Graphical Clone Checker

<정적분석 대응 보고서>

Team #3

201211341 김태현

201411259 고수창

200911411 이상규

Contents

1. Check Style 대응

Rule	Indentation
Problem	Checks correct indentation of Java Code.
Solution	코드 가독성을 위해 수정하지 않는다.

Rule	Parameter Name
Problem	Checks that parameter names conform to the specified format
Solution	IDE에서 기본적으로 생성되는 Parameter이므로 수정하지 않는다.

Rule	Method Name
Problem	Checks that method names conform to the specified format
Solution	코드의 가독성을 위해 수정하지 않는다.

Rule	Custom Import Order
Problem	Checks that the groups of import declarations appear in the order specified by the user. If there is an import but its group is not specified in the configuration such an import should be placed at the end of the import list.
Solution	import의 순서에 관련되어 필요성을 느끼지 못하여 수정하지 않는다.

Rule	Line Length
Problem	Checks for long lines.
Solution	코드의 가독성을 위해 수정하지 않는다.

Rule	Multiple Variable Declarations
Problem	Checks that each variable declaration is in its own statement and on its own line.
Solution	코드의 가독성을 위해 수정하지 않는다.

Rule	Variable Declaration Usage Distance
Problem	Checks the distance between declaration of variable and its first usage.
Solution	코드의 가독성을 위해 수정하지 않는다.

Rule	Comments Indentation
Problem	Controls the indentation between comments and surrounding code. Comments are indented at the same level as the surrounding code. Detailed info about such convention can be found here
Solution	코드의 가독성을 위해 수정하지 않는다.

2. Find Bug 대응

Rule	Dodgy - Write to static field from instance method
Problem	This instance method writes to a static field. This is tricky to get correct if multiple instances are being manipulated, and generally bad practice.
Solution	구현상에 어려움이 많기 때문에 수정하지 않는다.

Rule	Method names should start with a lower case letter
Problem	Methods should be verbs, in mixed case with the first letter lowercase, with the first letter of each internal word capitalized.
Solution	코드의 가독성을 위해 수정하지 않는다.

Rule	Performance - Method concatenates strings using + in a loop
Problem	The method seems to be building a String using concatenation in a loop. In each iteration, the String is converted to a StringBuffer/StringBuilder, appended to, and converted back to a String. This can lead to a cost quadratic in the number of iterations, as the growing string is recopied in each iteration. Better performance can be obtained by using a StringBuffer (or StringBuilder in Java 1.5) explicitly.
Solution	단기 프로젝트이므로 개발의 속도를 위해 수정하지 않는다.

Rule	Dodgy - Dead store to local variable
Problem	This instruction assigns a value to a local variable, but the value is not read or used in any subsequent instruction. Often, this indicates an error, because the value computed is never used. Note that Sun's javac compiler often generates dead stores for final local variables. Because FindBugs is a bytecode-based tool, there is no easy way to eliminate these false positives.
Solution	사용되지 않는 지역 변수를 제거한다.

Rule	Reliance on default encoding
Problem	Found a call to a method which will perform a byte to String (or String to byte) conversion, and will assume that the default platform encoding is suitable. This will cause the application behaviour to vary between platforms. Use an alternative API and specify a charset name or Charset object explicitly.
Solution	Windows와 Mac에서 정상 작동하므로 수정하지 않는다.

3. PMD 대응

Rule	Law Of Demeter
Problem	The Law of Demeter is a simple rule, that says "only talk to friends". It helps to reduce coupling between classes or objects. See also the references: Andrew Hunt, David Thomas, and Ward Cunningham. The Pragmatic Programmer. From Journeyman to Master. Addison-Wesley Longman, Amsterdam, October 1999.; K.J. Lieberherr and I.M. Holland. Assuring good style for object-oriented programs. Software, IEEE, 6(5):38–48, 1989.; http://www.ccs.neu.edu/home/lieber/LoD.html; http://en.wikipedia.org/wiki/Law_of_Demeter
Solution	구현상의 어려움 때문에 수정하지 않는다.

Rule	Short Variable
Problem	Detects when a field, local, or parameter has a very short name.
Solution	IDE에서 기본적으로 생성되는 Parameter이므로 수정하지 않는다.

Rule	Naming - Variable naming conventions
Problem	A variable naming conventions rule - customize this to your liking. Currently, it checks for final variables that should be fully capitalized and non-final variables that should not include underscores.
Solution	코드의 가독성을 위해 수정하지 않았다.

Rule	Naming - Method naming conventions
Problem	Method names should always begin with a lower case character, and should not contain underscores.
Solution	코드의 가독성을 위해 수정하지 않았다.

Rule	Bean Members Should Serialize
Problem	If a class is a bean, or is referenced by a bean directly or indirectly it needs to be serializable. Member variables need to be marked as transient, static, or have accessor methods in the class. Marking variables as transient is the safest and easiest modification. Accessor methods should follow the Java naming

	conventions, i.e.if you have a variable foo, you should provide getFoo and setFoo methods.
Solution	단기 프로젝트에 작은 규모이기 때문에 수정하지 않는다.

Rule	Loose coupling
Problem	Avoid using implementation types (i.e., HashSet); use the interface (i.e, Set) instead.
Solution	구현상의 어려움 때문에 수정하지 않는다.

Rule	System Println
Problem	System.(out|err).print is used, consider using a logger.
Solution	단기 프로젝트에 작은 규모이기 때문에 수정하지 않는다.

Rule	If Stmts Must Use Braces
Problem	Avoid using if statements without using curly braces.
Solution	코드의 가독성을 위해 수정하지 않는다.

Rule	Use Locale With Case Conversions
Problem	When doing a String.toLowerCase()/toUpperCase() call, use a Locale. This avoids problems with certain locales, i.e. Turkish.
Solution	C 프로그래밍 언어 코드이기 때문에 해당되지 않으므로 무시한다.