FUJITSU

FUJITSU Software
Agile+ Relief J V1.1.1

コードチェック操作手引書

B1WD-3623-01Z0(00)
2023年1月

まえがき

ソフトウェア開発においては、開発工程のできるだけ早い段階で品質を向上させることが後工程での手戻りを少なくし、開発コストの削減にもつながります。Agile+ Relief Jは、開発工程の中のコーディング工程における品質向上を支援する製品です。Java言語で記述されたプログラムを静的解析することにより、目視でのレビューでは見落としてしまうような問題点も徹底的に洗い出します。その結果、品質向上と開発コストの削減を同時に実現します。

Agile+ Relief Jとは

Agile+ Relief Jには、Agile+ Relief Jカスタマイザ（以降、カスタマイザと呼びます）とAgile+ Relief Jコードチェック（以降、コードチェックと呼びます）の2つのツールがあります。

- カスタマイザ

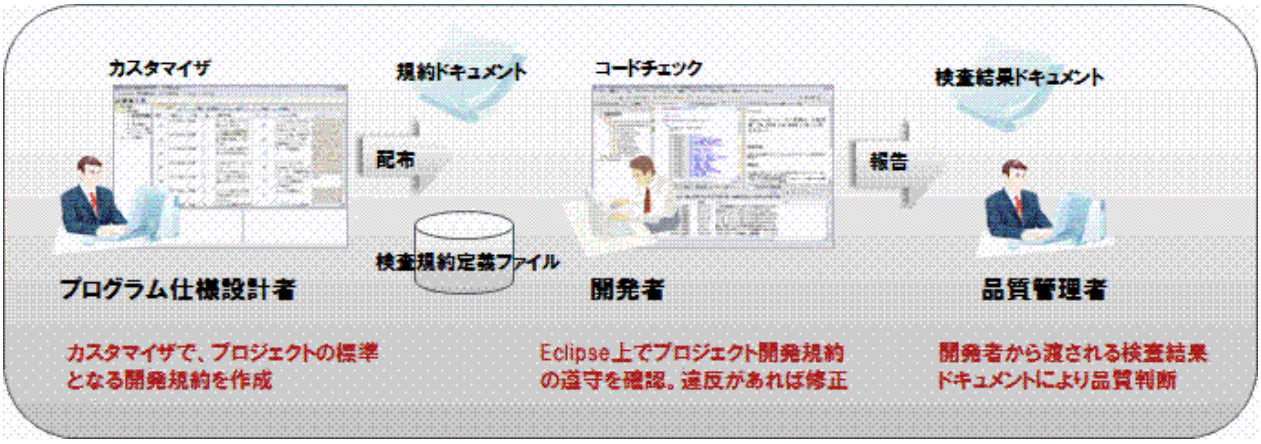
  開発プロジェクトの開発標準となるコーディング規約の作成を支援します。

- コードチェック

  Java言語で記述されたソースコードを静的解析し、カスタマイザで作成されたコーディング規約違反を検出します。

Agile+ Relief Jの利用者と役割

Agile+ Relief Jで想定する利用者とそれぞれの役割を以下に示します。



- プログラム仕様設計者

  開発プロジェクトの標準となるコーディング規約を決定します。Agile+ Relief Jの検査規約の内容を理解し、カスタマイザを使用してコーディング規約を作成します。作成したコーディング規約は、開発者および品質管理者に配布します。

- 開発者

  コーディング規約を遵守してコーディングします。コードチェックを使用して規約違反がないかをチェックし、結果をファイルに保存して品質管理者に渡します。

- 品質管理者

  コーディング規約を遵守しているかを最終的に判断します。開発者から渡されるコードチェック結果から品質に異常がないかを判断します。また、コーディング規約の妥当性についても確認します。

これらの利用者は、それぞれ違う人である必要はありません。「開発者が独自にコーディング上の決め事を作り、それをうっかり破っていないか自身で確認する」といった使い方もできます。

特に、開発者がコーディング規約を拡張したい場合には、利用者間で十分に意思疎通を図る必要があるでしょう。

本マニュアルについて

本マニュアルでは、Kiyacker互換パッケージインストール後のコードチェックの概要および操作方法について説明しています。

Kiyacker互換パッケージのインストールにより追加された機能については、各見出しに「【Kiyacker互換】」を付加しています。

なお、Kiyacker互換以外の機能説明では、画面イメージがKiyacker互換パッケージ適用前のものをそのまま使用しています。規約コードの表示イメージなど実際の画面と異なる場合がありますが、説明など内容について特に問題はありません。

カスタマイザについての詳細は、「Agile+ Relief Jカスタマイザ操作手引書」を参照してください。

本マニュアルの操作仕様は、Eclipse 3.4をベースに記載しています。他のバージョンのEclipseやInterstage Studioをご利用の場合、ご利用の開発環境に合わせてお読み替えください。

商標について

- Microsoft、Windows、Windows Serverは、米国Microsoft Corporationの米国およびその他の国における登録商標または商標です。
- Oracle とJava は、Oracle Corporation およびその子会社、関連会社の米国およびその他の国における登録商標です。文中の社名、商品名等は各社の商標または登録商標である場合があります。
- Interstageは富士通株式会社の登録商標です。
- Interstage Studio は、富士通株式会社の製品です。
- The name FindBugs™ and the FindBugs logo are trademarked by The University of Maryland
- Androidは、Google Inc.の登録商標または商標です。
- その他の名称については、各社の登録商標または商標です。

目　次

第1章 概要

ここでは、コードチェックの概要、および対象資産について説明します。

1.1 コードチェックとは

コードチェックは、プログラム仕様設計者がカスタマイザを使用して作成したコーディング規約を入力とし、Java言語で記述されたソースプログラムを静的解析することで、コーディング規約違反をチェックします。コードチェックは、Java開発環境であるEclipse上で動作する、Eclipseプラグインとして提供します。開発者は、Eclipse上でのプログラム開発を行いながら、いつでもコードチェックを実行し、コーディング規約違反についてチェックすることができます。

コードチェックでは、以下の観点からチェックが行われます。

- 信頼性

  複雑な制御構造などバグを誘発する可能性のある記述やAPIの利用誤りなどを検査する規約です。

- 保守性／可読性

  クラスやメソッドなどの命名規則やコーディング規則などの違反を検査する規約です。

- 効率性

  文字列やファイルの操作、DBやXMLへのアクセスなど性能劣化の原因となる箇所を検査する規約です。

- 機能性

  SQLインジェクションやXSS（クロスサイトスクリプティング）などのセキュリティ脆弱性を検査する規約です。

- 移植性

  OS依存コードなど他のプラットフォームへの移植で問題となる箇所を検査する規約です。

1.2 対象資産

コードチェックの対象となる資産は以下のとおりです。

1.2.1 コードチェックの対象となるEclipseプロジェクト

以下のEclipseプロジェクトがコードチェックの対象となります。

Javaプロジェクト

Javaアプリケーションの開発用に作成されたEclipseプロジェクトです。

Android Applicationプロジェクト

Androidアプリケーションの開発用に作成されたEclipseプロジェクトです。

1.2.2 コードチェックの対象となるファイル

「1.2.1 コードチェックの対象となるEclipseプロジェクト」で示すEclipseプロジェクト内の以下のファイルがコードチェックの対象となります。

Javaファイル

- 拡張子は「.java」です。
- JDK 1.4／5.0／6／7／8のJava言語仕様に準拠したJavaプログラムが対象です。
- 文字コードはMS932、EUC-JP、またはUTF-8にしてください。

- チェック対象となるJavaファイルは、すべて同じ文字コードにしてください。

XMLファイル

- 拡張子は「.xml」です。
- Androidプロジェクトの場合は、以下のXMLファイルが対象となり、Agile+ Relief Jによりチェックされます。
  - AndroidManifest.xml
  - リソースフォルダ(res/xml、res/menu、res/layout、res/layout-xxx)配下のXMLファイル
- PMDとの連携が有効で、かつPMDのバージョンが5.0以降の場合、チェック対象のフォルダ配下のすべてのXMLファイルが対象となり、PMDによりチェックされます。

JSPファイル

- 以下の拡張子を対象とします。
  - PMD 4.3以前の場合：「.jsp、.jspx」
  - PMD 5.0以降の場合：「.jsp」
- PMDとの連携が有効の場合に対象となり、PMDによりチェックされます。
- Kiyacker互換パッケージを適用した場合、Agile+ Relief Jのチェック対象となりますが、以下の条件を満たしている必要があります。
  - JSP 1.2までに準拠したJSPファイルです。
  - 文字コードはWindows-31J、EUC-JP、またはUTF-8にしてください。
  - JSPファイルの拡張子は「.jsp」です。
  - チェック対象となるJSPファイルは、すべて同じ文字コードにしてください。
  - JavaファイルとJSPファイルは同じ文字コードにしてください。ただし、JavaファイルがMS932の場合は、JSPファイルはWindows-31Jにしてください。

スタイルシートファイル

- 拡張子は「.xsl、.xslt」を対象とします。
- PMDとの連携が有効で、かつPMDのバージョンが5.0以降の場合に対象となり、PMDによりチェックされます。

ECMAScriptファイル

- 拡張子は「.js」を対象とします。
- PMDとの連携が有効で、かつPMDのバージョンが5.0以降の場合に対象となり、PMDによりチェックされます。

PLSQLファイル

- 拡張子は「.sql、.trg、.prc、.fnc、.pld、.pls、.plh、.plb、.pck、.pks、.pkh、.pkb、.typ、.tyb、.tps、.tpb」を対象とします。
- PMDとの連携が有効で、かつPMDのバージョンが5.1以降の場合に対象となり、PMDによりチェックされます。

Apache Velocityファイル

- 拡張子は「.vm」を対象とします。
- PMDとの連携が有効で、かつPMDのバージョンが5.1以降の場合に対象となり、PMDによりチェックされます。

第2章 コードチェック準備作業

ここでは、Agile+ Relief JをEclipse上で利用するためのインストール手順について説明します。

2.1 Agile+ Relief Jプラグインのインストール

Agile+ Relief Jは、動作環境としてJDKおよびEclipseが必要です。あらかじめこれらの環境を準備してください。

Agile+ Relief JプラグインをEclipse環境にインストールすることにより、Agile+ Relief Jの機能が利用できるようになります。

2.1.1 環境変数JAVA_HOME を設定する

Windowsの環境変数にJAVA_HOMEを設定します。JAVA_HOMEには、JDKのインストールパスを設定します。

JDKがインストールされていない場合は、OracleのWebページからJDK5.0以降を入手し、インストールしてください。

OracleのWebページ：http://www.oracle.com/technetwork/java/index.html

＜操作＞

1. [コンピュータ]を右クリック※－[プロパティ]を選択します。
2. [システム]ウィンドウの左側の[システムの詳細設定]をクリックします。[システムのプロパティ]ダイアログが表示されます。
3. [システムのプロパティ]ダイアログで[詳細設定]タブを選択し、[環境変数]ボタンをクリックします。[環境変数]ダイアログが表示されます。
4. 環境変数 JAVA_HOME にインストールしたJDKのパスを指定します。

※Windows 8.1、およびWindows Server 2012 R2以降は、[PC]を右クリックします。

2.1.2 Eclipseを準備する

Eclipse Projectのダウンロードページからeclipseを取得し、インストールします。

Eclipse Projectのダウンロードページ：http://www.eclipse.org/downloads/

＜操作＞

1. ダウンロードしたEclipseのアーカイブを解凍します。
   例えば「c:¥」に解凍した場合、「c:¥eclipse」にeclipse.exeとEclipseの動作に必要なファイルやフォルダが展開されます。

※ Interstage Studioをご利用の場合、お手持ちのInterstage Studioをインストールしてご利用ください。インストールの詳細は、Interstage Studioのマニュアルをご参照ください。

2.1.3 Agile+ Relief Jプラグインをインストールする

Agile+ Relief JプラグインをEclipseにインストールします。

Agile+ Relief Jプラグイン

" Agile+ Reliefインストールフォルダ¥PGReliefJava¥eclipse¥plugins" フォルダに、以下のAgile+ Relief Jプラグインが格納されています。

```
com.fujitsu.jp.fst.pgrelief.java.eclipse.doc.detail_x.x.x.jar
com.fujitsu.jp.fst.pgrelief.java.eclipse.doc.guide_x.x.x.jar
com.fujitsu.jp.fst.pgrelief.java.eclipse_x.x.x.jar
com.fujitsu.jp.fst.pgrelief.java.eclipse.kiyacker_x.x.x.jar
```

※x.x.xはプラグインの版数です。

＜操作＞

1. Agile+ Relief Jインストール先のpluginsフォルダを、Eclipseインストールフォルダ内のpluginsフォルダ、またはdropinsフォルダにコピーします。

   例）

   Eclipse展開先が"C:¥eclipse"、Agile+ Relief Jインストール先が"C:¥Program Files¥AgilePlus¥PGReliefJava"の場合

```
copy "C:¥Program Files¥AgilePlus¥PGReliefJava¥eclipse¥plugins" "C:¥eclipse¥plugins"
```

2. eclipse.exeを起動します。

```
eclipse.exe
```

Eclipseメニューバーに[Agile+ Relief J]が表示されている場合は、プラグインのインストールは終了です。

参考

Eclipseメニューバーに[Agile+ Relief J]が表示されない場合は、Eclipseのパースペクティブのカスタマイズを確認してください。

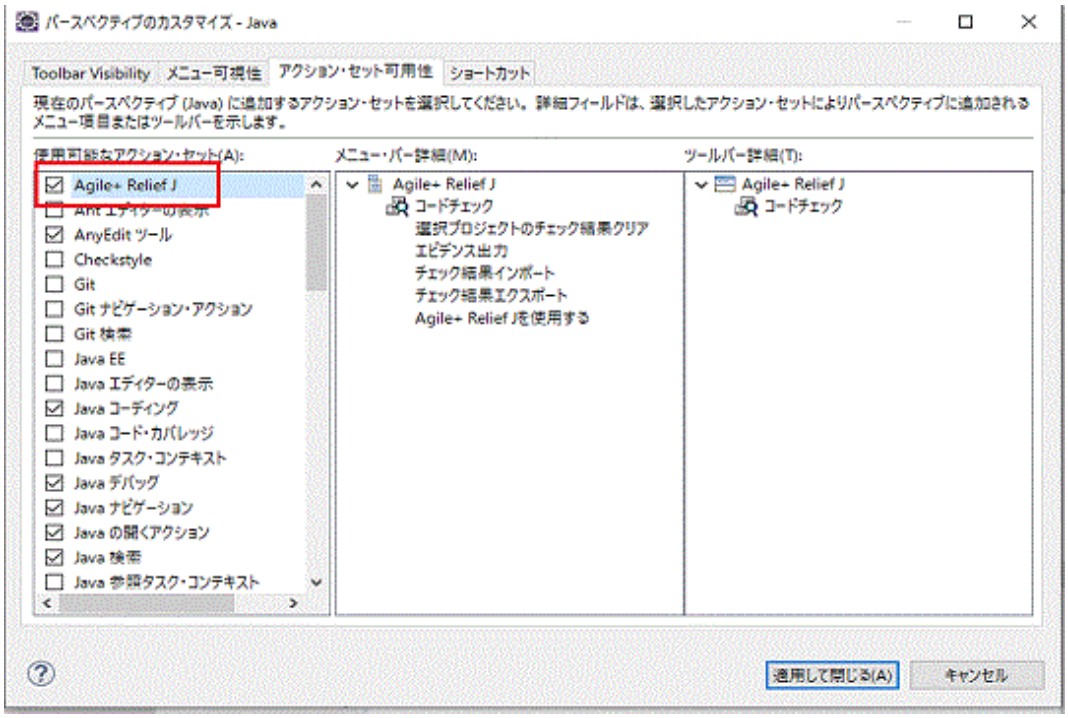
Eclipseの[パースペクティブのカスタマイズ]画面で、[Agile+ Relief J]のアクション・セットが有効になっているかを確認し、無効な場合には選択して有効にします。

＜操作＞

1. Eclipseメニューバーから[ウィンドウ(W)]＞[パースペクティブのカスタマイズ(Z)]を選択します。

   [パースペクティブのカスタマイズ]ウィンドウが表示されます。

2. [コマンド]－[使用可能なアクション・セット]の一覧で、[Agile+ Relief J]が選択されているか確認します。

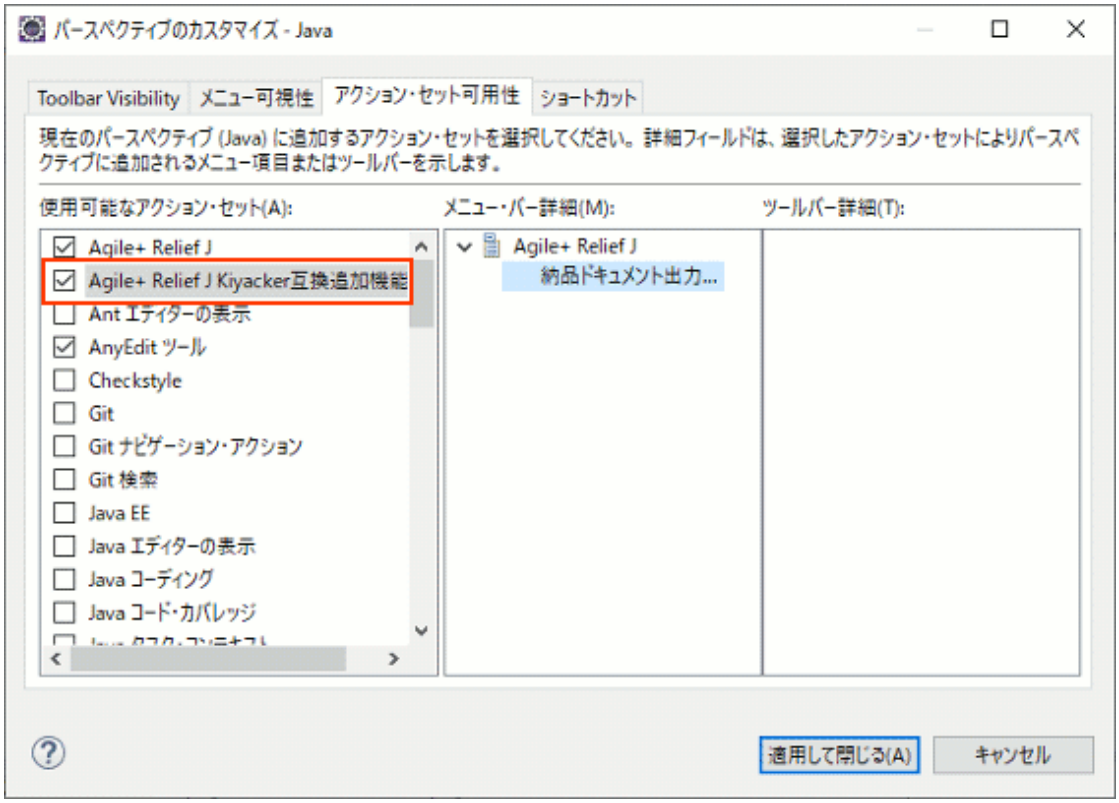


3. [Agile+ Relief J]が選択されていない場合には、選択して[OK]ボタンをクリックします。

   メインウィンドウのメニューバーに[Agile+ Relief J(J)]が表示されます。

同様に、[Agile+ Relief J Kiyacker互換追加機能]のアクション・セットについても、有効に設定します。

1. [コマンド]ー[使用可能なアクション・セット]の一覧で、[Agile+ Relief J Kiyacker互換追加機能]が選択されているか確認します。



2. [Agile+ Relief J Kiyacker互換追加機能]が選択されていない場合には、選択して[OK]ボタンをクリックします。

   [Agile+ Relief J(J)]メニューバーに[納品ドキュメント出力(O)]が表示されます。

2.1.4 Eclipseでビューの表示を設定する

Agile+ Relief Jプラグインで利用するビューを表示するには、以下の操作を行ってください。

＜操作＞

1. [ウィンドウ(W)]メニュー＞[ビューの表示(V)]＞[その他(O)]を選択します。

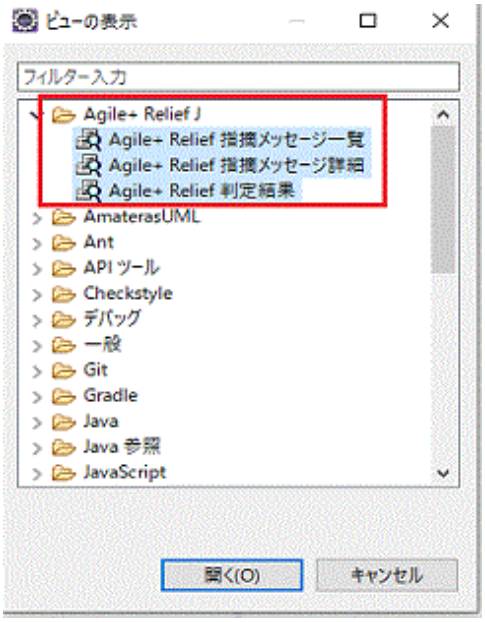
   [ビューの表示]画面が表示されます。

2. [Agile+ Relief J]の以下の項目から、表示するビューを選択して[OK]ボタンをクリックします。

   ー [Agile+ Relief 指摘メッセージ一覧]

   ー [Agile+ Relief 指摘メッセージ詳細]

ー [Agile+ Relief 判定結果]



2.2 コードチェックの利用環境を設定する

Eclipse上で、Agile+ Relief Jを利用するための環境設定を行います。

環境設定では、以下の項目を設定します。

- 検査規約定義ファイルの設定
- コードチェック結果出力先フォルダの設定
- ビルド時の設定
- コードチェック時の最大ヒープサイズの設定

2.2.1 Agile+ Relief Jを利用できる状態にする

Agile+ Relief Jの環境設定を行うには、事前にAgile+ Relief Jを利用できる状態にする必要があります。

＜操作＞

1. Eclipseメニューバーから[Agile+ Relief J(J)]＞[Agile+ Relief Jを使用する(L)]を選択します。

Eclipseメニューバーの[Agile+ Relief J(J)]＞[Agile+ Relief Jを使用する(L)]が選択されている場合には、Agile+ Relief Jが利用できる状態です。

2.2.2 検査規約定義ファイルを設定する

コードチェックで利用するプロジェクト開発規約となる検査規約定義ファイルを指定します（検査規約定義ファイルの拡張子は「pgrj」です）。

ここで設定された内容は、現在開かれているワークスペース上のすべてのプロジェクトのデフォルトになります。

プロジェクト単位で固有の設定を行う場合は、「2.2.6 プロジェクト固有設定」を参照してください。

＜操作＞

1. [ウィンドウ(W)]メニュー＞[設定(P)]を選択します。

   [設定]画面が表示されます。

2. ツリーから[Agile+ Relief J]を選択します。

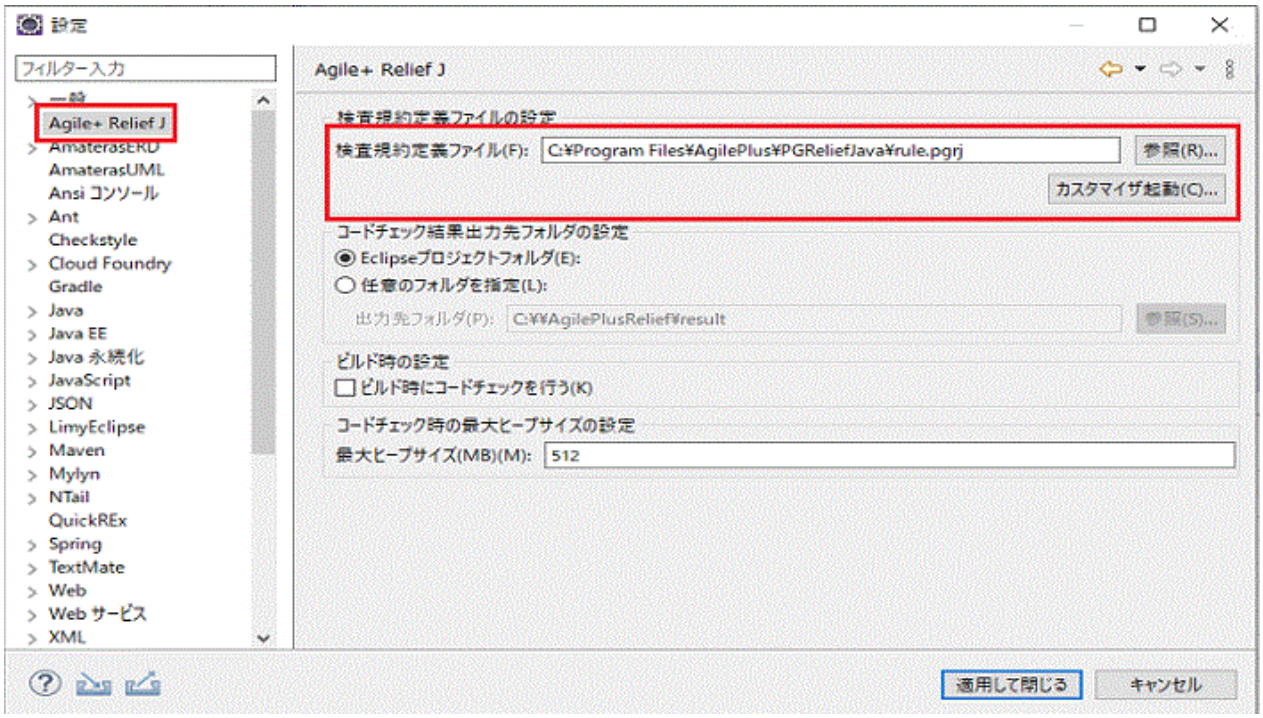
   Agile+ Relief Jの設定画面が表示されます。

3. [検査規約定義ファイルの設定]で検査規約定義ファイルを以下のいずれかの方法で指定します。

   ー [参照(R)]ボタンを押し、ファイル選択ダイアログを利用してファイルを選択

   ー 入力欄にフルパスで直接入力

   例：検査規約定義ファイルが"C:¥Program Files¥AgilePlus¥PGReliefJava¥rule.pgrj"の場合



参照

[カスタマイザ起動(C)]ボタンを押すと、カスタマイザが起動され、設定されている検査規約定義ファイルを編集することができます。

検査規約定義ファイルの編集の詳細については、「Agile+ Relief Jカスタマイザ操作手引書」を参照してください。

2.2.3 コードチェック結果出力先フォルダを設定する

コードチェックのチェック結果の出力先フォルダを指定します。出力先としては、Eclipseプロジェクトフォルダと任意のフォルダのいずれかを指定することができます。各出力先への出力形式は以下のようになります。

- Eclipseプロジェクトフォルダを指定した場合

  チェック対象のプロジェクトのフォルダ配下に出力されます。

- 任意のフォルダを指定した場合

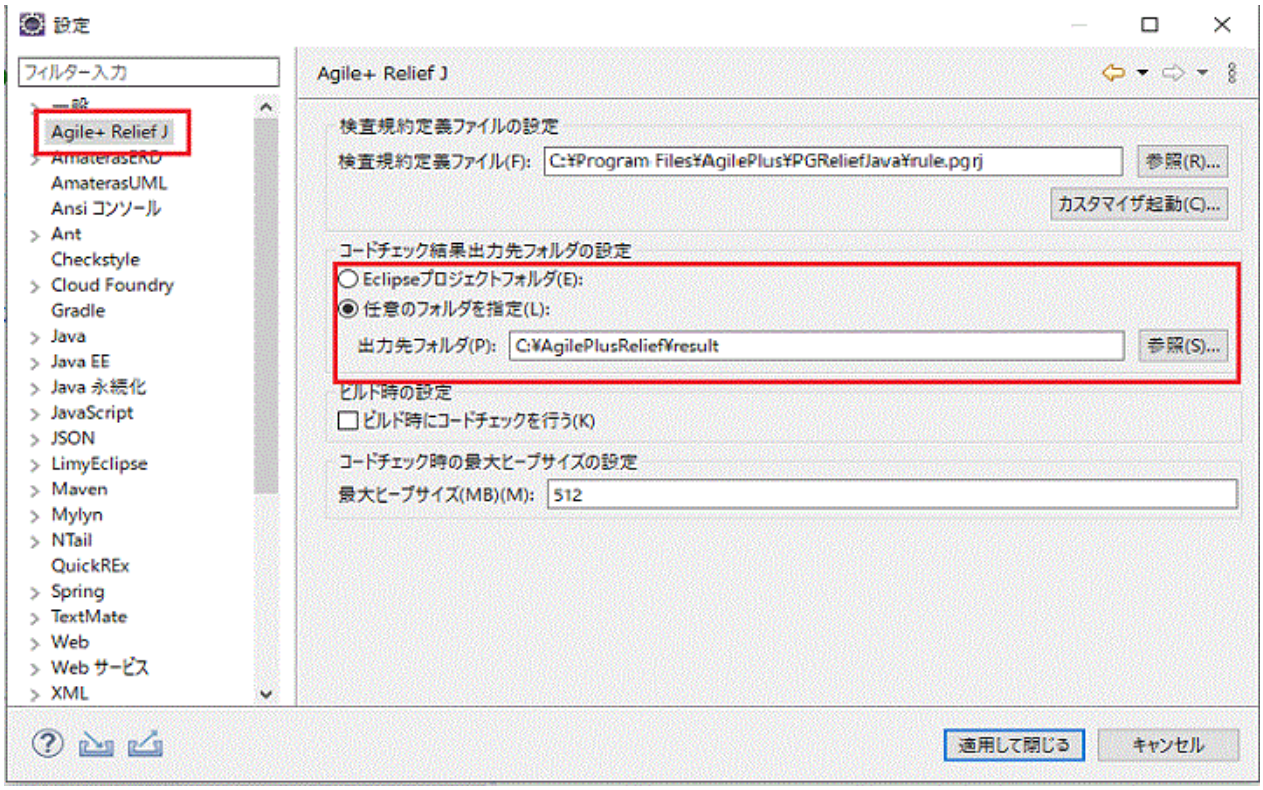
  指定されたフォルダ配下にチェック対象のプロジェクトと同名のフォルダが作成され、そのフォルダ配下に出力されます。

ここで設定された内容は、現在開かれているワークスペース上のすべてのプロジェクトのデフォルトになります。
プロジェクト単位で固有の設定を行う場合は、「2.2.6 プロジェクト固有設定」を参照してください。

＜操作＞

1. [ウィンドウ(W)]メニュー＞[設定(P)]を選択します。
   設定画面が表示されます。
2. ツリーから[Agile+ Relief J]を選択します。
   Agile+ Relief Jの設定画面が表示されます。
3. [コードチェック結果出力先フォルダの設定]で出力先フォルダを以下のいずれかの方法で指定します。
   - Eclipseプロジェクトフォルダ配下に出力する場合
     [Eclipseプロジェクトフォルダ(E)]を選択します。
   - Eclipseプロジェクトフォルダ以外の任意のフォルダに出力する場合
     1. [任意のフォルダを指定(L)]を選択します。
     2. 出力先フォルダを以下のいずれかの方法で指定します。
        - [参照(S)]ボタンを押し、フォルダ選択ダイアログを利用してフォルダを選択
        - 入力欄にフルパスで直接入力

   例:コードチェック結果の出力先フォルダが任意のフォルダ"C:¥AgilePlusRelief¥result"の場合



注意

コードチェック結果出力先フォルダの設定を変更すると、変更前や変更後のフォルダにコードチェック結果が存在する場合、そのチェック結果を削除します。そのため、再度コードチェックの実行が必要になるので注意してください。また、閉じられた状態にあるプロジェクトについては、コードチェック結果が存在しても削除されないため、プロジェクトを開いたときにコードチェック結果の表示が正常に行われない場合があります。このような場合も、再度コードチェックを実行してください。

2.2.4 ビルド時のオプションを設定する

ビルド時のコードチェックの動作を設定します。

[ビルド時にコードチェックを行う]を選択すると、ビルド完了後に自動的にコードチェックが実行されます。

ここで設定された内容は、現在開かれているワークスペース上のすべてのプロジェクトのデフォルトになります。

プロジェクト単位で固有の設定を行う場合は、「2.2.6 プロジェクト固有設定」を参照してください。

＜操作＞

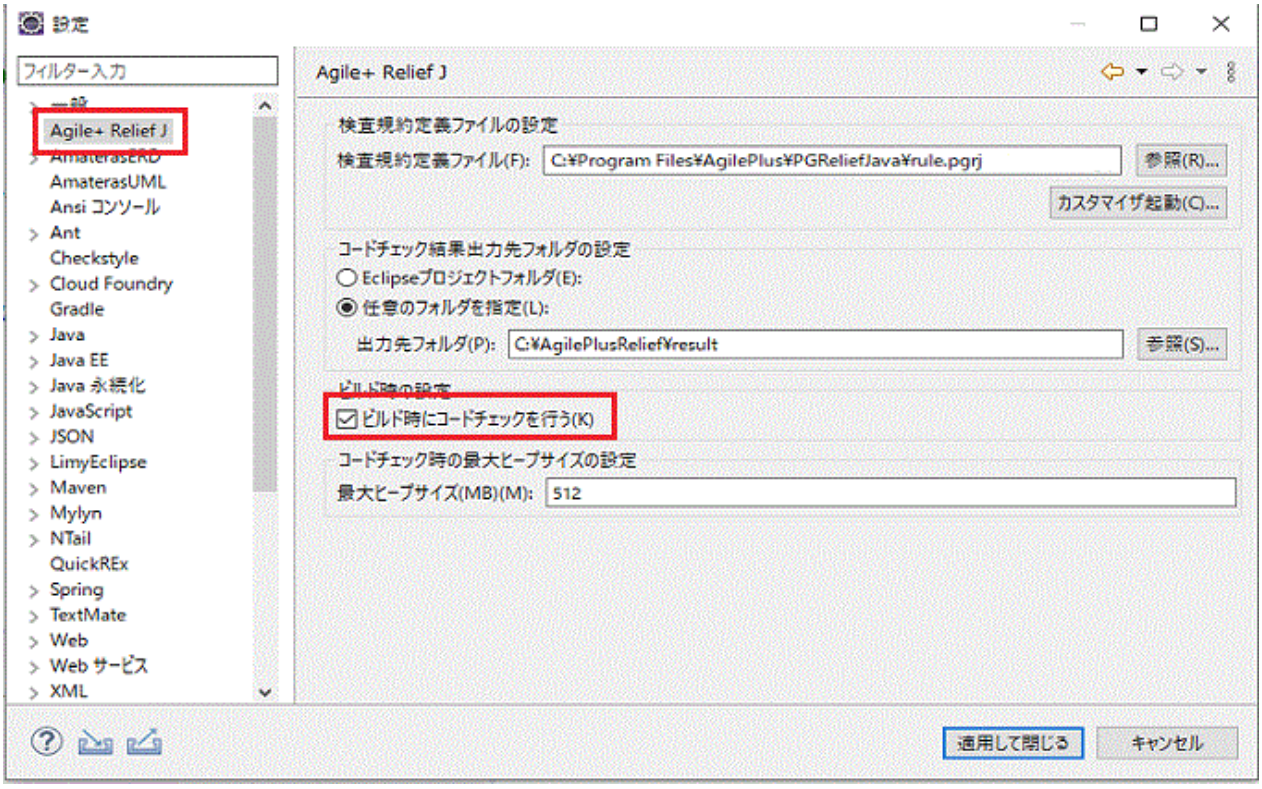
1. [ウィンドウ(W)]メニュー＞[設定(P)]を選択します。

   設定画面が表示されます。

2. ツリーから[Agile+ Relief J]を選択します。

   Agile+ Relief Jの設定画面が表示されます。

3. ビルド時にコードチェックを行う場合は、[ビルド時にコードチェックを行う(K)]チェックボックスをチェックします。



2.2.5 コードチェック時の最大ヒープサイズを設定する

コードチェック時のJavaの最大ヒープサイズを設定します。大規模資産に対するコードチェックなどでメモリ不足が発生する場合は、最大ヒープサイズとして大きな値を設定してください。

ここで設定された内容は、現在開かれているワークスペース上のすべてのプロジェクトのデフォルトになります。

プロジェクト単位で固有の設定を行う場合は、「2.2.6 プロジェクト固有設定」を参照してください。

＜操作＞

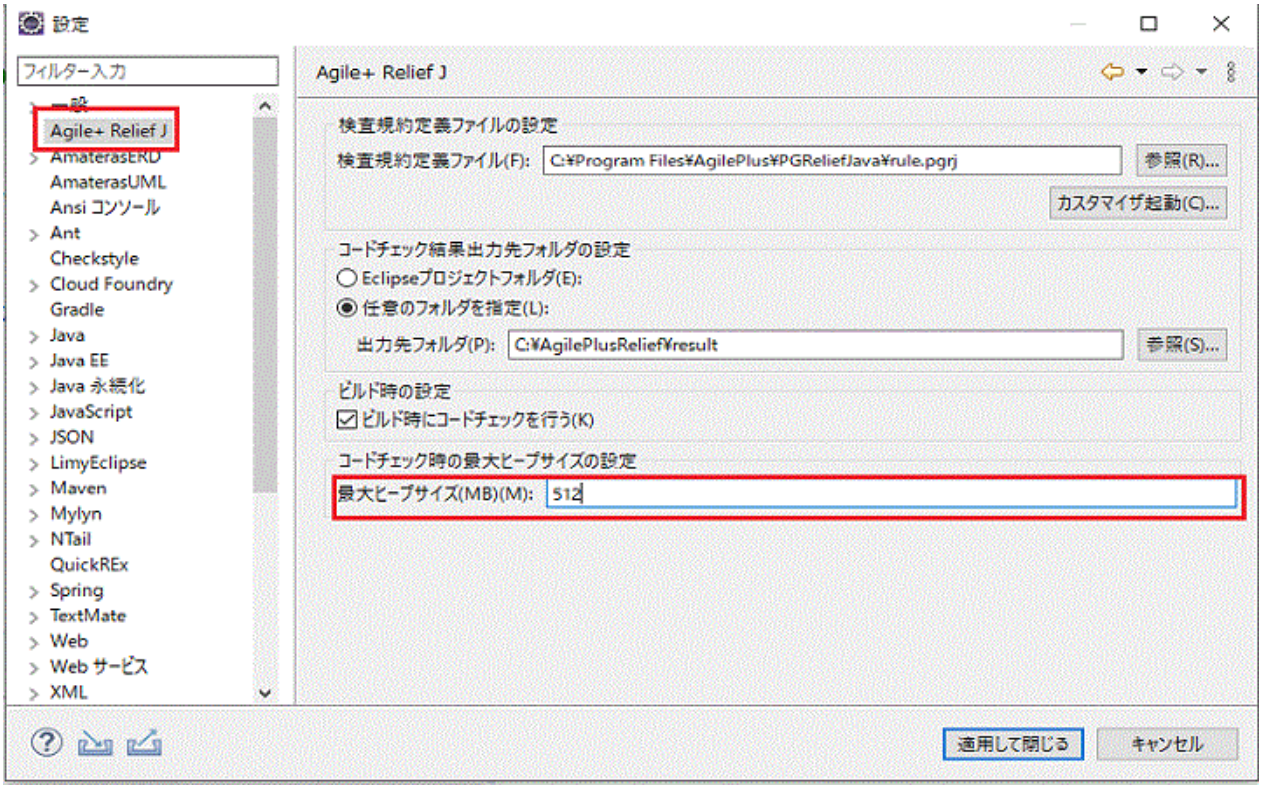
1. [ウィンドウ(W)]メニュー＞[設定(P)]を選択します。

   設定画面が表示されます。

2. ツリーから[Agile+ Relief J]を選択します。

   Agile+ Relief Jの設定画面が表示されます。

3. [最大ヒープサイズ(MB)(M)]入力欄に数値を設定します。



参照

コードチェック結果表示などEclipseに関するメモリ不足の問題については、「7.1 コードチェック」の「メモリ不足について」を参照してください。

2.2.6 プロジェクト固有設定

検査規約定義ファイル、コードチェック結果出力先フォルダ、ビルド時のオプション、およびコードチェック時の最大ヒープサイズは、プロジェクト固有に設定することができます。

プロジェクト固有設定をするには、以下の操作を行ってください。

＜操作＞

1. [プロジェクト(P)]メニュー＞[プロパティー(P)]を選択します。

   対象プロジェクトの[プロパティー]画面が表示されます。

2. ツリーから[Agile+ Relief J]を選択します。

   Agile+ Relief Jの設定画面が表示されます。

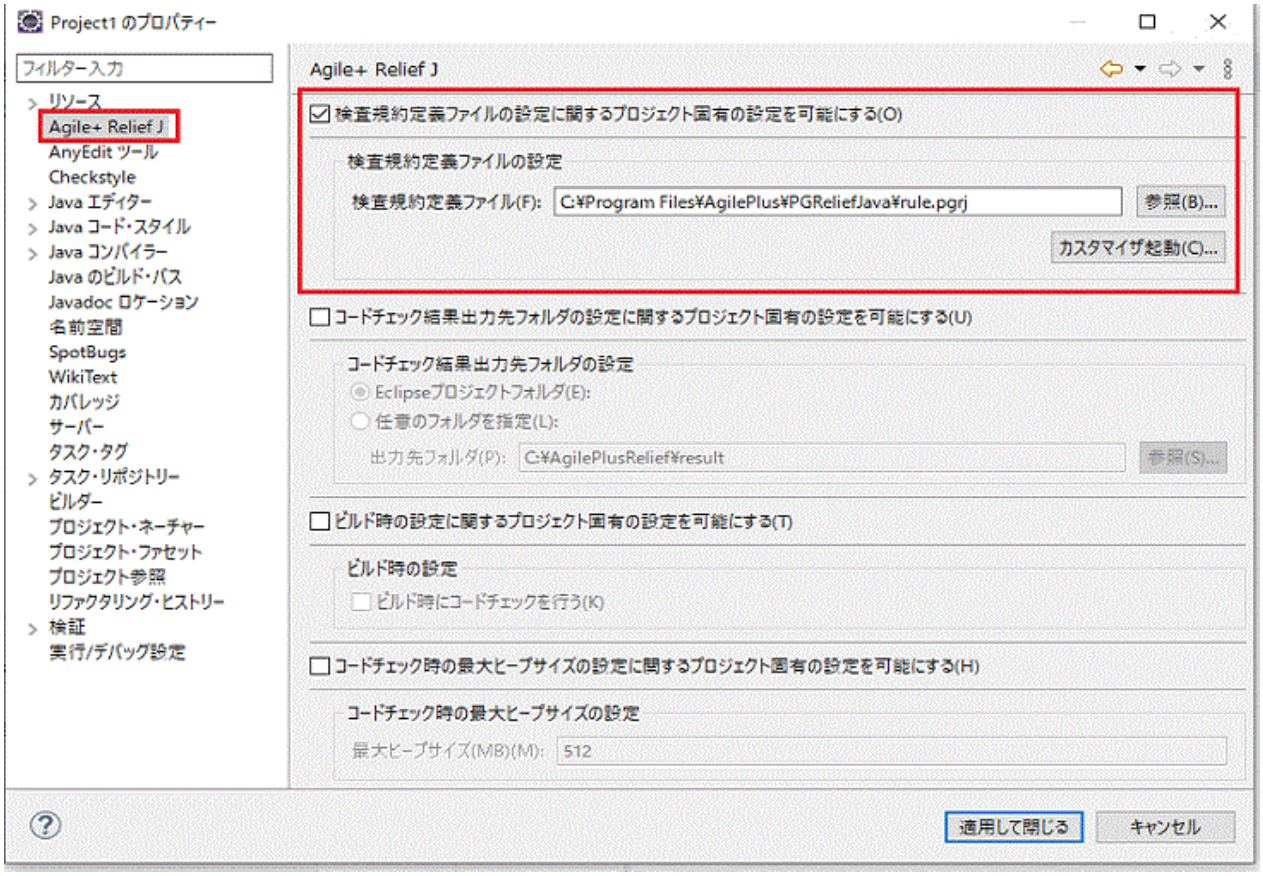
3. 検査規約定義ファイルを指定する場合、[検査規約定義ファイルの設定に関するプロジェクト固有の設定を可能にする(O)]チェックボックスをチェックします。

   [検査規約定義ファイルの設定]が有効になります。

4. 検査規約定義ファイルを以下いずれかの方法で指定します。

   ー [参照(B)]ボタンを押し、ファイル選択ダイアログを利用してファイルを選択

   ー 入力欄にフルパスで直接入力



5. コードチェック結果出力先フォルダを指定する場合、[コードチェック出力先フォルダの設定に関するプロジェクト固有の設定を可能にする(U)]チェックボックスをチェックします。

   [コードチェック結果出力先フォルダの設定]が有効になります。

6. [コードチェック結果出力先フォルダの設定]で出力先フォルダを以下のいずれかの方法で指定します。

   ー Eclipseプロジェクトフォルダ配下に出力する場合

   [Eclipseプロジェクトフォルダ(E)]を選択します。

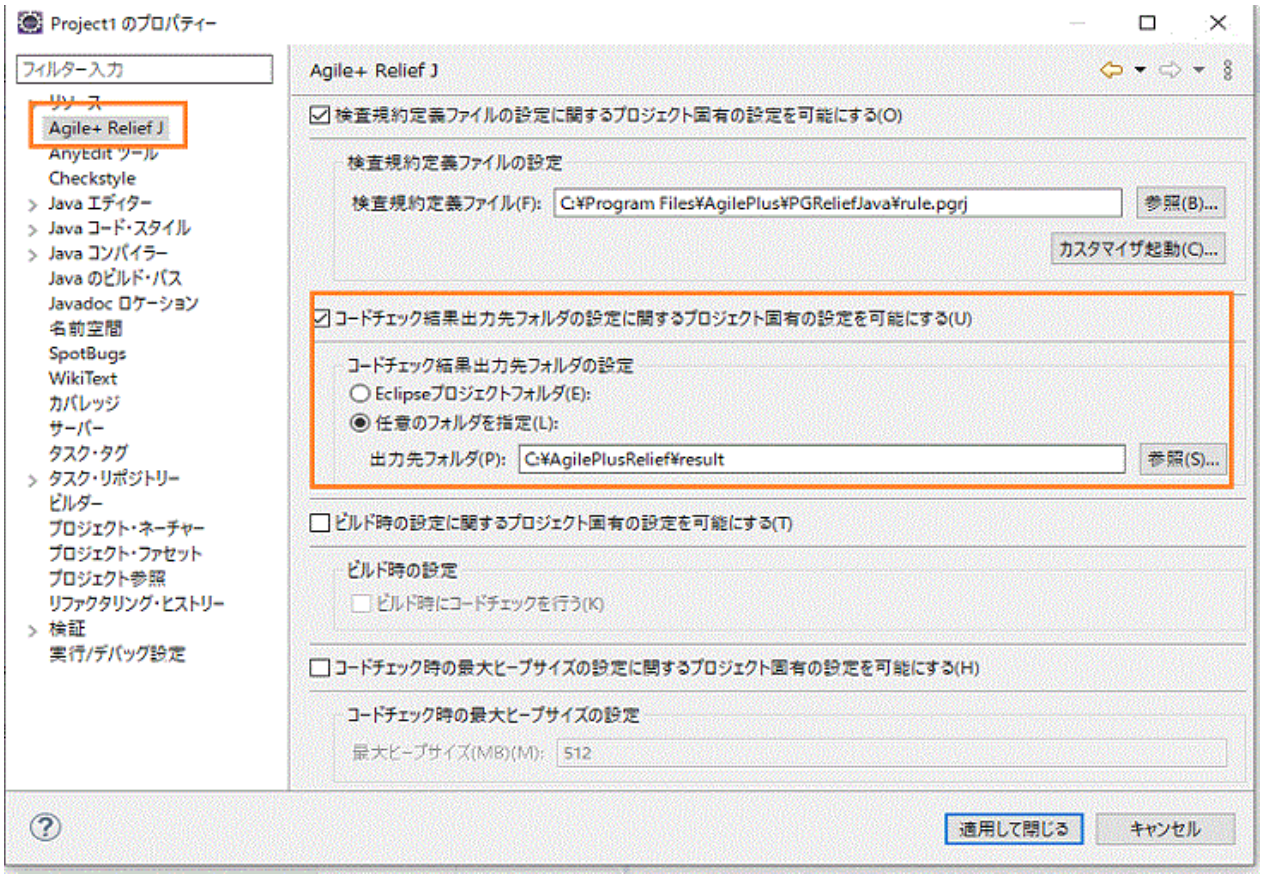
   ー Eclipseプロジェクトフォルダ以外の任意のフォルダに出力する場合

      1. [任意のフォルダを指定(L)]を選択します。

2. 出力先フォルダを以下のいずれかの方法で指定します。

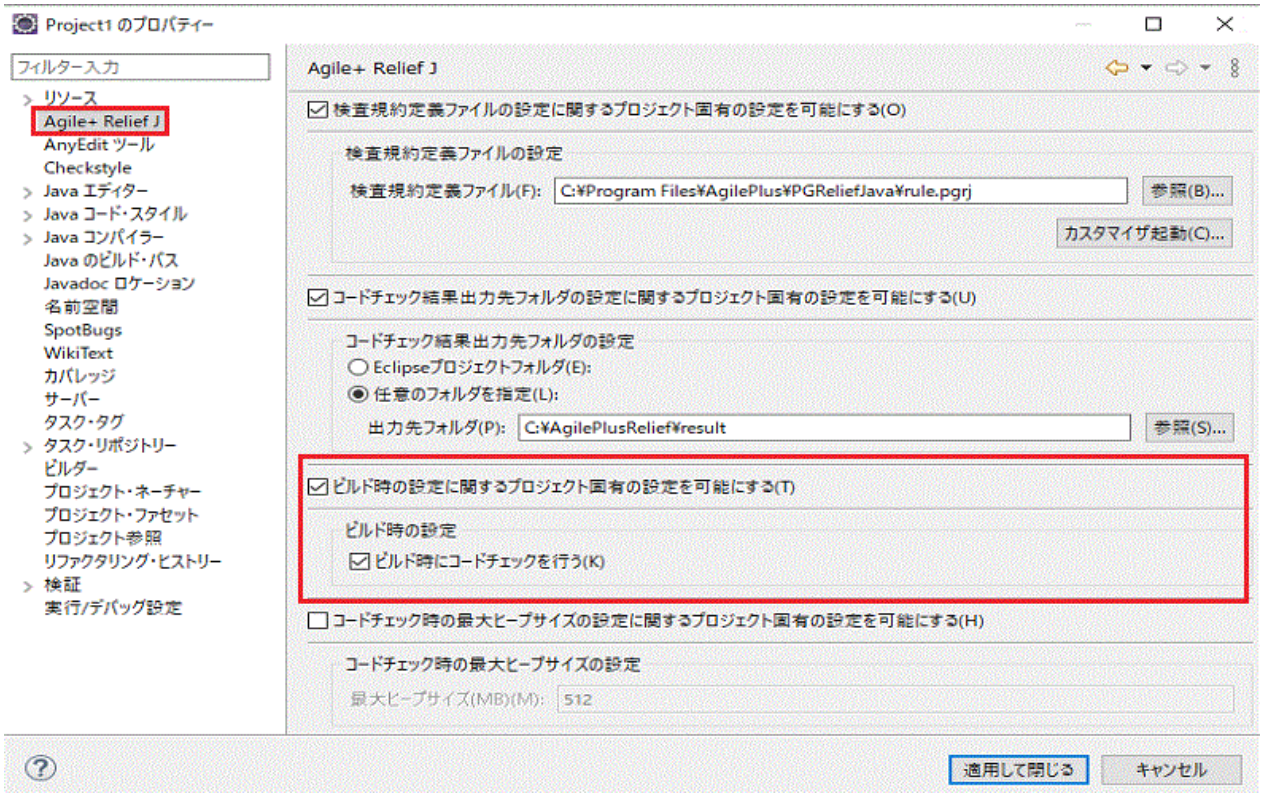
   ー [参照(S)]ボタンを押し、フォルダ選択ダイアログを利用してフォルダを選択

   ー 入力欄にフルパスで直接入力



7. ビルド時設定をする場合、[ビルド時の設定に関するプロジェクト固有の設定を可能にする(T)]チェックボックスをチェックします。
   [ビルド時の設定]が有効になります。

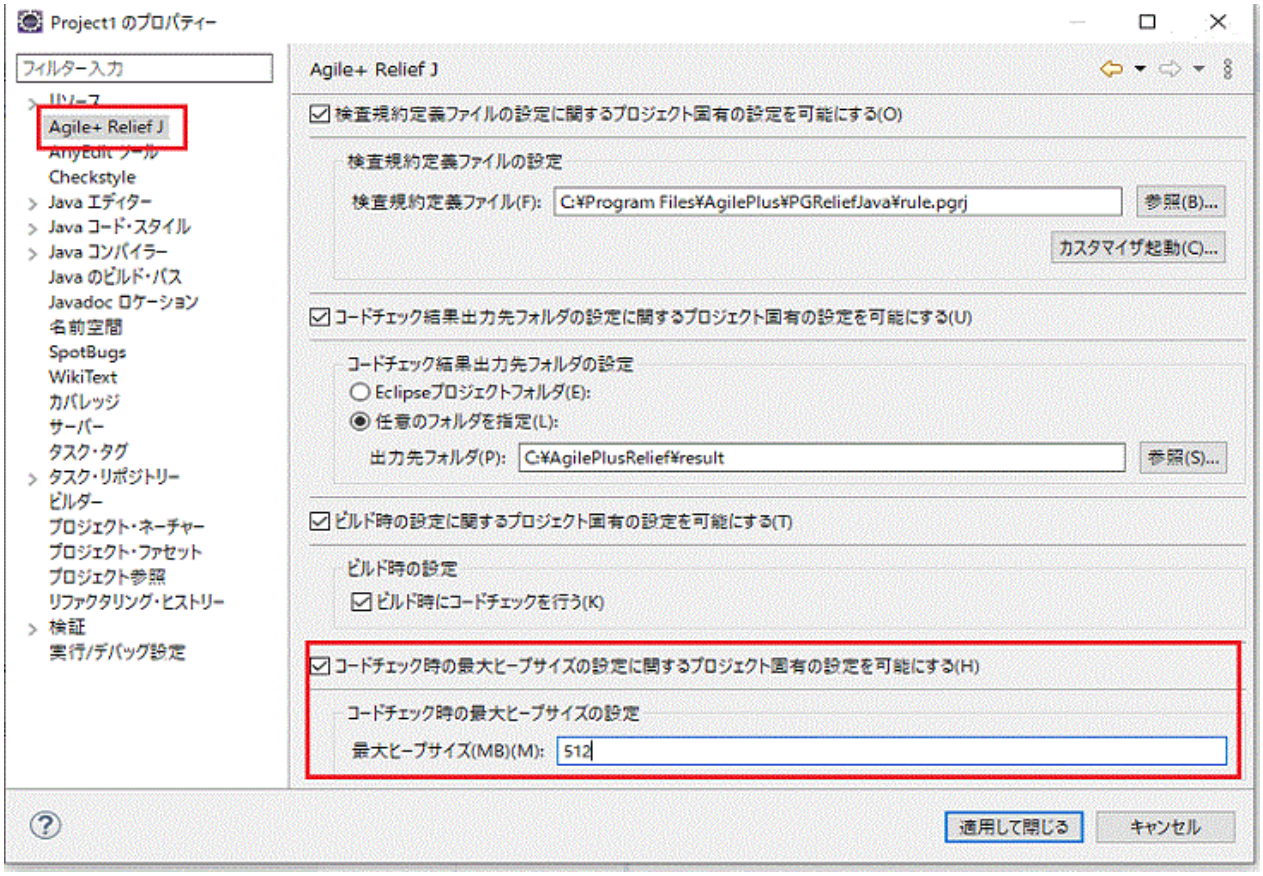
8. [ビルド時にコードチェックを行う(K)]チェックボックスをチェックします。



9. コードチェック時の最大ヒープサイズの設定をする場合、[コードチェック時の最大ヒープサイズの設定に関するプロジェクト固有の設定を可能にする(H)]チェックボックスをチェックします。

   [コードチェック時の最大ヒープサイズの設定]が有効になります。

10. [最大ヒープサイズ(MB)(M)]入力欄に数値を設定します。



2.3 Agile+ Relief Jプラグインをアンインストールする

Agile+ Relief Jプラグインをアンインストールするには、以下の操作を行ってください。

Agile+ Relief Jプラグイン

" Agile+ Reliefインストールフォルダ¥PGReliefJava¥eclipse¥plugins" フォルダに、以下のAgile+ Relief Jプラグインが格納されています。

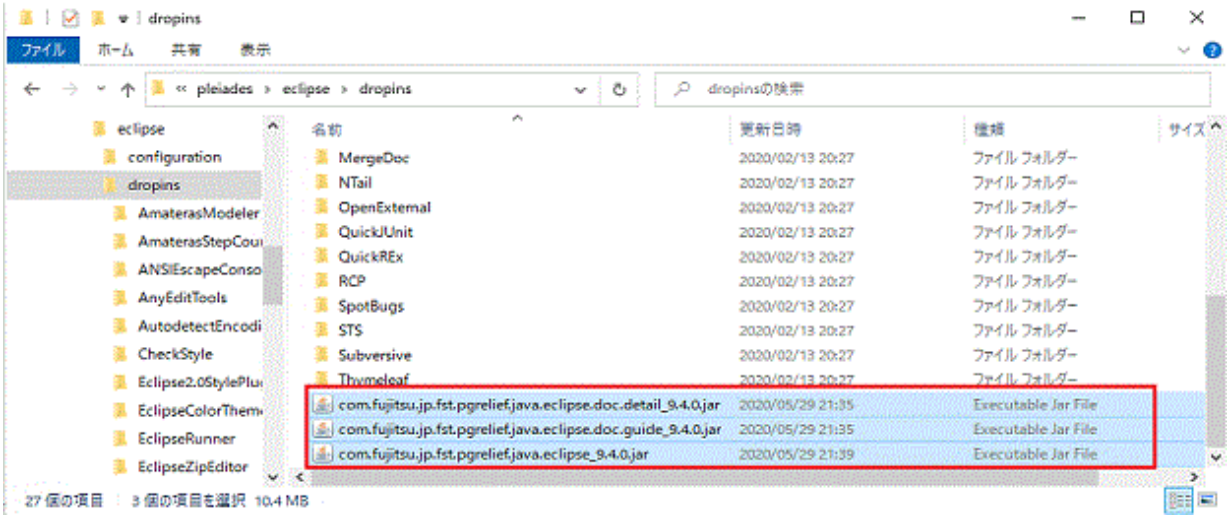
```
com.fujitsu.jp.fst.pgrelief.java.eclipse.doc.detail_x.x.x.jar
com.fujitsu.jp.fst.pgrelief.java.eclipse.doc.guide_x.x.x.jar
com.fujitsu.jp.fst.pgrelief.java.eclipse_x.x.x.jar
com.fujitsu.jp.fst.pgrelief.java.eclipse.kiyacker_x.x.x.jar
```

※x.x.xはプラグインの版数です。

＜操作＞

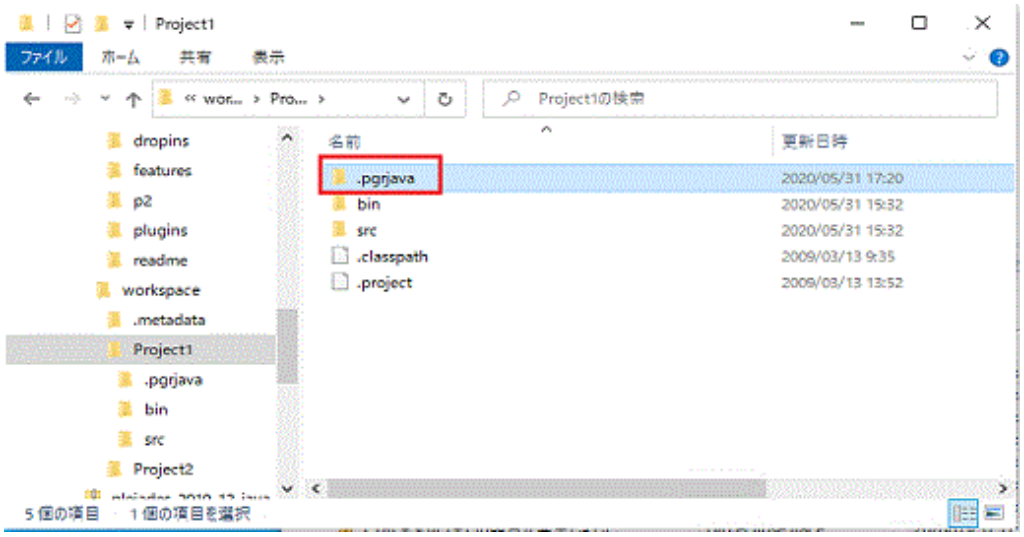
1. Eclipseを起動している場合は終了します。

2. Eclipseインストールフォルダ内のpluginsフォルダ、またはdropinsフォルダにコピーしたAgile+ Relief Jプラグインを削除します。



※Kiyacker互換ではAgile+ Relief Jプラグインの数が異なります。前述のAgile+ Relief Jプラグインを確認してください。

3. Eclipseワークスペースのプロジェクトフォルダ配下にある".pgrjava"フォルダを削除します。



第3章 コードチェック利用手順

ここでは、Agile+ Relief Jを利用したコードチェック手順を説明します。

3.1 コードチェックを実行する

コードチェックの実行は、プロジェクトを開いた状態で行います。必ず、コンパイルエラーがない状態で行ってください。

オートビルドを設定していない場合、ビルドを実行してからコードチェックを実施してください。

コードチェックの実行には以下の3つがあります。

- **プロジェクトを選択してコードチェック**

  ワークスペース内のチェック対象プロジェクトを複数選択できます。

  チェック対象プロジェクトは、パッケージ・エクスプローラ、プロジェクト・エクスプローラ、またはナビゲータのビューから選択します。

  選択したプロジェクト内にチェック対象プロジェクト以外のプロジェクトが含まれる場合は、[コードチェック]コマンドが無効になります。

- **ファイルを選択してコードチェック**

  ワークスペース内のチェック対象ファイルを複数選択できます。

  チェック対象ファイルは、パッケージ・エクスプローラ、プロジェクト・エクスプローラ、ナビゲータ、または判定結果のビューから選択します。

  選択したファイル内にチェック対象ファイル以外が含まれる場合、または、プロジェクトが異なるファイルが選択されている場合は、[コードチェック]コマンドが無効になります。

  また、チェック対象ファイルを開いたエディタがアクティブであれば、コードチェックの実行が可能です。

- **ビルド時に自動でコードチェック**

  Eclipseのビルドと連動して、ビルド完了後に対象となったチェック対象ファイルのコードチェックを自動的に実施します。

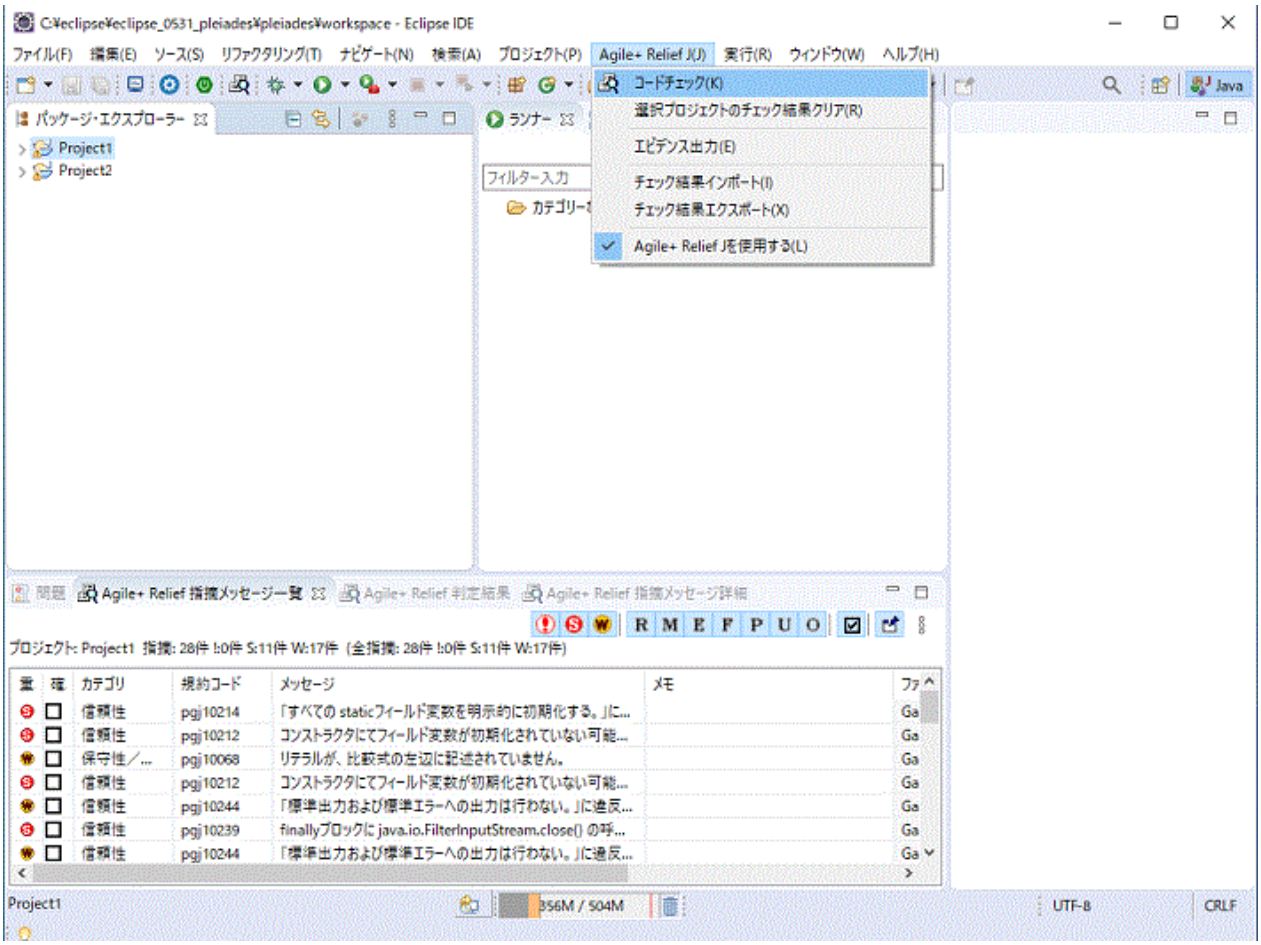
  ビルド時のオプション設定で、[ビルド時にコードチェックを行う]を選択することにより、Javaビルドのタイミングでコードチェックが行われます。オプションの設定方法は、「ビルド時の設定」を参照してください。

コードチェックが完了すると、[Agile+ Relief 指摘メッセージ一覧]ビュー、および[Agile+ Relief 判定結果]ビューの表示内容が自動的に切り替わります。

なお、コードチェックを実行すると、プロジェクトフォルダ配下に「.pgrjava」フォルダが生成されます。このフォルダはAgile+ Relief Jが利用するため、フォルダを変更したりしないでください。

以降で、プロジェクトやファイル選択でのコードチェックの実行方法を説明します。

3.1.1 メニューバーからのコードチェック



選択されたチェック対象プロジェクトまたはチェック対象ファイルに対してコードチェックを実行します。複数のチェック対象プロジェクトまたはチェック対象ファイル選択が可能です。

プロジェクトが選択された場合は、プロジェクト内のすべてのチェック対象ファイルに対してコードチェックを実行します。

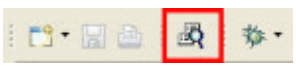
※チェック対象プロジェクトまたはチェック対象ファイルを選択していない場合は、機能を利用できません。

※チェック対象プロジェクトとチェック対象ファイルを同時に選択した場合は、機能を利用できません。

＜操作＞

1. コードチェックの対象資産を、以下のいずれかの方法で選択します。
   - パッケージ・エクスプローラ、プロジェクト・エクスプローラ、またはナビゲータのビューでチェック対象プロジェクトまたはチェック対象ファイルを選択します。
   - エディタでチェック対象プロジェクト配下のチェック対象ファイルを表示します。
   - 判定結果ビューでファイルを選択します。
2. [Agile+ Relief J(J)]メニュー＞[コードチェック(K)]を選択します。

3.1.2 ツールバーからのコードチェック



選択されたチェック対象プロジェクトまたはチェック対象ファイルに対してコードチェックを実行します。複数のチェック対象プロジェクトまたはチェック対象ファイル選択が可能です。

プロジェクトが選択された場合は、プロジェクト内のすべてのチェック対象ファイルに対してコードチェックを実行します。

※チェック対象プロジェクトまたはチェック対象ファイルを選択していない場合は、機能を利用できません。

※チェック対象プロジェクトとチェック対象ファイルを同時に選択した場合は、機能を利用できません。

＜操作＞

1. コードチェックの対象資産を、以下のいずれかの方法で選択します。
   - パッケージ・エクスプローラ、プロジェクト・エクスプローラ、またはナビゲータのビューでチェック対象プロジェクトまたはチェック対象ファイルを選択します。
   - エディタでチェック対象プロジェクト配下のチェック対象ファイルを表示します。
   - 判定結果ビューでファイルを選択します。
2. ツールバーの[コードチェック]をクリックします。

3.1.3 ポップアップメニューからのコードチェック

選択されたチェック対象プロジェクトまたはチェック対象ファイルに対してコードチェックを実行します。複数のチェック対象プロジェクトまたはチェック対象ファイル選択が可能です。

プロジェクトが選択された場合は、プロジェクト内のすべてのチェック対象ファイルに対してコードチェックを実行します。

※チェック対象プロジェクトまたはチェック対象ファイルを選択していない場合は、機能を利用できません。

※チェック対象プロジェクトとチェック対象ファイルを同時に選択した場合は、機能を利用できません。

＜操作＞

1. パッケージ・エクスプローラ、プロジェクト・エクスプローラ、またはナビゲータのビューでチェック対象プロジェクトまたはチェック対象ファイルを選択し、ポップアップメニューを表示します。
2. [Agile+ Relief J(J)]＞[コードチェック(K)]を選択します。

3.1.4 判定結果からのコードチェック

判定結果内の選択した行に対応するファイルを対象としてコードチェックを実行します。

行は複数選択できます。

＜操作＞

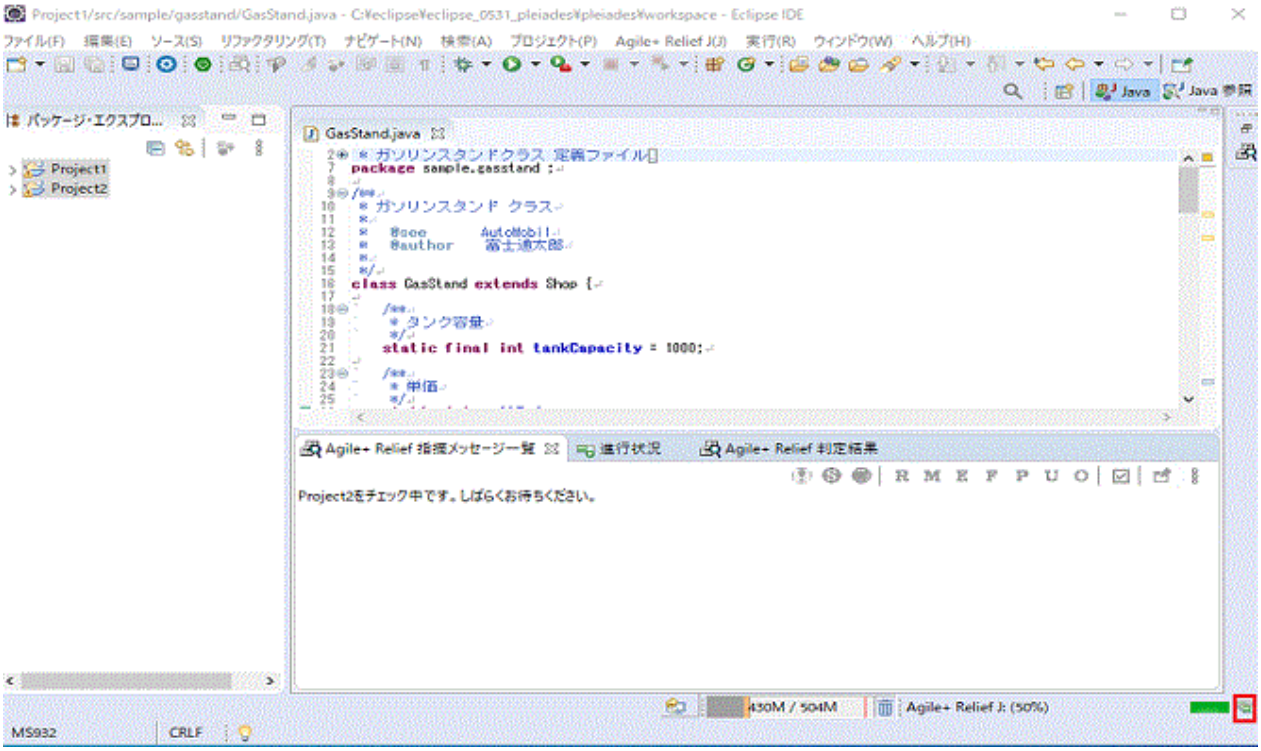
1. 判定結果でファイルを選択します。
2. 右クリックによりポップアップメニューを表示して、[コードチェック(K)]を選択します。

判定結果からのコードチェックは、メニューバーやツールバーを利用した操作もできます。

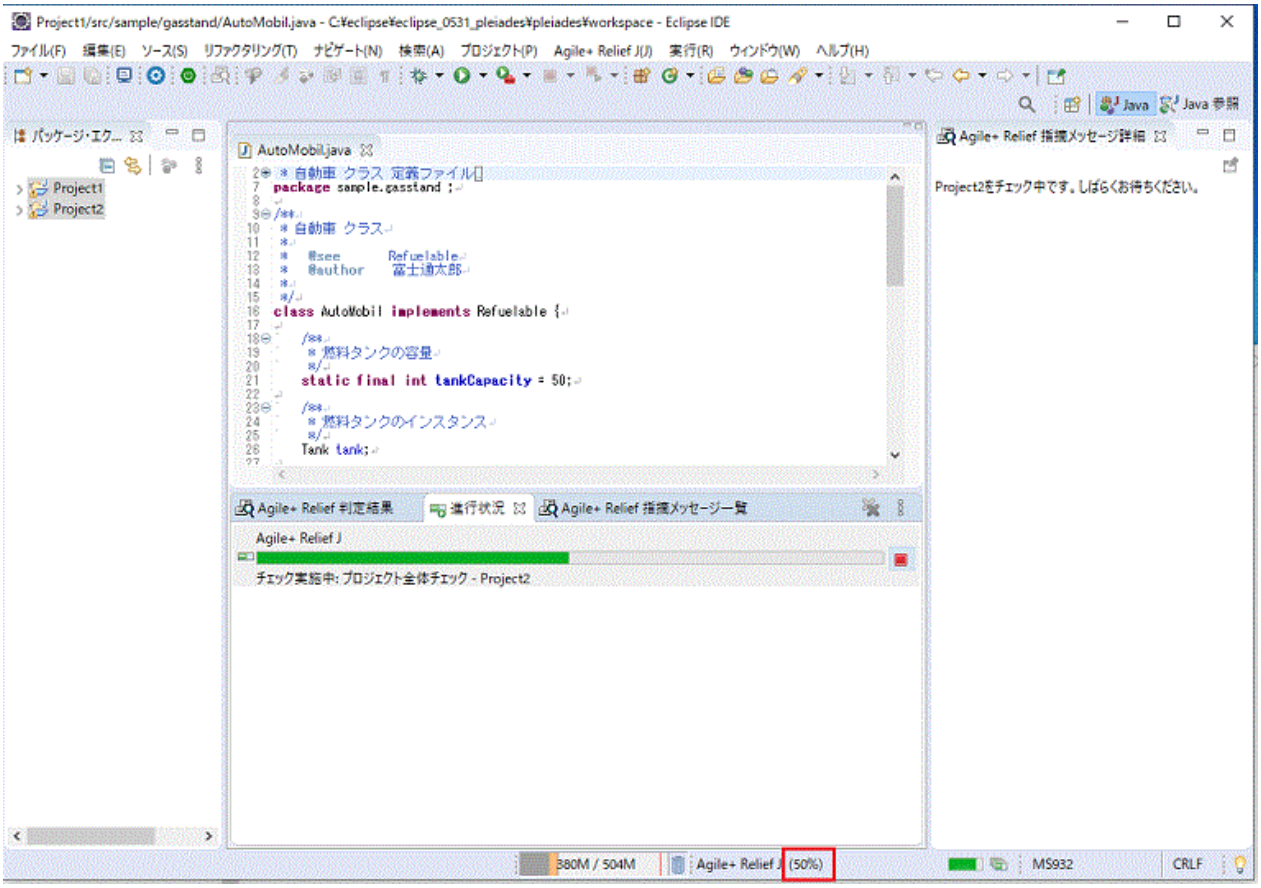
3.1.5 コードチェックの経過表示

コードチェックの実行中、進行状況ビューでコードチェックの経過を確認できます。

Eclipseのウィンドウ右下にある  をクリックすると、進行状況ビューが表示されます。



プロジェクトを複数選択した場合は、ステータスバーにプロジェクト単位の進行状況がパーセント表示されます。なお、プロジェクト単位にコードチェック処理を中断することができます。



3.2 コードチェック結果を確認する

コードチェックの結果は、以下の表示ビューを利用して確認します。

あらかじめ対象となるチェック対象プロジェクトを開き、かつ[Agile+ Relief J]メニューの[Agile+ Relief Jを使用する]を有効にしてください。

- 指摘メッセージ一覧

  コードチェック結果として、指摘メッセージの内容(重要度、確認済、カテゴリ、規約コード、メッセージ、メモ、ファイル名、パス名、および行番号)を一覧で確認することができます。

  ー 確認済、メモは、指摘メッセージの確認記録に利用します。

  ー ファイル名、パス名、行番号はエディタ連携に利用します。

- 指摘メッセージ詳細

  指摘メッセージの詳細(規約、カテゴリ、チェック内容、意味、対処方法)を確認することができます。

- 判定結果

  コードチェック結果として、判定結果の内容(パス名、ファイル名、判定、チェック、指摘件数)を一覧で確認することができます。

  指摘件数は、ファイル単位に集計したメトリクスにより、重要度、カテゴリ、未確認などの項目に分けられて表示されます。

  ー ファイル名、パス名はエディタ連携に利用します。

3.2.1 指摘メッセージを確認する

指摘メッセージは、コードチェックで規約違反として指摘されます。指摘メッセージの確認には、[Agile+ Relief 指摘メッセージ一覧]ビューを利用します。

以下の手順により、指摘メッセージを確認します。

1. 指摘メッセージ一覧を表示する
2. 指摘メッセージを確認する
3. 指摘メッセージの詳細を確認する
4. 指摘箇所のソースコードを確認する
5. 確認状態を記録する
6. 確認メモを記録する

指摘メッセージ一覧では、選択したチェック対象プロジェクトまたはチェック対象ファイルに対応する指摘メッセージを表示します。指摘メッセージ一覧の確認済、およびメモ欄を利用することにより、指摘の確認結果を記録できます。

3.2.1.1 指摘メッセージ一覧を表示する

指摘メッセージ一覧は、以下のビューにおける項目の選択と連動して、対象となる指摘を表示することができます。

- **パッケージ・エクスプローラ**

  選択したチェック対象プロジェクト、ソースフォルダ、パッケージ、チェック対象ファイルに対する指摘メッセージを表示します。

- **プロジェクト・エクスプローラ**

  選択したチェック対象プロジェクト、ソースフォルダ、パッケージ、チェック対象ファイルに対する指摘メッセージを表示します。

- **ナビゲータ**

  選択したチェック対象プロジェクト、チェック対象ファイル、フォルダに対する指摘メッセージを表示します。

- **判定結果**

  選択したファイルの指摘メッセージを表示します。

上記ビューの選択項目単位の動作は、以下のとおりです。

選択項目	説明
プロジェクト	選択したプロジェクト配下すべての指摘メッセージを表示します。 複数選択した場合は、初めに選択したプロジェクトが対象となります。
パッケージ	選択したパッケージ直下の指摘メッセージを表示します。 複数選択した場合は、初めに選択したパッケージが対象となります。
ファイル	選択したファイルの指摘メッセージを表示します。 複数選択した場合は、初めに選択したファイルが対象となります。
ソースフォルダ	選択したソースフォルダ配下すべての指摘メッセージを表示します。 複数選択した場合は、初めに選択したソースフォルダが対象となります。
フォルダ	選択したフォルダ直下の指摘メッセージを表示します。 複数選択した場合は、初めに選択したフォルダが対象となります。

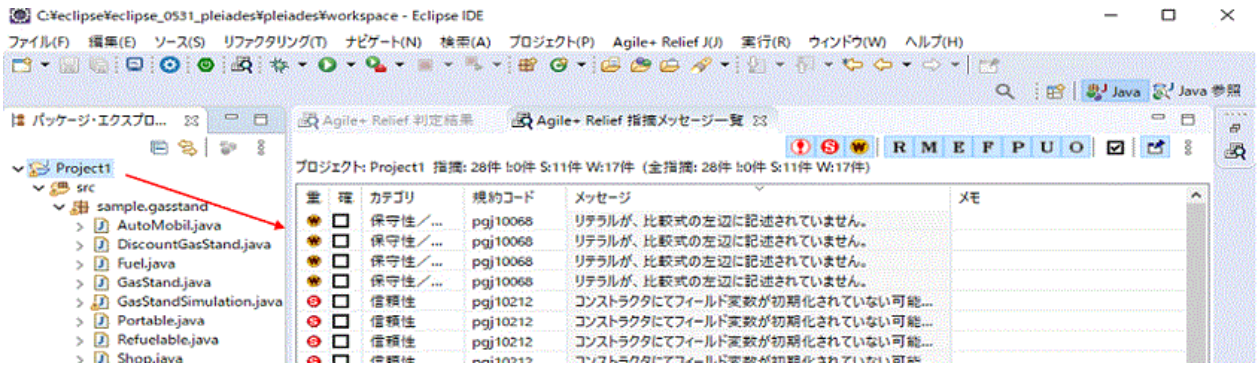
初期状態では他のビューとの連動はオンになっています。

指摘メッセージ一覧の表示内容をそのままで他のビューを操作したい場合には、ツールバーの (他のビューに連動する)をオフにすると他のビューと連動されなくなります。

例

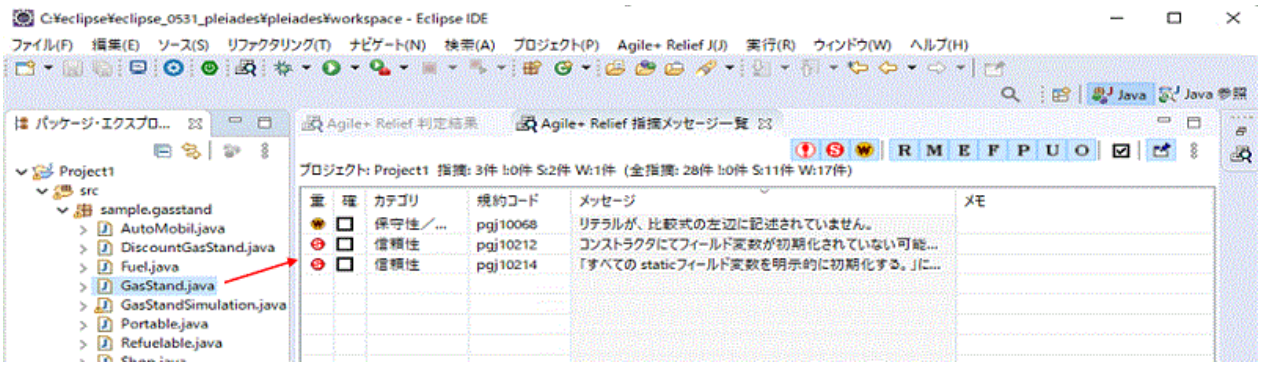
- パッケージ・エクスプローラ上のプロジェクト選択による表示

  パッケージ・エクスプローラ上でProject1を選択すると、指摘メッセージ一覧にProject1プロジェクトのJavaファイルに対する指摘が表示されます。



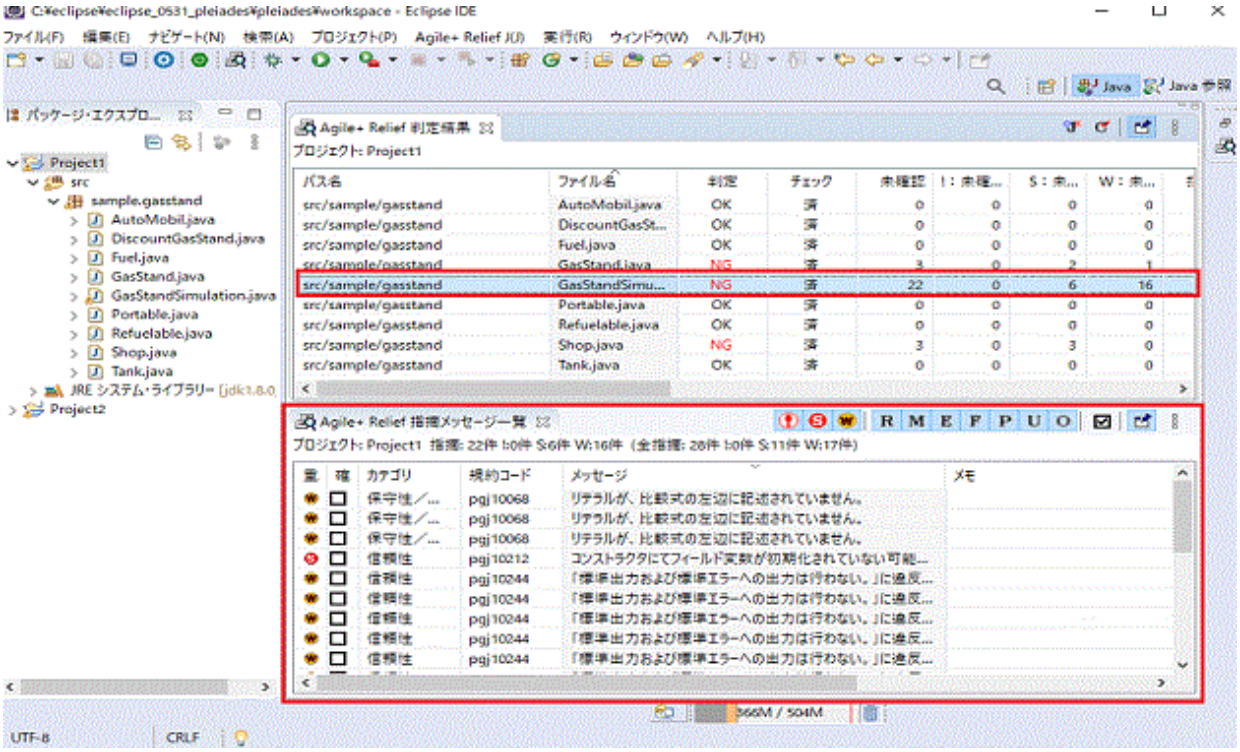
・ パッケージ・エクスプローラ上のJavaファイル選択による表示

パッケージ・エクスプローラ上のGasStand.javaを選択すると、指摘メッセージ一覧にGasStand.javaに対する指摘が表示されます。



・ 判定結果上のファイル選択による表示

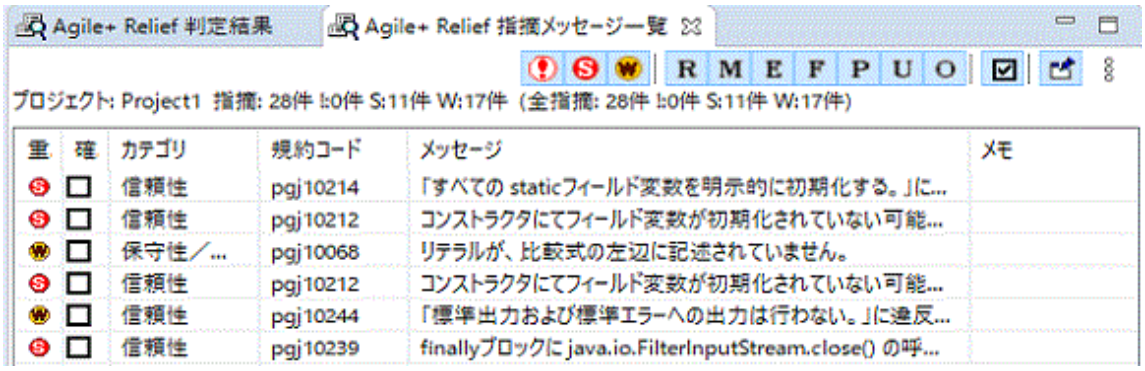
判定結果上でGasStand.javaの行を選択すると、指摘メッセージ一覧にGasStand.java に対する指摘が表示されます。



3.2.1.2 指摘メッセージを確認する

指摘メッセージ一覧により、指摘されたメッセージの内容を確認します。

指摘メッセージ一覧では、指摘内容および未確認の指摘件数を確認することができます。



指摘メッセージ一覧の表示項目は以下のとおりです。

ヘッダ部

表示項目	説明
プロジェクト	対象プロジェクト名を表示します。
指摘数	選択されたプロジェクトまたはファイルの指摘数を表示します。 重要度別(！、S、W)の内訳も表示します。
全指摘数	プロジェクト全体の指摘数を表示します。 重要度別(！、S、W)の内訳も表示します。

リスト部

表示項目	説明
重要度	規約の重要度をアイコンで表示します。アイコンには以下の種類があります。 :[重要度:！]:動作環境エラー :[重要度:S]:最重要と評価する規約 :[重要度:W]:重要と評価する規約
確認済	指摘箇所の確認状態を表示します。チェックボックスにより確認済のオン／オフが操作できます。 フィルタとの組み合わせで、不要なメッセージ(確認済など)を表示させないようにすることが可能です。
カテゴリ	規約のカテゴリを表示します。 カテゴリには以下の種類があります。 ・信頼性 ・保守性／可読性 ・効率性 ・機能性 ・移植性 ・ユーザ規約 ・その他(注1)
規約コード	規約コードを表示します。
メッセージ	規約違反の内容を表示します。
メモ	メモの内容を表示します。 修正検討中のコメントや、対処が不要な指摘に対する理由などをメモとして残すことに利用できます。

表示項目	説明
ファイル名	ファイル名を表示します。
パス名	パス名を表示します。
行番号	行番号を表示します。

(注1) FindBugsやPMDの一部の指摘は、「その他」のカテゴリになる場合があります。

3.2.1.3 指摘メッセージの詳細を確認する

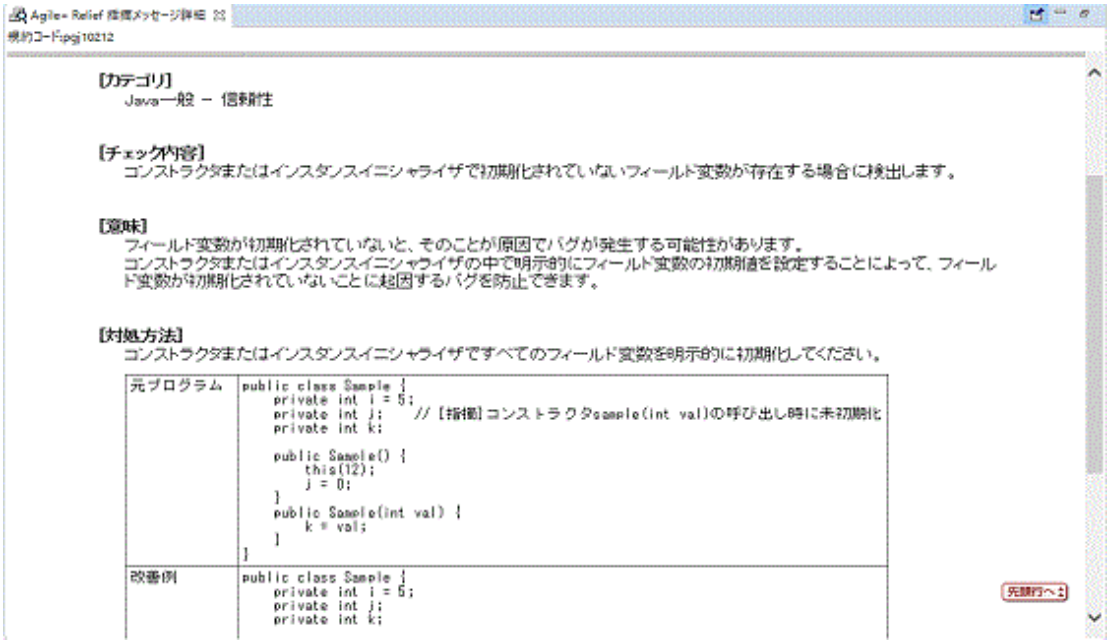
指摘メッセージに関する問題点や改善方法などの詳細を確認します。

指摘メッセージ一覧で詳細を確認したい指摘を選択すると、[Agile+ Relief 指摘メッセージ詳細]ビューに指摘の詳細内容が表示されます。

初期状態では他のビューとの連動はオンになっています。

指摘メッセージ詳細の表示内容をそのままで他のビューを操作したい場合には、ツールバーの (他のビューに連動する)をオフにすると他のビューと連動されなくなります。

表示内容



表示項目は以下のとおりです。

表示項目	説明
規約コード	規約のコードを表示します。
規約内容	規約の内容を表示します。
カテゴリ	規約のカテゴリを表示します。
チェック内容	規約を検出するためのチェック内容を表示します。
意味	規約の意味と問題点を表示します。
対処方法	規約への対処方法を例とともに表示します。

3.2.1.4 指摘箇所のソースコードを確認する

コードチェックで指摘された箇所をソースコードから確認します。

エディタ連携により、指摘メッセージから対象となるソースコードをエディタで開き、指摘箇所に位置づけることができます。

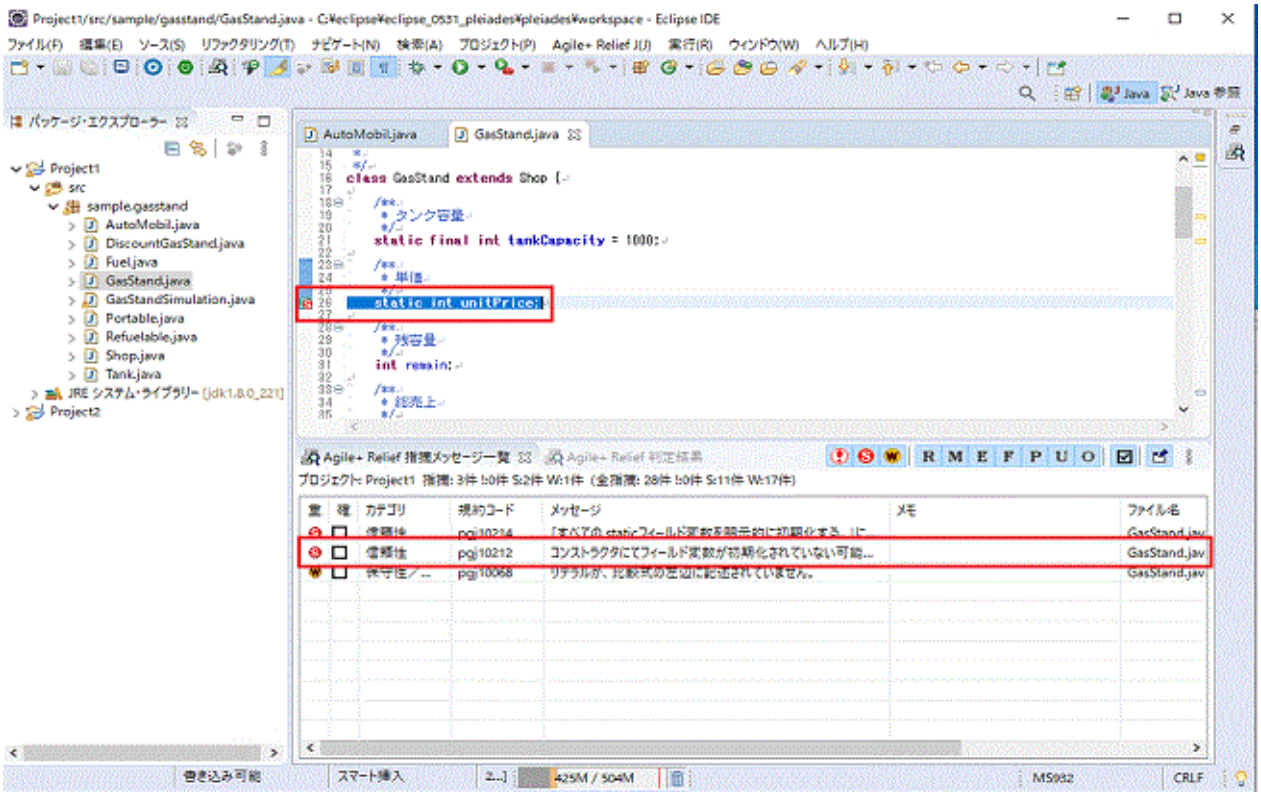
指摘メッセージの行番号が0の場合は、エディタ連携は以下のようになります。

- 対象ソースコードをエディタで開いていない場合、ソースコードをエディタで開いて先頭行に位置づけます。
- 対象ソースコードをすでにエディタで開いている場合、対象ソースコードを開いているエディタがそのまま表示されます。

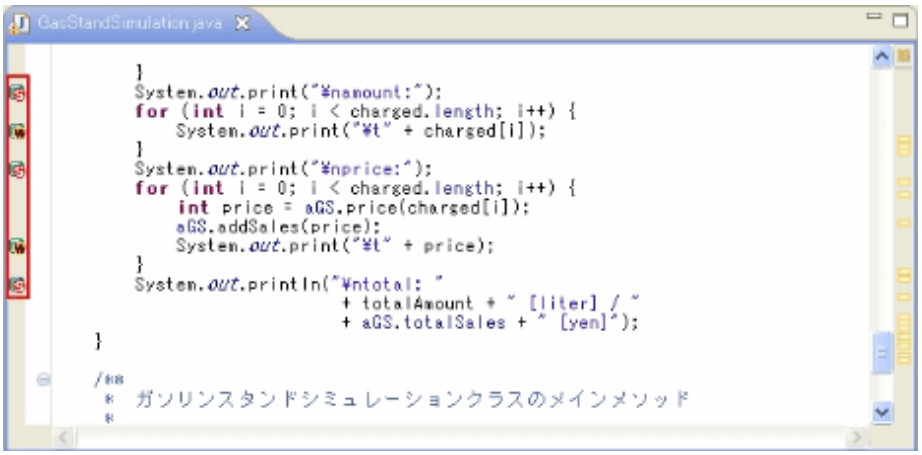
＜操作＞

1. 指摘メッセージ一覧の行を選択し、以下のいずれかの操作を行います。
   - ダブルクリック
   - ポップアップメニューを表示し、エディタ連携を選択します。

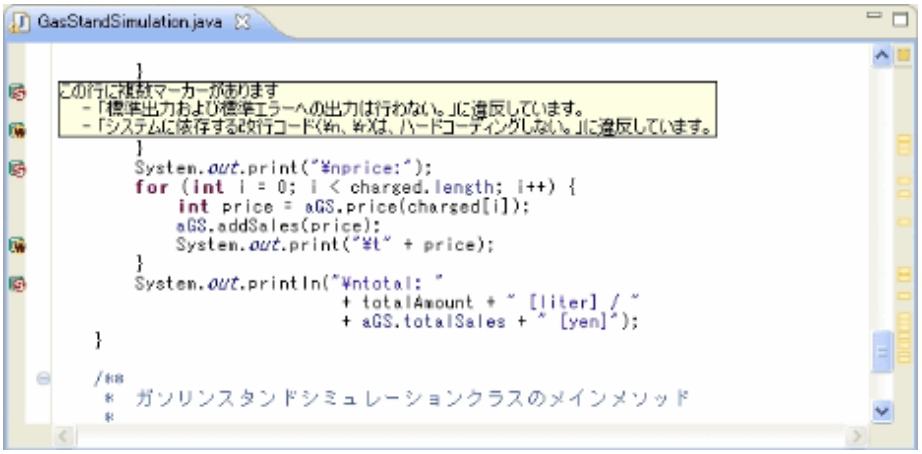
例:指摘メッセージ一覧上で指摘行のダブルクリックによる対象ファイルの指摘行にフォーカスを移動する場合



エディタのマーカー表示エリアには、コードチェック結果で規約違反のあった箇所が指摘の重要度を示すアイコンとしてマーカー表示されます。



マーカーにマウスカーソルを合わせると、指摘内容を表示できます。同一行に複数の指摘がされている場合には指摘内容はまとめて表示されます。



参考

Eclipse3.4では、マーカービューによりAgile* Relief指摘メッセージを確認することもできます。

3.2.1.5 確認状態を記録する

指摘メッセージの確認記録を、指摘メッセージの確認状態として記録します。

確認状態の記録には、指摘メッセージ一覧の[確認済]欄のチェックボックスを使います。確認済みはチェックし、確認の取り消しはチェックを外します。

確認の終わっていない指摘のみ表示したい場合には、確認済みフィルタをオフにすると指摘を絞り込むことができます。

＜操作＞

1. 指摘メッセージ一覧から記録する指摘メッセージを選択します。
2. 以下いずれかの操作により確認状態を記録します。
   - [確認済]欄のチェックボックスをチェックまたはチェックを外します。
   - ポップアップメニューを表示し、[確認済み]または[確認取り消し]を選択する。

確認済みとした指摘メッセージは、コードチェックを再実行した結果、再指摘された場合に確認済みのまま記録が残りますが、指摘された行のソースコードが前回と異なる場合には、確認済みのチェックが外されます。

3.2.1.6 確認メモを記録する

指摘メッセージの確認記録としてメモが必要な場合、指摘メッセージの確認メモとして記録します。

対処が不要な指摘の理由を記録すると、次回コードチェック結果で同じ箇所が指摘されたときに、メモにより対処が不要なことが分かるため、指摘内容の再確認が不要となります。

＜操作＞

1. 指摘メッセージを選択します。
2. メモ欄をクリックします。

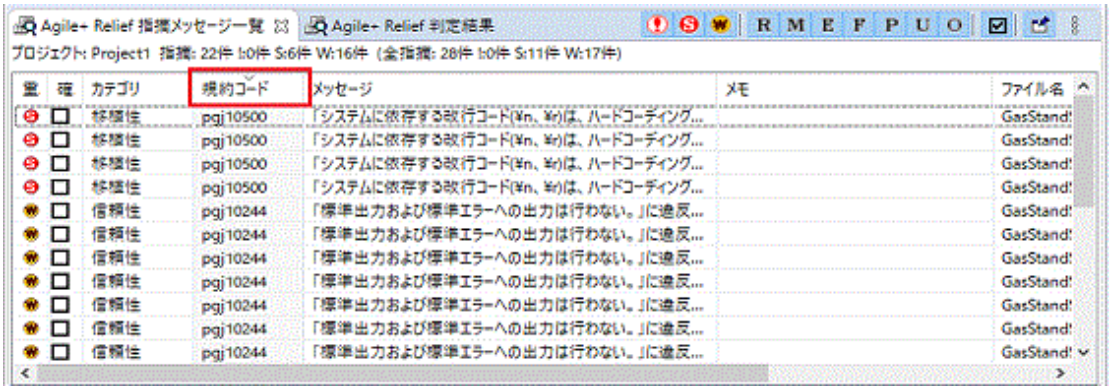
   入力フィールドが表示されます。
3. メモを入力し、Enterキーを押下します。

3.2.1.7 指摘メッセージ一覧をソートする

指摘メッセージの表示内容をソートします。

指摘メッセージ一覧のヘッダ上の各項目名をクリックすることで表示内容をソートすることができます。

例：規約コードでソートした場合



3.2.1.8 指摘メッセージ一覧をフィルタリングする

指摘メッセージの表示内容の条件によってフィルタリングします。

フィルタリングを行うための条件には、大きく重要度、カテゴリ、確認の3つの分類があります。

フィルタの種類

分類	アイコン	フィルタ項目
重要度		重要度:I
		重要度:S
		重要度:W
カテゴリ	R	信頼性
	M	保守性／可読性

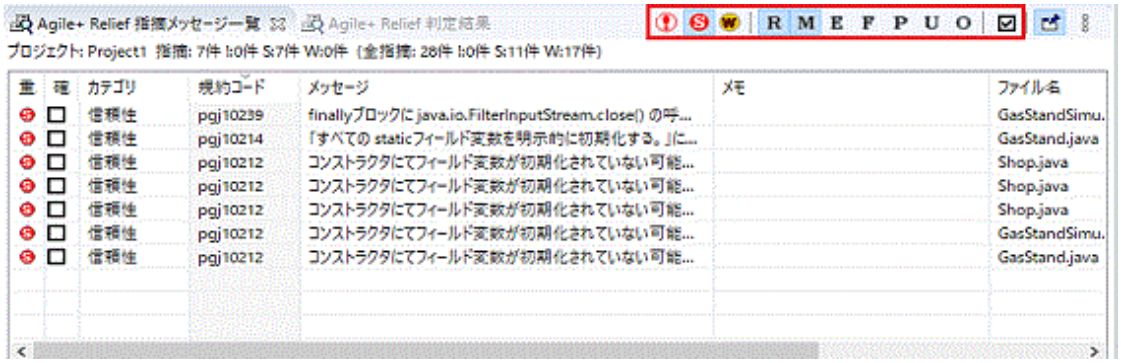
分類	アイコン	フィルタ項目
	E	効率性
	F	機能性
	P	移植性
	U	ユーザ規約
	O	その他
確認	☑	確認済

フィルタリングの指定は、ツールバーのアイコン、ビューメニュー、ポップアップメニューの3つの方法があります。

- ツールバーのアイコンで指定

  ツールバーから任意のフィルタ用アイコンを選択します。

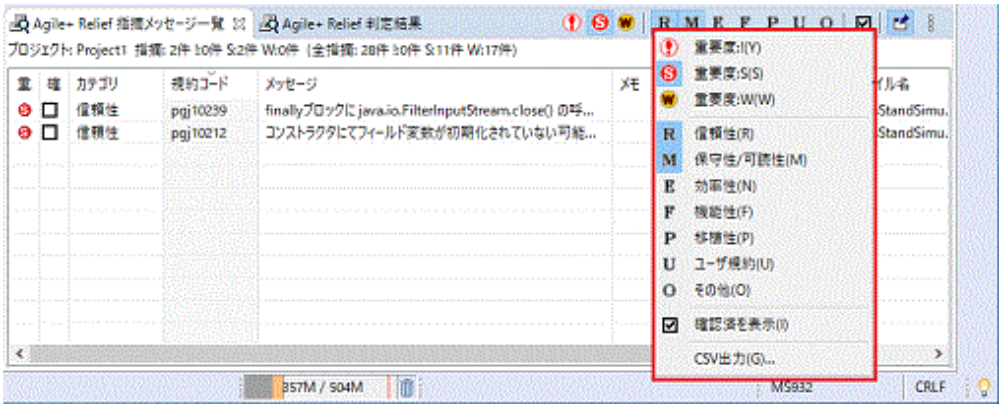
  例：重要度:S、カテゴリ：R（信頼性）、M（保守性／可読性）を指定した場合



- ビューメニューで指定

  ビューメニューから任意のフィルタ項目を選択します。

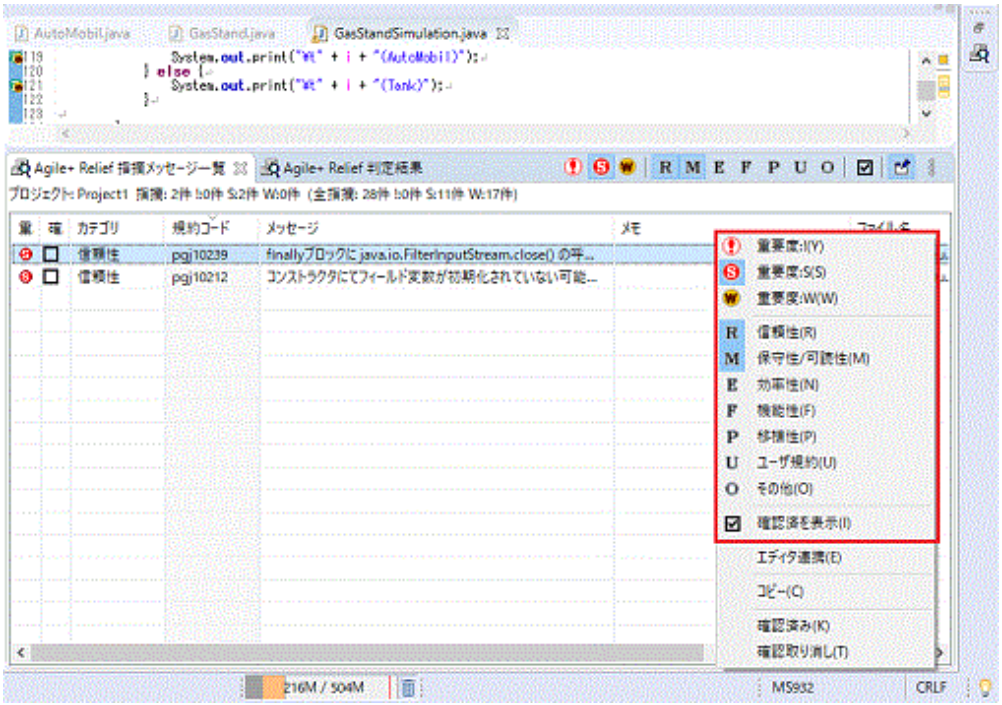
  例：重要度:S、カテゴリ：R（信頼性）、M（保守性／可読性）を指定した場合



- ポップアップメニューで指定

  ポップアップメニューから任意のフィルタ項目を選択します。

  例：重要度:S、カテゴリ:R（信頼性）、M（保守性／可読性）を指定した場合



3.2.2 判定結果を確認する

判定結果では、コードチェック結果、および指摘メッセージの確認状態をもとに以下の項目を表示します。

- 判定結果（OK／NG）
- コードチェック実行の有無
- 未確認指摘数（重要度）
- 指摘数（重要度別、カテゴリ別）

判定結果の診断は、重要度が"！"または"S"の未確認の指摘が1件以上の残っている場合はNGを表示し、それ以外の場合はOKを表示します。

判定結果の確認では、ファイル毎の判定結果がOKかNGかを確認し、NGのファイルについては指摘メッセージを確認してください。

3.2.2.1 判定結果を表示する

判定結果は、以下のビューにおける選択と連動して、対象となる資産の判定結果を表示することができます。

- **パッケージ・エクスプローラ**

  選択したチェック対象プロジェクト、ソースフォルダ、パッケージ、チェック対象ファイルに対する判定結果を表示します。

- **プロジェクト・エクスプローラ**

  選択したチェック対象プロジェクト、ソースフォルダ、パッケージ、チェック対象ファイルに対する判定結果を表示します。

・ **ナビゲータ**

選択したチェック対象プロジェクト、チェック対象ファイル、フォルダに対する判定結果を表示します。

上記ビューの選択項目単位の動作は、以下のとおりです。

選択項目	説明
プロジェクト	選択したプロジェクト配下すべての判定結果を表示します。 複数選択した場合は、初めに選択したプロジェクトが対象となります。
パッケージ	選択したパッケージ直下の判定結果を表示します。 複数選択した場合は、初めに選択したパッケージが対象となります。
ファイル	選択したファイルの判定結果を表示します。 複数選択した場合は、初めに選択したファイルが対象となります。
ソースフォルダ	選択したソースフォルダ配下すべての判定結果を表示します。 複数選択した場合は、初めに選択したソースフォルダが対象となります。
フォルダ	選択したフォルダ直下の判定結果を表示します。 複数選択した場合は、初めに選択したフォルダが対象となります。

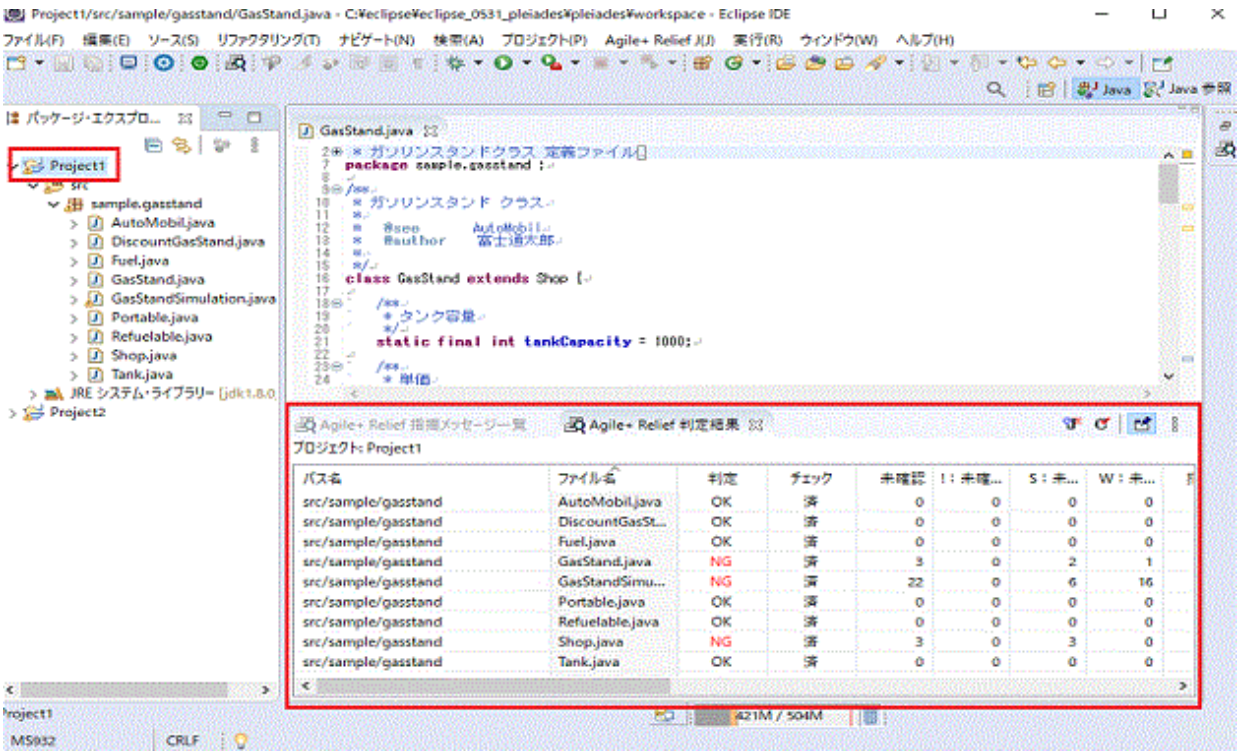
初期状態では他のビューとの連動はオンになっています。

判定結果の表示内容をそのままで他のビューを操作したい場合には、ツールバーの  (他のビューに連動する) をオフにすると他のビューと連動されなくなります。

例

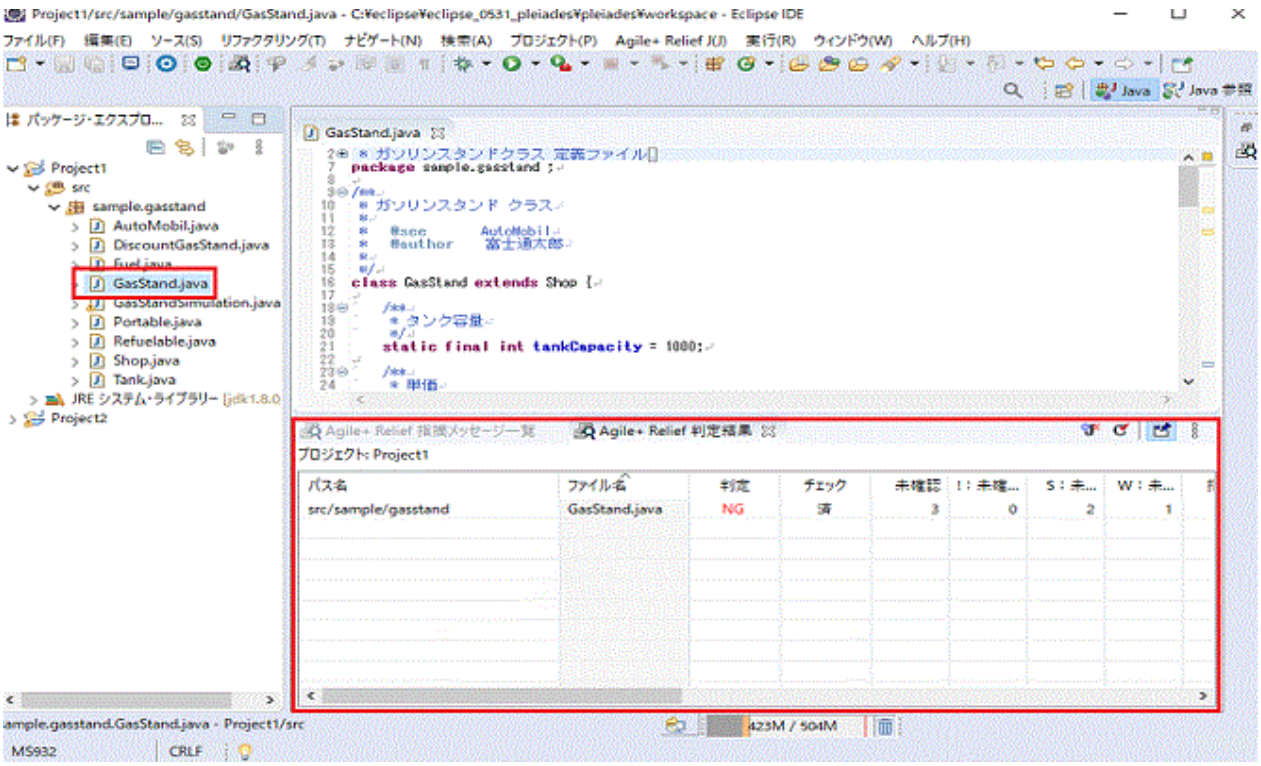
- パッケージ・エクスプローラ上のプロジェクト選択による表示

  パッケージ・エクスプローラ上でProject1を選択すると、判定結果にProject1のJavaファイルに対する判定結果が表示されます。



- パッケージ・エクスプローラ上のJavaファイル選択による表示

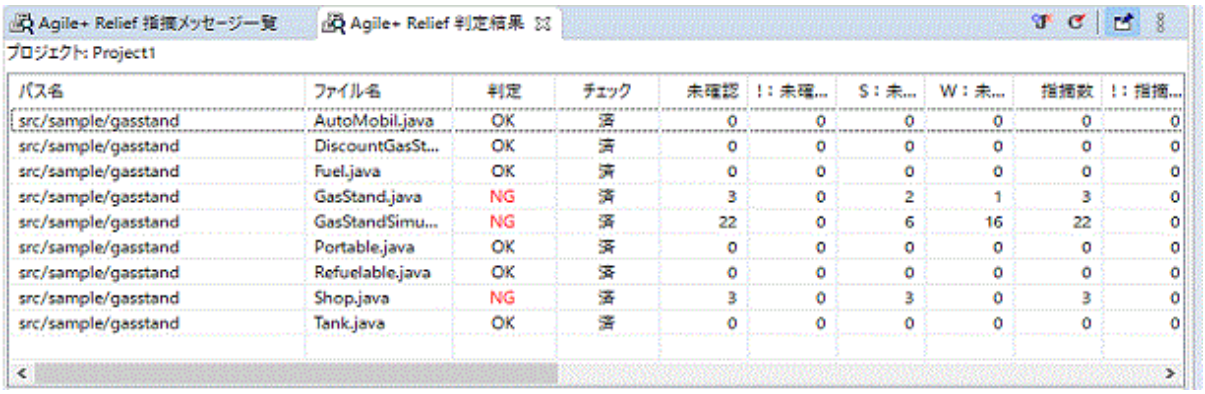
  パッケージ・エクスプローラ上でGasStand.javaを選択すると、判定結果にGasStand.javaに対する判定結果が表示されます。



3.2.2.2 判定結果を確認する

チェック対象ファイル単位に判定結果がOKかNGかを確認します。

NGの場合は、重要度が"S"または"!"の未確認の指摘が残っているため、再度、指摘メッセージ一覧で対象の指摘について検討してください。



表示内容は以下のとおりです。

ヘッダ部

表示項目	説明
プロジェクト	対象プロジェクト名を表示します。

リスト部

表示項目	説明
パス名	パス名を表示します。
ファイル名	ファイル名を表示します。
判定	未確認の指摘数をもとに合否判定を表示します。 "OK":[!:未確認]、および[S:未確認]が0件の場合 "NG":[!:未確認]、または[S:未確認]が1件以上の場合 "ー":コードチェック未実施、またはコードチェック対象外ファイル(注1)の場合
チェック	コードチェックの実施状態を表示します。 "済":コードチェック実施済の場合 "未":コードチェック未実施、またはチェック結果をクリアした場合 "ー":コードチェック対象外ファイル(注1)の場合
未確認	未確認の指摘総数を表示します。 指摘メッセージ一覧で[確認済]をチェックしていない指摘が、未確認の指摘としてカウントされます。
!:未確認	重要度:!の未確認指摘数を表示します。
S:未確認	重要度:Sの未確認指摘数を表示します。
W:未確認	重要度:Wの未確認指摘数を表示します。
指摘数	指摘総数を表示します。
!:指摘数	重要度:!の指摘数を表示します。
S:指摘数	重要度:Sの指摘数を表示します。
W:指摘数	重要度:Wの指摘数を表示します。
S:信頼性	信頼性の重要度:Sの指摘数を表示します。
S:保守性／可読性	保守性／可読性の重要度:Sの指摘数を表示します。
S:効率性	効率性の重要度:Sの指摘数を表示します。
S:機能性	機能性の重要度:Sの指摘数を表示します。
S:移植性	移植性の重要度:Sの指摘数を表示します。
S:ユーザ規約	ユーザ規約の重要度:Sの指摘数を表示します。
S:その他	その他の重要度:Sの指摘数を表示します。
W:信頼性	信頼性の重要度:Wの指摘数を表示します。
W:保守性／可読性	保守性／可読性の重要度:Wの指摘数を表示します。
W:効率性	効率性の重要度:Wの指摘数を表示します。
W:機能性	機能性の重要度:Wの指摘数を表示します。
W:移植性	移植性の重要度:Wの指摘数を表示します。
W:ユーザ規約	ユーザ規約の重要度:Wの指摘数を表示します。
W:その他	その他の重要度:Wの指摘数を表示します。

（注1）コードチェック対象外のファイルとは、カスタマイザで定義された[ファイル除外]の条件に一致したファイルを指します。詳しくは、[Agile* Relief Jカスタマイザ操作手引書]を参照してください。

3.2.2.3 判定対象のソースコードを確認する

判定対象のソースコードをエディタで確認します。

エディタ連携により、対象ファイルが以下のように表示されます。

- 対象ソースコードをエディタで開いていない場合、ソースコードをエディタで開いて先頭行に位置づけます。
- 対象ソースコードをすでにエディタで開いている場合、対象ソースコードを開いているエディタがそのまま表示されます。

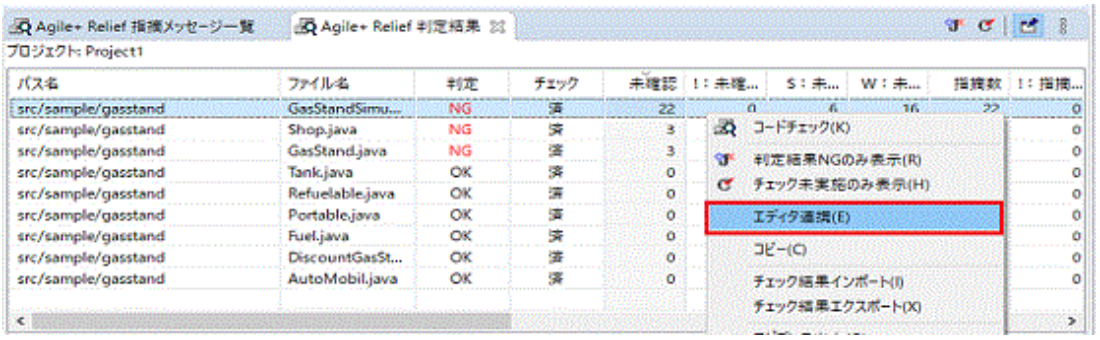
エディタ連携の操作は以下のとおりです。

- ファイル名をダブルクリック

  ファイル名をダブルクリックし、ソースファイルを表示します。

- ポップアップメニューからのジャンプ

  ポップアップメニューから[エディタ連携]を選択してソースファイルを表示します。



3.2.2.4 判定結果をソートする

判定結果の表示項目名単位に、判定結果の内容をソートできます。

ヘッダ上の表示項目名をクリックすると昇順／降順が切り替わり、判定結果の内容がソートされて表示されます。

例：未確認総指摘数でソートした場合

3.2.2.5 判定結果をフィルタリングする

判定結果の表示内容を、以下の条件によりフィルタリングします。

アイコン	条件	説明
	判定結果NGのみ表示	[判定]が"NG"の判定結果のみ表示します。
	チェック未実施のみ表示	[チェック]が"未"の判定結果のみ表示します。

フィルタリングを設定するには、ツールバーのアイコン、ビューメニュー、ポップアップメニューの3つの方法があります。

- ツールバーからフィルタリング

  ツールバーから条件のアイコンを選択すると、それぞれの条件にしたがって判定結果がフィルタリングされます。

  例：判定結果NGのみ表示、チェック未実施のみ表示とも指定しない場合

パス名	ファイル名	判定	チェック	未確認	! : 未確...	S : 未...	W : 未...	指摘数	! : 指摘...
src/sample/gasstand	AutoMobil.java	OK	済	0	0	0	0	0	0
src/sample/gasstand	DiscountGasSt...	OK	済	0	0	0	0	0	0
src/sample/gasstand	Fuel.java	OK	済	0	0	0	0	0	0
src/sample/gasstand	GasStand.java	NG	済	3	0	2	1	3	0
src/sample/gasstand	GasStandSimu...	NG	済	22	0	6	16	22	0
src/sample/gasstand	Portable.java	OK	済	0	0	0	0	0	0
src/sample/gasstand	Refuelable.java	OK	済	0	0	0	0	0	0
src/sample/gasstand	Shop.java	NG	済	3	0	3	0	3	0
src/sample/gasstand	Tank.java	OK	済	0	0	0	0	0	0

- ビューメニューからフィルタリング

  ビューメニューから条件の項目を選択すると、それぞれの条件にしたがって判定結果がフィルタリングされます。

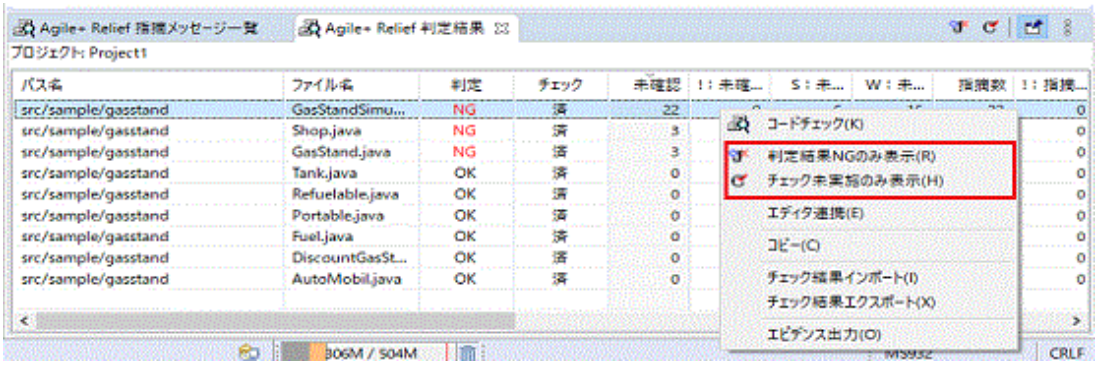
  例：判定結果NGのみ表示、チェック未実施のみ表示とも指定しない場合

パス名	ファイル名	判定	チェック	未確認	! : 未確...
src/sample/gasstand	GasStandSimu...	NG	済	22	0
src/sample/gasstand	Shop.java	NG	済	3	0
src/sample/gasstand	GasStand.java	NG	済	3	0
src/sample/gasstand	Tank.java	OK	済	0	0
src/sample/gasstand	Refuelable.java	OK	済	0	0
src/sample/gasstand	Portable.java	OK	済	0	0
src/sample/gasstand	Fuel.java	OK	済	0	0
src/sample/gasstand	DiscountGasSt...	OK	済	0	0
src/sample/gasstand	AutoMobil.java	OK	済	0	0

- ポップアップメニューからフィルタリング

  ポップアップメニューから条件のメニューを選択すると、それぞれの条件にしたがって判定結果がフィルタリングされます。

  例：判定結果NGのみ表示、チェック未実施のみ表示とも指定しない場合



パス名	ファイル名	判定	チェック	未確認	! : 未確...	S : 未...	W : 未...	指摘数	! : 指摘...
src/sample/gasstand	GasStandSimu...	NG	済	22					0
src/sample/gasstand	Shop.java	NG	済	3					0
src/sample/gasstand	GasStand.java	NG	済	3					0
src/sample/gasstand	Tank.java	OK	済	0					0
src/sample/gasstand	Refuelable.java	OK	済	0					0
src/sample/gasstand	Portable.java	OK	済	0					0
src/sample/gasstand	Fuel.java	OK	済	0					0
src/sample/gasstand	DiscountGasSt...	OK	済	0					0
src/sample/gasstand	AutoMobil.java	OK	済	0					0

3.2.3 他のビューとの連動を抑止する

[Agile+ Relief 指摘メッセージ一覧]ビュー、[Agile+ Relief 指摘メッセージ詳細]ビュー、および[Agile+ Relief 判定結果]ビューは、初期状態で他のビューと連動して表示を切り換えるように設定されています。

他のビューとの連動を抑止したい場合は、以下の操作を行ってください。

＜操作＞

1. ツールバーの (他のビューに連動する) をオフにします。

3.3 コードチェック結果を出力する

コードチェック結果をファイルに出力します。

あらかじめ対象となるチェック対象プロジェクトを開き、かつ[Agile+ Relief J]メニューの[Agile+ Relief Jを使用する]を有効にしてください。

コードチェック結果の出力には、[Agile+ Relief 指摘メッセージ一覧]ビュー、および[Agile+ Relief 判定結果]ビューを利用します。

3.3.1 指摘メッセージを出力する

[Agile+ Relief 指摘メッセージ一覧]ビューの現在の表示内容(ソート、フィルタリングされた状態)を出力します。
各項目の表示内容は、カンマ区切りによるCSV形式でファイルに保存されます。以下の項目については、ビューの表示内容と異なる点があります。

- 確認済

  チェックボックスのチェックにより以下の形式で出力されます。

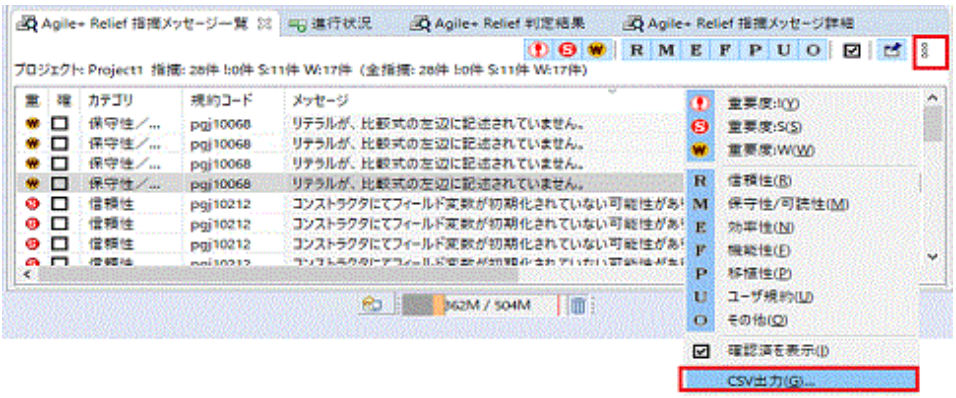
確認済	出力
チェックなし	0
チェックあり	1

- メッセージ

  メッセージにカンマ「,」、ダブルクォート「"」が含まれる場合、メッセージはダブルクォートで囲んで出力されます。また、メッセージは複数行に渡ることがあり、この場合にもメッセージはダブルクォートで囲んで出力されます。

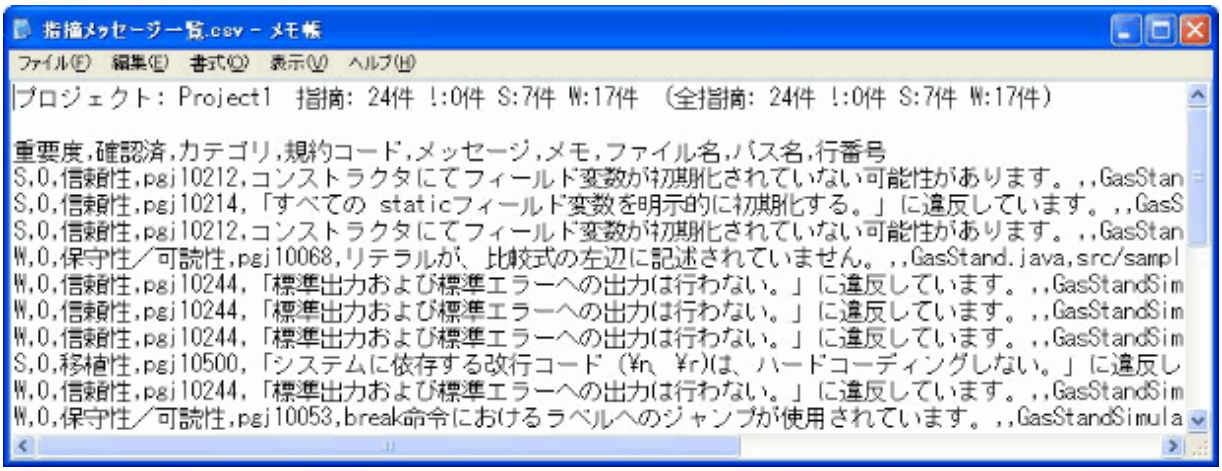
＜操作＞

1. [Agile+ Relief 指摘メッセージ一覧]ビューのビューメニューを表示し、[CSV出力(G)]を選択します。



2. [指摘メッセージの出力]ダイアログで、指摘メッセージ(*.csv)の出力名を指定して、[保存(S)]ボタンを選択します。

CSV出力イメージ



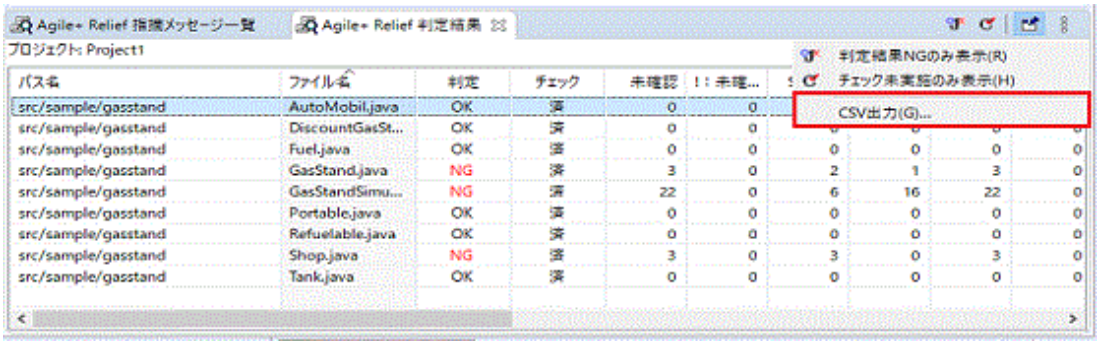
3.3.2 判定結果を出力する

[Agile+ Relief 判定結果]ビューで表示されている内容(ソート、フィルタリングされた状態)を出力します。

各項目の表示内容はカンマ区切りによるCSV形式でファイルに保存します。

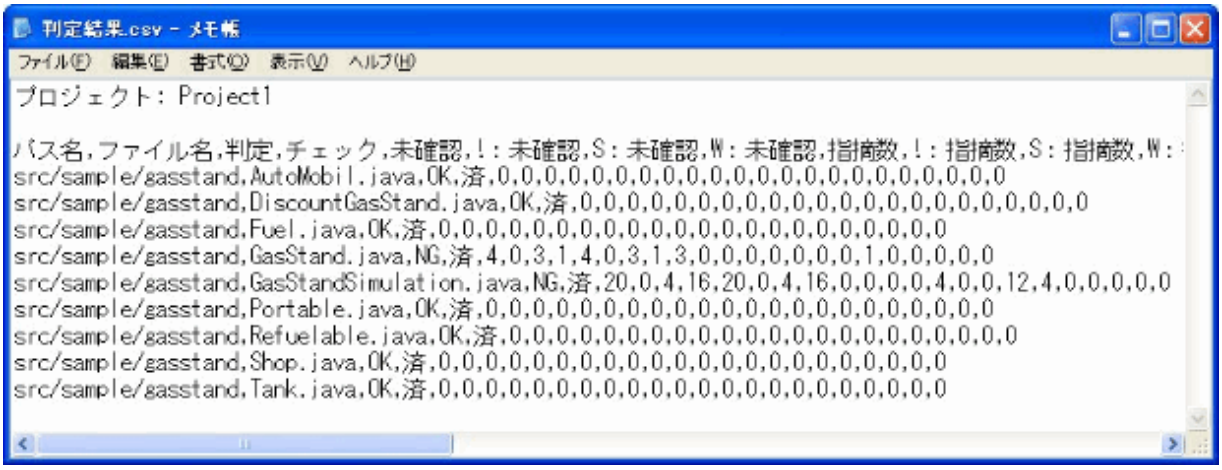
＜操作＞

1. [Agile+ Relief 判定結果]ビューのビューメニューを表示し、[CSV出力(G)]を選択します。



2. [判定結果の出力]ダイアログで判定結果(*.csv)の出力名を指定して、[保存(S)]ボタンを選択します。

CSV出力イメージ

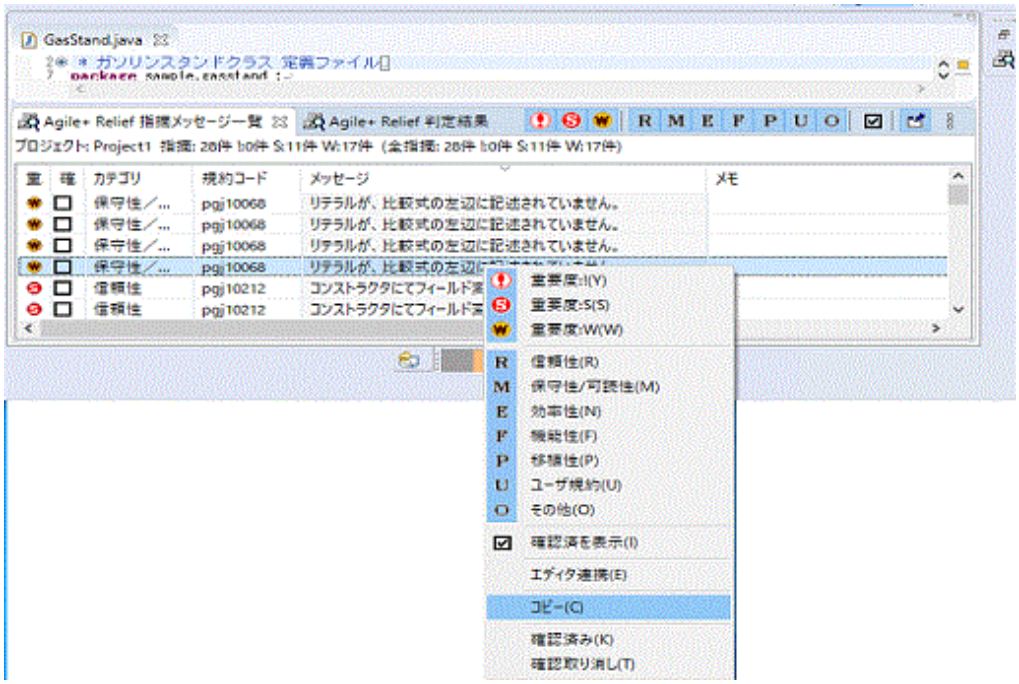


3.3.3 指摘メッセージをコピーする

選択された指摘メッセージの内容をクリップボードにコピーします。また、複数のメッセージを同時に選択することもできます。

＜操作＞

1. [Agile+ Relief 指摘メッセージ一覧]ビューでコピーしたい指摘メッセージを選択します。
2. ポップアップメニューを表示し、[コピー(C)]を選択します。

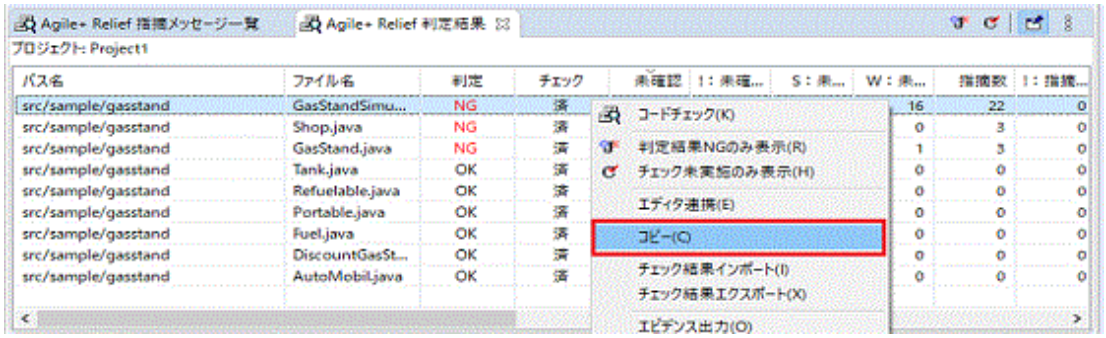


3.3.4 判定結果をコピーする

選択された判定結果の内容をクリップボードにコピーします。また、複数の判定結果を同時に選択することもできます。

＜操作＞

1. [Agile+ Relief 判定結果]ビューでコピーしたい判定結果を選択します。
2. ポップアップメニューを表示し、[コピー(C)]を選択します。



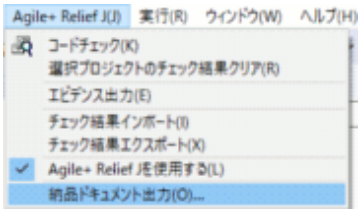
3.3.5 納品ドキュメントを出力する【Kiyacker互換】

コードチェックの結果をまとめて納品ドキュメント(CSV形式)として出力します。

納品ドキュメントファイルでは、各フォルダのチェック対象ファイルが、コーディング規約のどの規約コードで違反したかを確認できます。プロジェクト単位で、コーディング規約を遵守しているかをまとめるのが容易になります。

＜操作＞

1. パッケージ・エクスプローラ、プロジェクト・エクスプローラ、またはナビゲータのビューでチェック対象プロジェクトを1つ選択します。
2. [Agile+ Relief J(J)]メニュー＞[納品ドキュメント出力(O)]を選択します。



3. [納品ドキュメントの出力]ダイアログで、納品ドキュメント(*.csv)の出力名を指定して[保存(S)]ボタンを選択します。

納品ドキュメントには、規約適用された規約コード、規約内容、およびフォルダ内の各ファイルの規約違反数を1行1規約コードにして、フォルダ単位にまとめて出力します。

対象となるフォルダは、Eclipseプロジェクト配下のサブフォルダを含めたすべてのフォルダです。

納品ドキュメントの詳細は、以下を参照してください。

出力フォーマット

```
プロジェクト名,(1: プロジェクト名),日付,(2: 日付),(3: 時間)

フォルダ,(4: フォルダ)

規約コード,規約内容,(5:ファイル名 ...)
(6: 規約コード),(7: 規約内容),(8: 規約違反数)
  ....
```

出力項目一覧

項番	項目	出力内容
1	プロジェクト名	検査規約定義ファイル(pgrj)に設定されているプロジェクト名が出力されます。
2	日付	納品形式ドキュメントが作成された日付を出力します。 "YYYY/MM/DD"の形式で出力します。
3	時間	納品形式ドキュメントが作成された時間を出力します。 "HH:MM"の形式で出力します。
4	フォルダ	ファイルが格納されているフォルダパスを出力します。 フォルダはサブフォルダを含めて、複数出力します。
5	ファイル名	フォルダ内のファイル名をカンマ区切りですべて出力します。 ・コードチェックが実行されていない、またはコードチェック実行後にファイルが更新された場合は「ファイル名（未実施資産）」と出力します。 ・チェック対象外のファイルの場合は「ファイル名（チェック対象外）」と出力します。 ・1行に出力されるファイル名の個数は、254ファイルです。
6	規約コード	規約適用されている規約コードを、1行1コード単位で出力します。
7	規約内容	規約コードに対する規約内容を出力します。
8	規約違反数	コードチェック対象ファイルの規約コードに対する規約違反数を出力します。 ・チェック適用されていない規約に対しては、"-"(ハイフン)を出力します。

		・コードチェック未実施資産、またはチェック対象外のファイルの場合、規約違反数の欄には何も出力されません。

出力例

```
プロジェクト名,デフォルト,日付,2009/04/06,11:44

フォルダ,C:¥workspace¥Project2¥src¥sample¥student

規約コード,規約内容,Main.java,Person.java,Student.java,Teacher.java
K1C10026,switch文で breakしないでその下へ抜ける場合は、その位置に必ずコメントを記述する。,0,0,0,0
K1C10045,do-while文は使用しない。,0,0,0,0
K1C10046,break命令によるラベルへのジャンプは使用しない。,0,0,0,0
K1C10047,switch文の最後の caseは、defaultラベルを必ず記述する。,0,0,0,0
K1C10050,for文では、評価部に複雑な式を記述しないこと。,0,0,0,0     ...

フォルダ, C:¥workspace¥Project2¥src¥sample¥student¥type

規約コード,規約内容,Addr.java,Info.java,Sex.java,Subject.java,Town.java
K1C10026,switch文で breakしないでその下へ抜ける場合は、その位置に必ずコメントを記述する。,0,0,0,0,0
K1C10045,do-while文は使用しない。,0,0,0,0,0
K1C10046,break命令によるラベルへのジャンプは使用しない。,0,0,0,0,0
K1C10047,switch文の最後の caseは、defaultラベルを必ず記述する。,0,0,0,0,0
K1C10050,for文では、評価部に複雑な式を記述しないこと。,0,0,0,0,0
    ...
```

参考

- フォルダ内のファイル数が254を超える場合

  254ファイルの規約サマリが出力された後に、255以上のファイルについて規約サマリを続けて出力します。

  例）sample001.java～sample256.javaの連番ファイルがある場合の規約サマリの出力

```
規約コード,規約内容,sample001.java,sample002.java,sample003.java,sample004.java, ... ,sample254.java
    ...

規約コード,規約内容,sample255.java,sample256.java
    ...
```

- 行が65,536行を超えてしまう場合

  超えた行からは、以下の命名規則にしたがって出力します。

```
csvファイル名_枝番.csv
```

  ※枝番は2から振られます。

  例）sample-report.csvを出力先に指定し、65,536行を超えた場合の出力ファイル名

```
sample-report.csv
sample-report_2.csv
```

3.4 コードチェック結果をクリアする

選択したチェック対象プロジェクトのチェック結果をクリアします。また複数のチェック対象プロジェクトを選択することもできます。

なお、当機能を利用するには、あらかじめ対象となるプロジェクトを開き、かつ[Agile+ Relief J]メニューの[Agile+ Relief Jを使用する]を有効にしてください。

※指摘メッセージに対する「確認済」、「メモ」などの情報もクリアされます。

※チェック対象プロジェクトを選択していないときは機能を利用できません。

＜操作＞

1. パッケージ・エクスプローラ、プロジェクト・エクスプローラ、またはナビゲータのビューでチェック対象プロジェクトを選択します。
2. 次のいずれかの操作を行います。
   - [Agile+ Relief J(J)]メニュー＞[選択プロジェクトのチェック結果クリア(R)]を選択します。
   - ポップアップメニューから[Agile+ Relief J(J)]＞[選択プロジェクトのチェック結果クリア(R)]を選択します。

3.5 コードチェックのエビデンスを出力する

コードチェック結果のエビデンスを出力します。出力するエビデンスには、以下の5種類のファイルがあります。これらのエビデンスファイルは、すべてカンマ区切りによるCSV形式のファイルとして出力されます。お使いの品質管理システムなどに取り込むことで、Agile+ Relief Jによるソースコード解析プロセスのエビデンスとして利用することができます。

No	エビデンスファイル名	説明
1	MessageList.csv	検出された指摘メッセージの一覧を出力します。
2	CheckStatus_File.csv	チェック対象ファイル単位に、コードチェックの実行状況を出力します。実行状況には、コードチェック実施の有無やチェックオプションなどがあります。
3	CheckStatus_Rule.csv	チェック対象ファイル単位に、チェックが実行された規約の一覧と検出指摘数を出力します。
4	Metrics_File.csv	チェック対象ファイル単位に、行数などの情報を出力します。
5	Metrics_Func.csv	チェック対象ファイルがJavaファイルの場合、その中で定義されているメソッド単位に行数や複雑度などの情報を出力します。

注意

Agile+ Relief Jには、[Agile+ Relief 指摘メッセージ一覧]ビューや[Agile+ Relief 判定結果]ビューから指摘結果を出力する機能があり、エビデンスの一部の情報を出力できます。しかし、ビューのフィルタ状態により出力結果が異なるなど、利用者の出力ミスを引き起こす可能性があります。それに対し、エビデンス出力機能は、必ずすべてのコードチェック結果情報を出力するため、そのような出力ミスが起こりません。第三者にコードチェック結果を確認してもらうなどの場合、エビデンス出力機能の利用を推奨します。

参考

Agile+ Relief Jでは、これらのエビデンスファイルを入力し、その情報を分析した結果をMicrosoft(R) Office Excel(R)のブック形式のレポートとして出力する「診断レポート生成ツール」を提供しています。「診断レポート生成ツール」は、Windowsのスタートメニューまたはスタート画面(Windows(R) 8、Windows Server(R) 2012以降の場合)から利用してください。

3.5.1 エビデンスを出力する

選択したチェック対象プロジェクト、またはチェック対象ファイルのコードチェックによる検証結果のエビデンスを出力します。それらは各々を複数選択してエビデンスを出力することができます。ただし、以下のような選択はできません。

- チェック対象プロジェクトとチェック対象ファイルを一緒に選択する

・ 異なるチェック対象プロジェクトのファイルを一緒に選択する

なお、当機能を利用するには、あらかじめ対象となるプロジェクトを開き、かつ[Agile+ Relief J]メニューの[Agile+ Relief Jを使用する]を有効にしてください。

＜操作＞ チェック対象プロジェクトを選択してエビデンス出力する場合

1. パッケージ・エクスプローラ、プロジェクト・エクスプローラ、またはナビゲータのビューでチェック対象プロジェクトを選択します。
2. 次のいずれかの操作を行います。
   - [Agile+ Relief J(J)]メニュー＞[エビデンス出力(E)]を選択します。
   - ポップアップメニューから[Agile+ Relief J(J)]＞[エビデンス出力(E)]を選択します。
3. [エビデンス出力先フォルダ選択]ダイアログで、出力先となるフォルダを指定して、[OK]ボタンをクリックします。
4. 以下のフォルダに出力されます。

＜操作＞3. で指定したフォルダ¥*＜操作＞1. で選択したチェック対象プロジェクト名*

例）

＜操作＞1. で選択したチェック対象プロジェクト名	Sample
＜操作＞3. で指定したフォルダ	C:¥AgilePlusReliefEvidence
出力先フォルダ	C:¥AgilePlusReliefEvidence¥Sample

＜操作＞ チェック対象ファイルを選択してエビデンス出力する場合

1. パッケージ・エクスプローラ、プロジェクト・エクスプローラ、ナビゲータ、またはAgile+ Relief 判定結果のビューでチェック対象ファイルを選択します。
2. 次のいずれかの操作を行います。
   - [Agile+ Relief J(J)]メニュー＞[エビデンス出力(E)]を選択します。
   - パッケージ・エクスプローラ、プロジェクト・エクスプローラ、またはナビゲータの場合、ポップアップメニューから[Agile+ Relief J(J)]＞[エビデンス出力(E)]を選択します。
   - Agile+ Relief 判定結果の場合、ポップアップメニューから [エビデンス出力(O)]を選択します。
3. [エビデンス出力先フォルダ選択]ダイアログで、出力先となるフォルダを指定して、[OK]ボタンをクリックします。
4. 以下のフォルダに出力されます。

＜操作＞3. で指定したフォルダ¥*＜操作＞1. で選択したチェック対象ファイルが属するEclipseプロジェクト名*

例）

＜操作＞1. で選択したチェック対象ファイルが属するEclipseプロジェクト名	Sample
＜操作＞3. で指定したフォルダ	C:¥AgilePlusReliefEvidence
出力先フォルダ	C:¥AgilePlusReliefEvidence¥Sample

3.5.2 エビデンスファイルのフォーマット

エビデンスの各ファイルは、カンマ区切りによるCSV形式で保存されます。以降に、各エビデンスファイルのフォーマットについて説明します。

3.5.2.1 ファイルヘッダ部（共通）

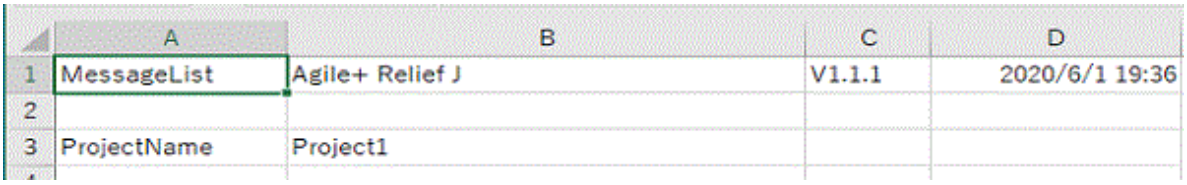
各エビデンスファイルの先頭4行に出力されます。すべてのエビデンスファイルに共通に出力されます。各エビデンスの情報は、5行目以降に出力されます。

以下に、各行およびカラムについて説明します。

行	カラム	出力項目	説明
1	1	エビデンス種別	エビデンスの種別を表す文字列です。 エビデンス種別の文字列については、以降の各エビデンスの説明を参照してください。
1	2	"Agile+ Relief J"	左記の文字列が設定されます。
1	3	製品バージョン	Agile+ Relief Jのバージョンが設定されます。
1	4	出力日時	エビデンスを出力した日時（YYYY/MM/DD hh:mm:ss）が設定されます。
2	空行		
3	1	"ProjectName"	左記の文字列が設定されます。
3	2	プロジェクト名	出力対象のEclipseのプロジェクト名が設定されます。
4	空行		

＜出力例＞

※MessageList.csvをMicrosoft(R) Office Excel(R)を利用して読み込んでいます。



	A	B	C	D
1	MessageList	Agile+ Relief J	V1.1.1	2020/6/1 19:36
2				
3	ProjectName	Project1		

3.5.2.2 MessageList.csv

ファイルヘッダ部に続いて、選択したチェック対象プロジェクト、またはチェック対象ファイルのコードチェックで検出された指摘メッセージの一覧が出力されます。

コードチェックで検出されたすべての指摘メッセージが確認できるため、各指摘メッセージへの対処について検証することができます。

注意

CheckStatus_File.csvの「チェック状況」が「未」のファイルでも、過去のコードチェックの結果が残っている場合、その結果が出力されます。

＜エビデンス種別＞

ファイルヘッダ部の出力項目であるエビデンス種別には、以下が出力されます。

MessageList

＜フォーマット＞

5行目に、指摘の各カラムの意味を表すカラムヘッダが出力されます。6行目以降に、検出指摘が行単位に出力されます。以下に、各カラムについて説明します。

<table>
<tr><th>カラム</th><th>カラムヘッダ名</th><th>説明</th></tr>
<tr><td>1</td><td>確認</td><td>[Agile+ Relief 指摘メッセージ一覧]ビューの[確認済]欄の情報が設定されます。

<table><tr><th>[確認済]欄</th><th>設定値</th></tr><tr><td>チェックなし</td><td>0</td></tr><tr><td>チェックあり</td><td>1</td></tr></table></td></tr>
<tr><td>2</td><td>ファイル</td><td>指摘が検出されたファイルのフルパス名が設定されます。</td></tr>
<tr><td>3</td><td>行番号</td><td>指摘が検出された行番号が設定されます。

動作環境エラーなど、その指摘が特定の行ではなくファイル全般に関わることがあります。そのような場合は、0が設定されます。</td></tr>
<tr><td>4</td><td>重要度</td><td>指摘の重要度が設定されます。</td></tr>
<tr><td>5</td><td>規約コード</td><td>指摘の規約コードが設定されます。</td></tr>
<tr><td>6</td><td>メッセージ</td><td>指摘のメッセージ文字列が設定されます。</td></tr>
<tr><td>7</td><td>メモ</td><td>[Agile+ Relief 指摘メッセージ一覧]ビューの[メモ]欄の情報が設定されます。</td></tr>
<tr><td>8</td><td>ツール</td><td>指摘を検出した以下のいずれかのツール名が設定されます。

・Agile+ Relief

・FindBugs

・PMD

※PGRelief J 2012 autumn以前のコードチェックの結果からエビデンス出力した場合は、このカラムは空欄になります。</td></tr>
<tr><td>9</td><td>カテゴリ</td><td>指摘に対応した以下のいずれかのカテゴリが設定されます。

・保守性／可読性

・信頼性

・効率性

・機能性

・移植性

・ユーザ規約

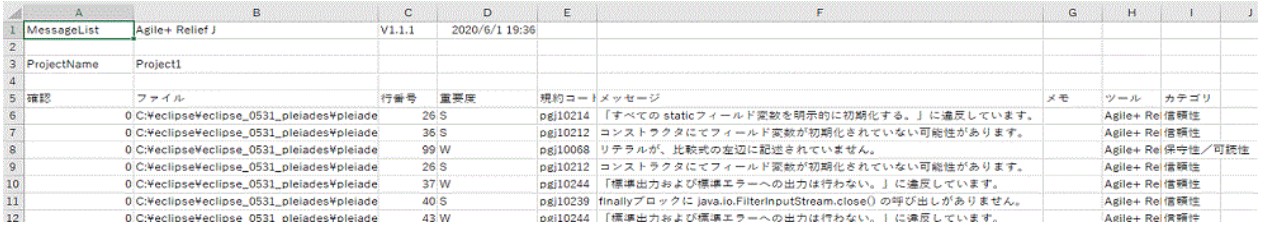
・動作環境エラー

・その他

※PGRelief J 2012 autumn以前のコードチェックの結果からエビデンス出力した場合は、このカラムは空欄になります。</td></tr>
</table>

＜出力例＞

※MessageList.csvをMicrosoft(R) Office Excel(R)を利用して読み込んでいます。



	A	B	C	D	E	F	G	H	I	J
1	MessageList	Agile+ Relief J	V1.1.1	2020/6/1 19:36						
2										
3	ProjectName	Project1								
4										
5	確認	ファイル	行番号	重要度	規約コード	メッセージ	メモ	ツール	カテゴリ	
6	0	C:¥eclipse¥eclipse_0531_pleiades¥pleiade	26	S	pgj10214	「すべての staticフィールド変数を明示的に初期化する。」に違反しています。		Agile+ Re	信頼性	
7	0	C:¥eclipse¥eclipse_0531_pleiades¥pleiade	36	S	pgj10212	コンストラクタにてフィールド変数が初期化されていない可能性があります。		Agile+ Re	信頼性	
8	0	C:¥eclipse¥eclipse_0531_pleiades¥pleiade	99	W	pgj10068	リテラルが、比較式の左辺に記述されていません。		Agile+ Re	保守性／可読性	
9	0	C:¥eclipse¥eclipse_0531_pleiades¥pleiade	26	S	pgj10212	コンストラクタにてフィールド変数が初期化されていない可能性があります。		Agile+ Re	信頼性	
10	0	C:¥eclipse¥eclipse_0531_pleiades¥pleiade	37	W	pgj10244	「標準出力および標準エラーへの出力は行わない。」に違反しています。		Agile+ Re	信頼性	
11	0	C:¥eclipse¥eclipse_0531_pleiades¥pleiade	40	S	pgj10239	finallyブロックに java.io.FilterInputStream.close() の呼び出しがありません。		Agile+ Re	信頼性	
12	0	C:¥eclipse¥eclipse_0531_pleiades¥pleiade	43	W	pgj10244	「標準出力および標準エラーへの出力は行わない。」に違反しています。		Agile+ Re	信頼性	

3.5.2.3 CheckStatus_File.csv

ファイルヘッダ部に続いて、チェック対象ファイル単位にコードチェックの実行状況の一覧が出力されます。

コードチェック実行の有無やチェックオプションなどの実行環境を確認できるため、コードチェックが適切に実施されているか検証することができます。

＜エビデンス種別＞

ファイルヘッダ部の出力項目であるエビデンス種別には、以下が出力されます。

CheckStatus_File

＜フォーマット＞

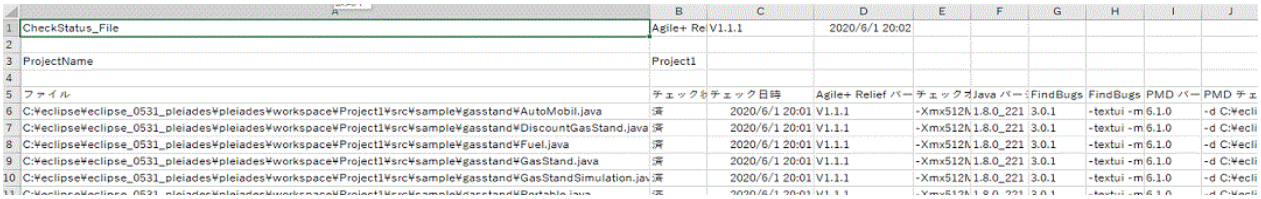
5行目に、チェック状況の各カラムの意味を表すカラムヘッダが出力されます。6行目以降に、チェック対象ファイルのチェック状況が行単位に出力されます。以下に、各カラムについて説明します。

カラム	カラムヘッダ名	説明
1	ファイル	チェック対象ファイルのフルパス名が設定されます。
2	チェック状況	コードチェックの実行有無が設定されます。 設定値 / 意味 未 / 未実行 済 / 実行済み - / コードチェック除外ファイル コードチェックが「未」の場合でも、3カラム目以降に情報が出力される場合があります。これは、以前にコードチェックされた時の結果です。
3	チェック日時	コードチェックが実行された日時（YYYY/MM/DD hh:mm:ss）が設定されます。 ※PGRelief J 2012 autumn以前のコードチェックの結果からエビデンス出力した場合、またはコードチェック除外ファイルの場合は、このカラムは空欄になります。
4	Agile+ Reliefバージョン	コードチェック時のAgile+ Relief Jのバージョンが設定されます。 ※PGRelief J 2012 autumn以前のコードチェックの結果からエビデンス出力した場合、またはコードチェック除外ファイルの場合は、このカラムは空欄になります。
5	チェックオプション	コードチェックで指定されたチェックオプションが設定されます。 ※PGRelief J 2012 autumn以前のコードチェックの結果からエビデンス出力した場合、またはコードチェック除外ファイルの場合は、このカラムは空欄になります。
6	Java バージョン	コードチェック時にAgile+ Relief Jが動作したJavaのバージョンが設定されます。 ※PGRelief J 2012 autumn以前のコードチェックの結果からエビデンス出力した場合、またはコードチェック除外ファイルの場合は、このカラムは空欄になります。
7	FindBugs バージョン	FindBugsとの連携が有効な場合、FindBugsのバージョンが設定されます。 ※PGRelief J 2012 autumn以前のコードチェックの結果からエビデンス出力した場合、またはコードチェック除外ファイルの場合は、このカラムは空欄になります。
8	FindBugs チェックオプション	FindBugsとの連携時に指定された、FindBugsのオプションが設定されます。 ※PGRelief J 2012 autumn以前のコードチェックの結果からエビデンス出力した場合、またはコードチェック除外ファイルの場合は、このカラムは空欄になります。
9	PMD バージョン	PMDとの連携が有効な場合、PMDのバージョンが設定されます。 ※PGRelief J 2012 autumn以前のコードチェックの結果からエビデンス出力した場合、またはコードチェック除外ファイルの場合は、このカラムは空欄になります。

カラム	カラムヘッダ名	説明
10	PMD チェックオプション	PMDとの連携時に指定された、PMDのオプションが設定されます。 ※PGRelief J 2012 autumn以前のコードチェックの結果からエビデンス出力した場合、またはコードチェック除外ファイルの場合は、このカラムは空欄になります。

＜出力例＞

※CheckStatus_File.csvをMicrosoft(R) Office Excel(R)を利用して読み込んでいます。



	A	B	C	D	E	F	G	H	I	J
1	CheckStatus_File	Agile+ Re	V1.1.1	2020/6/1 20:02						
2										
3	ProjectName	Project1								
4										
5	ファイル	チェック状	チェック日時	Agile+ Relief バー	チェックオ	Java バー	FindBugs	FindBugs	PMD バー	PMD チェ
6	C:¥eclipse¥eclipse_0531_pleiades¥pleiades¥workspace¥Project1¥src¥sample¥gasstand¥AutoMobil.java	済	2020/6/1 20:01	V1.1.1	-Xmx512M	1.8.0_221	3.0.1	-textui -m	6.1.0	-d C:¥ecli
7	C:¥eclipse¥eclipse_0531_pleiades¥pleiades¥workspace¥Project1¥src¥sample¥gasstand¥DiscountGasStand.java	済	2020/6/1 20:01	V1.1.1	-Xmx512M	1.8.0_221	3.0.1	-textui -m	6.1.0	-d C:¥ecli
8	C:¥eclipse¥eclipse_0531_pleiades¥pleiades¥workspace¥Project1¥src¥sample¥gasstand¥Fuel.java	済	2020/6/1 20:01	V1.1.1	-Xmx512M	1.8.0_221	3.0.1	-textui -m	6.1.0	-d C:¥ecli
9	C:¥eclipse¥eclipse_0531_pleiades¥pleiades¥workspace¥Project1¥src¥sample¥gasstand¥GasStand.java	済	2020/6/1 20:01	V1.1.1	-Xmx512M	1.8.0_221	3.0.1	-textui -m	6.1.0	-d C:¥ecli
10	C:¥eclipse¥eclipse_0531_pleiades¥pleiades¥workspace¥Project1¥src¥sample¥gasstand¥GasStandSimulation.jav	済	2020/6/1 20:01	V1.1.1	-Xmx512M	1.8.0_221	3.0.1	-textui -m	6.1.0	-d C:¥ecli

3.5.2.4 CheckStatus_Rule.csv

ファイルヘッダ部に続いて、チェック対象ファイル単位に、チェックが実行された規約の一覧と検出指摘数が出力されます。

チェックされた規約とチェックされなかった規約が確認できるため、チェック漏れの規約がないかを検証することができます。

注意

CheckStatus_File.csvの「チェック状況」が「未」のファイルでも、過去のコードチェックの結果が残っている場合、その結果が出力されます。

＜エビデンス種別＞

ファイルヘッダ部の出力項目であるエビデンス種別には、以下が出力されます。

CheckStatus_Rule

＜フォーマット＞

5行目以降に、規約コードを行、チェック対象ファイルを列に配置したマトリクス形式で、指摘件数が出力されます。以下に、各カラムについて説明します。

カラム	カラムヘッダ名	説明
1	ルール	規約コードが設定されます。 FindBugsおよびPMDの指摘は、それぞれのツール単位にまとめて設定されます。
2	概要	規約内容が設定されます。 規約コードが、FindBugsまたはPMDの場合は、設定されません。
3～	ファイルパス名	すべてのチェック対象ファイルのフルパス名が、3カラム目以降に設定されます。

規約コードおよびチェック対象ファイルに対応するカラムには、指摘件数が設定されます。指摘件数の設定値には、以下の2種類があります。

設定値	意味
0以上の数値	検出された指摘数です。
空欄	チェック対象ファイルのコードチェック時に、対象の規約コードに対するチェックが実行されなかった。 ※PGRelief J 2012 autumn以前のコードチェックの結果からエビデンス出力した場合は、検出された指摘数が0件の場合も、空欄になります。

以下の出力イメージで説明します。

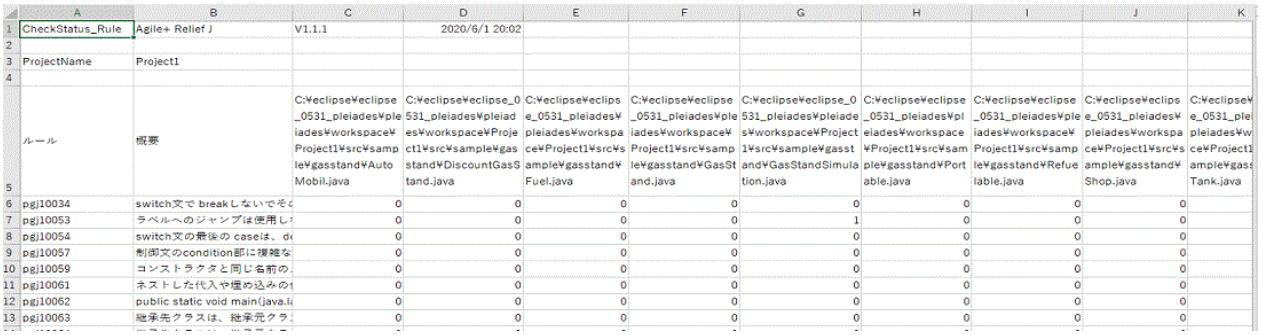
File_1は、pgj10000の指摘が0件、pgj10001の指摘が1件検出されたことを意味しています。一方、File_2は、pgj10000の指摘が1件検出されていますが、pgj10001のチェックが実行されていません。

【出力イメージ】

ルール	概要	File_1	File_2	File_3	・・・	File_n
pgj10000	ファイル名は、「クラス名.java」または「インタフェース名.java」にする。	0	1	0		1
pgj10001	ファイル名は、日本語を使用してはいけない。	1		0		1

＜出力例＞

※CheckStatus_Rule.csvをMicrosoft(R) Office Excel(R)を利用して読み込んでいます。



	A	B	C	D	E	F	G	H	I	J	K
1	CheckStatus_Rule	Agile+ Relief J	V1.1.1	2020/6/1 20:02							
2											
3	ProjectName	Project1									
4											
5	ルール	概要	C:¥eclipse¥eclipse_0531_pleiades¥pleiades¥workspace¥Project1¥src¥sample¥gasstand¥AutoMobil.java	C:¥eclipse¥eclipse_0531_pleiades¥pleiades¥workspace¥Project1¥src¥sample¥gasstand¥DiscountGasStand.java	C:¥eclipse¥eclipse_0531_pleiades¥pleiades¥workspace¥Project1¥src¥sample¥gasstand¥Fuel.java	C:¥eclipse¥eclipse_0531_pleiades¥pleiades¥workspace¥Project1¥src¥sample¥gasstand¥GasStand.java	C:¥eclipse¥eclipse_0531_pleiades¥pleiades¥workspace¥Project1¥src¥sample¥gasstand¥GasStandSimulation.java	C:¥eclipse¥eclipse_0531_pleiades¥pleiades¥workspace¥Project1¥src¥sample¥gasstand¥Portable.java	C:¥eclipse¥eclipse_0531_pleiades¥pleiades¥workspace¥Project1¥src¥sample¥gasstand¥Refuelable.java	C:¥eclipse¥eclipse_0531_pleiades¥pleiades¥workspace¥Project1¥src¥sample¥gasstand¥Shop.java	C:¥eclipse¥eclipse_0531_pleiades¥pleiades¥workspace¥Project1¥src¥sample¥gasstand¥Tank.java
6	pgj10034	switch文で breakしないでそ	0	0	0	0	0	0	0	0	
7	pgj10053	ラベルへのジャンプは使用し	0	0	0	0	1	0	0	0	
8	pgj10054	switch文の最後の caseは、d	0	0	0	0	0	0	0	0	
9	pgj10057	制御文のcondition部に複雑な	0	0	0	0	0	0	0	0	
10	pgj10059	コンストラクタと同じ名前の	0	0	0	0	0	0	0	0	
11	pgj10061	ネストした代入や埋め込みの	0	0	0	0	0	0	0	0	
12	pgj10062	public static void main(java.l	0	0	0	0	0	0	0	0	
13	pgj10063	継承先クラスは、継承元クラ	0	0	0	0	0	0	0	0	

3.5.2.5 Metrics_File.csv

ファイルヘッダ部に続いて、チェック対象ファイル単位に、ファイルの行数やメソッド数（Javaファイルの場合）に関する情報の一覧が出力されます。

行数など規模に関する情報から、生産性や保守性などについて検証することができます。

注意

- CheckStatus_File.csvの「チェック状況」が「未」のファイルでも、過去のコードチェックの結果が残っている場合、その結果が出力されます。
- Metrics_File.csvは、PGRelief J 2012 autumn以前のコードチェックの結果から出力できません。Metrics_File.csvが必要な場合は、PGRelief J 2013以降でコードチェックを実行してから、エビデンス出力を行ってください。
- PMDの入力となるファイル（XMLファイル、JSPファイル、スタイルシートファイル、ECMAScriptファイル、PLSQLファイル、Apache Velocityファイル）のメトリクス情報は出力されません。

＜エビデンス種別＞

ファイルヘッダ部の出力項目であるエビデンス種別には、以下が出力されます。

Metrics_File

＜フォーマット＞

5行目に、メトリクス情報の各カラムの意味を表すカラムヘッダが出力されます。6行目以降に、チェック対象ファイルのメトリクス情報が行単位に出力されます。以下に、各カラムについて説明します。

カラム	カラムヘッダ名	説明
1	ファイル	チェック対象ファイルのフルパス名が設定されます。

カラム	カラムヘッダ名	説明
2	総行数	空行やコメント行を含めた行数が設定されます。
3	実行数	空行やコメント行を除いた行数が設定されます。 チェック対象ファイルがxmlファイルの場合は、総行数と同じ値が設定されます。
4	メソッド数	ファイル内に定義されているメソッド数が設定されます。 チェック対象ファイルがxmlの場合は、0が設定されます。

＜出力例＞

※Metrics_File.csvをMicrosoft(R) Office Excel(R)を利用して読み込んでいます。

	A	B	C	D
1	Metrics_File	Agile+ Relief	V1.1.1	2020/6/1 20:02
2				
3	ProjectName	Project1		
4				
5	ファイル	総行数	実行数	メソッド数
6	C:¥eclipse¥workspace¥Project1¥src¥sample¥gasstand¥AutoMobil.java	60	14	3
7	C:¥eclipse¥workspace¥Project1¥src¥sample¥gasstand¥DiscountGasStand.java	77	18	3
8	C:¥eclipse¥workspace¥Project1¥src¥sample¥gasstand¥Fuel.java	24	4	0
9	C:¥eclipse¥workspace¥Project1¥src¥sample¥gasstand¥GasStand.java	101	31	5
10	C:¥eclipse¥workspace¥Project1¥src¥sample¥gasstand¥GasStandSimulation.java	152	98	4
11	C:¥eclipse¥workspace¥Project1¥src¥sample¥gasstand¥Portable.java	24	4	0
12	C:¥eclipse¥workspace¥Project1¥src¥sample¥gasstand¥Refuelable.java	42	5	0
13	C:¥eclipse¥workspace¥Project1¥src¥sample¥gasstand¥Shop.java	32	6	0
14	C:¥eclipse¥workspace¥Project1¥src¥sample¥gasstand¥Tank.java	63	15	3

3.5.2.6 Metrics_Func.csv

ファイルヘッダ部に続いて、Metrics_File.csvに出力されたファイルの内、Javaファイルについて、その中で定義されているメソッド単位に行数や複雑度などの情報の一覧が出力されます。

保守性や可読性などについて検証することができます。

注意

- CheckStatus_File.csvの「チェック状況」が「未」のファイルでも、過去のコードチェックの結果が残っている場合、その結果が出力されます。
- Metrics_Func.csvは、PGRelief J 2012 autumn以前のコードチェックの結果から出力できません。Metrics_Func.csvが必要な場合は、PGRelief J 2013以降でコードチェックを実行してから、エビデンス出力を行ってください。
- PMDの入力となるファイル（XMLファイル、JSPファイル、スタイルシートファイル、ECMAScriptファイル、PLSQLファイル、Apache Velocityファイル）のメトリクス情報は出力されません。

＜エビデンス種別＞

ファイルヘッダ部の出力項目であるエビデンス種別には、以下が出力されます。

Metrics_Func

＜フォーマット＞

5行目に、メトリクス情報の各カラムの意味を表すカラムヘッダが出力されます。6行目以降に、Javaファイルで定義されているメソッドのメトリクス情報が行単位に出力されます。以下に、各カラムについて説明します。

カラム	カラムヘッダ名	説明
1	ファイル	チェック対象ファイルのフルパス名が設定されます。
2	行番号	メソッドの行番号が設定されます。
3	メソッド名	メトリクス採取対象のメソッド名が設定されます。
4	総行数	空行やコメント行を含めた行数が設定されます。
5	実行数	空行やコメント行を除いた行数が設定されます。
6	ネスト数	if文、while文、do-while文、for文、switch文、try文、synchronized文、およびブロックによる階層の最大値が設定されます。
7	複雑度1	if1、while1、dowhile1、for1、switch1の和に、1を加算した値が設定されます。
8	複雑度2	if2、while2、dowhile2、for2、switch2の和に、1を加算した値が設定されます。
9	return	return文の総数が設定されます。
10	break	break文の総数が設定されます。
11	continue	continue文の総数が設定されます。
12	if1	if文の総数が設定されます。
13	while1	while文の総数が設定されます。
14	dowhile1	do-while文の総数が設定されます。
15	for1	for文の総数が設定されます。
16	switch1	switch文の総数が設定されます。
17	if2	if文の条件式内の&&および||の数を考慮した総数が設定されます。例えば、if(A && B)の場合は、2とカウントします。if1は、1とカウントします。
18	while2	while文の条件式内の&&および||の数を考慮した総数が設定されます。例えば、while(A && B)の場合は、2とカウントします。while1は、1とカウントします。
19	dowhile2	do-while文の条件式内の&&および||の数を考慮した総数が設定されます。例えば、while(A && B)の場合は、2とカウントします。dowhile1は、1とカウントします。
20	for2	for文の条件式内の&&および||の数を考慮した総数が設定されます。例えば、for(; A && B ;)の場合は、2とカウントします。for1は、1とカウントします。
21	switch2	case文およびdefault文の総数が設定されます。

＜出力例＞

※Metrics_Func.csvをMicrosoft(R) Office Excel(R)を利用して読み込んでいます。

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	Metrics_Func	Agile+ Relief J	V1.1.1	2020/6/1 20:02																	
2																					
3	ProjectName	Project1																			
4																					
5	ファイル	行番号	メソッド名	総行数	実行数	ネスト数	複雑度1	複雑度2	return	break	continue	if1	while1	dowhile1	for1	switch1	if2	while2	dowhile2	for2	switch2
6	C:¥eclipse¥workspace¥Project1¥src¥sample¥gasstand¥Auto	35	sample.ga	3	3	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
7	C:¥eclipse¥workspace¥Project1¥src¥sample¥gasstand¥Auto	46	sample.ga	3	3	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
8	C:¥eclipse¥workspace¥Project1¥src¥sample¥gasstand¥Auto	57	sample.ga	3	3	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
9	C:¥eclipse¥workspace¥Project1¥src¥sample¥gasstand¥Disc	30	sample.ga	3	3	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
10	C:¥eclipse¥workspace¥Project1¥src¥sample¥gasstand¥Disc	41	sample.ga	4	4	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
11	C:¥eclipse¥workspace¥Project1¥src¥sample¥gasstand¥Disc	53	sample.ga	24	7	1	2	2	1	0	0	1	0	0	0	0	1	0	0	0	0
12	C:¥eclipse¥workspace¥Project1¥src¥sample¥gasstand¥Gas	87	sample.ga	3	3	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
13	C:¥eclipse¥workspace¥Project1¥src¥sample¥gasstand¥Gas	45	sample.ga	7	7	2	2	2	0	0	0	1	0	0	0	0	1	0	0	0	0

3.6 コードチェック結果を共有する

コードチェック結果(確認状態・確認メモを含む)を開発者間で共有します。

以下の手順により、コードチェック結果を共有します。

1. 共有するコードチェック結果をエクスポートする。
2. 共有するコードチェック結果をインポートする。

他の開発者のコードチェック結果を共有でき、再解析の手間が削減できます。また、開発者間で分担して確認した結果を集約することもできます。

3.6.1 コードチェック結果をエクスポートする

選択したチェック対象プロジェクト、またはチェック対象ファイルのコードチェックによるチェック結果をエクスポートします。ただし、以下のような選択はできません。

- チェック対象プロジェクトを複数選択する
- チェック対象プロジェクトとチェック対象ファイルを一緒に選択する
- 異なるチェック対象プロジェクトのファイルを一緒に選択する

なお、当機能を利用するには、あらかじめ対象となるプロジェクトを開き、かつ[Agile+ Relief J]メニューの[Agile+ Relief Jを使用する]を有効にしてください。

＜操作＞ チェック対象プロジェクトを選択してチェック結果をエクスポートする場合

1. パッケージ・エクスプローラ、プロジェクト・エクスプローラ、またはナビゲータのビューでチェック対象プロジェクトを選択します。
2. 次のいずれかの操作を行います。
   - [Agile+ Relief J(J)]メニュー＞[チェック結果エクスポート(X)]を選択します。
   - ポップアップメニューから[Agile+ Relief J(J)]＞[チェック結果エクスポート(X)]を選択します。
3. [チェック結果エクスポート]ダイアログでエクスポートするチェック結果ファイル名(*.pgrjres)を指定して、[保存(S)]ボタンをクリックします。

＜操作＞ チェック対象ファイルを選択してチェック結果をエクスポートする場合

1. パッケージ・エクスプローラ、プロジェクト・エクスプローラ、ナビゲータ、またはAgile+ Relief 判定結果のビューでチェック対象ファイルを単一または複数選択します。
2. 次のいずれかの操作を行います。
   - [Agile+ Relief J(J)]メニュー＞[チェック結果エクスポート(X)]を選択します。
   - パッケージ・エクスプローラ、プロジェクト・エクスプローラ、またはナビゲータの場合、ポップアップメニューから[Agile+ Relief J(J)]＞[チェック結果エクスポート(X)]を選択します。
   - Agile+ Relief 判定結果の場合、ポップアップメニューから [チェック結果エクスポート(X)]を選択します。
3. [チェック結果エクスポート]ダイアログでエクスポートするチェック結果ファイル名(*.pgrjres)を指定して、[保存(S)]ボタンをクリックします。

3.6.2 コードチェック結果をインポートする

選択したチェック対象プロジェクト、またはチェック対象ファイルのコードチェックによるチェック結果をインポートします。ただし、以下のような選択はできません。

- チェック対象プロジェクトを複数選択する
- チェック対象プロジェクトとチェック対象ファイルを一緒に選択する
- 異なるチェック対象プロジェクトのファイルを一緒に選択する

なお、当機能を利用するには、あらかじめ対象となるプロジェクトを開き、かつ[Agile+ Relief J]メニューの[Agile+ Relief Jを使用する]を有効にしてください。

＜操作＞ チェック対象プロジェクトを選択してチェック結果をインポートする場合

1. パッケージ・エクスプローラ、プロジェクト・エクスプローラ、またはナビゲータのビューでチェック対象プロジェクトを選択します。
2. 次のいずれかの操作を行います。
   - [Agile+ Relief J(J)]メニュー＞[チェック結果インポート(I)]を選択します。
   - ポップアップメニューから[Agile+ Relief J(J)]＞[チェック結果インポート(I)]を選択します。
3. [チェック結果インポート]ダイアログでインポートするチェック結果ファイル名(*.pgrjres)を指定して、[開く(O)]ボタンをクリックします。

＜操作＞ チェック対象ファイルを選択してチェック結果をインポートする場合

1. パッケージ・エクスプローラ、プロジェクト・エクスプローラ、ナビゲータ、またはAgile+ Relief 判定結果のビューでチェック対象ファイルを単一または複数選択します。
2. 次のいずれかの操作を行います。
   - [Agile+ Relief J(J)]メニュー＞[チェック結果インポート(I)]を選択します。
   - パッケージ・エクスプローラ、プロジェクト・エクスプローラ、またはナビゲータの場合、ポップアップメニューから[Agile+ Relief J(J)]＞[チェック結果インポート(I)]を選択します。
   - Agile+ Relief 判定結果の場合、ポップアップメニューから [チェック結果インポート(I)]を選択します。
3. [チェック結果インポート]ダイアログでインポートするチェック結果ファイル名(*.pgrjres)を指定して、[開く(O)]ボタンをクリックします。

注意

- 選択したチェック対象ファイルにコードチェック結果が存在する場合、インポート方式についての確認ダイアログが表示されます。以下のいずれかのインポート方式を選択してください。

インポート方式	動作
すべて置き換え	コードチェック結果（確認状態・確認メモを含む）の置き換えを行う。
確認状態・確認メモのみ置き換え	インポート先の指摘メッセージに対して確認状態・確認メモのみ置き換えを行う。

- ソース修正やソース削除などによりインポートが行われなかった指摘メッセージについては、csv形式のファイルに出力されます。ログファイルは、インポートしたチェック結果ファイルと同じフォルダに出力されます。

＜ファイル名＞

*プロジェクト名*_import_*yyyyMMddhhmmss*.log

＜フォーマット＞

1行目に、各カラムの意味を表すカラムヘッダが出力されます。2行目以降に、インポートが行われなかった指摘メッセージが行単位に出力されます。以下に、各カラムについて説明します。

カラム	カラムヘッダ名	説明
1	確認済	[Agile+ Relief 指摘メッセージ一覧]ビューの[確認済]欄の情報が出力されます。 [確認済]欄：設定値 チェックなし：0 チェックあり：1
2	規約コード	指摘の規約コードが出力されます。
3	メッセージ	指摘のメッセージ文字列が出力されます。
4	メモ	[Agile+ Relief 指摘メッセージ一覧]ビューの[メモ]欄の情報が出力されます。

カラム	カラムヘッダ名	説明
5	ファイル名	指摘が検出されたファイル名が出力されます。
6	パス名	指摘が検出されたファイルのパス名が出力されます。
7	行番号	指摘が検出された行番号が出力されます。

3.7 ヘルプ表示

3.7.1 目次を表示する

ヘルプの目次を表示します。

＜操作＞

1. [ヘルプ]メニュー＞[ヘルプ目次(H)]でヘルプを選択します。
2. 目次から[Agile+ Relief J コードチェック操作手引書]を選択します。

3.7.2 関連トピックを表示する

以下のビューに関連するトピックを表示します。

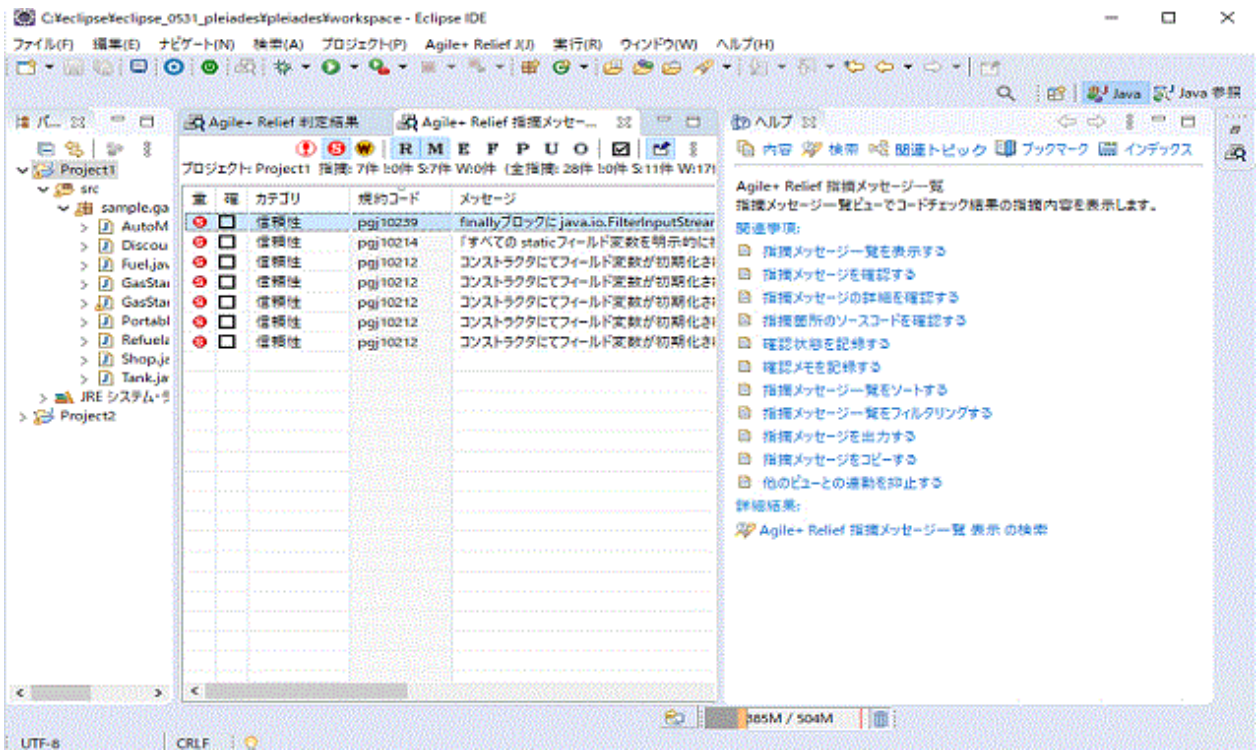
- 指摘メッセージ一覧
- 指摘メッセージ詳細
- 判定結果
- 設定－[Agile+ Relief J]
- プロジェクトのプロパティー－[Agile+ Relief J]

＜操作＞

1. ビューにフォーカスのある状態で[F1]キーを押下します。

   関連トピックが表示されます。

例）指摘メッセージ一覧で[F1]キーを押した場合



第4章 コードチェックの使い方

ここでは、Agile+ Relief Jの一連の操作を、サンプルを使って順に説明します。

" Agile+ Relief インストールフォルダ¥PGReliefJava¥sample¥JavaProject" フォルダに格納されているサンプルを利用します。
サンプルのJavaソースには、あらかじめ規約違反となるようにコードを記述してあります。

コードチェックの実行から結果の確認、出力までを以下のサンプルを例に説明します。

サンプル

- 格納場所:C:¥Program Files¥AgilePlus¥PGReliefJava¥sample¥JavaProject
- プロジェクト:Project2

4.1 サンプルプロジェクトを準備する

Eclipse上でサンプルのプロジェクトを作成します。

1. Eclipseの[ファイル]メニューから[インポート]を選択して、[インポート]ー[選択]画面を表示します。
2. [インポートソースの選択]から[一般]ー[既存プロジェクトをワークスペースへ]を選択し、[次へ]を押して、[インポート]ー[プロジェクトのインポート]画面を表示します。
3. [ルートフォルダの選択]でフォルダに
   " C:¥Program Files¥AgilePlus¥PGReliefJava¥sample¥JavaProject"を指定します。
4. プロジェクトからProject2を選択して、[完了]ボタンをクリックします。

   サンプルをワークスペースにコピーする場合は、[プロジェクトをワークスペースにコピー]をチェックします。

作成後はプロジェクトをビルドします。

4.2 Agile+ Relief Jを有効にする

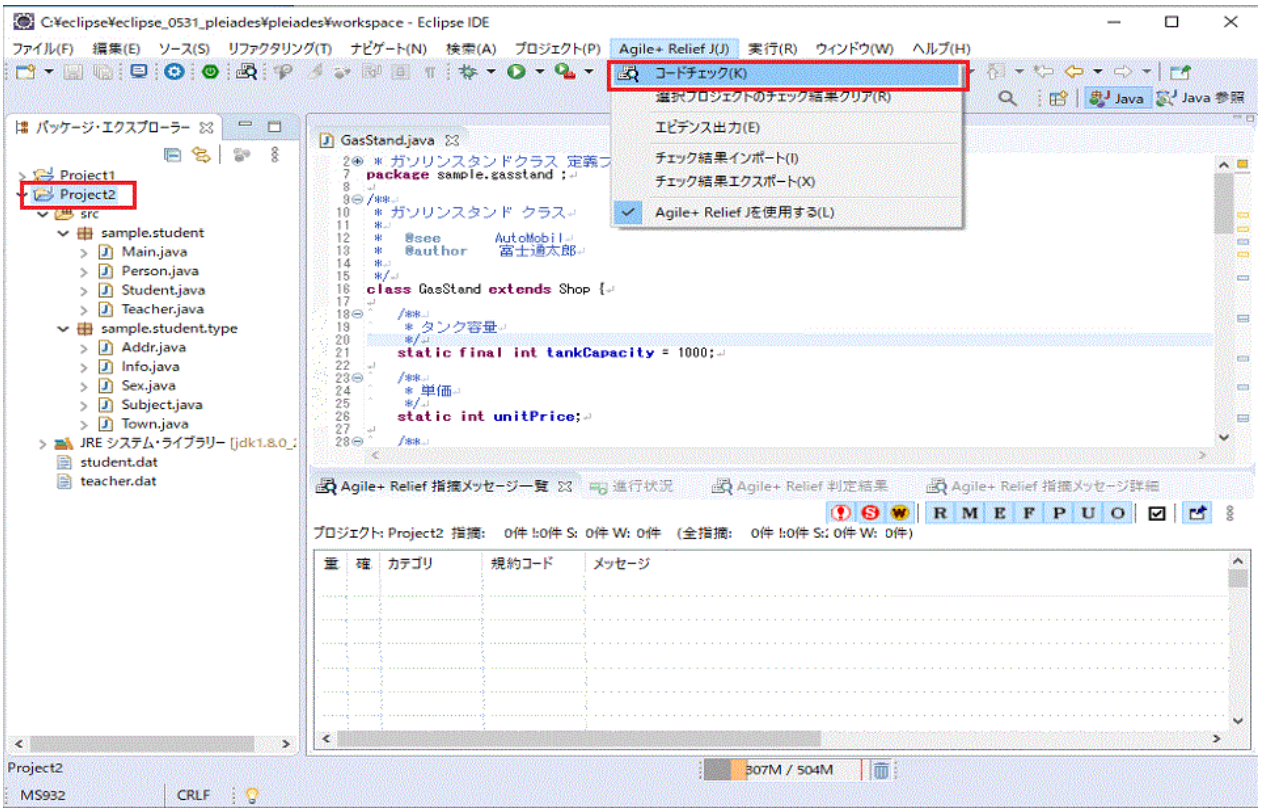
[Agile+ Relief J]メニューから[Agile+ Relief Jを使用する]を有効にして、Agile+ Relief Jを使用できる状態にします。

Agile+ Relief Jを有効にすると、[Agile+ Relief 指摘メッセージ一覧]ビュー、[Agile+ Relief 指摘メッセージ詳細]ビュー、[Agile+ Relief 判定結果]ビューが表示されます。

4.3 コードチェックを実行する

サンプルプロジェクトに対してコードチェックを実行します。

パッケージ・エクスプローラ上でProject2を選択し、[Agile+ Relief J]メニューの[コードチェック]を選択すると、コードチェックが開始されます。

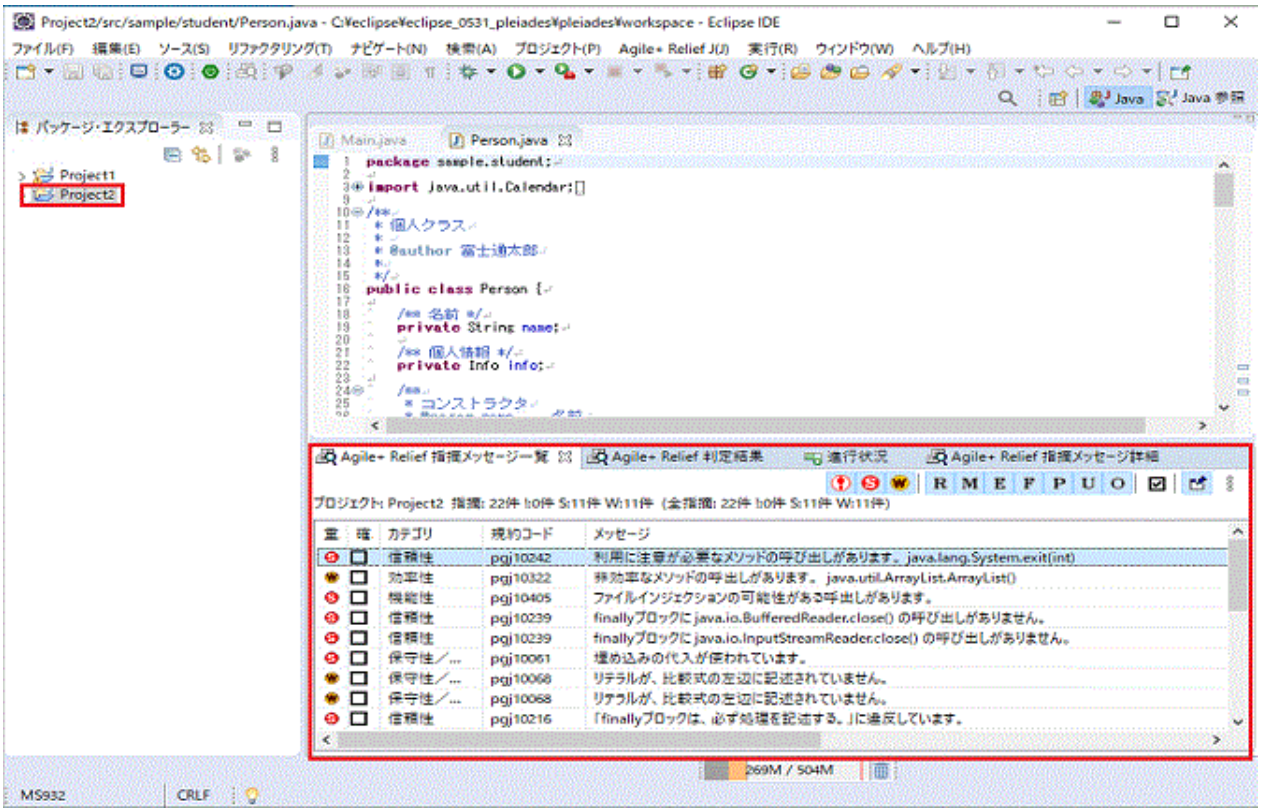


コードチェックが終了すると、[Agile+ Relief 指摘メッセージ一覧]ビュー、および[Agile+ Relief 判定結果]ビューにコードチェックの結果が表示されます。

4.4 指摘メッセージを確認する

コードチェックの結果として、指摘メッセージを確認します。

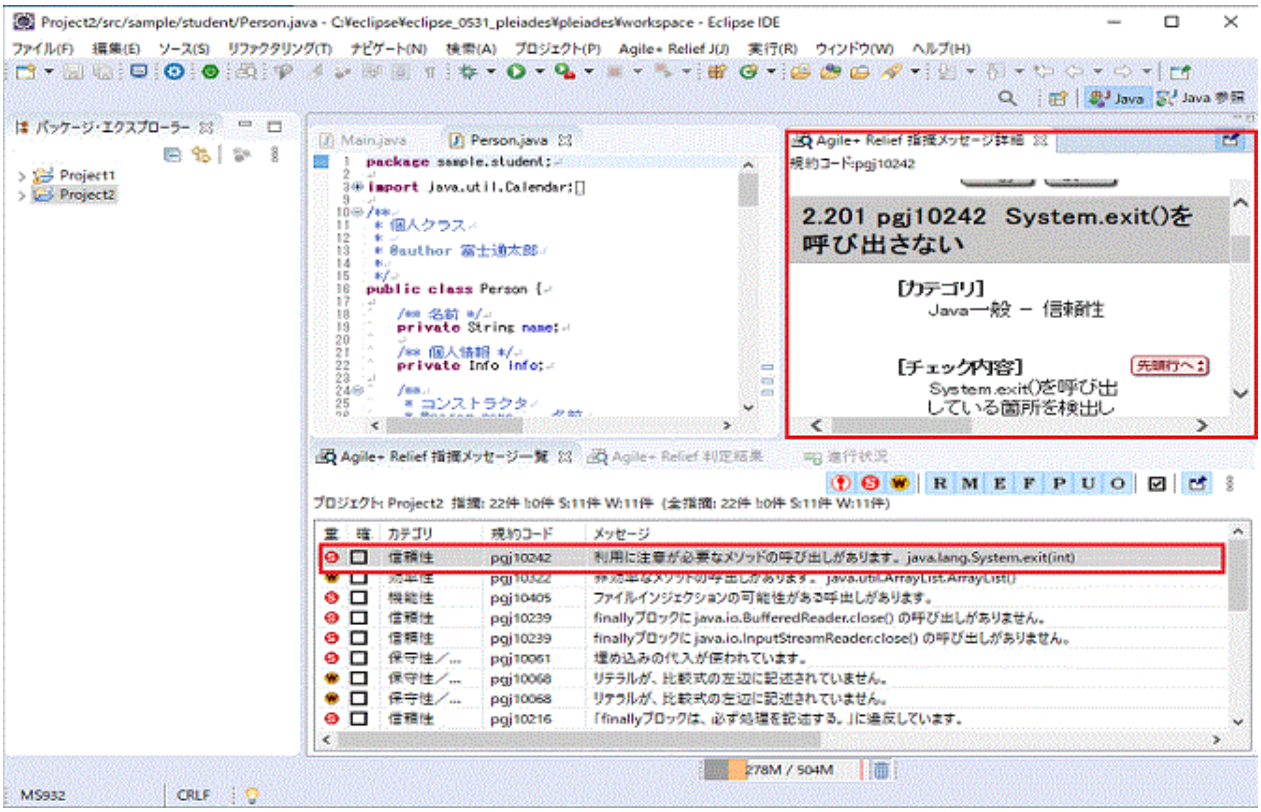
パッケージ・エクスプローラでProject2を選択すると、[Agile+ Relief 指摘メッセージ一覧]ビューにProject2のJavaソースに対する指摘メッセージが表示されます。



4.5 指摘メッセージの詳細を確認する

指摘メッセージの詳細（規約、カテゴリ、チェック内容、意味、対処方法）を確認します。

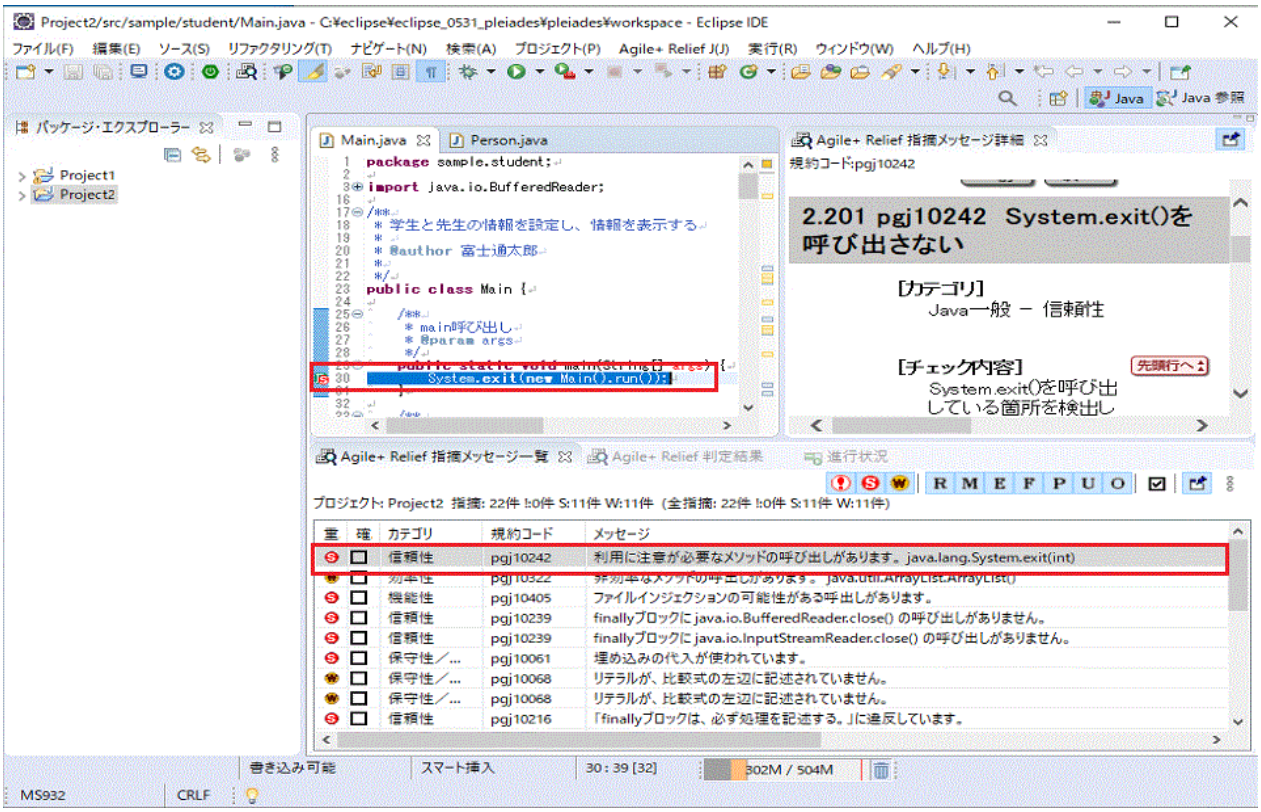
[Agile+ Relief 指摘メッセージ一覧]ビューでpgj10242の指摘メッセージを選択すると、[Agile+ Relief 指摘メッセージ詳細]ビューにpgj10242の詳細が表示されます。



4.6 ソースコードを確認する

エディタを開いてソースコードから指摘箇所を確認します。

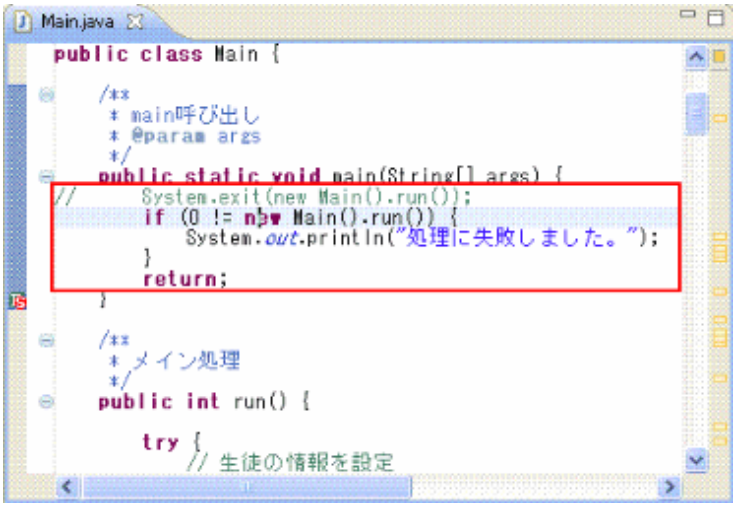
[Agile+ Relief 指摘メッセージ一覧]ビューでpgj10242の指摘メッセージ行をダブルクリックすると、Main.javaの指摘行に位置づけてエディタ上に表示されます。また、エディタ上では、指摘箇所にマーカーが付けられます。



4.7 ソースコードを修正する

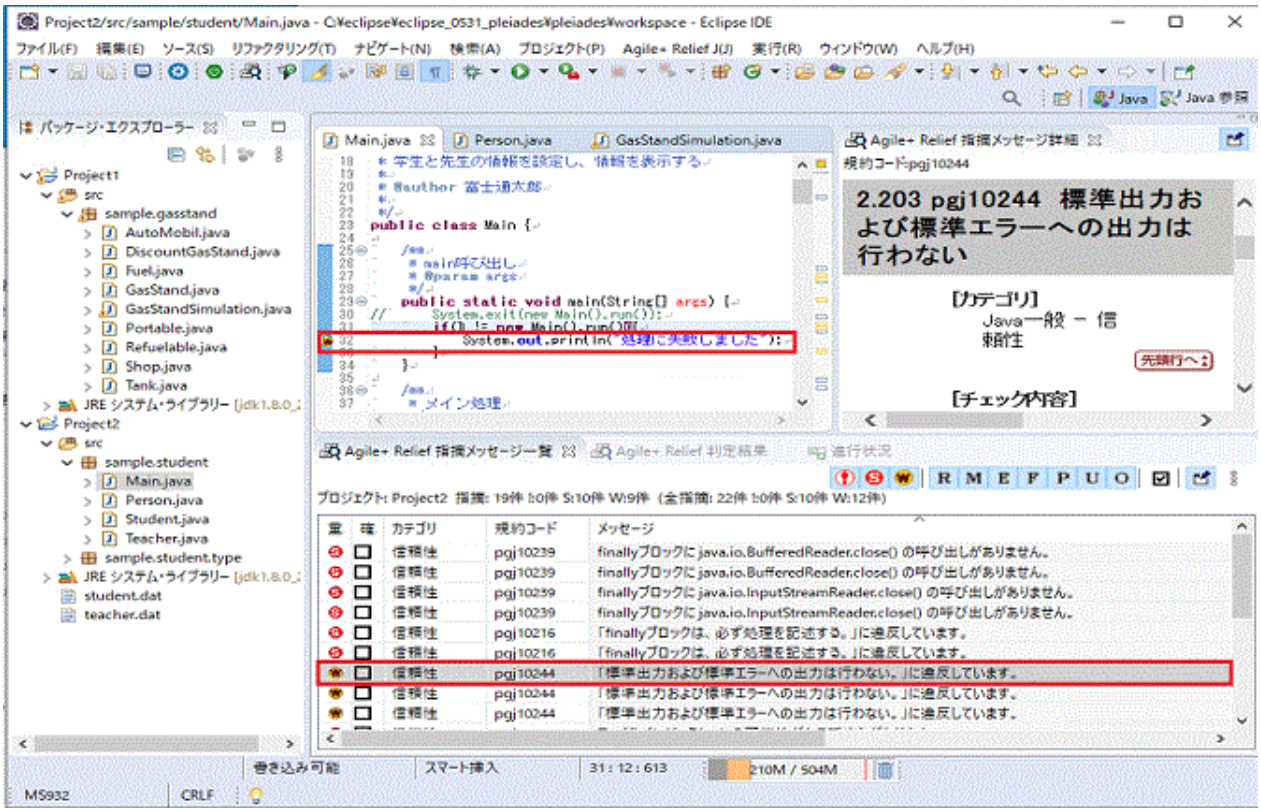
指摘メッセージ、および指摘詳細を参考にソースコードの指摘箇所を修正します。

pgj10242が「System.exit()を呼び出さない」の規約であることから、System.exit()メソッドの呼び出しをMainクラスのrun()メソッドの呼び出しに変更し、その復帰値が0でなかった場合にメッセージを出力して復帰するように修正しました。



ソースを保存して、再度コードチェックを実行します。

指摘内容が変わったことが確認できます。



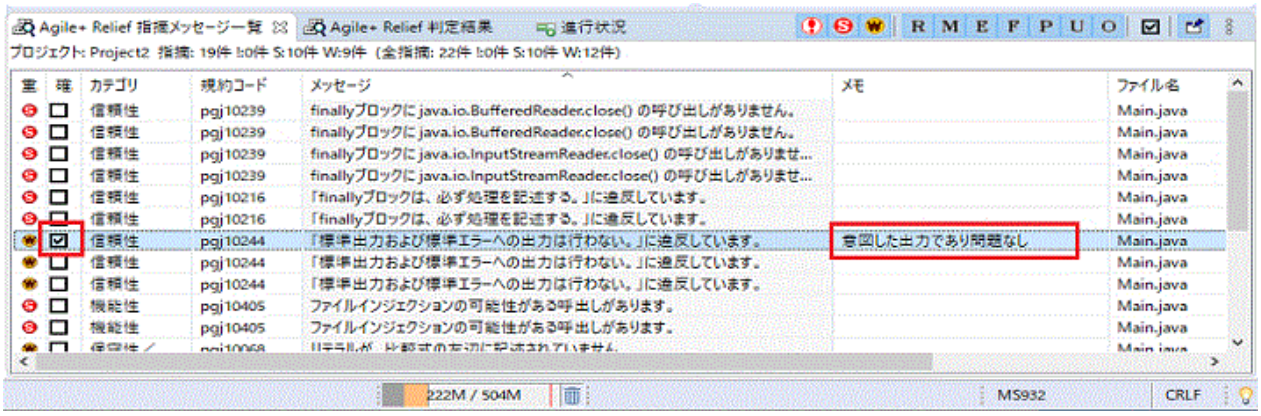
4.8 確認およびメモを記録する

ここでは、新しい指摘に対して対処します。

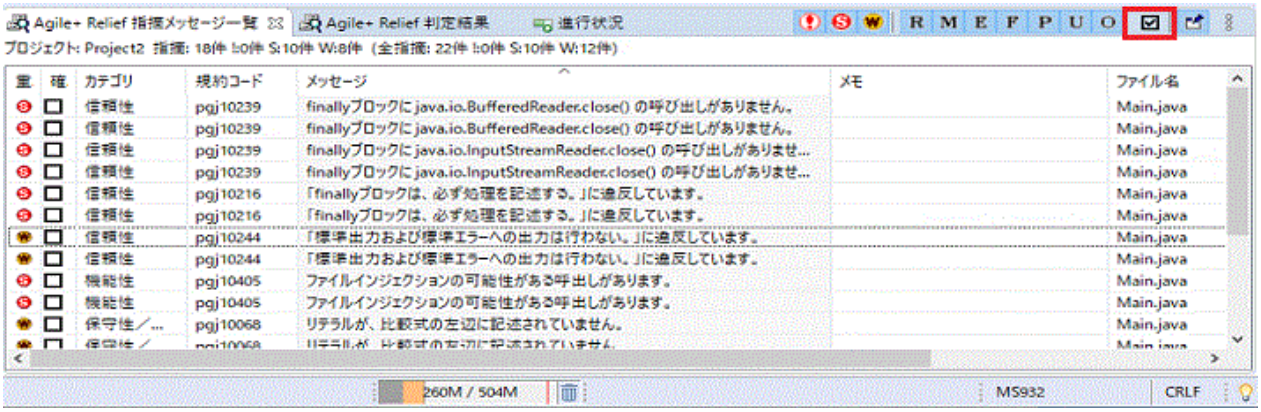
この指摘は、標準出力を行わない指摘ですが、当プログラムは標準出力を意図したものです。

そのため、当指摘は修正の対処は不要です。

そこで、[確認済]チェックボックスをチェックし、[メモ]欄にその理由を記録します。



以下のようにツールバーの ☑ (確認済を表示) を無効にすると、[確認済]チェックボックスをチェックした指摘については表示されなくなります。このような操作を行うことで、重要な指摘のみを表示することができ、効率的に確認することができます。



4.9 判定結果を確認する

[Agile+ Relief 判定結果]ビューでは、指摘への対応状況を確認することができます。

[判定]欄、[チェック]欄を確認します。

[判定]欄が NG のソースに対しては、[Agile+ Relief 指摘メッセージ一覧]ビューで指摘を確認し、修正を行ってください。

[チェック]欄が 未 のソースに対しては、コードチェックを行って、判定結果を確認してください。

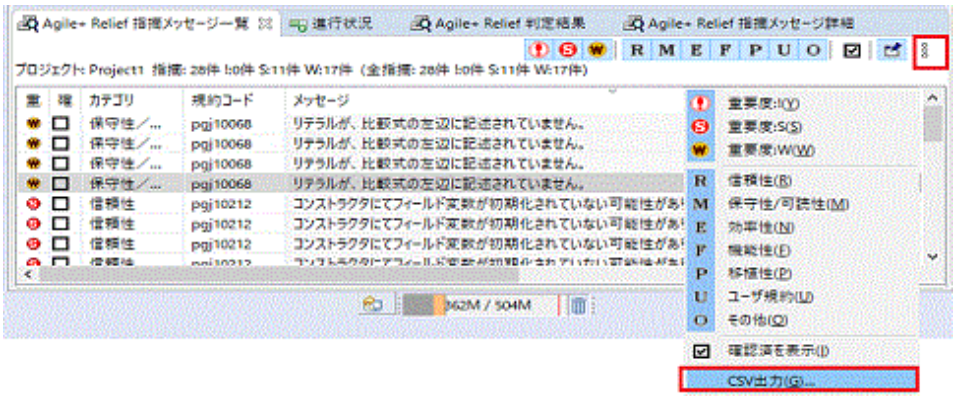
4.10 指摘結果を出力する

すべての指摘に対して修正が完了したら、指摘結果をファイルに出力し、レビュー時の補助資料として使用します。

すべての結果をレビューするために、ビューのフィルタ設定ですべての結果を表示した状態にしてから指摘結果を出力してください。

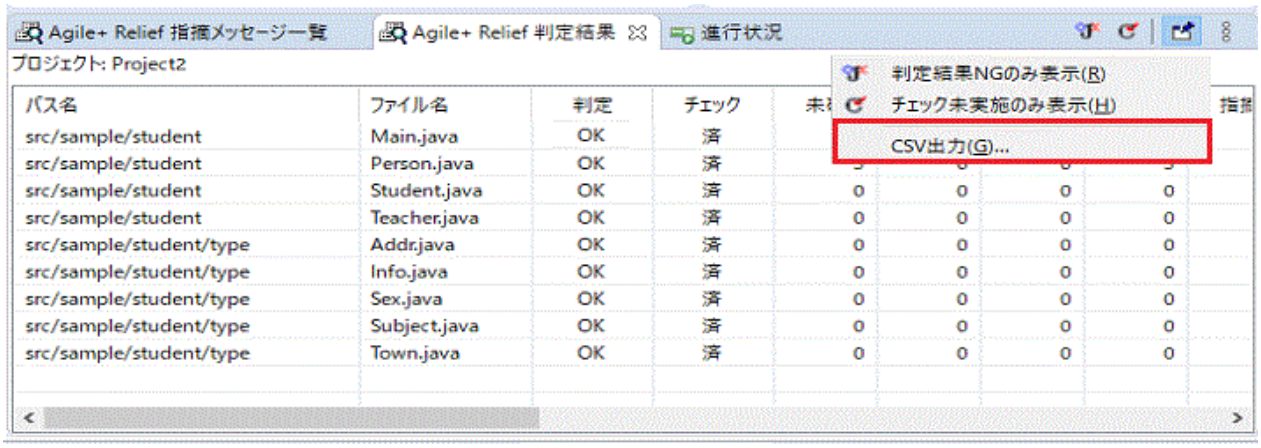
指摘メッセージ一覧の出力

指摘メッセージ一覧のビューメニューから[CSV出力]を選択し、保存ダイアログで出力先を指定します。
すべての指摘を表示するには、すべてのフィルタ項目をオンにしてください。



判定結果の出力

判定結果のビューメニューから[CSV出力]を選択し、保存ダイアログで出力先を指定します。
すべての判定結果を出力するには、すべてのフィルタ項目をオフにしてください。

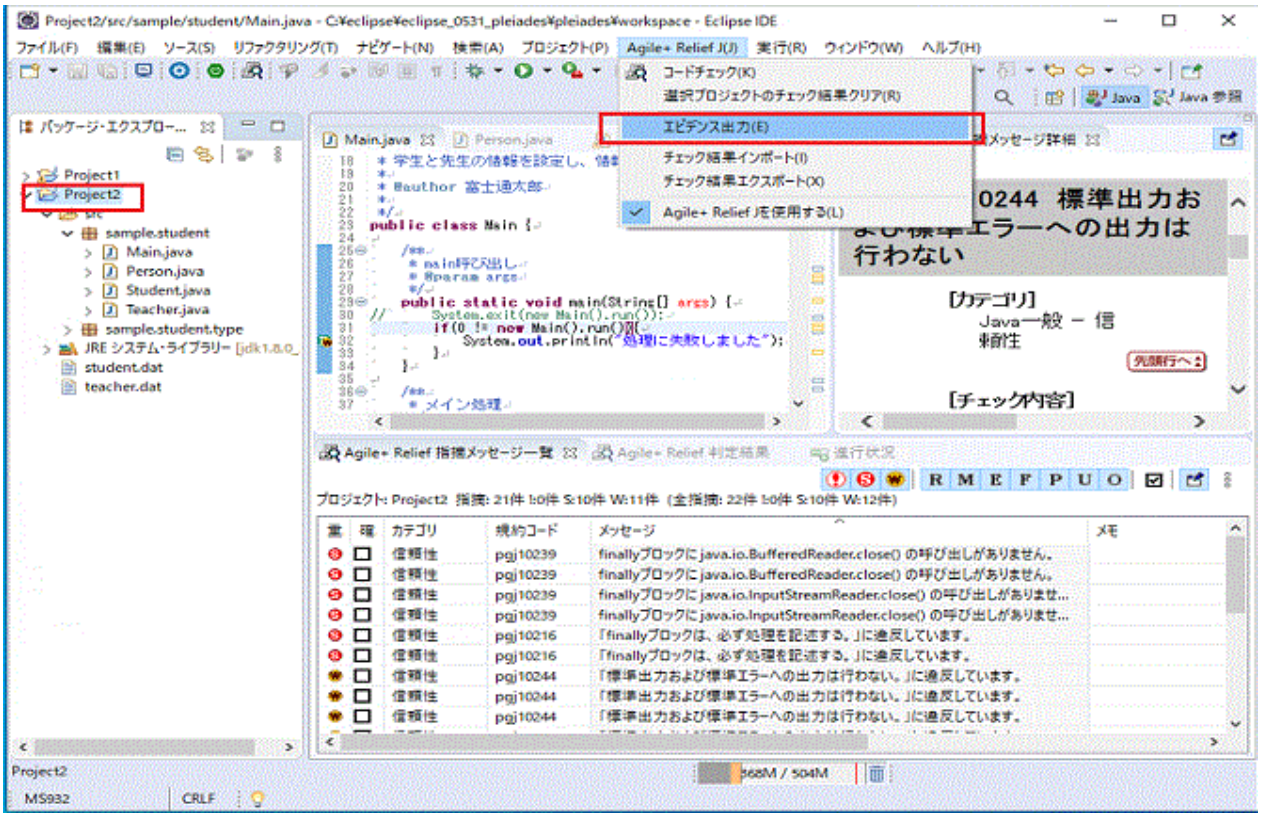


4.11 エビデンスを出力する

コードチェックにより検出された指摘への対処が完了したら、エビデンスを出力します。

エビデンスは、プログラムレビューなどで、コードチェックが適正に実施されていることの確認に使用します。

パッケージ・エクスプローラ上でProject2を選択し、[Agile+ Relief J]メニューの[エビデンス出力]を選択すると、エビデンス出力が開始されます。


Project2/src/sample/student/Main.java - C:¥eclipse¥eclipse_0531_pleiades¥pleiades¥workspace - Eclipse IDE
ファイル(F) 編集(E) ソース(S) リファクタリング(T) ナビゲート(N) 検索(A) プロジェクト(P) Agile+ Relief J(J) 実行(R) ウィンドウ(W) ヘルプ(H)
コードチェック(K)
選択プロジェクトのチェック結果クリア(R)
エビデンス出力(E)
チェック結果インポート(I)
チェック結果エクスポート(X)
Agile+ Relief Jを使用する(L)
パッケージ・エクスプロー...
Project1
Project2
src
sample.student
Main.java
Person.java
Student.java
Teacher.java
sample.student.type
JRE システム・ライブラリー [jdk1.8.0_
student.dat
teacher.dat
Main.java
Person.java
public class Main {
public static void main(String[] args) {
System.out.println("処理に失敗しました");
メッセージ詳細
0244 標準出力および標準エラーへの出力は行わない
[カテゴリ]
Java一般 － 信頼性
先頭行へ
[チェック内容]
Agile+ Relief 指摘メッセージ一覧
Agile+ Relief 判定結果
進行状況
プロジェクト: Project2 指摘: 21件 E:0件 S:10件 W:11件 (全指摘: 22件 E:0件 S:10件 W:12件)
重 確 カテゴリ 規約コード メッセージ メモ
信頼性 pgj10239 finallyブロックに java.io.BufferedReader.close() の呼び出しがありません。
信頼性 pgj10239 finallyブロックに java.io.BufferedReader.close() の呼び出しがありません。
信頼性 pgj10239 finallyブロックに java.io.InputStreamReader.close() の呼び出しがありませ...
信頼性 pgj10239 finallyブロックに java.io.InputStreamReader.close() の呼び出しがありませ...
信頼性 pgj10216 「finallyブロックは、必ず処理を記述する。」に違反しています。
信頼性 pgj10216 「finallyブロックは、必ず処理を記述する。」に違反しています。
信頼性 pgj10244 「標準出力および標準エラーへの出力は行わない。」に違反しています。
信頼性 pgj10244 「標準出力および標準エラーへの出力は行わない。」に違反しています。
信頼性 pgj10244 「標準出力および標準エラーへの出力は行わない。」に違反しています。
Project2
MS932
CRLF
568M / 504M

第5章 セキュリティ監査ドキュメントについて【Kiyacker互換】

ここでは、セキュリティ監査ドキュメントの生成方法、利用方法について説明します。

セキュリティ監査ドキュメント(以降、監査ドキュメントと呼びます)は、セキュリティのレビュー支援を目的としたもので、監査が必要な箇所を規約単位に報告書としてまとめます。

監査ドキュメントの生成では、以下のファイルを利用します。

- 監査ドキュメントテンプレート

  監査ドキュメント生成用のExcelテンプレートファイルです。
  "Agile+ Reliefインストールフォルダ¥PGReliefJava" フォルダに、report-generator.xltの名前でファイルが格納されています。

- Kiyackerログ出力ファイル

  コードチェックの実行結果ログが出力されるファイルです。
  "Eclipseプロジェクトフォルダ¥.pgrjava" フォルダに、project_***yyyymmdd***_coderev.log(*yyyymmdd*はコードチェックを実行した日付)の名前でファイルが出力されます。

5.1 監査ドキュメントの生成方法【Kiyacker互換】

監査ドキュメントの生成は、監査ドキュメントテンプレートをもとに監査ドキュメントを起動し、Kiyackerログ出力ファイルからセキュリティ関連の情報を自分自身のシートに取り込みます。

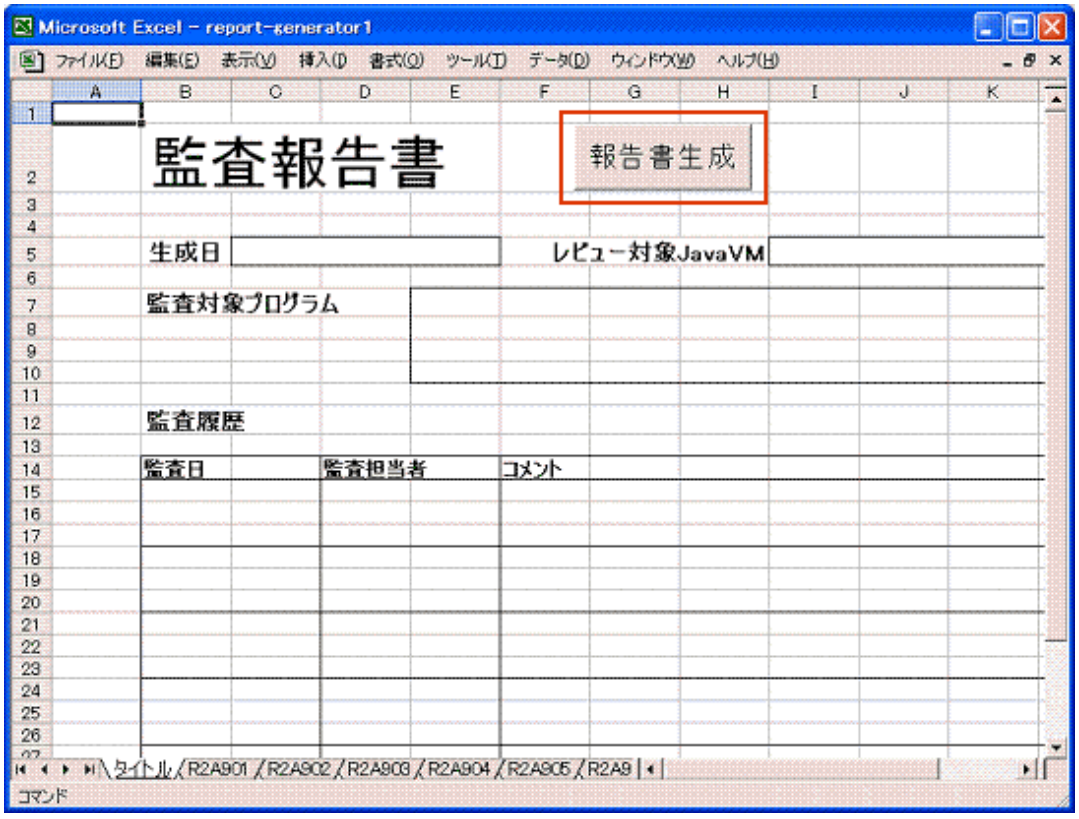
監査ドキュメントのレイアウトを変更したい場合は、テンプレートファイルを修正してご利用ください。

＜操作＞

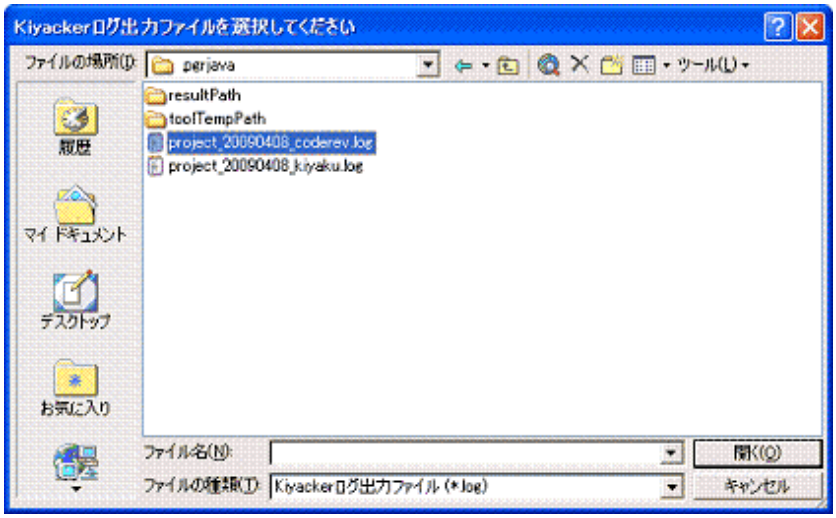
1. 監査ドキュメントテンプレート(report-generator.xlt)を起動します。

   report-generator1の名前で監査報告書が表示されます。

2. [タイトル] シートの[報告書生成] ボタンをクリックします。



3. 表示されたファイル選択ダイアログで、Kiyackerログ出力ファイルを選択してください。



選択と同時にログファイルが解析され、ドキュメントへの変換が始まります。しばらくお待ちください。

再度ドキュメントを生成したい場合には、監査ドキュメントの起動からやり直してください。

出力例）

5.2 監査ドキュメントの利用方法【Kiyacker互換】

監査ドキュメントには、セキュリティに関する規約単位に報告書が生成されます。

開発者は、監査方法にしたがって監査ドキュメントをチェックしてください。

監査方法は、「Agile+ Relief J規約詳細説明書」に記載される対象規約コードの監査方法を参照してください。

- 監査結果はExcelからドキュメントに入力できますが、いったん監査ドキュメントをExcelで印刷してから書き込んでもかまいません。
- すべての監査が終了しましたら、監査ドキュメントを印刷してください。
  印刷された監査ドキュメントはチェックを行なったプログラムの安全性の品質をあらわすものになります。
  この監査ドキュメントは、プログラムと一緒にテスト仕様書に含めて納品する、品質検証の証拠として提出する、などといった使い方が可能です。

第6章 動作環境エラー

Agile+ Relief Jで通知される動作環境エラーとその意味、対策について以下に示します。

動作環境エラーが発生した場合、Agile+ Relief Jは、要求された操作に対する処理を中断します。以下を参考に、動作環境エラーを解決してから、再度、同様の操作を実行してください。

表6.1 動作環境エラー一覧

コード	メッセージ文字列	意味	対策
ERR001	チェック実行中にエラーが発生しました。	Agile+ Relief Jに異常が発生しました。	弊社の問い合わせ窓口に連絡してください。
ERR002	メモリ不足が発生しました。	Agile+ Relief Jの実行中にメモリが不足しました。	・コードチェック時の最大ヒープサイズを大きくして、再度実行してください。 ・コードチェック対象のソース数を減らして、再度実行してください。
ERR003	Agile+ Relief Jのインストール環境に誤りがあります。	Agile+ Relief Jのインストール環境が壊れている可能性があります。	Agile+ Relief Jの再インストールを実施してください。
ERR004	JDK環境設定（JAVA_HOME）に誤りがあります。	・JDKがインストールされていません ・環境変数JAVA_HOMEが設定されていないまたは誤りがあります	・JDKをインストールしてください ・環境変数JAVA_HOMEを設定してください
ERR100	ファイルの入力に失敗しました。	・Javaソースの読み込み権限がありません ・Javaソースのエンコードと指定されたエンコードが違います	・ファイルの存在、アクセス権、ネットワークなどを確認し、ファイルが読み込める状態にしてください ・ファイルのエンコードをサポート対象のエンコードにしてください
ERR101	ファイルの解析に失敗しました。	Javaソースに文法エラーがあります。	文法エラーを修正して、再度コードチェックを実行してください。
ERR102	解析に必要なクラスライブラリが見つかりません。	クラスライブラリの指定が不足しています。	必要なクラスライブラリをすべて指定してください。
ERR103	チェック対象のファイルが見つかりません。	・指定されたファイルが存在しません ・指定されたプロジェクト配下にチェック対象のファイルが存在しません ・指定されたプロジェクトフォルダが存在しません	ファイルやフォルダの存在を確認してください。
ERR104	classファイルが見つかりません。	classファイルが1つも見つかりません。	チェック対象のJavaソースをビルドし、classファイルを作成してください。
ERR105	作業ファイルの入出力に失敗しました。	・読み込み／書き込み権限がありません ・ディスク容量が不足しています ・ネットワークが切断されています	対象のフォルダのアクセス権やディスク容量を確認し、読み込み／書き込みができる状態にしてください。
ERR106	classファイルが最新でない可能性があります。チェック対象のJavaソースをビルドし、classファイルを作成してください。	classファイルが最新でない可能性があります。	チェック対象のJavaソースをビルドし、classファイルを作成してください。
ERR203	チェック結果の出力に失敗しました。	・読み込み／書き込み権限がありません ・ディスク容量が不足しています	・対象のフォルダのアクセス権やディスク容量を確認し、読み込み／書き込みができる状態にしてください。

		・ネットワークが切断されています ・Javaの識別子に日本語が使用されています。(PMD連携使用時)	・Javaの識別子に日本語が使用されている場合は、PMD連携を使用できません。詳細は、「B.4 トラブルシューティング」を参照してください。
ERR204	エンコードの指定に誤りがあります。	サポート対象外のエンコードが指定されました。	サポート対象のエンコードを指定してください。
ERR205	検査規約定義ファイルに誤りがあります。	・code=001 存在しないパスが指定されました。 ・code=002 上位バージョンの検査規約定義ファイル、または検査規約定義ファイル以外のファイルが指定されました。	正しい検査規約定義ファイルを指定してください。
ERR500	連携ツールの実行中にエラーが発生しました。 エラー詳細はコンソールで確認してください。	FindBugsまたはPMDの実行でエラーが発生しました。	コンソールを確認し、連携ツールの出力したエラーメッセージに従って対処してください。
ERR501	連携ツールの起動に失敗しました。	プラグインファイル(plugins.xml)の定義に誤りがあります。	プラグインファイルの定義を確認してください。
ERR502	プラグインファイルに誤りがあります。	プラグインファイル(plugins.xml)の形式に誤りがあります。	プラグインファイルの形式を確認してください。
ERR900	ライセンス管理システムでエラーが検出されました。 インストールガイドの「ライセンス管理システムのメッセージと対処方法」にしたがって対処してください。		

第7章 注意事項

7.1 コードチェック

■環境変数について

環境変数JAVA_HOMEの登録が必要です。利用されるJDKのパスをJAVA_HOMEに登録してください。

■チェック対象Javaファイルについて

- JDK 1.4／5.0／6／7／8のJava言語仕様に準拠したJavaプログラムが対象です。
- 識別子として assert または enum を使用しているJavaプログラムは、解析できません。
- Javaファイルは 5,000 行までが対象となります。
- 文字コードがMS932、EUC-JP、またはUTF-8以外の場合はチェックが正しく実施されません。
- JIS2004で追加された環境依存文字(unicode)はサポートしていません。
- フルパス名が259文字を超えるファイルについては正常に読み込めません。
- ファイル名が255文字を超えるファイルについては正常に読み込めません。
- 匿名クラス、ローカルクラスのコードチェックは行えません。
  そのため、匿名クラス内、ローカルクラス内で変数やメソッドを定義している場合に違反として検出される場合があります。
- 改行コードにCRLF以外が使用されている場合、コードチェックが正しく行われません。
- チェック実行対象には、エラーのない状態のJavaファイルを指定してください。

■classファイルについて

- コードチェックでは、classファイルも必要です。必ずチェック対象のJavaソースをビルドしてclassファイルを作成してください。
- コンパイル時のオプションとして、以下を必ず指定してください。以下のオプションが指定されていない場合、正しくチェックできません。

  【javacコマンドラインオプション】

  -g:source,lines

  【Eclipseでの設定】

  プロジェクトの[プロパティ]－[Javaコンパイラー]で以下をチェックしてください。

  - 生成されたクラス・ファイルに行番号属性を追加
  - 生成されたクラス・ファイルにソース・ファイル名を追加

■結果ファイルの出力について

コードチェック時に「結果ファイルの出力に失敗しました。」が出力される場合は、通知されたフォルダの書き込み権限、ディスク容量を確認してください。

■メモリ不足について

大規模な開発資産のプロジェクトを指定してコードチェックを行った場合に「メモリが不足しています。」というメッセージが出力される場合があります。

【回避方法】

最大ヒープサイズを既定値より大きく指定（推奨 256MB以上）してEclipseを起動します。

－ Eclipseの場合

Eclipse の起動オプションに「**-vmargs -Xmx256M**」を指定します。

起動オプションの詳細については、Eclipseのヘルプから「ワークベンチユーザガイド」－「タスク」－「Eclipseの実行」を参照してください。

■正しく検出されないケースについて

－ ファイル先頭コメントとクラスコメントの間に宣言が存在しない場合

ファイル先頭コメントとクラスコメントの間に宣言が存在しないコードは、規約K1C20001で誤指摘されます。

＜例＞

下記のプログラム例では、ファイル先頭コメントとクラスコメントの間に、import文など宣言が1つも存在しないため、6行目のクラスコメントをファイル先頭コメントと認識し、「/**」になっているためK1C20001規約違反として指摘してしまいます。

規約K1C20001:「ソースの先頭部分には、ヘッダコメントを記述する」

```
 1  /*
 2   * ファイル名 : MyClass.java
 3   * ・・・
 4   */
 5
 6  /**  ←K1C20001規約違反として誤指摘される
 7   * クラス名 : MyClass
 8   * ・・・
 9   */
10  public class MyClass {
11      :
12  }
```

7.2 Eclipse操作

■Eclipseの起動オプションについて

Agile+ Relief Jプラグインを更新する場合は、"-clean"オプションを付けてEclipseを起動してください。

■リンク資産について

－ Eclipseプロジェクトでリンクされたソースファイルは、コードチェック対象外です。Eclipseプロジェクトフォルダ配下に資産を配置するようにしてください。

－ プロジェクトの新規作成で[既存Antビルドファイルからの Javaプロジェクト]を選択してJavaプロジェクトを作成した場合、対象となる資産がリンクされるためコードチェックの対象外となります。既存Antビルドファイルは使わずに、Javaプロジェクトを新規に作成してください。

■ビルド設定について

ビルド後の自動チェック指定([ウィンドウ(W)]メニュー＞[設定(P)]－[Agile+ Relief J]－[ビルド時の設定])は設定時のプロジェクトにのみ有効です。

新規プロジェクトには適用されませんので、プロジェクトを追加した場合は、再度ビルド時の設定を行ってください。

■表示切り替えについて

コードチェック結果の表示対象となる指摘数が多い場合は、[Agile+ Relief 指摘メッセージ一覧]ビュー、および[Agile+ Relief 判定結果]ビューの表示切り替えに時間が掛かります。

■結果表示について

コードチェックの結果が、[Agile+ Relief 指摘メッセージ一覧]ビュー、および[Agile+ Relief 判定結果]ビューに反映されない場合は、以下を確認してください。

- 対象資産の選択

  パッケージ・エクスプローラ、プロジェクト・エクスプローラ、またはナビゲータのビューで対象資産(プロジェクトまたはファイル)を選択します。

- 他のビューとの連動

  [Agile+ Relief 指摘メッセージ一覧]ビュー、および[Agile+ Relief 判定結果]ビューのツールバーにある (他のビューに連動する)をオンにして他のビューに連動するようにし、対象資産を再度選択します。

- フィルタリング

  [Agile+ Relief 指摘メッセージ一覧]ビュー、および[Agile+ Relief 判定結果]ビューのフィルタを再設定します。

■指摘メッセージ一覧について

指摘メッセージ一覧中の「確認済」、「メモ」欄の内容は、再度コードチェックを実行しても引き継ぐことができますが、以下の場合は内容が引き継がれない場合があります。

- 指摘を修正し、再度のコードチェックで指摘されなくなった場合
- ファイル名が変更された場合
- ファイルを移動した場合
- プロジェクト名が変更された場合

■エディタ連携について

- 動作環境エラーなどのファイルが表示されない指摘の場合は、エディタ連携できません。
- [Agile+ Relief 指摘メッセージ一覧]ビュー、または[Agile+ Relief 判定結果]ビューでツールバーの (他のビューに連動する)がオフ、かつ表示対象のプロジェクトが閉じられている場合は、エディタ連携できません。

■エディタのマーカーについて

リファクタリングでプロジェクト名を変更した場合、変更前にコードチェック実行した結果を正しく表示できなくなります。

この場合、チェック結果をクリアした後、コードチェックを再実行してください。

■コードチェック結果の出力について

[Agile+ Relief 指摘メッセージ一覧]ビュー、または[Agile+ Relief 判定結果]ビューのCSV出力を行い、CSVの出力件数が65536行を超えた場合、Excelで当該出力ファイルを開くことはできません(Excelのバージョンによっては可能)。

■再コードチェックが必要なケースについて

以下のような操作を実行すると、マーカー、確認済、およびメモなどの情報の再設定が正常に行われない場合があります。これは、ソースの該当箇所が変更により削除されているなどが原因です。
このような場合は、再度、コードチェックを実行してください。

- 構成管理(CVSなど)との同期処理(Commit/Updateなど)の実施

ー Eclipse上での構成変更

- ソースフォルダ名、パッケージ名、フォルダ名、ファイル名の変更
- ソースフォルダ、パッケージ、フォルダ、ファイルの移動

ー Eclipse以外のツールによる直接変更

- ソースフォルダ名、パッケージ名、フォルダ名、ファイル名の変更
- ソースフォルダ、パッケージ、フォルダ、ファイルの移動
- ファイル名またはファイルの中身の変更

また、以下の変更を行った場合には、コードチェックの対象資産とコードチェック結果との整合性がとれなくなるため、チェック結果のクリア後に再コードチェックを行ってください。

ー ワークスペースの移動

ー プロジェクト名の変更

ー プロジェクトの移動

ー Eclipse外で検査規約定義ファイルの置き換え

■処理中の操作について

コードチェックの実行中や結果読み込み中は、Agile+ Relief Jの各機能は操作できません。

■ポップアップメニュー表示について

パッケージ・エクスプローラ、またはプロジェクト・エクスプローラのビューでJavaファイルとJSPファイルとを複数選択してポップアップメニューを表示した場合、ポップアップメニューに[Agile+ Relief J(J)]の項目は表示されません。

■納品ドキュメント出力について

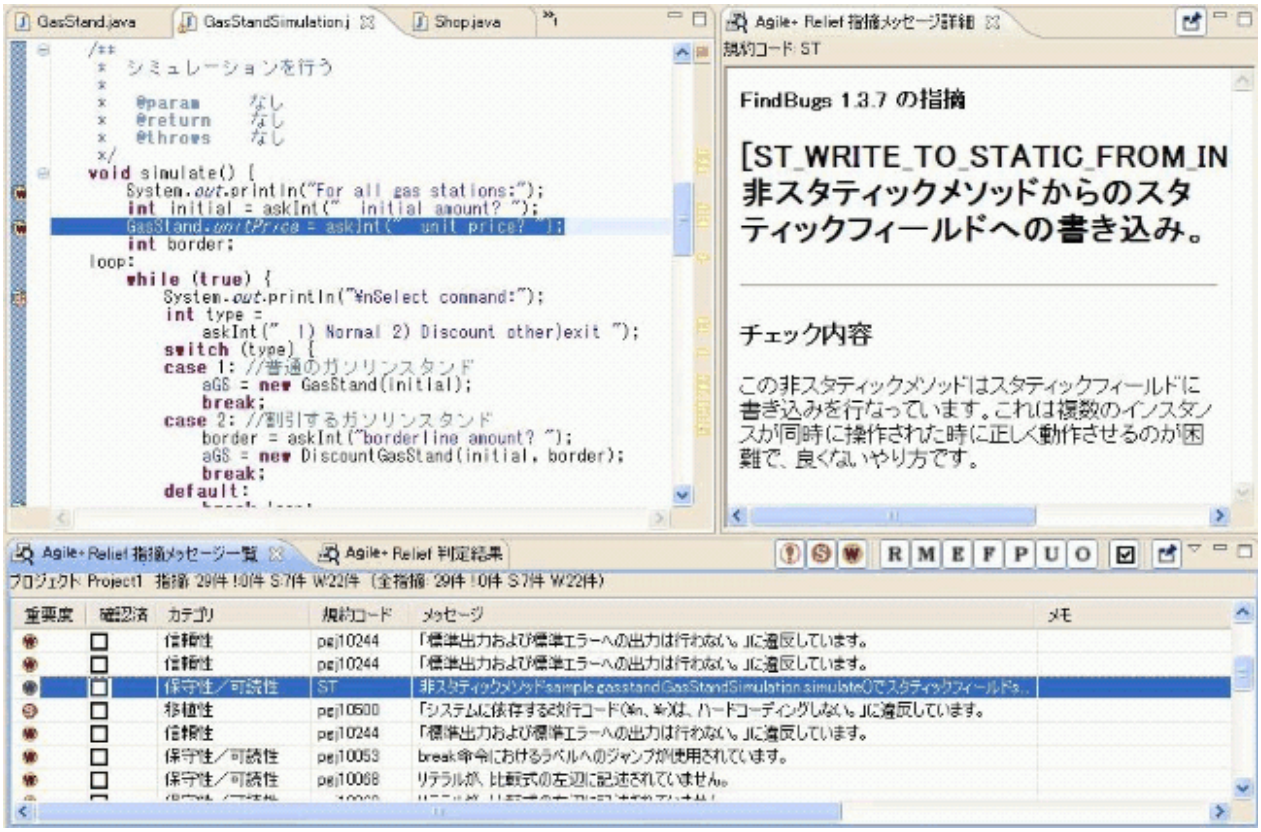
ファイル名に枝番が振られるケースでは、指定したファイル名が最大長の場合にファイルの出力に失敗します。納品ドキュメントに指定するファイル名の長さを短くしてください。

付録A　FindBugsとの連携について

ここでは、Agile+ Relief JのFindBugs連携とその使用方法について説明します。

A.1　FindBugs連携とは

FindBugs連携とは、オープンソースの静的解析ツールであるFindBugsと連携し、FindBugsの指摘をAgile+ Relief Jから操作できるようになります。



FindBugs連携では以下の機能を提供します。

- コードチェック実行時に自動でFindBugsを実行する
- FindBugsの指摘を、[Agile+ Relief 指摘メッセージ一覧]ビューで確認できる
- FindBugsの指摘詳細を、[Agile+ Relief 指摘メッセージ詳細]ビューで確認できる

FindBugs連携を利用することにより、コードチェックを強化し、品質向上を図ることができます。

A.2　FindBugsを使用する

FindBugsを利用するには、以下の手順による操作が必要です。

＜操作＞

1. FindBugsのインストール

   FindBugsの公開サイト(http://findbugs.sourceforge.net/)から配布物をダウンロードします。対象バージョンは、"Agile+ Reliefインストールフォルダ¥PGReliefJava" フォルダ配下にあるReadMe.txtを参照してください。

   ダウンロードした配布物を任意のフォルダに展開します。

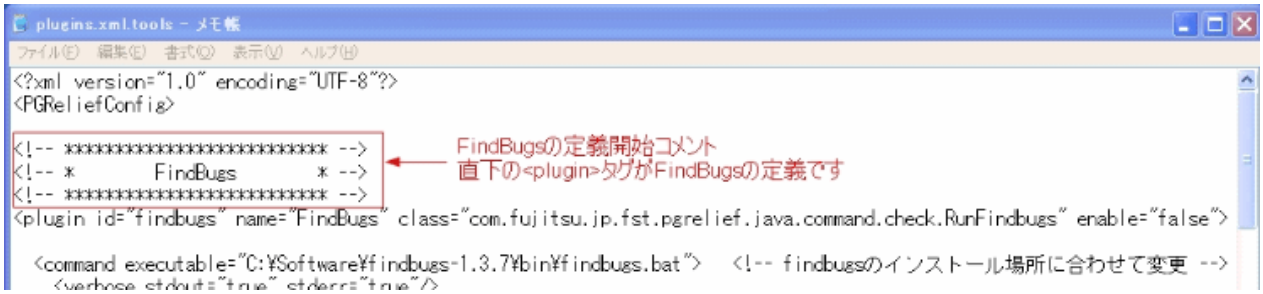
2. プラグインファイルの準備

   "Agile+ Reliefインストールフォルダ¥PGReliefJava" フォルダ配下にあるプラグインファイルの雛形ファイルを"plugins.xml.tools"から"plugins.xml"にリネームします。

   PMD連携の設定において、既にプラグインファイルのリネームをしている場合は、この操作は不要です。

3. プラグインファイルの編集

   FindBugs連携の有効／無効、FindBugsのコマンドパスおよびコマンドオプションを定義します。プラグインファイル(plugins.xml)内の以下のコメント以降の<plugin>タグがFindBugs連携に関する定義になります。



以下に定義内容の詳細を説明します。

設定内容

定義項目	タグ名	属性名	設定内容
ツール連携フラグ	plugin	enable	FindBugs連携の有効／無効を指定します。
コマンドパス	command	executable	FindBugsの実行ファイルのパスを指定します。
コマンドオプション	arg	value	FindBugsの実行ファイルに渡すコマンドオプションを指定します。

ツール連携フラグ

ツール連携フラグを変更することで、FindBugs連携を有効にしたり、無効にしたりできます。ツール連携フラグは、<plugin>タグのenable属性にtrueまたはfalseを指定します。

【FindBugs連携を有効にする場合】

```
<plugin id="findbugs" name="FindBugs" class="・・・" enable="true">
```

【FindBugs連携を無効にする場合】

```
<plugin id="findbugs" name="FindBugs" class="・・・" enable="false">
```

コマンドパス

FindBugsインストールフォルダに格納されているFindBugsの実行ファイル(バッチファイル)を指定します。

```
<command executable="FindBugsインストールフォルダ¥bin¥findbugs.bat">
```

コマンドオプション

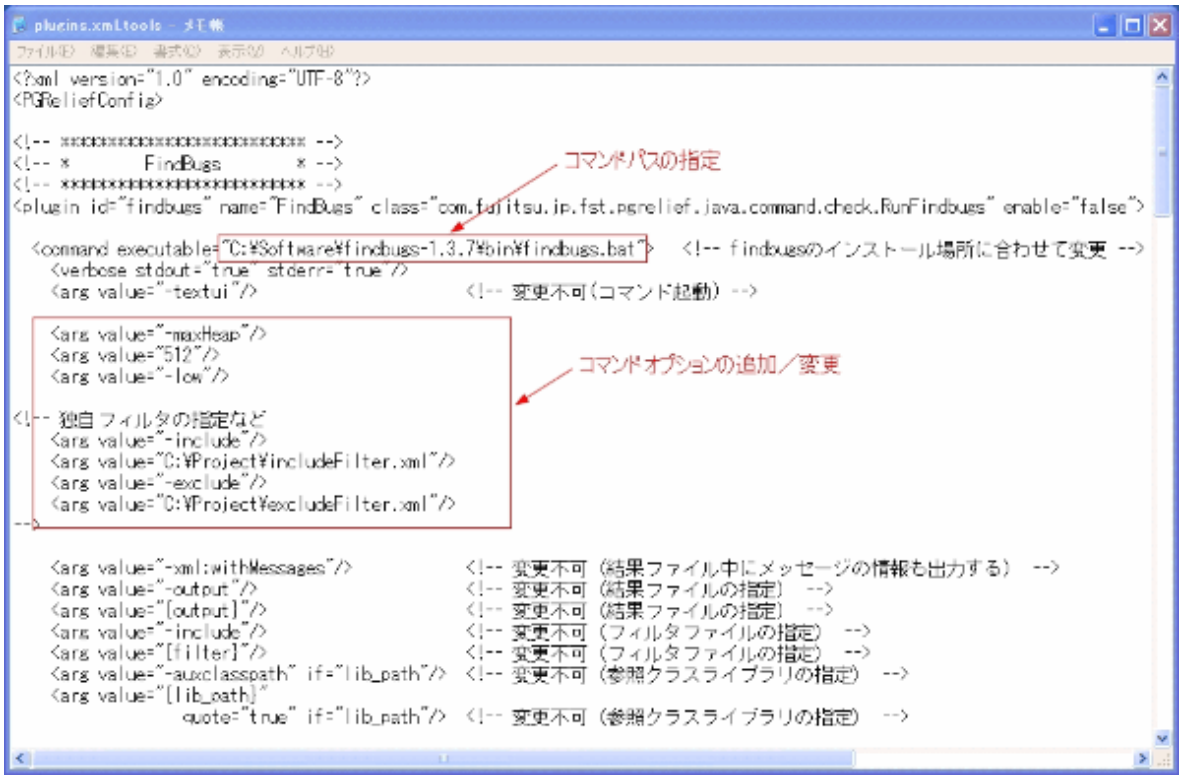
FindBugsの実行ファイルに渡すFindBugsオプションを指定します。オプションは、オプション名とそのパラメタの組み合わせを<arg>タグにより指定します。パラメタがない場合は、オプション名のみ<arg>タグで指定します。

例）"-maxHeap 1.5"というオプションの指定

```
<arg value="-maxHeap"/>
<arg value="512"/>
```

プラグインファイルには、予めオプションが定義されています。必要に応じて追加・変更してください。また、<!--変更不可-->というコメントがあるオプションについては、変更や削除をしないように注意してください。

追加する場合には、"独自フィルタの指定など"のコメントのある行を目安に、オプションを追加してください。



大量の参照クラスライブラリを指定して解析を実行するとFindBugsのコマンド文字数がシステムの制限を超過し、FindBugsが正しく動作しない場合があります。

FindBugs3.0.0以降をご利用で大量の参照クラスライブラリを指定する場合は、コマンドオプションを以下の様に変更してください。

【変更前】

```
<arg value="-auxclasspath" if="lib_path"/> <!-- 変更不可（参照クラスライブラリの指定） -->
<arg value="{lib_path}" quote="true" if="lib_path"/> <!-- 変更不可（参照クラスライブラリの指定） -->
```

【変更後】

```
<arg value="-auxclasspathFromFile" if="lib_path_file"/> <!-- 変更不可（参照クラスライブラリの指定） -->
<arg value="{lib_path_file}" quote="true" if="lib_path_file"/> <!-- 変更不可（参照クラスライブラリの指定）
-->
```

4. FindBugsの実行

   Agile+ Relief Jのコードチェックを実行することでFindBugsが実行されます。

   なお、plugins.xmlの編集は、Agile+ Relief Jを使用する前に済ませてください。Agile+ Relief J使用中にplugins.xmlを編集した場合は、[Agile+ Relief J] － [Agile+ Relief Jを使用する]メニューによりAgile+ Relief Jの使用を止め、再度使用状態にしてください。

5. FindBugsの結果確認

   FindBugsの指摘メッセージはAgile+ Relief J の指摘メッセージの形式でマッピングされ、[Agile+ Relief 指摘メッセージ一覧]ビュー、および[Agile+ Relief 指摘メッセージ詳細]ビューで確認することができます。

A.3 FindBugs指摘メッセージのマッピング

FindBugsの指摘メッセージは、以下のようにAgile+ Relief Jコードチェック結果にマッピングされ、指摘メッセージ一覧に表示されます。

カテゴリ

プラグインファイルの雛形ファイルである"plugins.xml.tools"では、FindBugsカテゴリとAgile+ Relief Jカテゴリの対応関係は、以下の表で示すように定義されています。表内にないFindBugsカテゴリは、Agile+ Relief Jカテゴリの「その他」に対応付けられます。

表A.1 "Pugins.xml.tools"でのFindBugsカテゴリとAgile+ Relief Jカテゴリのマッピング

FindBugs (日本語)	Agile+ Relief J（指摘一覧での表示形式）
Bad practice (良くない習慣)	信頼性
Performance (実行効率)	効率性
Correctness (正確性)	信頼性
Internationalization (国際化)	移植性
Multithreaded correctness (マルチスレッド環境での正確性)	信頼性
Malicious code vulnerability (脆弱性を持つコード)	機能性
Dodgy (回避可能)	保守性/可読性
Security (セキュリティ)	機能性
Experimental	その他

この対応関係は、プラグインファイル(plugins.xml)内の<option>タグで、カテゴリ単位に以下のように定義します。この定義を変更することで、対応関係をカスタマイズすることができます。

```
<option id="category.FindBugsカテゴリ名" value="Agile+ Relief Jカテゴリ名"/>
```

例) FindBugsの「BAD_PRACTICE」を、Agile+ Relief Jの「信頼性」に対応付ける

```
<option id="category.BAD_PRACTICE" value="CATEGORY_RELIABILITY"/>
```

Agile+ Relief Jのカテゴリ名は、以下を使用してください。

表A.2 Agile+ Relief Jカテゴリ名

Agile+ Relief Jカテゴリ名	意味
CATEGORY_MANITAINABILITY	保守性／可読性
CATEGORY_RELIABILITY	信頼性
CATEGORY_EFFICIENCY	効率性

Agile+ Relief Jカテゴリ名	意味
CATEGORY_FUNCTIONNALITY	機能性
CATEGORY_PORTABILITY	移植性
CATEGORY_OTHER	その他

重要度

重要度は、以下の表のように定義されています。変更することはできません。

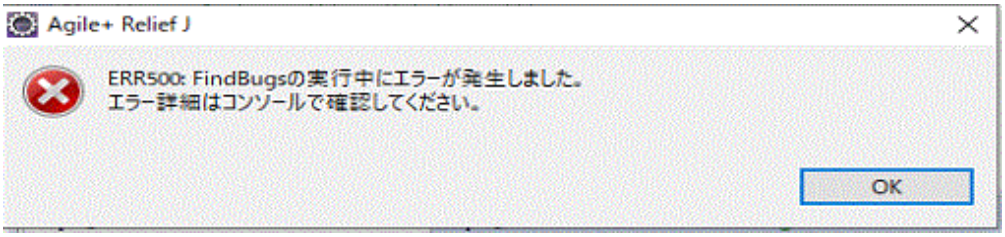
表A.3 FindBugsの重要度とAgile+ Relief Jの重要度のマッピング

FindBugs	Agile+ Relief J（指摘一覧での表示形式）
high-priority warning	重要度:S
medium-priority warning	重要度:W
low-priority warning	重要度:W

A.4 トラブルシューティング

FindBugs連携のよくある問題とその対処方法について説明します。

■コードチェック時に以下のエラーが表示される



コンソールビューに出力されている内容をもとに、FindBugsのマニュアルなどを参照して確認してください。

コンソールビューへの出力例をもとに、よくあるエラーを以下にあげます。

－FindBugsのオプションエラーが表示される

FindBugsのコマンド引数に無効なオプションを指定した場合に、以下のオプションエラーが表示されます。プラグイン用設定ファイル(plugins.xml)でコマンド引数を確認して正しいオプションに変更後、再度コードチェックを実行してください。

コンソール出力例（引数に-maxHeapと指定するところを、-maxHeaと指定した場合）

```
plugin: FindBugs

FindBugs: Unknown option: -maxHea
FindBugs: Usage: findbugs [general options] -textui [command line options...] [jar/zip/class files, directories...]
FindBugs: General options:
FindBugs:   -gui             Use the Graphical UI (default behavior)
FindBugs:   -gui1            Use the older Graphical UI
FindBugs:   -textui          Use the Text UI
    :
```

－FindBugsでメモリ不足が表示される

FindBugsの実行時にメモリ不足となった場合にエラーが表示されます。プラグイン用設定ファイル(plugins.xml)のコマンド引数(-maxHeapオプション)で、解析に必要なメモリ容量を指定してください。

コンソール出力例

```
plugin: FindBugs

FindBugs: Out of memory
FindBugs: Total memory: …
FindBugs:  free memory: OM
FindBugs: Analyzed: D:¥Work¥CT_Security¥PGRJava_CodeRev_Engine¥bin
FindBugs: Exception in thread "main" java.lang.OutOfMemoryError: Java heap space
FindBugs:  at edu.umd.cs.findbugs.asm.FBClassReader.readLabel(FBClassReader.java:53)
FindBugs:  at org.objectweb.asm.ClassReader.accept(Unknown Source)
    :
```

■指摘メッセージ一覧で、ERR500のメッセージが指摘される

前述のFindBugs実行時エラーの対処を参照してください。

■指摘メッセージ一覧で、FindBugsの指摘の行番号が0になる

FindBugsの一部の指摘において、FindBugsから行番号の情報を取得できない場合があります。このような場合、Agile+ Relief Jでは対象指摘の行番号を0として表示します。

■エディタで、FindBugsの指摘に対するマーカーが付かない

エディタで表示するファイルに指摘の行番号が見つからない場合は、エディタにマーカーは設定されません。指摘の行番号が0の場合も同様です(詳細は前述を参照)。

■指摘メッセージ一覧からのエディタ連携時に、エディタでファイルの先頭行に位置づけられる

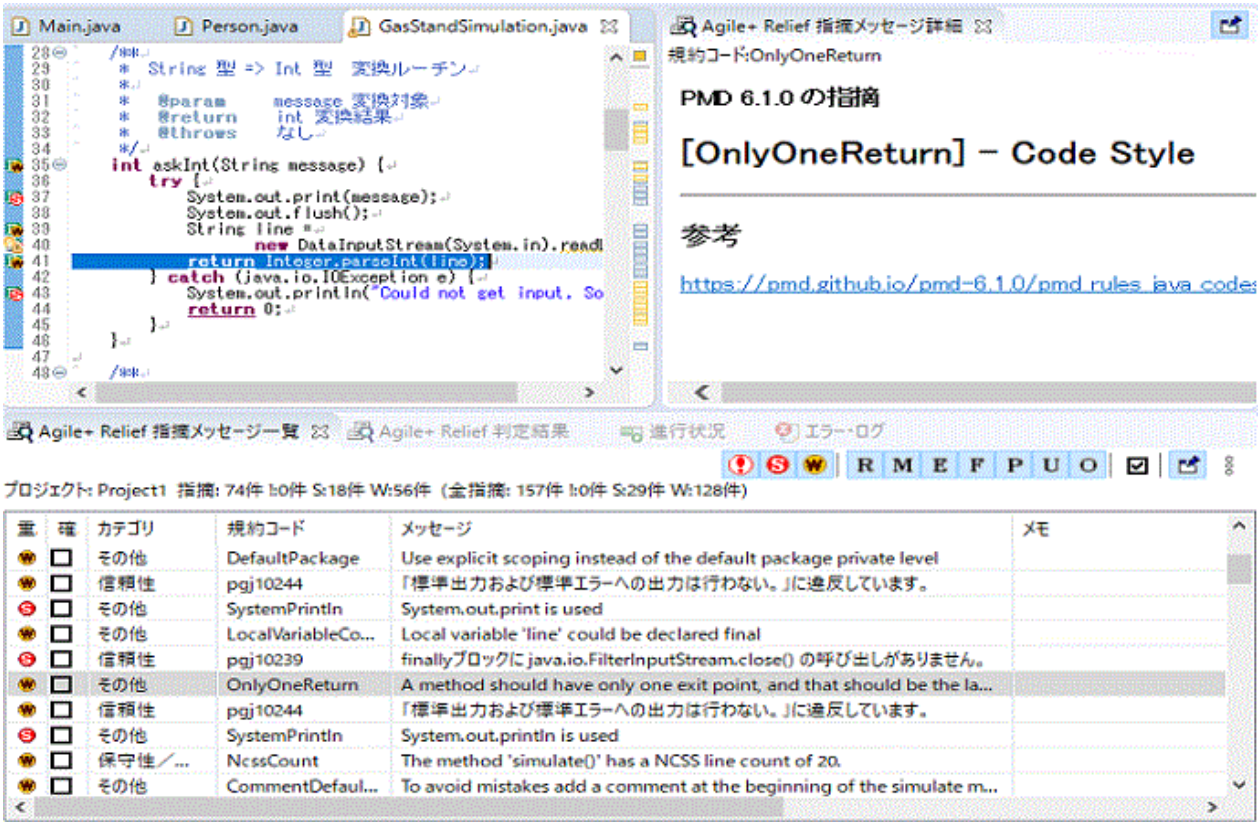
エディタで表示するファイルに指摘の行番号が見つからない場合は、ファイルの先頭行に位置づけられます。指摘の行番号が0の場合も同様です(詳細は前述を参照)。

付録B　PMDとの連携について

ここでは、Agile+ Relief JのPMD連携とその使用方法について説明します。

B.1　PMD連携とは

PMD連携とは、オープンソースの静的解析ツールであるPMDと連携し、PMDの指摘をAgile+ Relief Jから操作できるようになります。



PMD連携では以下の機能を提供します。

- コードチェック実行時に自動でPMDを実行する
- PMDの指摘を、[Agile+ Relief 指摘メッセージ一覧]ビューで確認できる
- PMDの指摘詳細を、[Agile+ Relief 指摘メッセージ詳細]ビューで確認できる

PMD連携を利用することにより、コードチェックを強化し、品質向上を図ることができます。

B.2　PMDを使用する

PMDを利用するには、以下の手順による操作が必要です。

＜操作＞

1. PMDのインストール

   PMDの公開サイト(http://pmd.sourceforge.net/)から配布物をダウンロードします。対象バージョンは、"Agile* Reliefインストールフォルダ¥PGReliefJava" フォルダ配下にあるReadMe.txtを参照してください。

   ダウンロードした配布物を任意のフォルダに展開します。

2. プラグインファイルの準備

   "Agile* Reliefインストールフォルダ¥PGReliefJava" フォルダ配下にあるプラグインファイルの雛形ファイルを"plugins.xml.tools"から"plugins.xml"にリネームします。

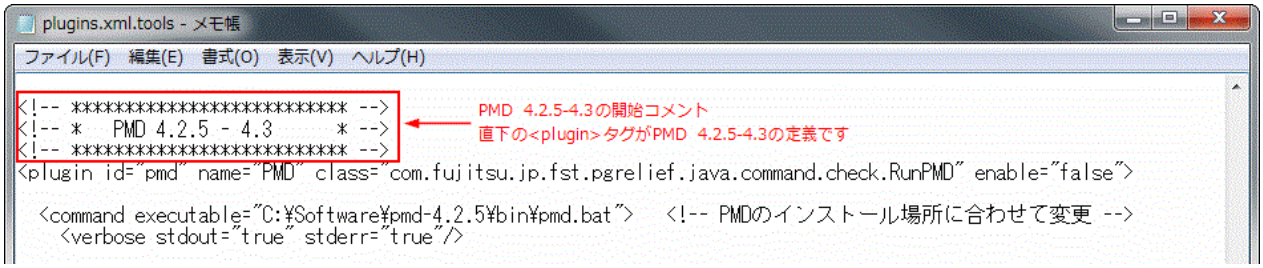
   FindBugs連携の設定において、既にプラグインファイルのリネームをしている場合は、この操作は不要です。

3. プラグインファイルの編集

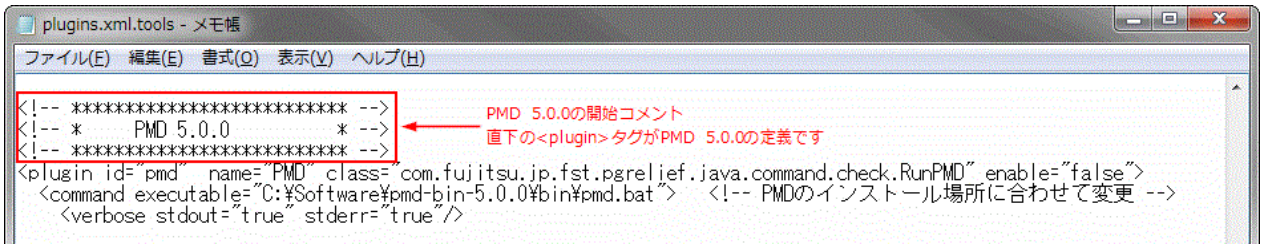
   PMD連携の有効／無効、PMDのコマンドパスおよびコマンドオプションを定義します。プラグインファイル(plugins.xml)内の以下のコメント以降の<plugin>タグがPMD連携に関する定義になります。

   なお、PMD連携のための<plugin>タグは、PMDのバージョン別に3種類あります。お使いのPMDのバージョンと一致する<plugin>タグを編集してください。

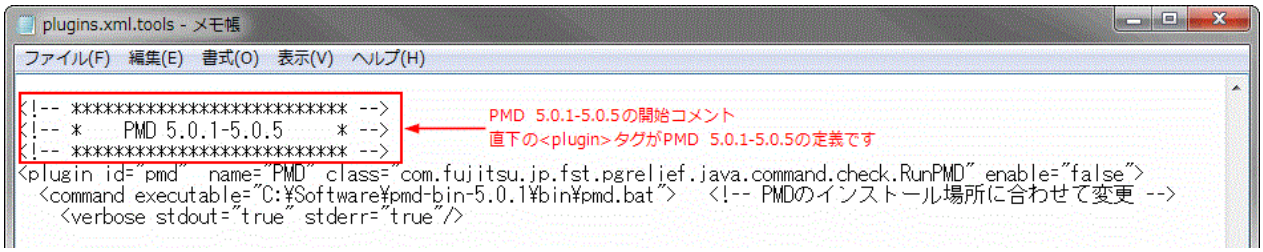
   [PMD 4.2.5-4.3の場合]



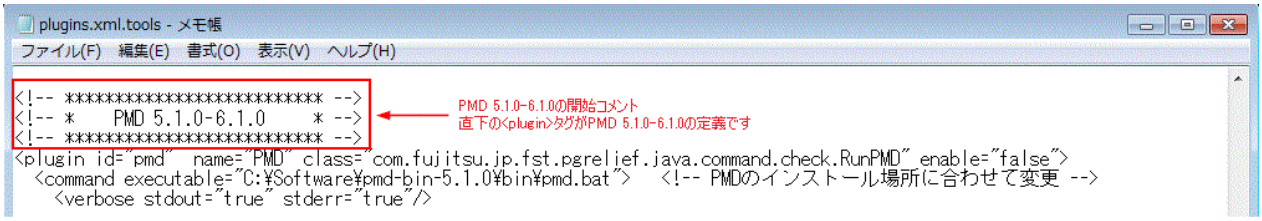
   [PMD 5.0.0の場合]



   [PMD 5.0.1-5.0.5の場合]



[PMD 5.1.0-6.1.0の場合]



以下に定義内容の詳細を説明します。

定義内容

定義項目	タグ名	属性名	設定内容
ツール連携フラグ	plugin	enable	PMD連携の有効／無効を指定します。
コマンドパス	command	executable	PMDの実行ファイルのパスを指定します。
コマンドオプション	arg	value	PMDの実行ファイルに渡すコマンドオプションを指定します。

ツール連携フラグ

ツール連携フラグを変更することで、PMD連携を有効にしたり、無効にしたりできます。ツール連携フラグは、<plugin>タグのenable属性にtrueまたはfalseを指定します。

【PMD連携を有効にする場合】

```
<plugin id="pmd" name="PMD" class="・・・" enable="true">
```

【PMD連携を無効にする場合】

```
<plugin id="pmd" name="PMD" class="・・・" enable="false">
```

コマンドパス

PMDインストールフォルダに格納されているPMDの実行ファイル（バッチファイル）を指定します。

```
<command executable="PMDインストールフォルダ¥bin¥pmd.bat">
```

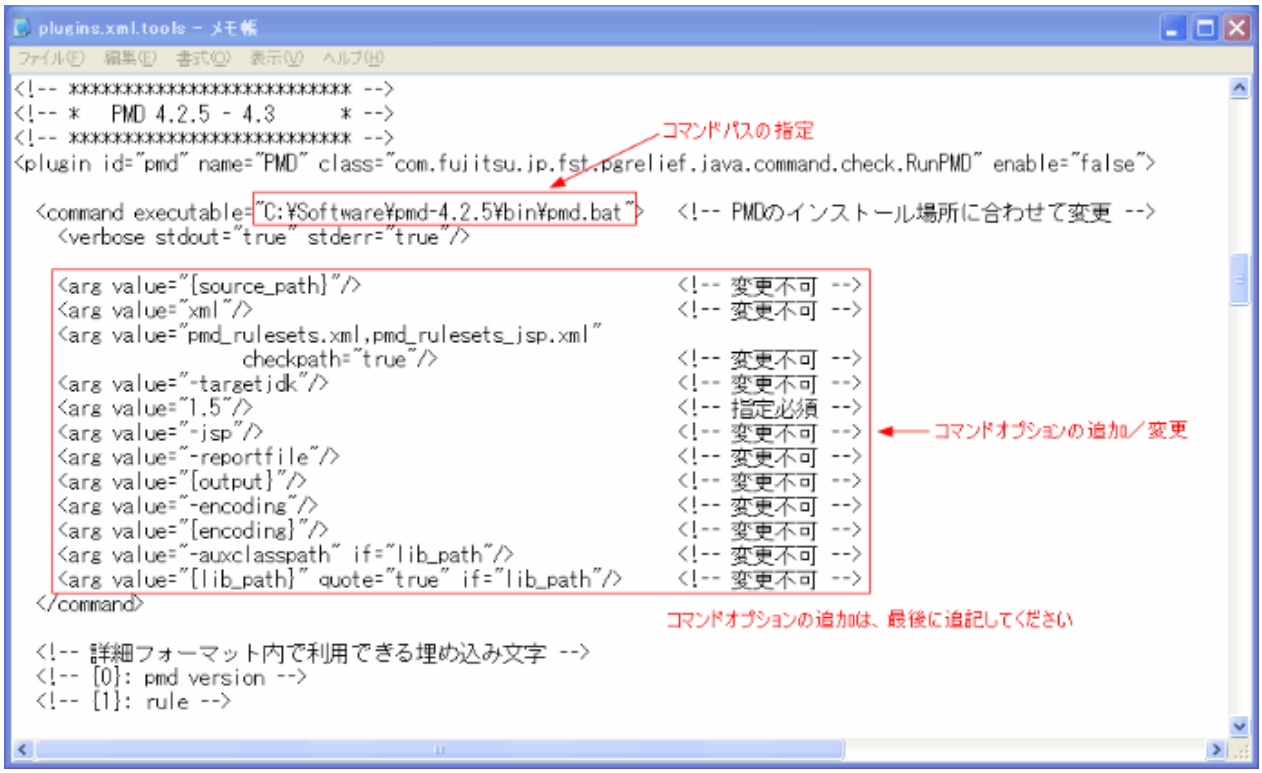
コマンドオプション

PMDの実行ファイルに渡すPMDオプションを指定します。オプションは、オプション名とそのパラメタの組み合わせを<arg>タグにより指定します。パラメタがない場合は、オプション名のみ<arg>タグで指定します。

例）"-targetjdk 1.5"というオプションの指定（PMD 4.2.5-4.3の場合）

```
<arg value="-targetjdk"/>
<arg value="1.5"/>
```

プラグインファイルには、予めオプションが定義されています。必要に応じて追加・変更してください。また、<!--変更不可-->というコメントがあるオプションについては、変更や削除をしないように注意してください。



なお、解析ソースのJavaバージョンを指定するオプションは、必ず確認してください。お使いの環境にあったJavaバージョンを指定してください。

以下は、PMDの各バージョンでのJavaバージョンの定義方法です。下線部で示すJavaバージョンを変更してください。

[PMD 4.2.5-4.3の場合]

```
<arg value="-targetjdk"/>          <!-- 変更不可 -->
<arg value="1.5"/>                 <!-- 指定必須 -->
```

[PMD 5.0.0の場合]

```
<arg value="-version"/>            <!-- 変更不可 -->
<arg value="java"/>                <!-- 変更不可 -->
<arg value="1.7"/>                 <!-- 指定必須 -->
```

[PMD 5.0.1-5.0.5の場合]

```
<arg value="-version"/>            <!-- 変更不可 -->
<arg value="1.7"/>                 <!-- 指定必須 -->
```

[PMD 5.1.0-6.1.0の場合]

```
<arg value="-version"/>            <!-- 変更不可 -->
<arg value="1.8"/>                 <!-- 指定必須 -->
```

大量の参照クラスライブラリを指定して解析を実行するとPMDのコマンド文字数がシステムの制限を超過し、PMDが正しく動作しない場合があります。

大量の参照クラスライブラリを指定する場合は、コマンドオプションを以下の様に変更してください。

【変更前】

```
<arg value="-auxclasspath" if="lib_path"/> <!-- 変更不可 -->
<arg value="{lib_path}" quote="true" if="lib_path"/> <!-- 変更不可 -->
```

【変更後】

```
<arg value="-auxclasspath" if="lib_path_file"/> <!-- 変更不可 -->
<arg value="file:///{lib_path_file}" quote="true" if="lib_path_file"/> <!-- 変更不可 -->
```

4. ルールセットファイルの編集

PMDのチェックルールは、"Agile+ Reliefインストールフォルダ¥PGReliefJava" フォルダ配下にある以下のルールセットファイルに定義されています。

[PMD 4.2.5-4.3の場合]

pmd_rulesets.xml	Javaのチェックルール
pmd_rulesets_jsp.xml	JSPのチェックルール

[PMD 5.0.0-5.0.5の場合]

pmd5_rulesets.xml	以下のルールセットが定義されています。 ・ Javaのチェックルール ・ JSPのチェックルール ・ XMLのチェックルール ・ XSLのチェックルール ・ ECMAScriptのチェックルール

[PMD 5.1.0-6.1.0の場合]

pmd5_1_rulesets.xml	以下のルールセットが定義されています。 ・ Javaのチェックルール ・ JSPのチェックルール ・ XMLのチェックルール ・ XSLのチェックルール ・ ECMAScriptのチェックルール ・ PLSQLのチェックルール ・ Apache Velocityのチェックルール

上記ルールセットファイルには、既にPMDで標準提供されているルールセットが定義されています。必要に応じて、削除または追加などしてください。

なお、これらのルールセットファイルの記述形式は、PMDのルールセットの記述形式に従っています。詳細は、PMDの公開サイト（http://pmd.sourceforge.net/）を参照してください。

5. PMDの実行

Agile+ Relief Jのコードチェックを実行することでPMDが実行されます。

なお、plugins.xmlの編集は、Agile+ Relief Jを使用する前に済ませてください。Agile+ Relief J使用中にplugins.xmlを編集した場合は、[Agile+ Relief J] － [Agile+ Relief Jを使用する]メニューによりAgile+ Relief Jの使用を止め、再度使用状態にしてください。

6. PMDの結果確認

PMDの指摘メッセージはAgile+ Relief J の指摘メッセージの形式でマッピングされ、[Agile+ Relief 指摘メッセージ一覧]ビュー、および[Agile+ Relief 指摘メッセージ詳細]ビューで確認することができます。

B.3 PMD指摘メッセージのマッピング

PMDの指摘メッセージは、以下のようにAgile+ Relief Jコードチェック結果にマッピングされ、指摘メッセージ一覧に表示されます。

カテゴリ

プラグインファイルの雛形ファイルである"plugins.xml.tools"では、PMDカテゴリとAgile+ Relief Jカテゴリの対応関係は、以下の表で示すように定義されています。表内にないPMDカテゴリは、Agile+ Relief Jカテゴリの「その他」に対応付けられます。

表B.1 "Plugins.xml.tools"でのPMDカテゴリとAgile+ Relief Jカテゴリのマッピング

言語	PMD 4.2.5-4.3	PMD 5.0.0-5.0.5	PMD 5.1.0-6.1.0	Agile+ Relief J（指摘一覧での表示形式）
Java	Android Rules	Android	Android	その他
Java	Basic Rules	Basic	Basic	保守性／可読性
Java	Braces Rules	Braces	Braces	保守性／可読性
Java	Clone Implementation Rules	Clone Implementation	Clone Implementation	信頼性
Java	Code Size Rules	Code Size	Code Size	保守性／可読性
Java	Controversial Rules	Controversial	Controversial	信頼性
Java	Coupling Rules	Coupling	Coupling	保守性／可読性
Java	Design Rules	Design	Design	保守性／可読性
Java	Finalizer Rules	Finalizer	Finalizer	信頼性
Java	Import Statement Rules	Import Statements	Import Statements	保守性／可読性
Java	J2EE Rules	J2EE	J2EE	移植性
Java	JavaBean Rules	JavaBean	JavaBean	その他
Java	JUnit Rules	JUnit	JUnit	その他
Java	Jakarta Commons Logging Rules	Jakarta Commons Logging	Jakarta Commons Logging	その他
Java	Java Logging Rules	Java Logging	Java Logging	信頼性
Java	Migration Rules	Migration	Migration	移植性
Java	Naming Rules	Naming	Naming	保守性／可読性
Java	Optimization Rules	Optimization	Optimization	効率性
Java	Strict Exception Rules	Strict Exception	Strict Exception	信頼性
Java	String and StringBuffer Rules	String and StringBuffer	String and StringBuffer	効率性
Java	Security Code Guidelines	Security Code Guidelines	Security Code Guidelines	機能性

言語	PMD 4.2.5-4.3	PMD 5.0.0-5.0.5	PMD 5.1.0-6.1.0	Agile+ Relief J（指摘一覧での表示形式）
Java	Type Resolution Rules	Type Resolution	Type Resolution	その他
Java	Unused Code Rules	Unused	Unused	保守性／可読性
Java	－	Comments	Comments	保守性／可読性
Java	－	Empty Code	Empty Code	保守性／可読性
Java	－	Unnecessary	Unnecessary	保守性／可読性
JSP	Basic JSF Rules	Basic JSF	Basic JSF	その他
JSP	Basic JSP Rules	Basic JSP	Basic JSP	その他
ECMAScript	－	Basic Ecmascript	Basic Ecmascript	その他
ECMAScript	－	Braces	Braces	保守性／可読性
ECMAScript	－	Unnecessary	Unnecessary	保守性／可読性
ECMAScript	－	－	Controversial Ecmascript	信頼性
XML	－	Basic XML	Basic XML	その他
XSL	－	XPath in XSL	XPath in XSL	その他
PLSQL	－	－	Tom Kyte's Despair	その他
PLSQL	－	－	Code Size	保守性／可読性
PLSQL	－	－	PLSQL DATETIME	信頼性
Apache Velocity	－	－	Basic Velocity	その他

この対応関係は、プラグインファイル(plugins.xml)内の<option>タグで、カテゴリ単位に以下のように定義します。この定義を変更することで、対応関係をカスタマイズすることができます。

```
<option id="category.PMDカテゴリ名" value="Agile+ Relief Jカテゴリ名"/>
```

例) PMDの「Basic Rules」を、Agile+ Relief Jの「保守性／可読性」に対応付ける

```
<option id="category.Basic Rules" value="CATEGORY_MANITAINABILITY"/>
```

Agile+ Relief Jのカテゴリ名は、以下を使用してください。

表B.2 Agile+ Relief Jカテゴリ名

Agile+ Relief Jカテゴリ名	意味
CATEGORY_MANITAINABILITY	保守性／可読性
CATEGORY_RELIABILITY	信頼性
CATEGORY_EFFICIENCY	効率性
CATEGORY_FUNCTIONNALITY	機能性
CATEGORY_PORTABILITY	移植性
CATEGORY_OTHER	その他

重要度

重要度は、以下の表のように定義されています。変更することはできません。

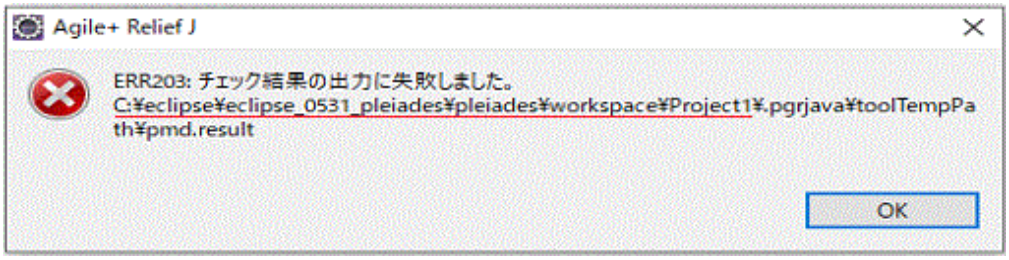
表B.3 PMDの重要度とAgile+ Relief Jの重要度のマッピング

PMD	Agile+ Relief J（指摘一覧での表示形式）
priority:1	重要度:S
priority:2	重要度:S
上記以外	重要度:W

B.4 トラブルシューティング

PMD連携のよくある問題とその対処方法について説明します。

■コードチェック時に以下のエラーが表示される



※下線部のパスは、使用環境により異なります。

コンソールビューに以下のようなメッセージが出力されていないか確認してください。

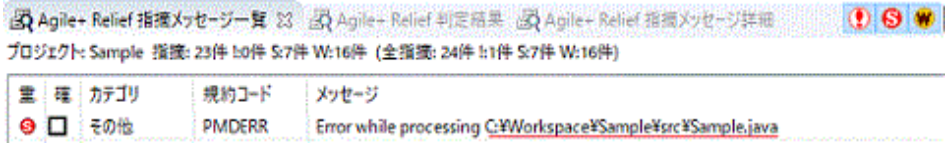
```
plugin: PMD

[Fatal Error] pmd.result:10:154: The entity "u65e5" was referenced, but not declared.
```

※下線部の文字列は、使用環境により異なります。

これは、解析対象となるJavaソースで、Javaの識別子に日本語を使用していることが原因として考えられます。PMDでは、Javaの識別子に日本語が使用されていると正常な解析結果が出力されない場合があります。

このような場合は、PMD連携を使用することはできません。

■指摘メッセージ一覧で、以下のPMDERRが指摘される



※下線部のパスは、使用環境により異なります。

解析対象の資産に構文的な誤りがないにも関わらず上記エラーが発生する原因として、以下の2つを確認しています。

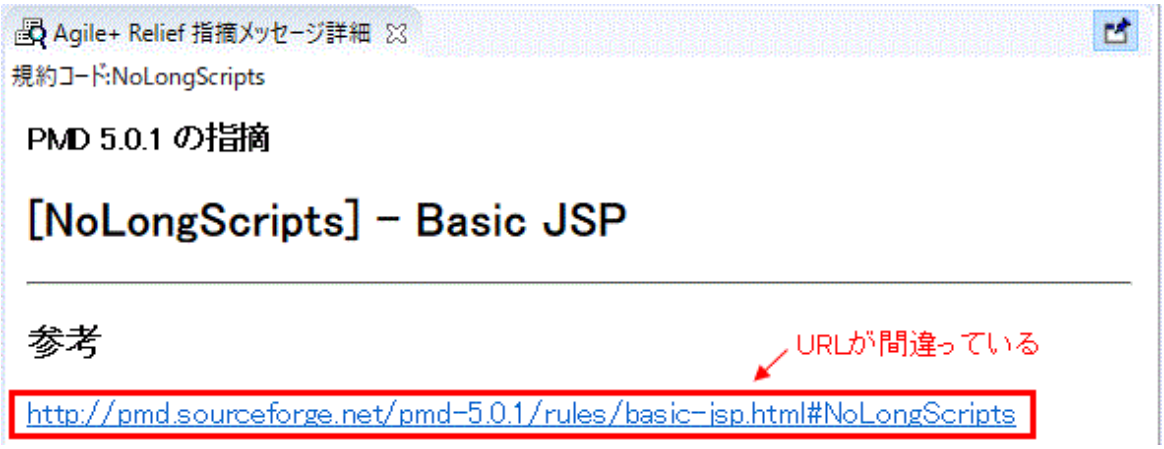
－Javaの識別子に日本語を使用している

－JavaファイルがBOM(Byte Order Mark)付きUTF-8エンコードのファイルである

PMDでは、上記のような場合に解析が正常に実行できません。

このような場合は、PMD連携を使用することはできません。

■指摘メッセージ詳細ビューに表示されるURLからPMDのサイトにジャンプできない



特定のPMDバージョンでは、指摘メッセージ詳細ビューに表示されるURLが間違っています。URLの情報はPMDの実行結果から取得しますが、PMD のバージョンによっては間違ったURLが返されます。

このような場合は、PMDのホームページから目的のドキュメントを参照してください。