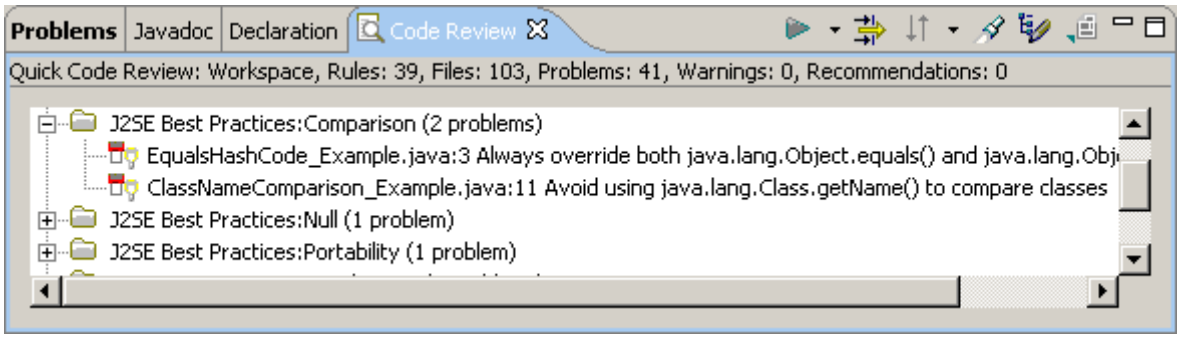
The Code Review view provides the following information:

- Code review statistics: The line above the findings displays information about the most recent code review: type, scope, number of rules and files included, and number and severity of findings.
- Code review findings: The findings from the code review are listed in the Code Review view, within folders. Each folder name tells you the category of rules applied and the number of findings.

Getting more information on a code review finding

To get more information on a finding in the code review:

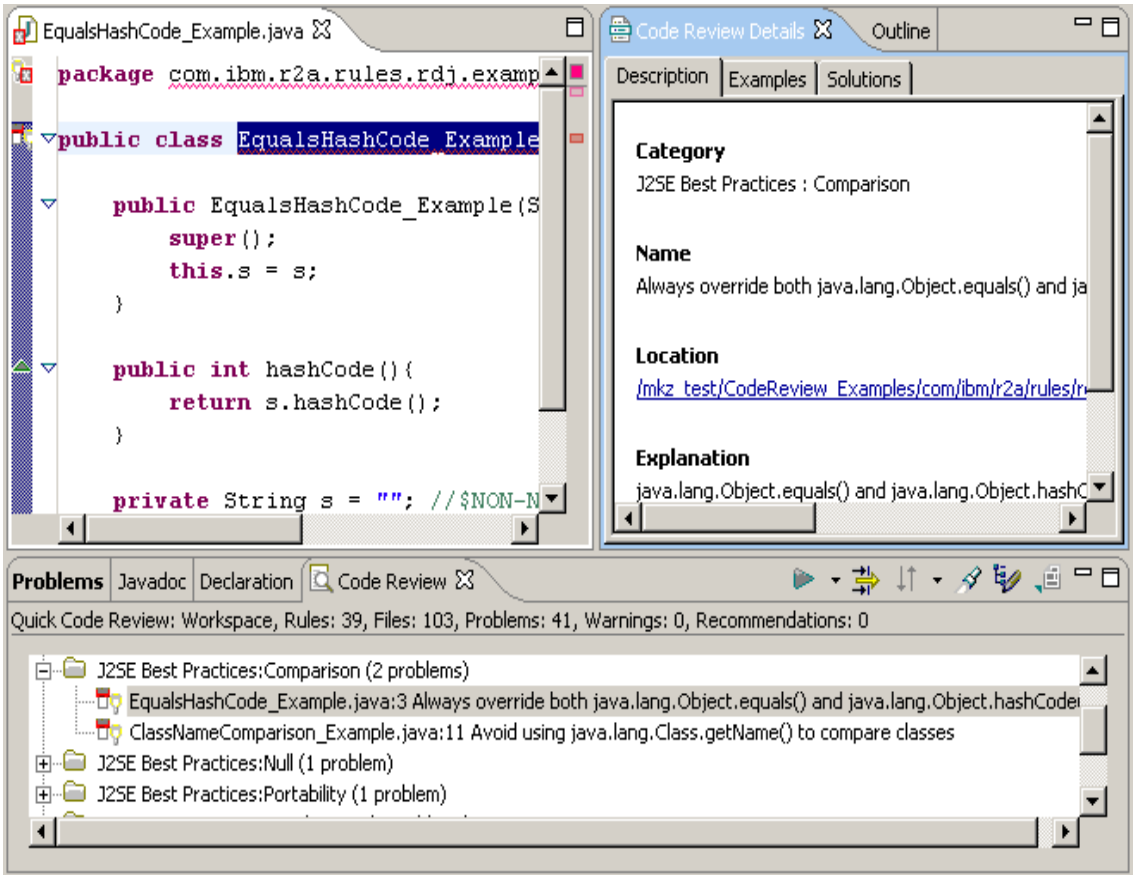
1. In the Code Review view, scroll to the **J2SE Best Practices: Comparison** folder. Then expand the folder to show the findings in it, as shown in the following screen capture:



2. The first finding begins with EqualsHashCode_Example.java. The rule that was applied is noted after it:

   Always override both java.lang.Object.equals() and java.lang.Object.hashCode()

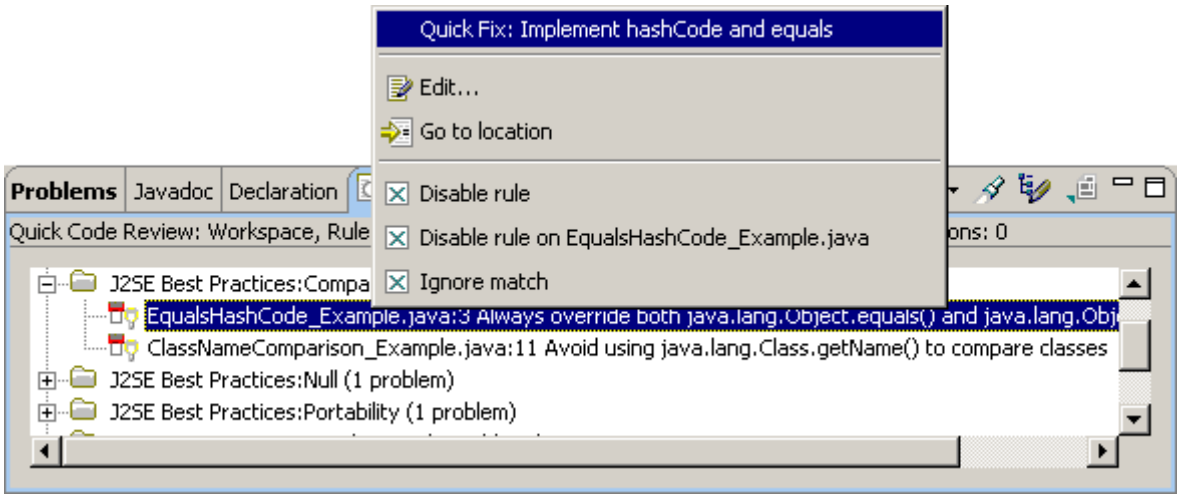
3. Double-click the first finding. Details about it appear in two places, as outlined in the following points and screen capture:
    - Source code: Displays the code where the finding occurs and highlights the exact location of it.
    - Code Review Details view: Describes the finding in more detail and provides examples and solutions to correct it.



Selecting a finding that has a quick fix

You can tell that both findings in the Best Practices: Comparison folder have a quick fix by their icons. As a review, the quick fix icons are shown in the following illustration:

1. Right-click the first finding in the list, as shown in the next screen capture.
2. The **Quick Fix** pop-up menu choice varies depending upon the solution. For the finding you selected, the fix is to implement hashCode and equals.



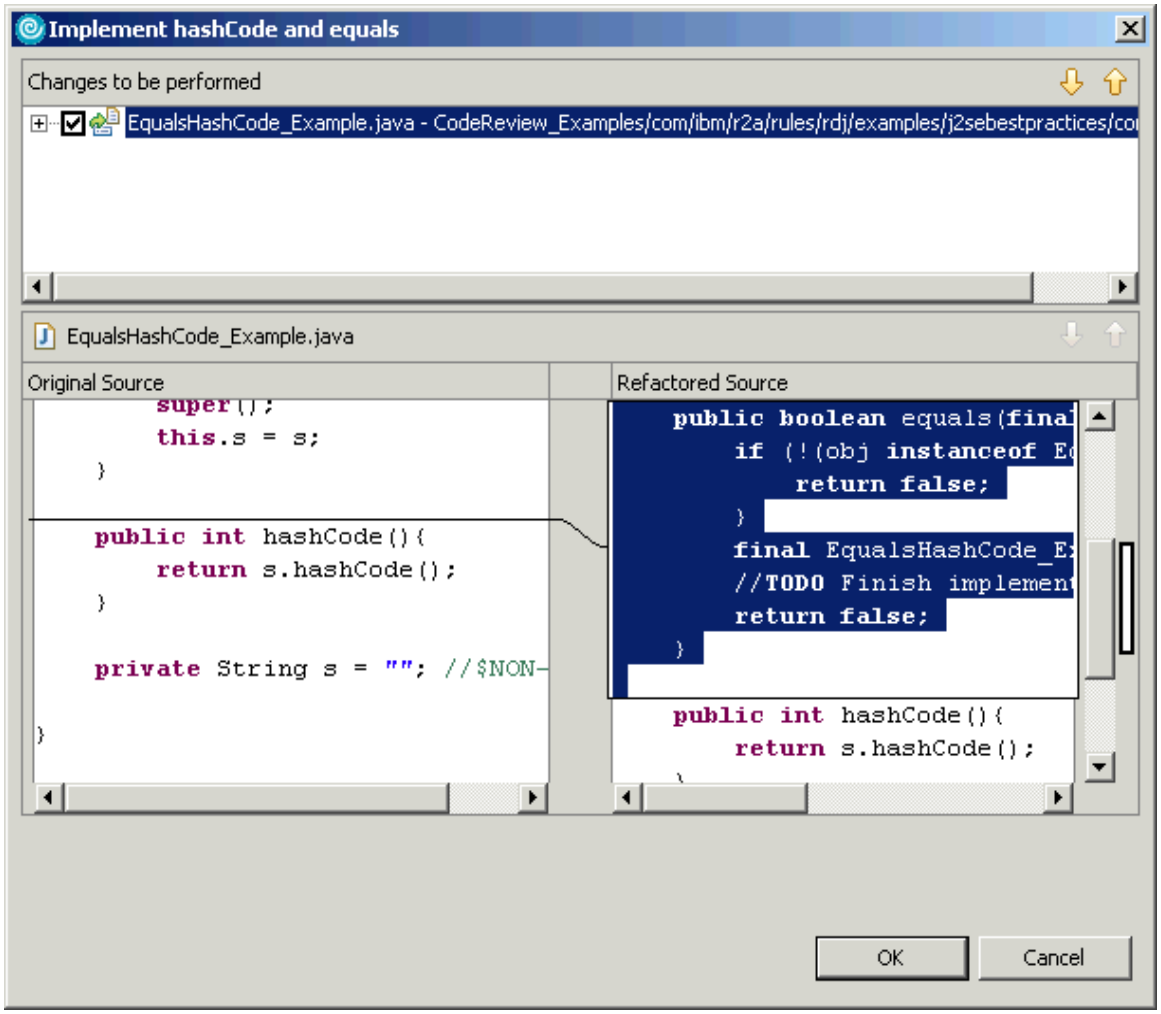
3. Click **Quick Fix: Implement hashCode and equals**.

Applying the quick fix

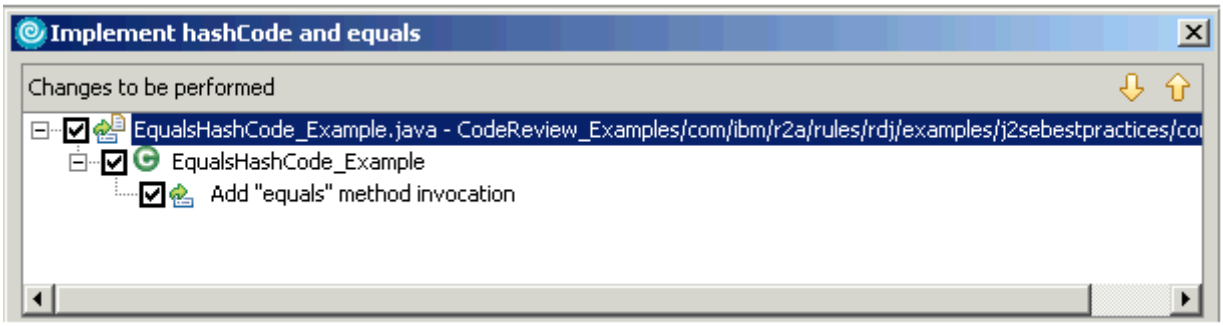
The quick fix for the finding you selected is to implement hashCode and equals.

To review and apply the quick fix to the finding:

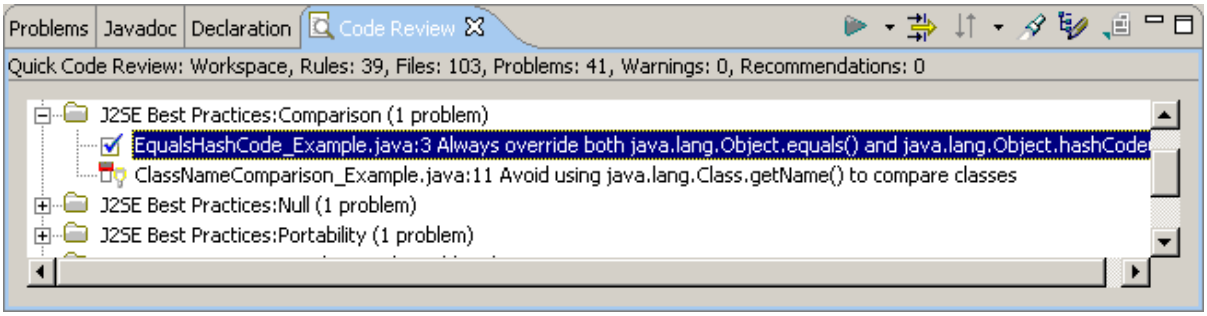
1. You see a side-by-side view of the code, as shown in the following screen capture. The original source code is on the left and the refactored source code that would be created by the quick fix is on the right. If you decide to apply the quick fix, it will append the missing lines of code that are highlighted.



2. In the **Changes to be performed** section expand the list to see exactly what the quick fix will change and how, as shown in the following screen capture:



3. Review the changes in the list. Then click **OK** to apply the quick fix to all the selected changes in the list.
4. After the quick fix has been applied, you see a checkmark next to the finding you resolved.



Next to the checkmark you see the following information:

- The quick fix you applied
- The line number in the source code where the finding is located
- The rule in the code review that had not been adhered to

You have completed Exercise: Running a code review and applying a quick fix.

Exercise wrap-up

You have performed all the tasks in Exercise: Running a code review and applying a quick fix.

Tasks for running a code review

When you ran this code review, you performed the following tasks:

1. Selected a code review to run.
2. Viewed rules applied in the code review.
3. Chose a body of code to run the review on.
4. Ran the code review.
5. Viewed the findings of the code review.
6. Selected a finding to see the following information for it:
    - Source code.
    - Description, examples, and solutions.

Tasks for applying a quick fix

When you applied the quick fix, you performed the next set of tasks:

1. Recognized when a quick fix is available for a finding.
2. Saw a list of changes that the quick fix would make to the code.
3. Previewed the original and refactored code.
4. Applied the quick fix to refactor the code.
5. Got a confirmation that the quick fix had been applied.

Leveraging the power of a code review

By proactively running code reviews, you are able to analyze the findings early. This means you can also address them early, before they lead to the following problems:

- Affect your application's performance, maintenance, or scalability
- Cost your company money, time, and resources

Leveraging the power of a quick fix

By applying a supplied quick fix, you have an automated way to resolve a common finding. Quick fixes help you in the following ways:

- Correct a problem consistently each time
- Free you up to code and spend less time fixing bugs

Finish the tutorial by reviewing the learning objectives in Summary: Running a code review.

Exercise 1.2: Running a code review and applying a quick fix

This exercise assumes you have completed Exercise 1.1: Importing the required resources. In the exercise you read a user scenario first. Then you assume the role of the software developer described in the user scenario.

User scenario

A large group of geographically dispersed developers is coding a new software application. It is important that the developers routinely run code reviews to check for problems in their code.

One of the developers wants to run a code review to see how the code is doing in general. To review newly written code to assess adherence to best practices in several areas, the developer runs an automated quick code review. This review applies several categories of supplied rules to code. Each category of rules checks the quality of the code in a specific area, such as performance.

When the code review finishes, you see a list of findings. Each finding represents a string of code that does not adhere strictly to an applied rule. One of the findings has a quick fix available for it, so the developer applies the automated solution and corrects the problem right away.

In the first part of the exercise, you perform the following tasks to run a code review:

1. Select a code review to run.
2. View the rules applied in the code review.
3. Choose what code to run the review on.
4. Run the code review.
5. View the findings of the code review.
6. Select a finding to see the following information for it:
    - Source code
    - Description, examples, and solutions

Next, to apply a quick fix to one finding in the code review you perform the following tasks:

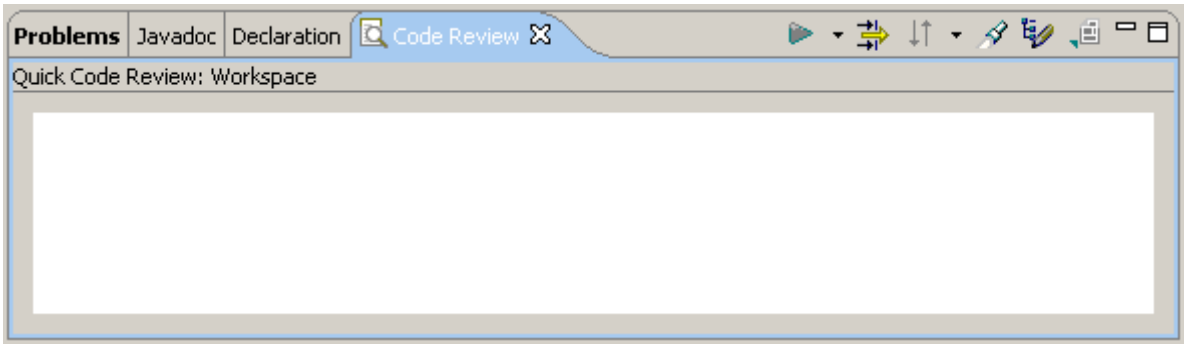
1. Recognize when a quick fix is available for a finding.
2. See a list of changes that the quick fix will make to the code.
3. Preview the original and refactored code before you apply the quick fix.
4. Apply the quick fix to refactor the code.
5. Get a confirmation after the quick fix has been applied.

Exercise

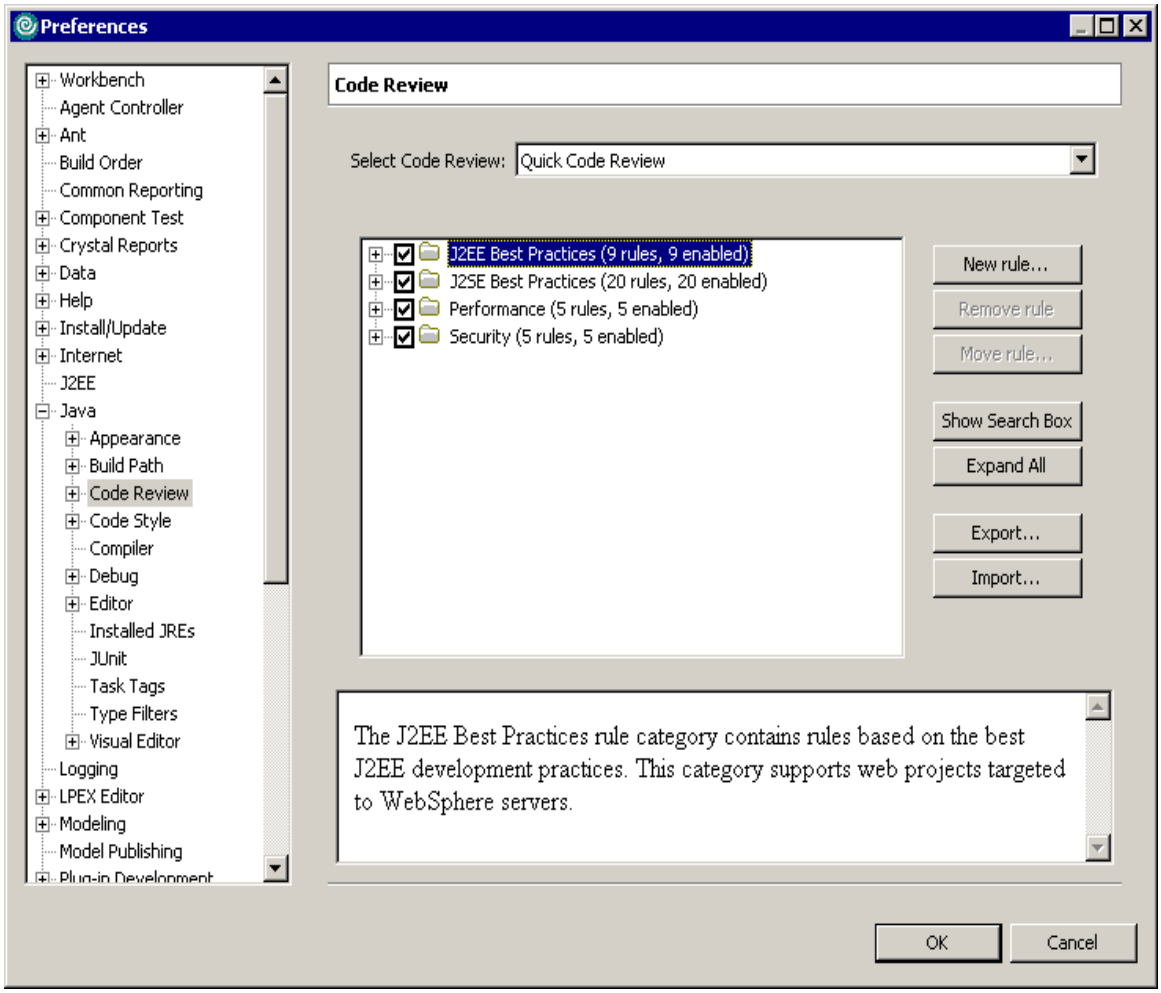
Selecting a code review

To select a quick code review:

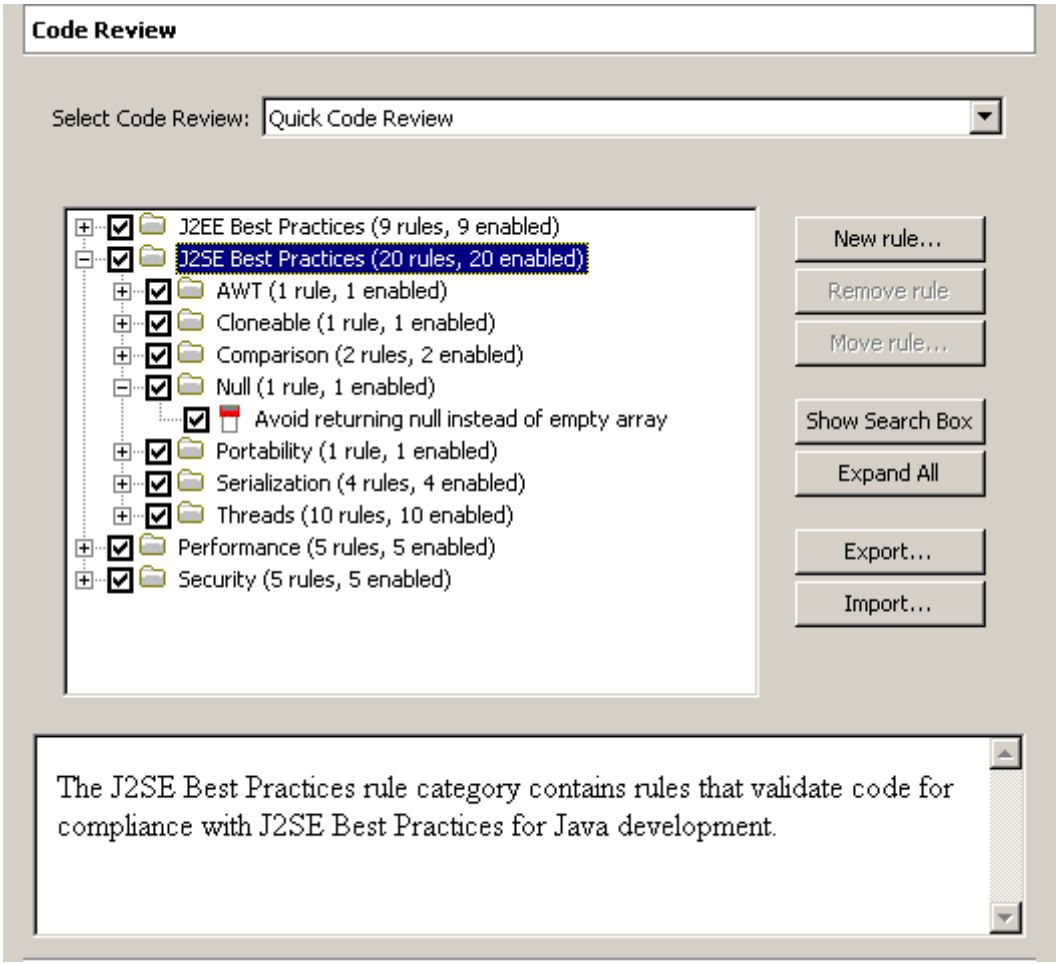
1. On the toolbar in the Code Review view click the **Manage Rules** icon,  .



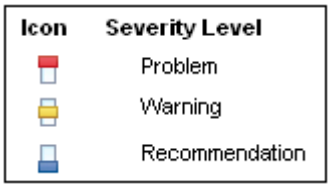
2. In the **Select Code Review** list, click **Quick Code Review**. The folders of rules for the code review you selected are displayed, as shown in the following screen capture:



3. To see one of the rules that will be applied in the code review, expand the **J2SE Best Practices** folder and then the **Null** subfolder. The Null folder shows one rule with a problem severity level, as shown in the following screen capture:



As a review, the severity level icons are shown in the following illustration:



4. Click **OK** to choose the Quick Code Review.

Selecting a code base to review

To select the project as the code base to review:

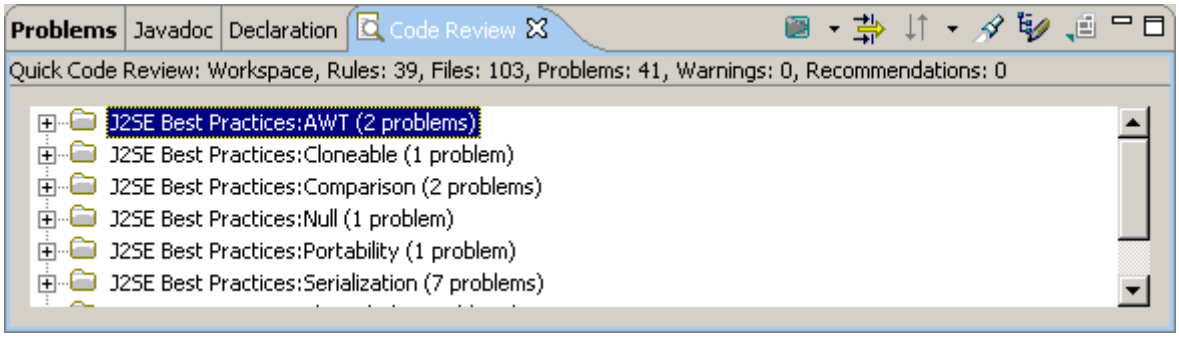
- On the toolbar in the Code Review view click the **Review** icon ( ) > **Projects** > **Review CodeReview_Examples**.

Running the code review

Once you select the code base to review, the code review runs. You can track its status by checking the progress bar in the lower-right corner of the view.

Viewing the code review findings

When the code review is finished, the findings are shown in the Code Review view,as shown in the following screen capture:



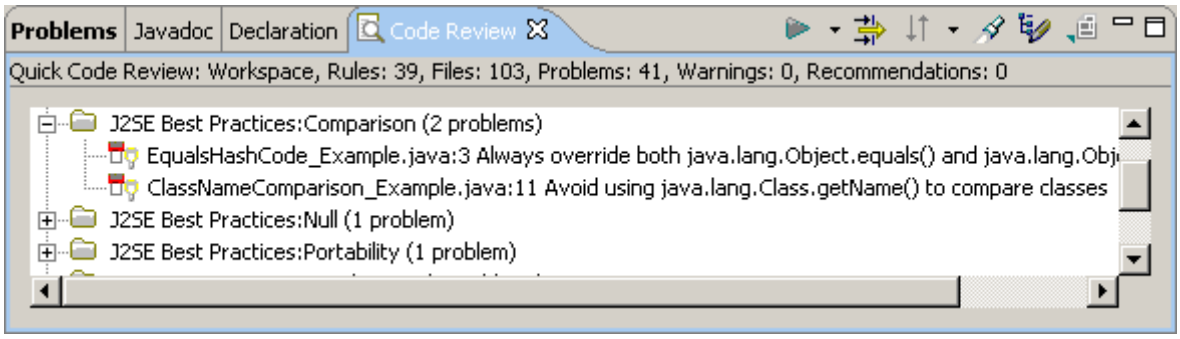
The Code Review view provides the following information:

- Code review statistics: The line above the findings displays information about the most recent code review: type, scope, number of rules and files included, and number and severity of findings.
- Code review findings: The findings from the code review are listed in the Code Review view, within folders. Each folder name tells you the category of rules applied and the number of findings.

Getting more information on a code review finding

To get more information on a finding in the code review:

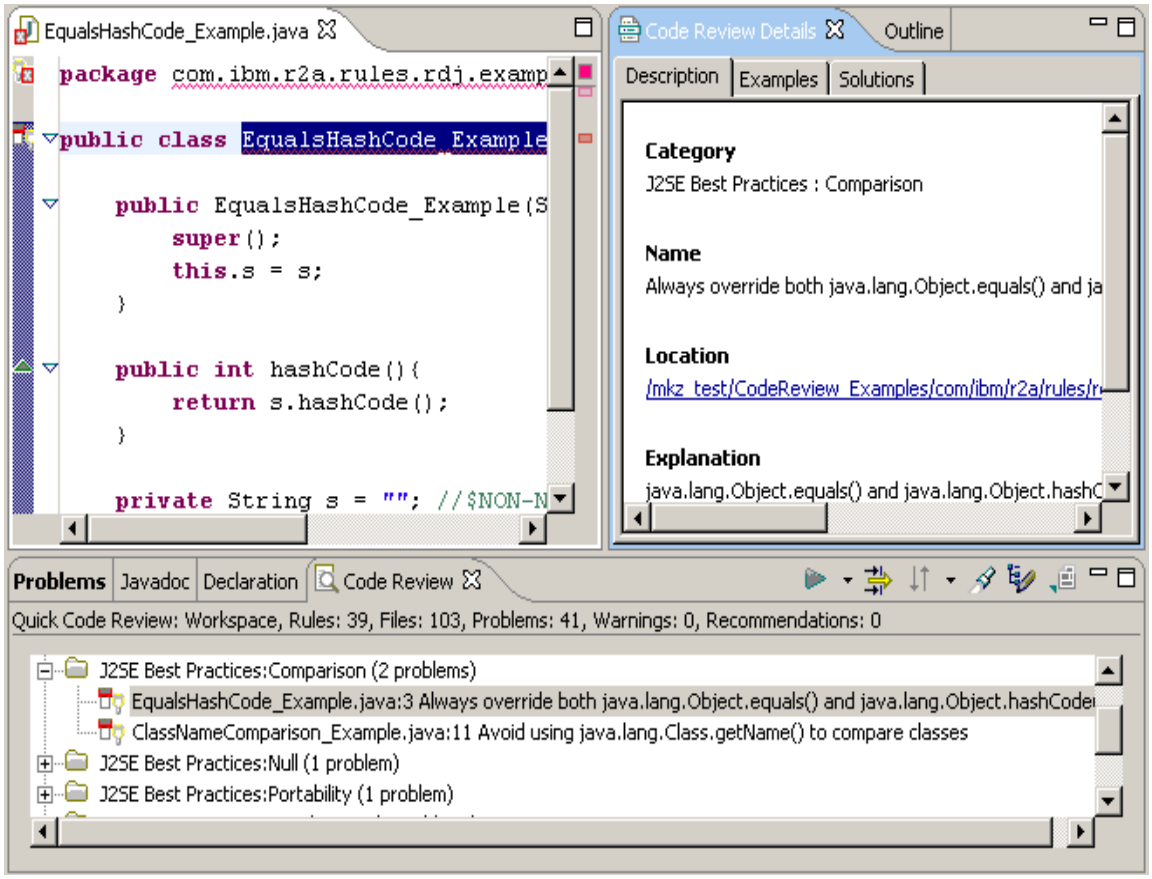
1. In the Code Review view, scroll to the **J2SE Best Practices: Comparison** folder. Then expand the folder to show the findings in it, as shown in the following screen capture:



2. The first finding begins with EqualsHashCode_Example.java. The rule that was applied is noted after it:

   Always override both java.lang.Object.equals() and java.lang.Object.hashCode()

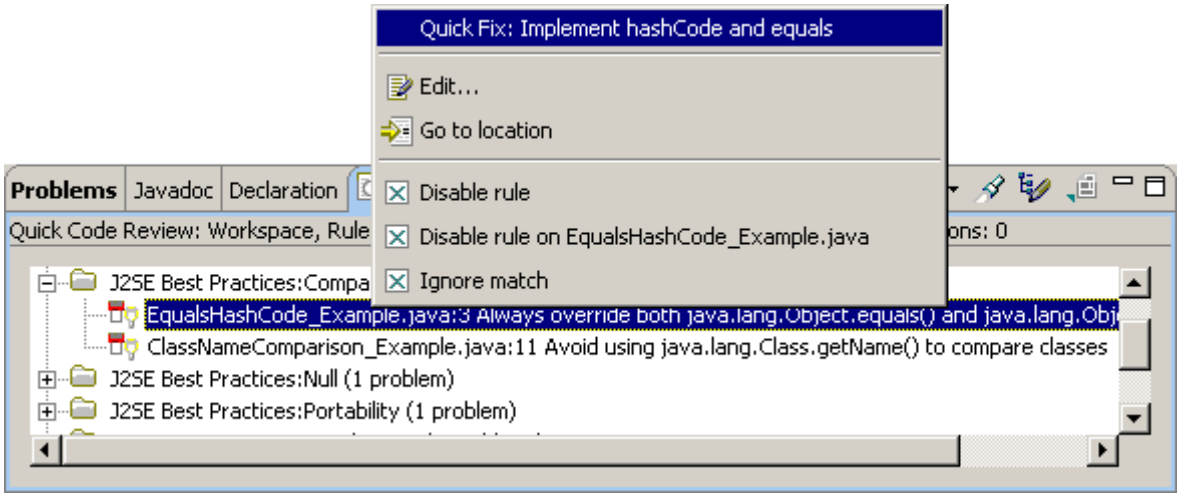
3. Double-click the first finding. Details about it appear in two places, as outlined in the following points and screen capture:
    - Source code: Displays the code where the finding occurs and highlights the exact location of it.
    - Code Review Details view: Describes the finding in more detail and provides examples and solutions to correct it.



Selecting a finding that has a quick fix

You can tell that both findings in the Best Practices: Comparison folder have a quick fix by their icons. As a review, the quick fix icons are shown in the following illustration:

1. Right-click the first finding in the list, as shown in the next screen capture.
2. The **Quick Fix** pop-up menu choice varies depending upon the solution. For the finding you selected, the fix is to implement hashCode and equals.



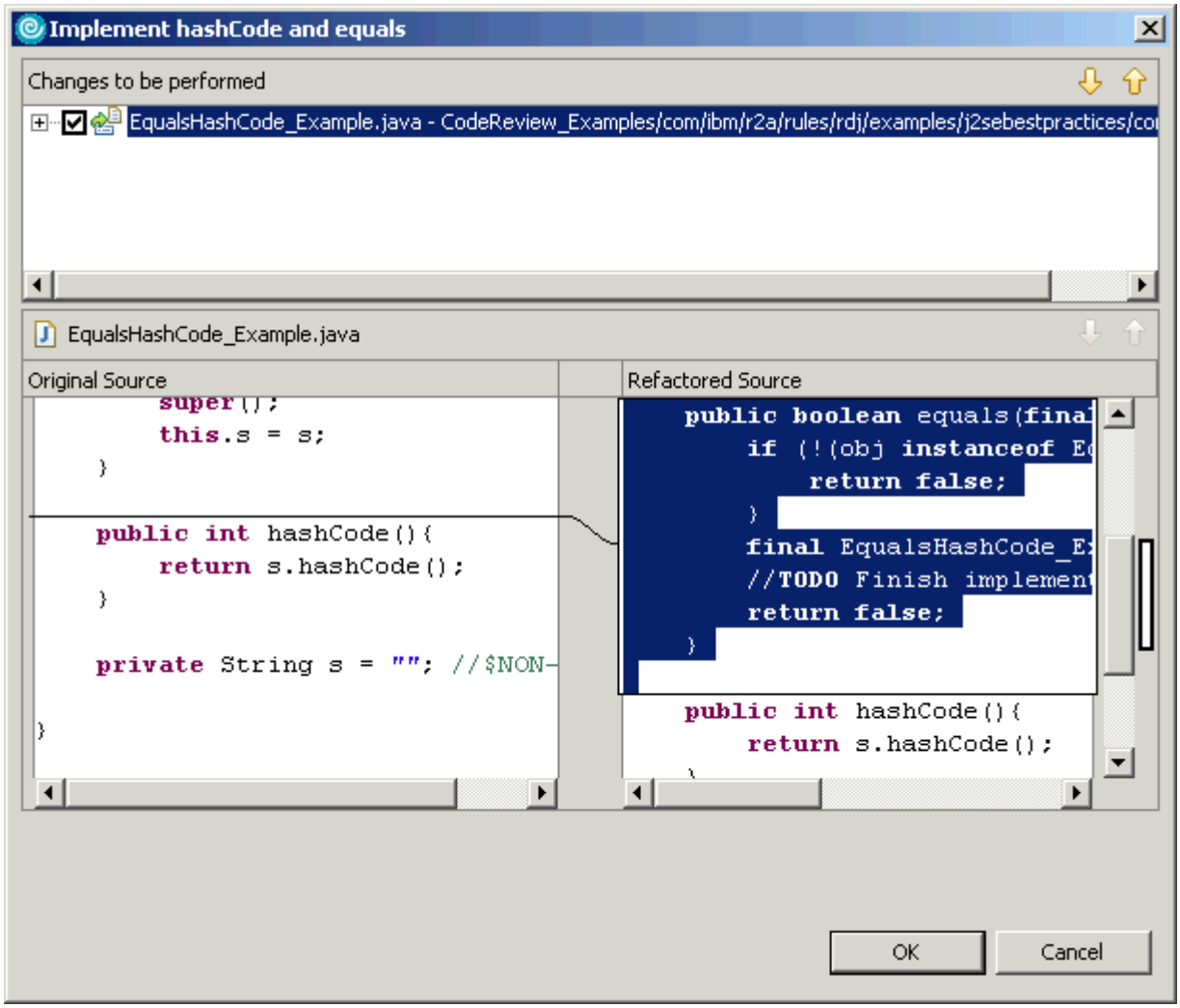
3. Click **Quick Fix: Implement hashCode and equals**.

Applying the quick fix

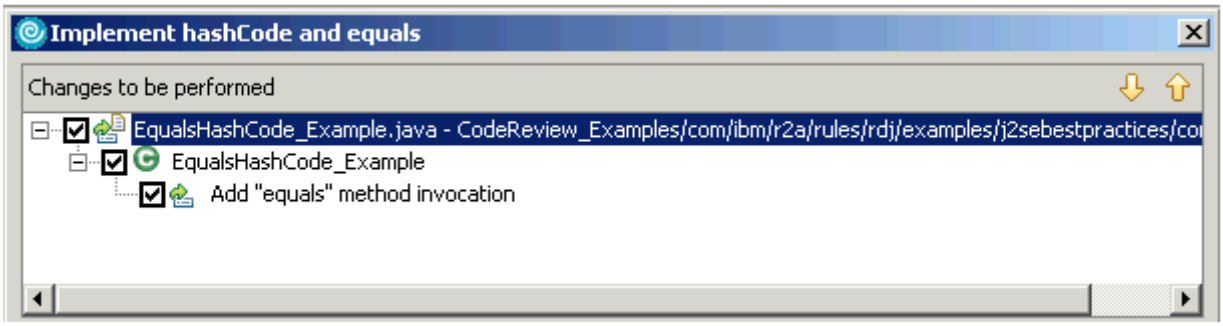
The quick fix for the finding you selected is to implement hashCode and equals.

To review and apply the quick fix to the finding:

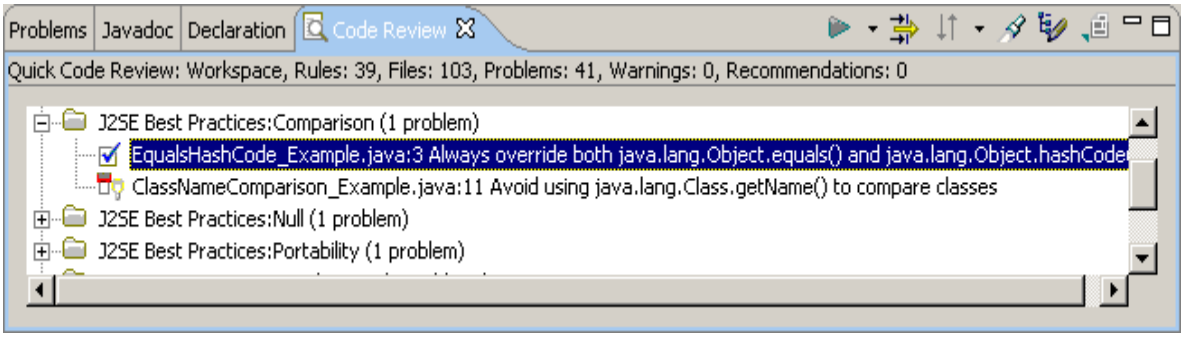
1. You see a side-by-side view of the code, as shown in the following screen capture. The original source code is on the left and the refactored source code that would be created by the quick fix is on the right. If you decide to apply the quick fix, it will append the missing lines of code that are highlighted.



2. In the **Changes to be performed** section expand the list to see exactly what the quick fix will change and how, as shown in the following screen capture:



3. Review the changes in the list. Then click **OK** to apply the quick fix to all the selected changes in the list.
4. After the quick fix has been applied, you see a checkmark next to the finding you resolved.



Next to the checkmark you see the following information:

- The quick fix you applied
- The line number in the source code where the finding is located
- The rule in the code review that had not been adhered to

You have completed Exercise: Running a code review and applying a quick fix.

Exercise wrap-up

You have performed all the tasks in Exercise: Running a code review and applying a quick fix.

Tasks for running a code review

When you ran this code review, you performed the following tasks:

1. Selected a code review to run.
2. Viewed rules applied in the code review.
3. Chose a body of code to run the review on.
4. Ran the code review.
5. Viewed the findings of the code review.
6. Selected a finding to see the following information for it:
    - Source code.
    - Description, examples, and solutions.

Tasks for applying a quick fix

When you applied the quick fix, you performed the next set of tasks:

1. Recognized when a quick fix is available for a finding.
2. Saw a list of changes that the quick fix would make to the code.
3. Previewed the original and refactored code.
4. Applied the quick fix to refactor the code.
5. Got a confirmation that the quick fix had been applied.

Leveraging the power of a code review

By proactively running code reviews, you are able to analyze the findings early. This means you can also address them early, before they lead to the following problems:

- Affect your application's performance, maintenance, or scalability
- Cost your company money, time, and resources

Leveraging the power of a quick fix

By applying a supplied quick fix, you have an automated way to resolve a common finding. Quick fixes help you in the following ways:

- Correct a problem consistently each time
- Free you up to code and spend less time fixing bugs

Finish the tutorial by reviewing the learning objectives in Summary: Running a code review.

Summary: Running a code review

This tutorial showed you how to run a code review.

Completed Learning Objectives

If you completed the exercise, you should now be able to do the following tasks:

- Run a code review.
- Apply a supplied quick fix to resolve a problem.

More information

If you want to learn more about the topics covered in this tutorial, please refer to the online Help for running code reviews.