Atelier 2

Améliorer la qualité du code Java

Tests unitaires avec JUnit

Unit est un framework open source pour réaliser des tests unitaires sur du code Java. Le principal intérêt est de s'assurer que le code répond toujours au besoin même après d'éventuelles modifications.

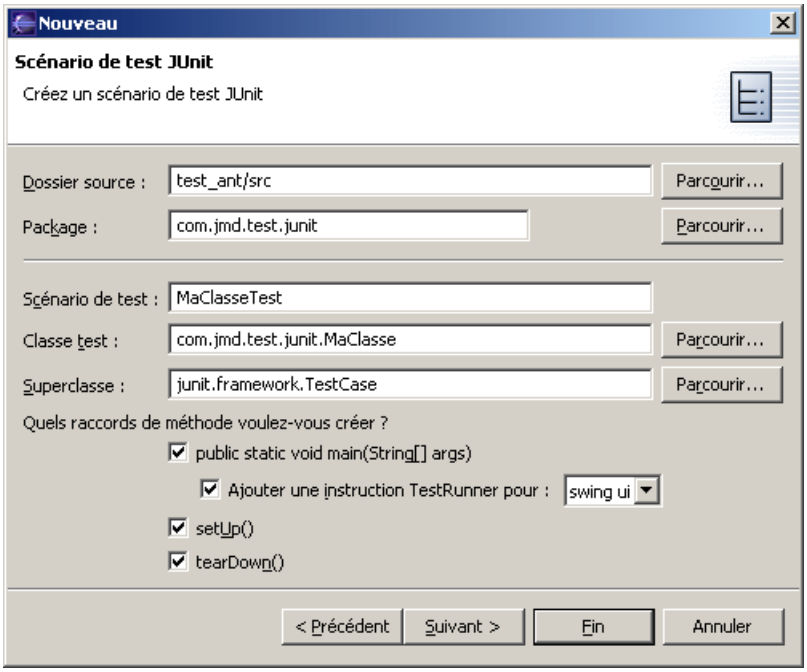
Le but est d'automatiser les tests. Ceux ci sont exprimés dans des classes sous la forme de cas de tests avec leurs résultats attendus. JUnit exécute ces tests et les comparent avec ces résultats.

Avec Junit, l'unité de tests est une classe dédiée qui regroupe des cas de tests. Ces cas de tests exécutent les tâches suivantes :

- création d'une instance de la classe et de tout autre objet nécessaire aux tests
- appel de la méthode à tester avec les paramètres du cas de test
- comparaison du résultat obtenu avec le résultat attendu : en cas d'échec, une exception est levée

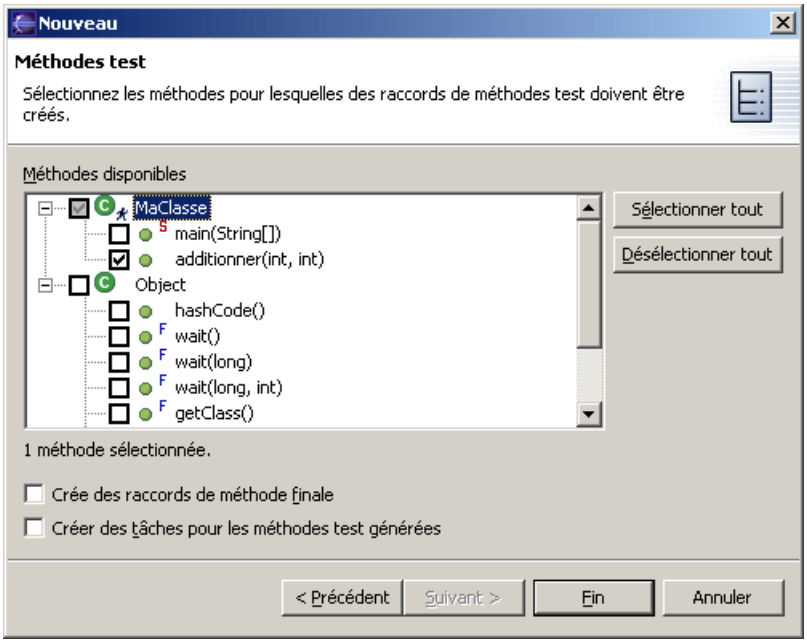
Ecriture des cas de tests

Pour utiliser JUnit, il faut créer une classe qui va contenir les cas de test. Il faut créer une nouvelle entité de type " Java / JUnit / Scénario de test ".



Si le fichier junit.jar n'est pas inclus dans le classpath du projet, un message d'erreur est affiché et il est impossible de poursuivre l'exécution de l'assistant.

Cliquez sur le bouton "Suivant".



Il faut compléter la classe générée selon les besoins : par exemple, ajouter un attribut qui va contenir une instance de la classe à tester, ajouter l'instanciation de cette classe dans la méthode setUp() et libérer cette instance dans la méthode tearDown().

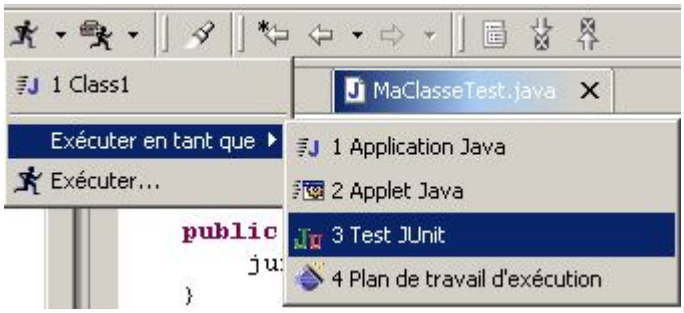
Il faut ajouter les traitements nécessaires dans les méthodes testXXX() en utilisant l'API de JUnit.

```
package com.moi.test.junit;
import junit.framework.TestCase;

public class MaClasseTest extends TestCase {
   private MaClasse maClasse = null;

   public MaClasseTest(String arg0) {
         super(arg0);
   }
   public static void main(String[] args) {
         junit.swingui.TestRunner.run(MaClasseTest.class);
   }
   protected void setUp() throws Exception {
         super.setUp();
         maClasse = new MaClasse();
   }
   protected void tearDown() throws Exception {
         super.tearDown();
         maClasse = null;
   }
   public void testAdditioner() {
         assertTrue(maClasse.additioner(2,2) == 4);
   }
}
```

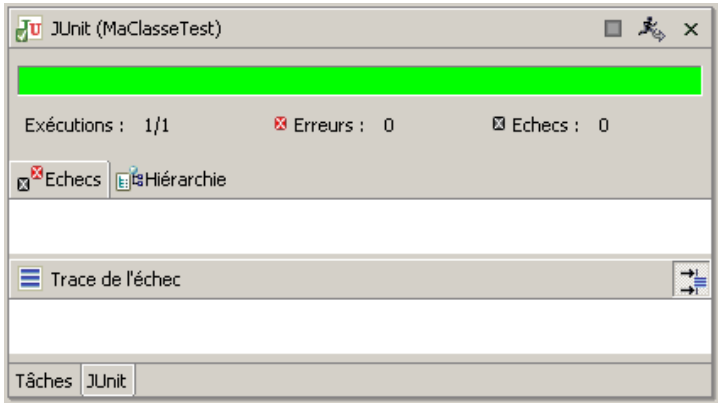
JUnit utilise l'instrospection pour exécuter les méthodes commençant par test.

Exécution des cas de tests

Pour exécuter les tests, il faut exécuter la classe en tant que « Test JUnit ».



Eclipse exécute les tests et affiche le résultat dans une vue dédiée.



Si tous les cas de tests ont été exécutés avec succès, une ligne verte est affichée.

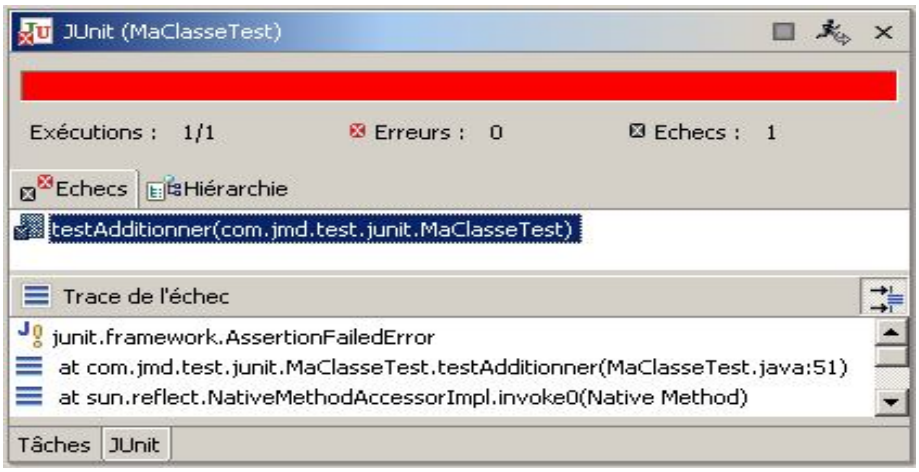
Cette vue contient deux onglets :

- « Failures » : contient la liste des cas de tests qui ont échoués
- « Hierarchy » : contient une arborescence des cas de tests

Dans le cas où un ou plusieurs tests échouent, la ligne est rouge.

```
public void testAdditioner() {
      assertTrue(maClasse.additioner(2,2) == 4);
      assertTrue(maClasse.additioner(2,3) == 4);

}
```

Une exécution de l'exemple précédent permet d'avoir le résultat suivant :



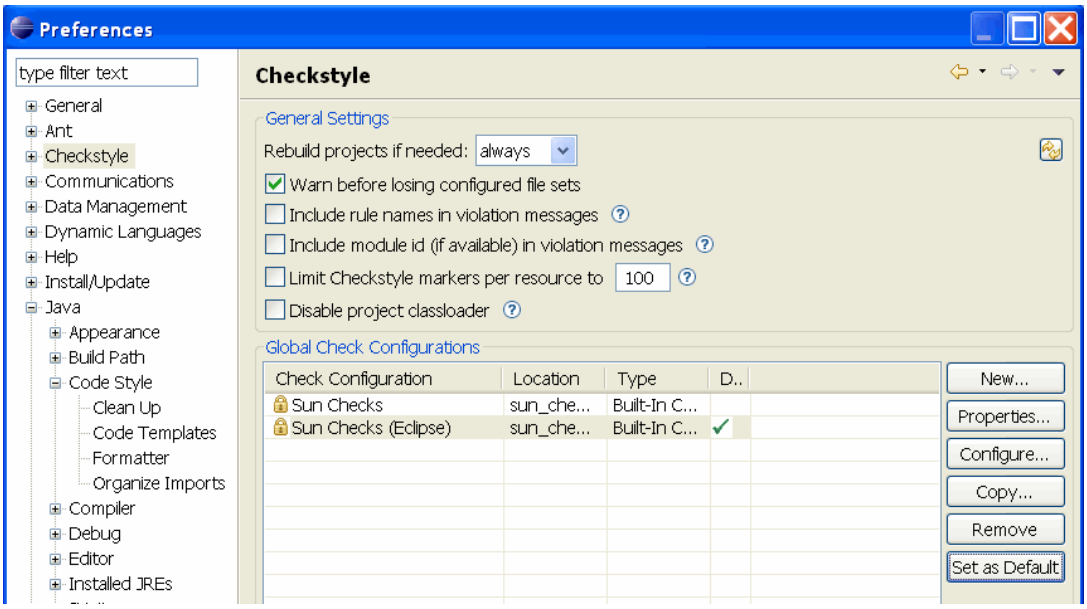
Qualité de codage avec CheckStyle

Checkstyle checks if the Java code is according to certain standards hence ensuring a certain quality of the coding.

Installation

Update site for the Eclipse Checkstyle Plugin : http://eclipse-cs.sourceforge.net/update.

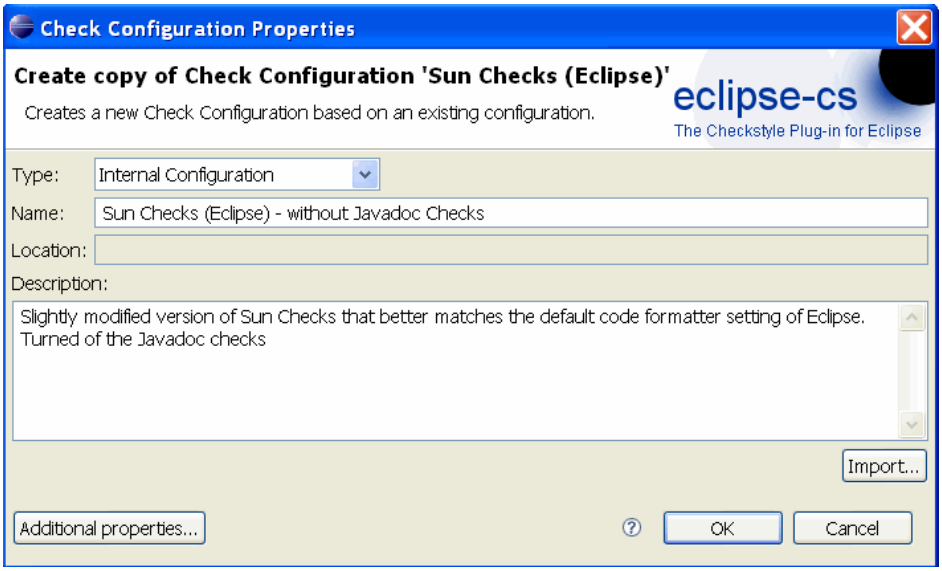
If you developing with Eclipse, make sure to select the Sun Conventions (Eclipse) under Window -> Preferences -> Checkstyle. Press the "Set as Default" after selecting the right entry.



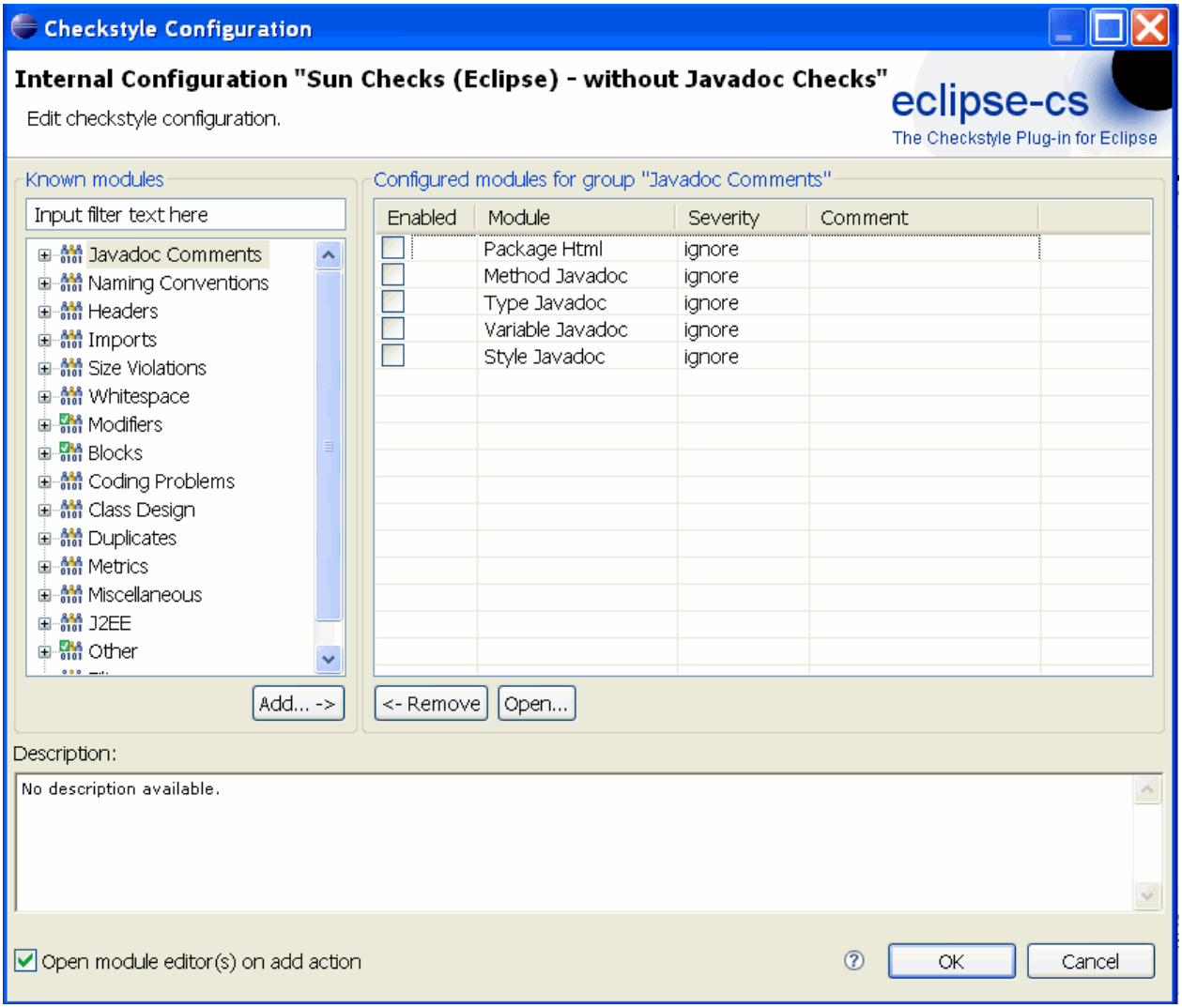
Configuration

You can turn of certain checks. If you change settings from the standard profile you should always make a copy of the existing profile.

To customizine your check, first make a copy of the checks



Select your new configuration and press Configure. De-activate for example the checks for Javadoc Comments.



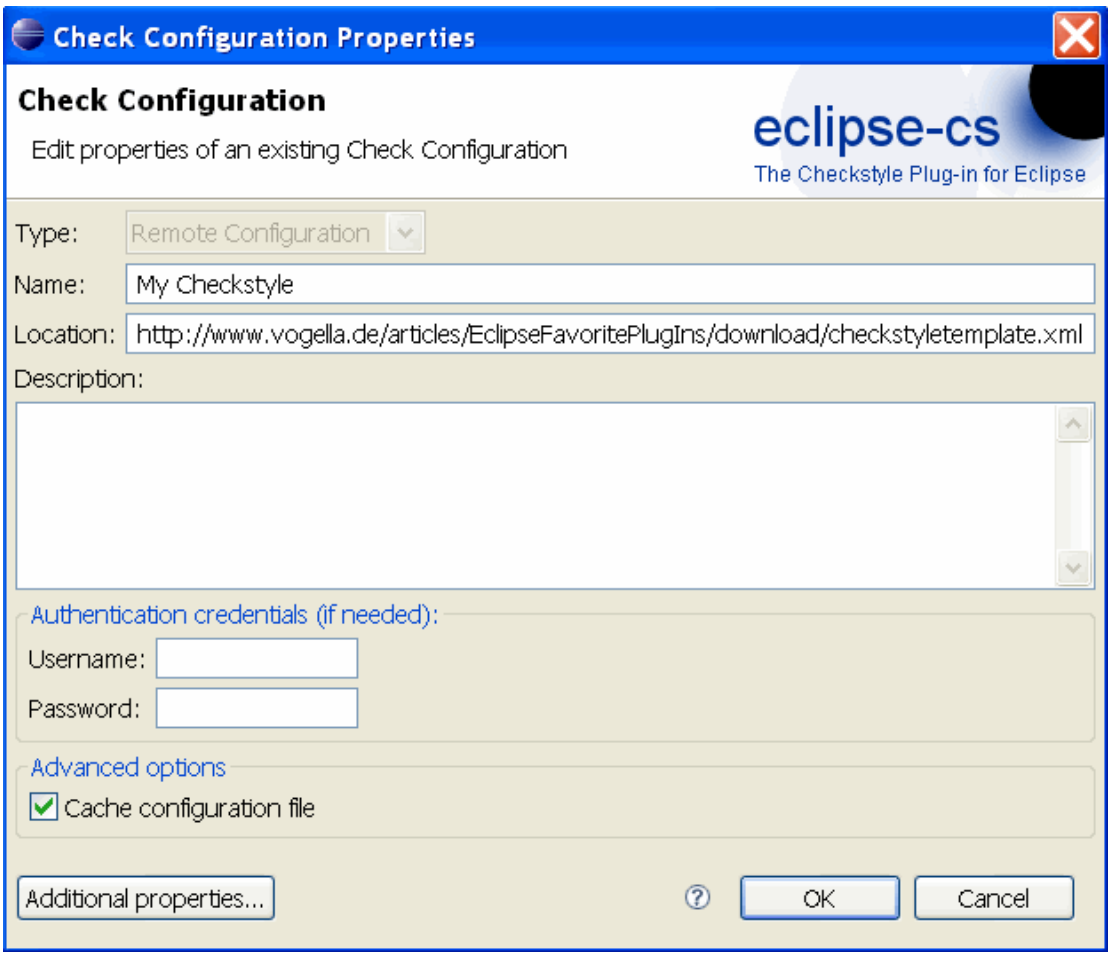
Make this new setting your default one.

2.4. Using common Checkstyle rules for teams

For teams it is good to follow the same coding style rules.

The Eclipse checkstyle plug-in allows this by proving a remote site for the checkstyle settings.

Press new in the settings. Select "Remote Configuration". Give the rule set a description and then type in the URL you want to use for the settings.

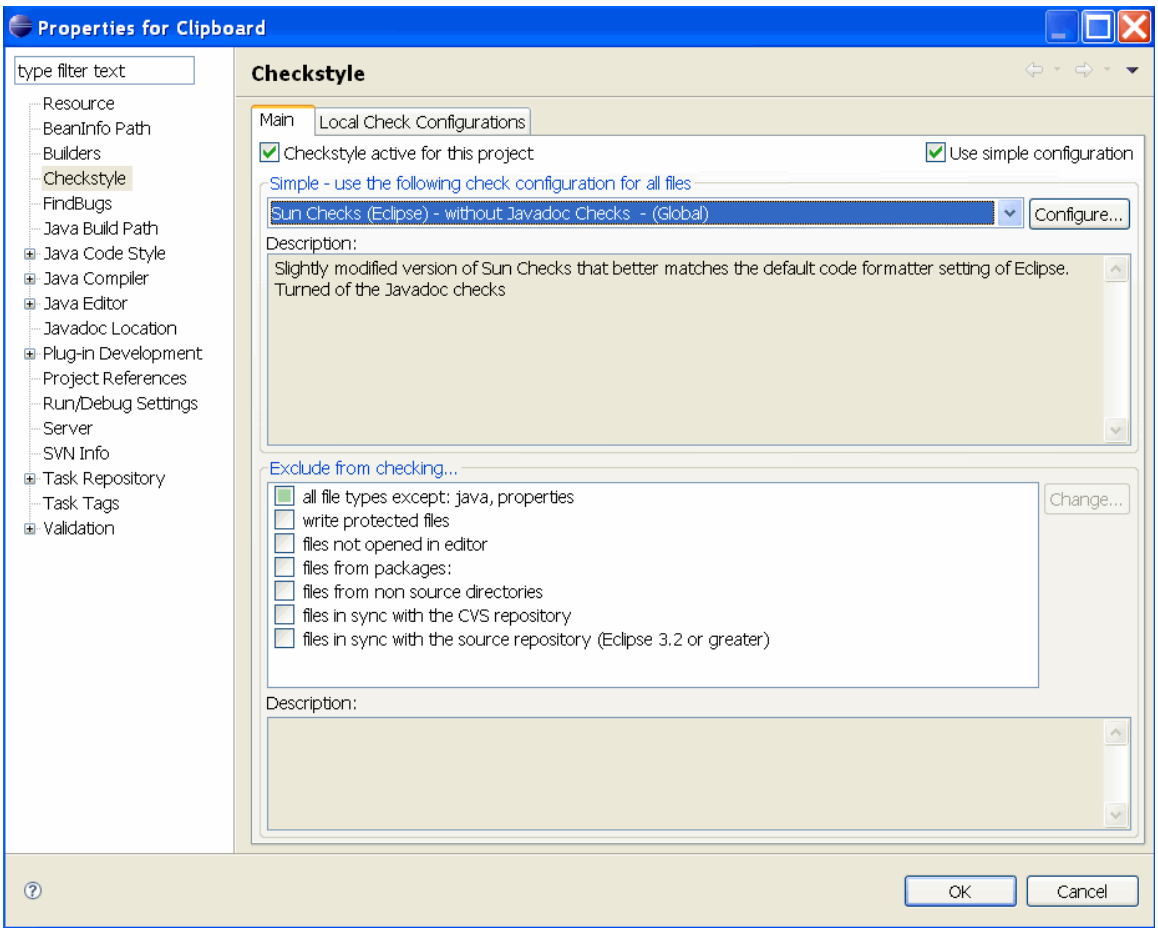


Make this new setting your default one.

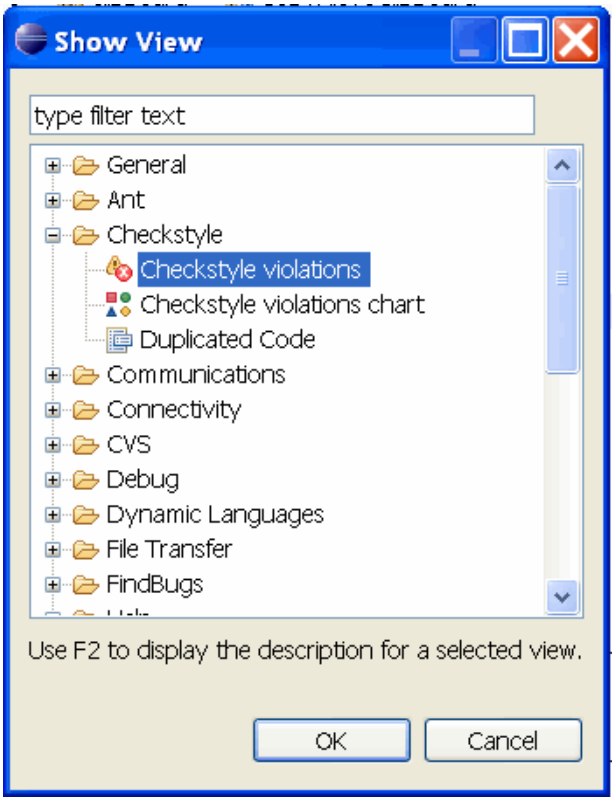
Using Checkstyle in your projects

Make your new profile the default one.

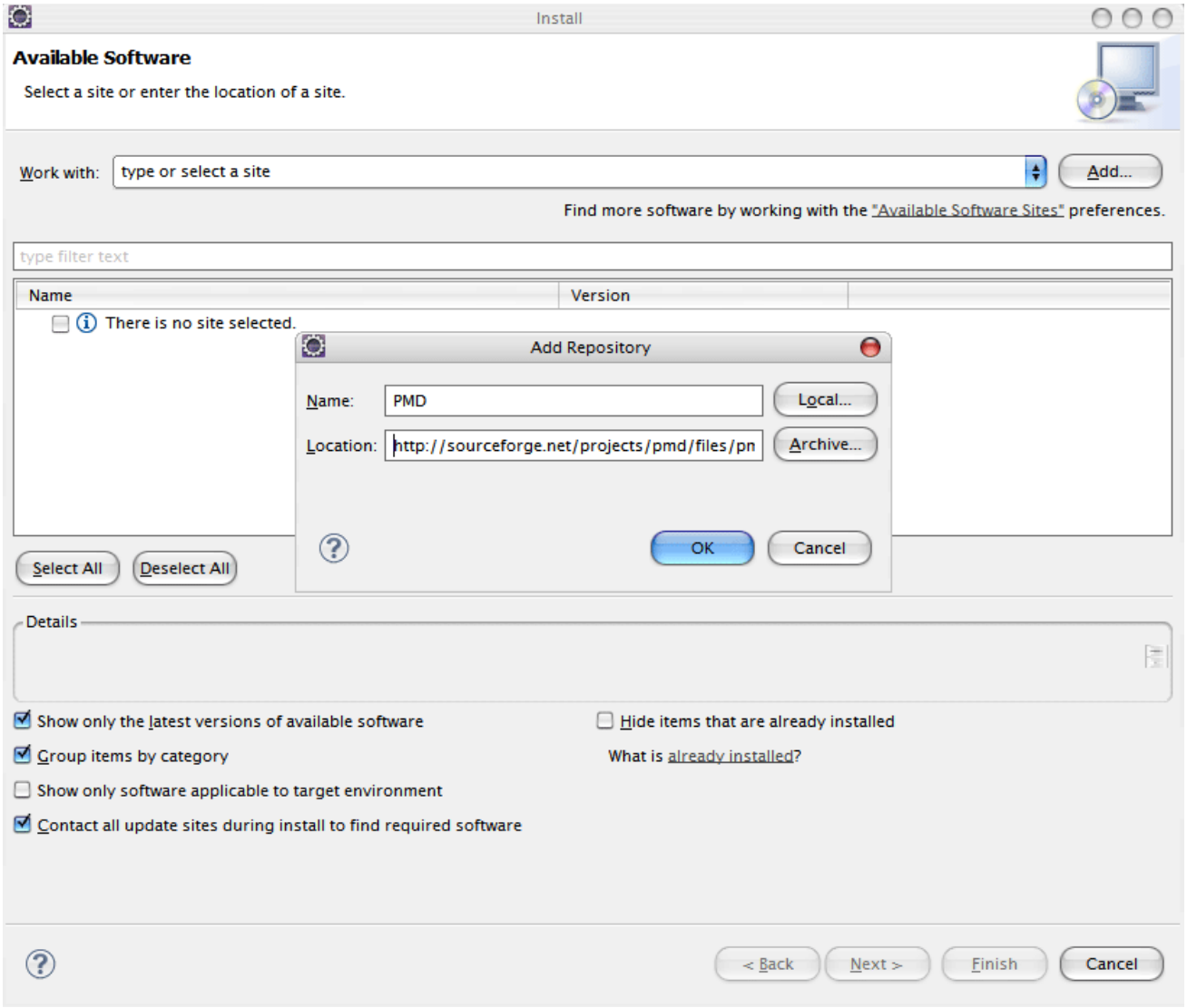
You need to activate the Eclipse Checkstyle Plugin for your project. Right click on your project and search for Checkstyle. Select the checkbox "Checkstyle active for this project".



You can use the checkstyle browser view to display the violations.



For Installing PMD Plugin from eclipse, select **Help->Install New Software** Then click add, then provide Name and Location according to the following screenshot

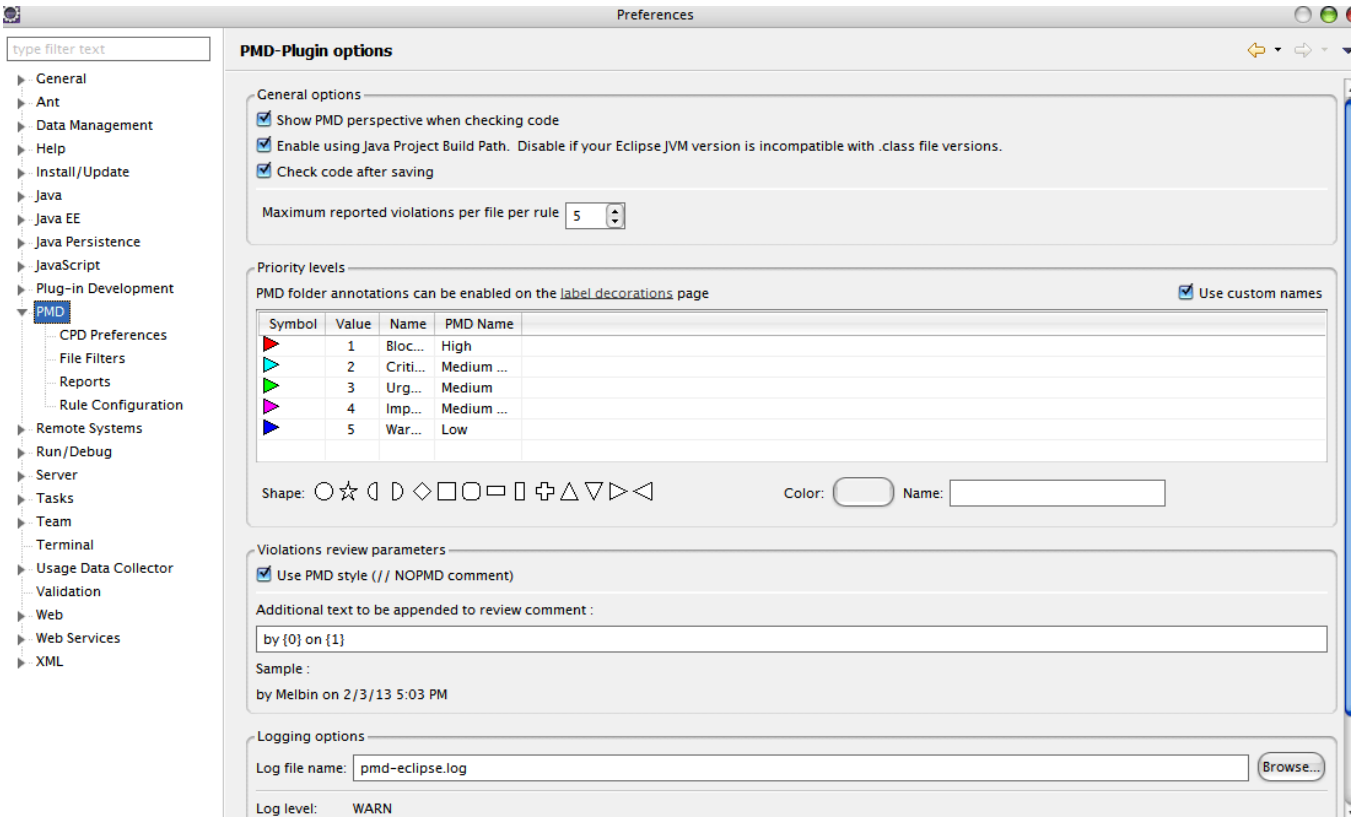


Then select the PMD version click next and accept the license details, it will install the PMD

PMD Eclipse Configuration

After install, you can configure PMD from eclipse, Window->Preferences->Java->PMD

Here you can apply **PMD Rules**, in order to utilizing different rules on **PMD bug patterns**, you can also configure different reports here



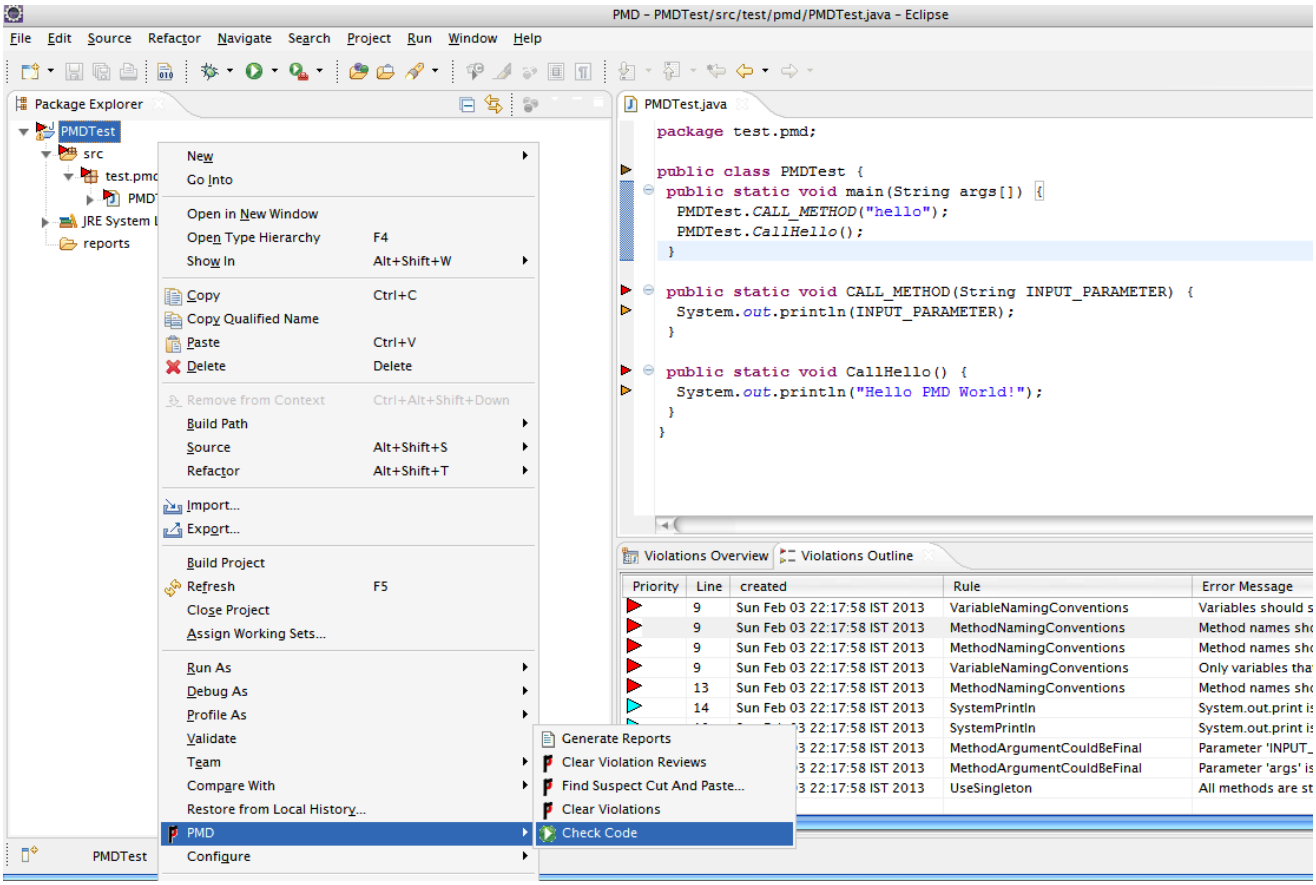
Using PMD Eclipse Plugin

Finding bugs using PMD, you need to create a java project and add the following class into src

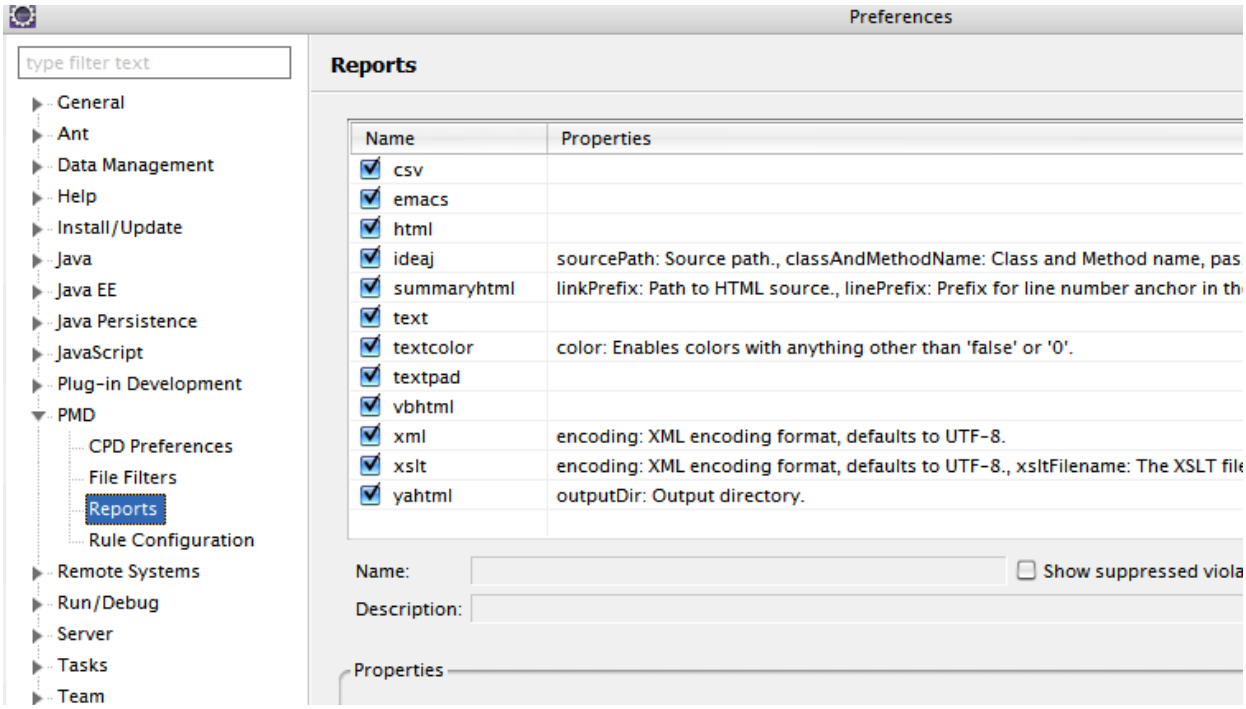
```
package test.pmd;

//PMD Eclipse Tutorial
public class PMDTest {
  public static void main(String args[]) {
    PMDTest.CALL_METHOD("hello");
    PMDTest.CallHello();
  }

  public static void CALL_METHOD(String INPUT_PARAMETER) {
    System.out.println(INPUT_PARAMETER);
  }

  public static void CallHello() {
    System.out.println("Hello PMD World!");
  }
}
```

Running PMD

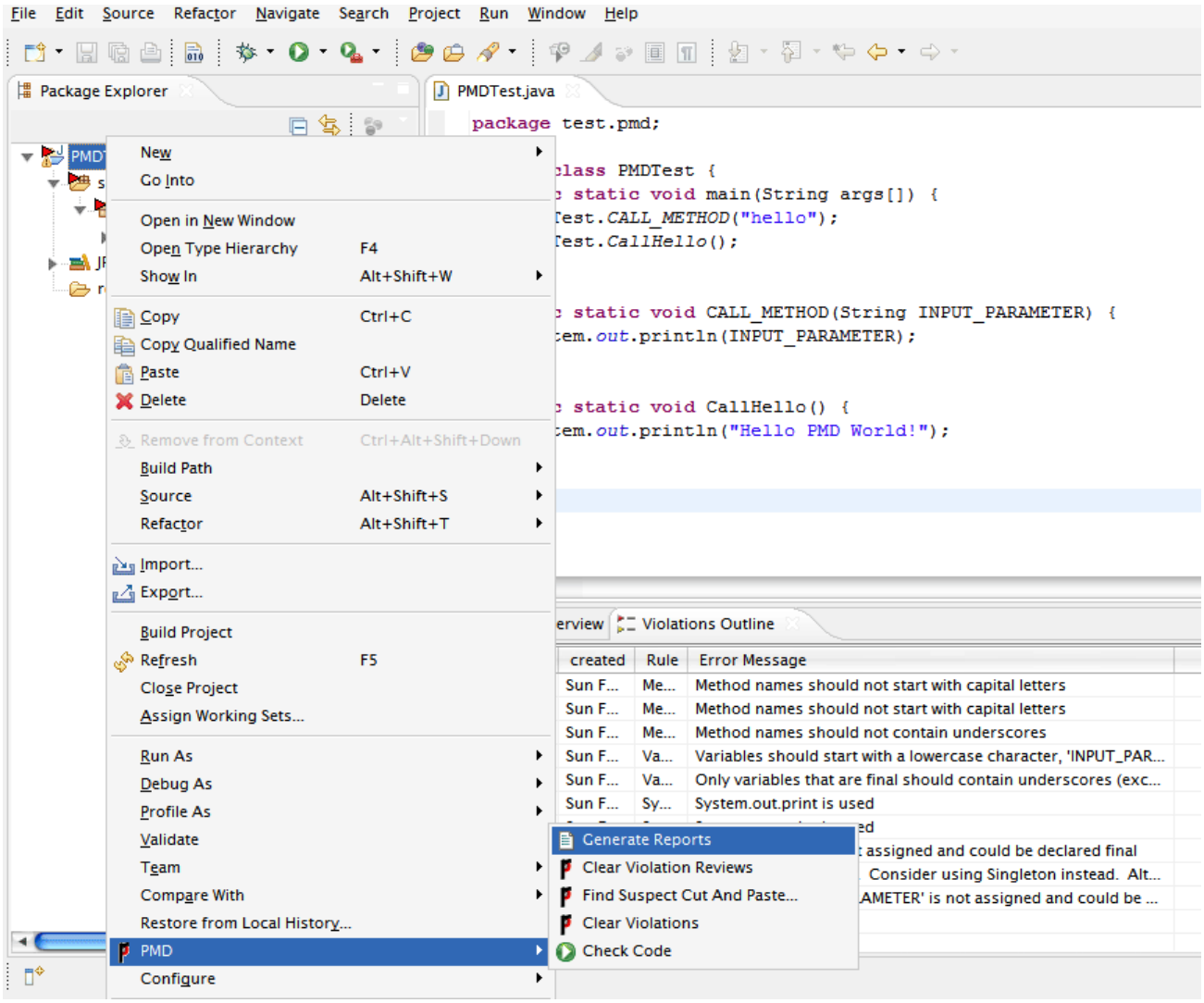
Then, Build the project using **Project->Build Project**. Right click on the project, and run the PMD according to following screenshot.



You can generate reports in various formats by using PMD, you can configure this in PMD settings (please check the screenshot below)



You can see below how to generate reports using PMD (please check the screenshot below)



Now you can see the generated reports inside the project folder (in our case we can see a separate reports folder inside PMDTest project)