FIO02-J. Detect and handle file-related errors

Java's file-manipulation methods often indicate failure with a return value instead of throwing an exception. Consequently, programs that ignore the return values from file operations often fail to detect that those operations have failed. Java programs must check the return values of methods that perform file I /O. This is a specific instance of EXP00-J. Do not ignore values returned by methods.

Noncompliant Code Example (`delete()`)

This noncompliant code example attempts to delete a specified file but gives no indication of its success. The Java platform requires `File.delete()` to throw a `SecurityException` only when the program lacks authorization to delete the file [API 2014]. No other exceptions are thrown, so the deletion can silently fail.

```
File file = new File(args[0]);
file.delete();
```

Compliant Solution

This compliant solution checks the return value of `delete()`:

```
File file = new File("file");
if (!file.delete()) {
  System.out.println("Deletion failed");
}
```

Compliant Solution

This compliant solution uses the `java.nio.file.Files.delete()` method from Java SE 7 to delete the file:

```
Path file = new File(args[0]).toPath();
try {
  Files.delete(file);
} catch (IOException x) {
  System.out.println("Deletion failed");
  // Handle error
}
```

The Java SE 7 Documentation [J2SE 2011] defines `Files.delete()` to throw the following exceptions:

Exception	Reason
`NoSuchFileException`	File does not exist
`DirectoryNotEmptyException`	File is a directory and could not otherwise be deleted because the directory is not empty
`IOException`	An I/O error occurs
`SecurityException`	In the case of the default provider and a security manager is installed, the `SecurityManager.checkDelete(String)` method is invoked to check delete access to the file

Because `SecurityException` is a runtime exception, it need not be declared. Because `NoSuchFileException` and `DirectoryNotExmptyExcepti` on both inherit from `IOException`, they will be caught by the compliant solution's `catch` clause.

Risk Assessment

Failure to check the return values of methods that perform file I/O can result in unexpected behavior.

Rule	Severity	Likelihood	Remediation Cost	Priority	Level
FIO02-J	Medium	Probable	Medium	P8	L2

Automated Detection

Tool	Version	Checker	Description
SonarQube	6.7	**S899**	

Related Guidelines

SEI CERT C++ Coding Standard	VOID FIO04-CPP. Detect and handle input and output errors

Bibliography

[API 2014]	`File.delete()`
[J2SE 2011]	`Files.delete()`
[Seacord 2013]	Chapter 8, "File I/O"