ERR04-J. Do not complete abruptly from a finally block

Never use `return`, `break`, `continue`, or `throw` statements within a `finally` block. When program execution enters a `try` block that has a `finally` block, the `finally` block always executes regardless of whether the `try` block (or any associated `catch` blocks) executes to normal completion. Statements that cause the `finally` block to complete abruptly also cause the `try` block to complete abruptly and consequently suppress any exception thrown from the `try` or `catch` blocks. According to *The Java Language Specification*, §14.20.2, "Execution of `try-finally` and `try-catch-finally`" [JLS 2015]:

> *If execution of the `try` block completes abruptly for any other reason R, then the `finally` block is executed. Then there is a choice:*
>
> - *If the `finally` block completes normally, then the `try` statement completes abruptly for reason R.*
> - *If the `finally` block completes abruptly for reason S, then the `try` statement completes abruptly for reason S (and reason R is discarded).*

Noncompliant Code Example

In this noncompliant code example, the `finally` block completes abruptly because of a `return` statement in the block:

```
class TryFinally {
  private static boolean doLogic() {
    try {
      throw new IllegalStateException();
    } finally {
      System.out.println("logic done");
      return true;
    }
  }
}
```

The `IllegalStateException` is suppressed by the abrupt completion of the `finally` block caused by the `return` statement.

Compliant Solution

This compliant solution removes the `return` statement from the `finally` block:

```
class TryFinally {
  private static boolean doLogic() {
    try {
      throw new IllegalStateException();
    } finally {
      System.out.println("logic done");
    }
    // Any return statements must go here;
    // applicable only when exception is thrown conditionally
  }
}
```

Exceptions

ERRO4-J-EX0: Control flow statements whose destination is within the `finally` block are perfectly acceptable. For example, the following code does not violate this rule because the `break` statement exits within the `while` loop but not within the `finally` block:

```
class TryFinally {
  private static boolean doLogic() {
    try {
      throw new IllegalStateException();
    } finally {
      int c;
      try {
        while ((c = input.read()) != -1) {
          if (c > 128) {
            break;
          }
        }
      } catch (IOException x) {
        // Forward to handler
      }
      System.out.println("logic done");
    }
    // Any return statements must go here; applicable only when exception is thrown conditionally
  }
}
```

Risk Assessment

Abrupt completion of a `finally` block masks any exceptions thrown inside the associated `try` and `catch` blocks.

Rule	Severity	Likelihood	Remediation Cost	Priority	Level
ERR04-J	Low	Probable	Medium	**P4**	**L3**

Automated Detection

Tool	Version	Checker	Description
CodeSonar	5.1p0	**PMD.Strict-Exceptions.DoNotThrowExceptionInFinally**	Do not throw exception in finally
Coverity	7.5	**PW.ABNORMAL_TERMINATION_ OF_FINALLY_BLOCK**	Implemented
Parasoft Jtest	10.3	**PB.CUB.ARCF, PB.CUB.ATSF**	
SonarQube	6.7	**S1143**	Jump statements should not occur in "finally" blocks

Related Guidelines

MITRE CWE	CWE-459, Incomplete Cleanup CWE-584, Return Inside `finally` Block

Bibliography

[Bloch 2005]	Puzzle 36. Indecision
[Chess 2007]	Section 8.2, "Managing Exceptions, The Vanishing Exception"
[JLS 2015]	§14.20.2, "Execution of `try-finally` and `try-catch-finally`"