GRAMMATECH

SEI CERT-JAVA RULES AND RECOMMENDATIONS

MAPPED TO CODESONAR® 6.2 WARNING CLASSES

TRUSTED LEADERS OF SOFTWARE ASSURANCE AND ADVANCED CYBER-SECURITY SOLUTIONS

WWW.GRAMMATECH.COM

INTRODUCTION

The SEI CERT Oracle Coding Standard for Java (CERT-Java) provides rules and recommendations for secure coding in the Java programming language. The goal of these rules and recommendations is to develop safe, reliable, and secure systems, for example by eliminating undefined behaviors that can lead to undefined program behaviors and exploitable vulnerabilities. Conformanceto the coding rules defined in this standard is necessary (but not sufficient) to ensure the safety, reliability, and security of software systems developed in the Java programming language.

CodeSonar 6.2 includes a large number of warning classes that support checking for the CERT-Java rules and recommendations. Every CodeSonar warning report includes the identifiers of any CERT-Java rules and recommendations that are closely mapped to the warning's class. (The closemapping for a warning class is the set of categories—including CERT-Java rules and recommendations—that most closely match the class, if any).

You can configure CodeSonar to enable and disable warning classes mapped to specific CERT-Java rules and recommendations, or use build presets to enable all warning classes that are closely mapped to any CERT-Java rules and recommendations. In addition, you can use the CodeSonar search function to find warnings related to specific CERT-Java rules or recommendations, or to any CERT-Java rule or recommendation.

For more information on the SEI CERT-Java Coding Standard:

https://wiki.sei.cmu.edu/confluence/display/java/SEI+CERT+Oracle+Coding+Standard+for+Java

The remainder of this document comprises two tables:

- A table showing the close mapping between CodeSonar warning classes and the SEI CERTOracle Coding Standard for Java.

- A table showing the broad mapping between CodeSonar warning classes and the SEI CERTOracle Coding Standard for Java. The broad CERT-Java mapping for a CodeSonar warning class includes the close CERT-Java mapping for the class, plus any other CERT-Java rules and recommendations that are related to the class in a meaningful way, but not eligible for theclose mapping.

GrammaTech, Inc. is a leading developer of software-assurance tools and advanced cyber-security solutions. GrammaTech helps organizations develop and release high quality software, free of harmful defects that cause system failures, enable data breaches, and increase corporateliabilities in today's connected world. GrammaTech's CodeSonar is used by embedded developers worldwide.

SEI CERT ORACLE CODING STANDARD FOR JAVA CLOSE MAPPING (CODESONAR V6.2P0)

The following table contains CodeSonar warning classes that are closely mapped to CERT-Javarules and recommendations.

CERT-Java Rules and Recommendations		Closely Mapped CodeSonar Warning Classes
DCL00-J	Prevent class initialization cycles	Assertion Contains Side Effects (Java) Useless Assignment (Java) Useless Assignment to Default (Java)
DRD00	Do not store sensitive information on external storage (SD card) unless encrypted first	Sensitive Data Written to External Storage (Java)
DRD13	Do not provide addJavascriptInterface method access in a WebView which could contain untrusted content. (API level JELLY_BEAN or below)	Risky JavaScript Interface (Java)
DRD17-J	Do not use the Android cryptographic security provider encryption default for AES	Deprecated Cryptography Provider (Java)
DRD18	Do not use the default behavior in a cryptographic library if it does not use recommended practices	Cryptographic Algorithm with Risky Default Cipher (Java) Cryptographic Algorithm with Weak Cipher (Java) Cryptographic Algorithm with Weak Hash (Java) Deprecated Cryptography Provider (Java) Inadequate Salt (Java) Insecure Key Derivation (Java) Risky Cipher Algorithm (Java) Risky Cipher Field (Java) Risky Cryptographic Field (Java) Unsafe Base64 Encoding (Java) Weak Cryptographic Value (Java) Weak Hash Algorithm (Java
DRD22	Do not cache sensitive information	Sensitive Data Cached (Java) Sensitive Data Written to External Storage (Java) Sensitive Data Written to Local File (Java)
ENV01-J	Place all security-sensitive code in a single JAR and sign and seal it	LDAP Authentication Disabled (Java) Potential LDAP Poisoning (Java)
ENV03-J	Do not grant dangerous combinations of permissions	Accessing File in Permissive Mode (Java) Permissive File Mode (Java)
ENV06-J	Production code must not contain debugging entry points	Class Enables Debug Features (Java) Method Enables Debug Features (Java)
ERR00-J	Do not suppress or ignore checked exceptions	Empty Exception Handler (Java)
ERR02-J	Prevent exceptions while logging data	Debug Warning (Java)
ERR07-J	"Do not throw RuntimeException, Exception, or Throwable"	Broad Throws Clause (Java)
ERR08-J	Do not catch NullPointerException or any of its ancestors	Generic Exception Handler (Java) Inappropriate Exception Handler (Java)
ERR09-J	Do not allow untrusted code to terminate the JVM	Debug Call (Java)
EXP00-J	Do not ignore values returned by methods	Call Might Return Null (Java) Ignored Return Value (Java)
EXP01-J	Do not use a null in a case where an object is required	Actual Parameter Element may be null (Java) Field Element may be null (deep) (Java) Field may be null (deep) (Java) Null Parameter Dereference (Java) Null Pointer Dereference (Java) Null Pointer Dereference (deep) (Java) Return Value may Contain null Element (Java) Return Value may be null (Java) Return null Array (Java) Return null Boolean (Java) Return null Optional (Java) Unchecked Parameter Dereference (Java) Unchecked Parameter Dereference (deep) (Java) Unchecked Parameter Element Dereference (deep) (Java) null Passed to Method (deep) (Java)
EXP02-J	Do not use the Object.equals() method to compare two arrays	Should Use equals() Instead of == (Java) equals on Array (Java)

EXP03-J	Do not use the equality operators when comparing values of boxed primitives	Comparison to Empty String (Java) Should Use equals() Instead of == (Java) equals on Array (Java)
EXP06-J	Expressions used in assertions must not produce side effects	Assertion Contains Side Effects (Java)
FIO01-J	Create files with appropriate access permissions	Accessing File in Permissive Mode (Java) Permissive File Mode (Java)
FIO02-J	Detect and handle file-related errors	Ignored Return Value (Java)
FIO04-J	Release resources when they are no longer needed	Closeable Not Closed (Java) Closeable Not Stored (Java)
FIO09-J	Do not rely on the write() method to output integers outside the range 0 to 255	Call Might Return Null (Java)
IDS00-J	Prevent SQL injection	SQL Injection (Java)
IDS03-J	Do not log unsanitized user input	Tainted Log (Java)
IDS07-J	Sanitize untrusted data passed to the Runtime.exec() method	Command Injection (Java)
IDS08-J	Sanitize untrusted data included in a regular expression	Tainted Regular Expression (Java)
IDS14-J	Do not trust the contents of hidden form fields	Code Injection (Java) Command Injection (Java) Cross Site Scripting (Java) DLL Injection (Java) DOS Injection (Java) Reflection Injection (Java) SQL Injection (Java) Tainted @Trusted Value (Java) Tainted Bundle (Java) Tainted Control (Java) Tainted Expression Evaluation (Java) Tainted HTTP Response (Java) Tainted Hardware Device Property (Java) Tainted LDAP Attribute (Java) Tainted LDAP Filter (Java) Tainted Log (Java) Tainted Message (Java) Tainted Network Address (Java) Tainted Path (Java) Tainted Regular Expression (Java) Tainted Resource (Java) Tainted Session (Java) Tainted URL (Java) Tainted XAML (Java) Tainted XML (Java) Tainted Xpath (Java)
LCK00-J	Use private final lock objects to synchronize classes that may interact with untrusted code	Synchronization on Interned String (Java)
LCK05-J	Synchronize access to static fields that can be modified by untrusted code	Unguarded Method (Java)
LCK09-J	Do not perform operations that can block while holding a lock	Blocking in Critical Section (Java)
LCK10-J	Use a correct form of the double-checked locking idiom	Double-Checked Locking (Java)
MET08-J	Preserve the equality contract when overriding the equals() method	Asymmetric compareTo (Java) Missing Equals Override (Java) compareTo without equals (Java) compareTo/equals mismatch (Java)
MET09-J	Classes that define an equals() method must also define a hashCode() method	Defines equals but not hashCode (Java) Defines hashCode but not equals (Java)
MET10-J	Follow the general contract when implementing the compareTo() method	Missing Call to super (Java)
MSC02-J	Generate strong random numbers	Hardcoded Random Seed (Java) Insecure Random Number Generator (Java) Risky Cipher Field (Java) Risky Cryptographic Algorithm (Java) Risky Cryptographic Field (Java) Unsafe Base64 Encoding (Java) Weak Hash Algorithm Field (Java)
MSC03-J	Never hard code sensitive information	Hardcoded Password (Java) Sensitive Data Written to External Storage (Java)
MSC05-J	Do not exhaust heap space	Closeable Not Stored (Java) Inefficient Instantiation (Java)

NUM00-J	Detect or prevent integer overflow	Abs on random (Java) Cast: int Computation to long (Java)
NUM12-J	Ensure conversions of numeric types to narrower types do not result in lost or misinterpreted data	Approximate e Constant (Java) Approximate pi Constant (Java) Cast: Integer to Floating Point (Java) Floating Point Equality (Java)
NUM13-J	Avoid loss of precision when converting primitive integers to floating-point	Cast: Integer to Floating Point (Java)
OBJ07-J	Sensitive classes must not let themselves be copied	clone Non-cloneable (Java) clone Subclass of Non-clonable (Java) clone not final (Java)
OBJ08-J	Do not expose private members of an outer class from within a nested class	Inner Class Should be Static (Java)
SEC01-J	Do not allow tainted variables in privileged blocks	Accessing File in Permissive Mode (Java) Permissive File Mode (Java)
SEC05-J	"Do not use reflection to increase accessibility of classes, methods, or fields"	Reflection Bypasses Member Accessibility (Java) Reflection Modifies Member Accessibility (Java)
SEC06-J	Do not rely on the default automatic signature verification provided by URLClassLoader and java.util.jar	Android Message Injection (Java) Tainted Message (Java)
SER00-J	Enable serialization compatibility during class evolution	Missing Serial Version Field (Java)
SER01-J	Do not deviate from the proper signatures of serialization methods	Serialization Not Disabled (Java)
SER02-J	Sign then seal objects before sending them outside a trust boundary	Android Message Injection (Java) Tainted Message (Java)
SER03-J	Do not serialize unencrypted sensitive data	Serialization Not Disabled (Java)
SER06-J	Make defensive copies of private mutable components during deserialization	Serialization Not Disabled (Java)
SER07-J	Do not use the default serialized form for classes with implementation-defined invariants	Serialization Not Disabled (Java)
SER10-J	Avoid memory and resource leaks during serialization	Closeable Not Stored (Java) Inefficient Instantiation (Java)
SER12-J	Prevent deserialization of untrusted data	Serialization Not Disabled (Java)
THI00-J	Do not invoke Thread.run()	Synchronous Call to Thread Body (Java)
VNA00-J	Ensure visibility when accessing shared primitive variables	Impossible Client Side Locking (Java) Missing synchronized Statement (Java) Synchronization on static (Java) Unguarded Field (Java) Unguarded Parameter (Java) Useless volatile Modifier (Java)
VNA03-J	Do not assume that a group of calls to independently atomic methods is atomic	Useless volatile Modifier (Java)

SEI CERT ORACLE CODING STANDARD FOR JAVA BROAD MAPPING (CODESONAR V6.2P0)

The following table contains CodeSonar warning classes that are broadly mapped to CERT-Javarules and recommendations.

Warning classes that are also in the close mapping are shown in bold text.

CERT-Java Rules and Recommendations		Closely Mapped CodeSonar Warning Classes
DCL00-J	Prevent class initialization cycles	**Assertion Contains Side Effects (Java)** **Useless Assignment (Java)** **Useless Assignment to Default (Java)**
DRD00	Do not store sensitive information on external storage (SD card) unless encrypted first	**Sensitive Data Written to External Storage (Java)**
DRD13	Do not provide addJavascriptInterface method access in a WebView which could contain untrusted content. (API level JELLY_BEAN or below)	**Risky JavaScript Interface (Java)**
DRD17-J	Do not use the Android cryptographic security provider encryption default for AES	**Deprecated Cryptography Provider (Java)**
DRD18	Do not use the default behavior in a cryptographic library if it does not use recommended practices	**Cryptographic Algorithm with Risky Default Cipher (Java)** **Cryptographic Algorithm with Weak Cipher (Java)** **Cryptographic Algorithm with Weak Hash (Java)** **Deprecated Cryptography Provider (Java)** **Inadequate Salt (Java)** **Insecure Key Derivation (Java)** **Risky Cipher Algorithm (Java)** **Risky Cipher Field (Java)** **Risky Cryptographic Field (Java)** **Unsafe Base64 Encoding (Java)** **Weak Cryptographic Value (Java)** **Weak Hash Algorithm (Java**
DRD22	Do not cache sensitive information	**Sensitive Data Cached (Java)** **Sensitive Data Written to External Storage (Java)** **Sensitive Data Written to Local File (Java)**
ENV01-J	Place all security-sensitive code in a single JAR and sign and seal it	**LDAP Authentication Disabled (Java)** **Potential LDAP Poisoning (Java)**
ENV03-J	Do not grant dangerous combinations of permissions	**Accessing File in Permissive Mode (Java)** **Permissive File Mode (Java)**
ENV06-J	Production code must not contain debugging entry points	**Class Enables Debug Features (Java)** **Method Enables Debug Features (Java)**
ERR00-J	Do not suppress or ignore checked exceptions	**Empty Exception Handler (Java)**
ERR02-J	Prevent exceptions while logging data	**Debug Warning (Java)**
ERR07-J	"Do not throw RuntimeException, Exception, or Throwable"	**Broad Throws Clause (Java)**
ERR08-J	Do not catch NullPointerException or any of its ancestors	**Generic Exception Handler (Java)** **Inappropriate Exception Handler (Java)**
ERR09-J	Do not allow untrusted code to terminate the JVM	**Debug Call (Java)**
EXP00-J	Do not ignore values returned by methods	**Call Might Return Null (Java)** **Ignored Return Value (Java)**

EXP01-J	Do not use a null in a case where an object is required	**Actual Parameter Element may be null (Java)** **Field Element may be null (deep) (Java)** **Field may be null (deep) (Java)** **Null Parameter Dereference (Java)** **Null Pointer Dereference (Java)** **Null Pointer Dereference (deep) (Java)** **Return Value may Contain null Element (Java)** **Return Value may be null (Java)** **Return null Array (Java)** **Return null Boolean (Java)** **Return null Optional (Java)** **Unchecked Parameter Dereference (Java)** **Unchecked Parameter Dereference (deep) (Java)** **Unchecked Parameter Element Dereference (deep) (Java)** **null Passed to Method (deep) (Java)**
EXP02-J	Do not use the Object.equals() method to compare two arrays	**Should Use equals() Instead of == (Java)**
		equals on Array (Java)
EXP03-J	Do not use the equality operators when comparing values of boxed primitives	**Comparison to Empty String (Java)** **Should Use equals() Instead of == (Java)** **equals on Array (Java)**
EXP06-J	Expressions used in assertions must not produce side effects	**Assertion Contains Side Effects (Java)**
FIO01-J	Create files with appropriate access permissions	**Accessing File in Permissive Mode (Java)** **Permissive File Mode (Java)**
FIO02-J	Detect and handle file-related errors	**Ignored Return Value (Java)**
FIO04-J	Release resources when they are no longer needed	**Closeable Not Closed (Java)** **Closeable Not Stored (Java)**
FIO09-J	Do not rely on the write() method to output integers outside the range 0 to 255	**Call Might Return Null (Java)**
IDS00-J	Prevent SQL injection	**SQL Injection (Java)**
IDS03-J	Do not log unsanitized user input	**Tainted Log (Java)**
IDS07-J	Sanitize untrusted data passed to the Runtime.exec() method	**Command Injection (Java)**
IDS08-J	Sanitize untrusted data included in a regular expression	**Tainted Regular Expression (Java)**
IDS14-J	Do not trust the contents of hidden form fields	**Code Injection (Java)** **Command Injection (Java)** **Cross Site Scripting (Java)** **DLL Injection (Java)** **DOS Injection (Java)** **Reflection Injection (Java)** **SQL Injection (Java)** **Tainted @Trusted Value (Java)** **Tainted Bundle (Java)** **Tainted Control (Java)** **Tainted Expression Evaluation (Java)** **Tainted HTTP Response (Java)** **Tainted Hardware Device Property (Java)** **Tainted LDAP Attribute (Java)** **Tainted LDAP Filter (Java)** **Tainted Log (Java)** **Tainted Message (Java)** **Tainted Network Address (Java)** **Tainted Path (Java)** **Tainted Regular Expression (Java)** **Tainted Resource (Java)** **Tainted Session (Java)** **Tainted URL (Java)** **Tainted XAML (Java)** **Tainted XML (Java)** **Tainted Xpath (Java)**
LCK00-J	Use private final lock objects to synchronize classes that may interact with untrusted code	**Synchronization on Interned String (Java)**
LCK05-J	Synchronize access to static fields that can be modified by untrusted code	**Unguarded Method (Java)**
LCK09-J	Do not perform operations that can block while holding a lock	**Blocking in Critical Section (Java)**

LCK10-J	Use a correct form of the double-checked locking idiom	**Double-Checked Locking (Java)**
MET08-J	Preserve the equality contract when overriding the equals() method	**Asymmetric compareTo (Java)** **Missing Equals Override (Java)** **compareTo without equals (Java)** **compareTo/equals mismatch (Java)**
MET09-J	Classes that define an equals() method must also define a hashCode() method	**Defines equals but not hashCode (Java)** **Defines hashCode but not equals (Java)**
MET10-J	Follow the general contract when implementing the compareTo() method	**Missing Call to super (Java)**
MSC02-J	Generate strong random numbers	**Hardcoded Random Seed (Java)** **Insecure Random Number Generator (Java)** **Risky Cipher Field (Java)** **Risky Cryptographic Algorithm (Java)** **Risky Cryptographic Field (Java)** **Unsafe Base64 Encoding (Java)** **Weak Hash Algorithm Field (Java)**
MSC03-J	Never hard code sensitive information	**Hardcoded Password (Java)** **Sensitive Data Written to External Storage (Java)**
MSC05-J	Do not exhaust heap space	**Closeable Not Stored (Java)** **Inefficient Instantiation (Java)**
NUM00-J	Detect or prevent integer overflow	**Abs on random (Java)** **Cast: int Computation to long (Java)**
NUM12-J	Ensure conversions of numeric types to narrower types do not result in lost or misinterpreted data	**Approximate e Constant (Java)** **Approximate pi Constant (Java)** **Cast: Integer to Floating Point (Java)** **Floating Point Equality (Java)**
NUM13-J	Avoid loss of precision when converting primitive integers to floating-point	**Cast: Integer to Floating Point (Java)**
OBJ07-J	Sensitive classes must not let themselves be copied	**clone Non-cloneable (Java)** **clone Subclass of Non-clonable (Java)** **clone not final (Java)**
OBJ08-J	Do not expose private members of an outer class from within a nested class	**Inner Class Should be Static (Java)**
SEC01-J	Do not allow tainted variables in privileged blocks	**Accessing File in Permissive Mode (Java)** **Permissive File Mode (Java)**
SEC05-J	"Do not use reflection to increase accessibility of classes, methods, or fields"	**Reflection Bypasses Member Accessibility (Java)** **Reflection Modifies Member Accessibility (Java)**
SEC06-J	Do not rely on the default automatic signature verification provided by URLClassLoader and java.util.jar	**Android Message Injection (Java)** **Tainted Message (Java)**
SER00-J	Enable serialization compatibility during class evolution	**Missing Serial Version Field (Java)**
SER01-J	Do not deviate from the proper signatures of serialization methods	**Serialization Not Disabled (Java)**
SER02-J	Sign then seal objects before sending them outside a trust boundary	**Android Message Injection (Java)** **Tainted Message (Java)**
SER03-J	Do not serialize unencrypted sensitive data	**Serialization Not Disabled (Java)**
SER06-J	Make defensive copies of private mutable components during deserialization	**Serialization Not Disabled (Java)**
SER07-J	Do not use the default serialized form for classes with implementation-defined invariants	**Serialization Not Disabled (Java)**
SER10-J	Avoid memory and resource leaks during serialization	**Closeable Not Stored (Java)** **Inefficient Instantiation (Java)**
SER12-J	Prevent deserialization of untrusted data	**Serialization Not Disabled (Java)**
THI00-J	Do not invoke Thread.run()	**Synchronous Call to Thread Body (Java)**
VNA00-J	Ensure visibility when accessing shared primitive variables	**Impossible Client Side Locking (Java)** **Missing synchronized Statement (Java)** **Synchronization on static (Java)** **Unguarded Field (Java)** **Unguarded Parameter (Java)** **Useless volatile Modifier (Java)**
VNA03-J	Do not assume that a group of calls to independently atomic methods is atomic	**Useless volatile Modifier (Java)**