FIO50-J. Do not make assumptions about file creation

Although creating a file is usually accomplished with a single method call, this single action raises multiple security-related questions. What should be done if the file cannot be created? What should be done if the file already exists? What should be the file's initial attributes, such as permissions?

Java provides several generations of file-handling facilities. The original input/output facilities, which included basic file handling, are in the package `java.io`. More comprehensive facilities were included in JDK 1.4 with the New I/O package `java.nio` (see New I/O APIs [Oracle 2010b]). Still more comprehensive facilities were included in JDK 1.7 with the New I/O 2 package `java.nio.file`. Both packages introduced a number of methods to support finer-grained control over file creation.

FIO01-J. Create files with appropriate access permissions explains how to specify the permissions of a newly created file.

Noncompliant Code Example

This noncompliant code example tries to open a file for writing:

```
public void createFile(String filename)
    throws FileNotFoundException{
  OutputStream out = new FileOutputStream(filename);
  // Work with file
}
```

If the file existed before being opened, its former contents will be overwritten with the contents provided by the program.

Noncompliant Code Example (TOCTOU)

This noncompliant code example tries to avoid altering an existing file by creating an empty file using `java.io.File.createNewfile()`. If a file with the given name already exists, then `createNewFile()` will return `false` without destroying the named file's contents.

```
public void createFile(String filename)
    throws IOException{
  OutputStream out = new FileOutputStream(filename, true);
  if (!new File(filename).createNewFile()) {
      // File cannot be created...handle error
  } else {
      out = new FileOutputStream(filename);
      // Work with file
  }
}
```

Unfortunately, this solution is subject to a TOCTOU (time-of-check, time-of-use) race condition. It is possible for an attacker to modify the file system after the empty file is created but before the file is opened, such that the file that is opened is distince from the file that was created.

Compliant Solution (`Files`)

This compliant solution uses the `java.nio.file.Files.newOutputStream()` method to atomically create the file and throws an exception if the file already exists:

```
public void createFile(String filename)
    throws FileNotFoundException{
  try (OutputStream out = new BufferedOutputStream(
        Files.newOutputStream(Paths.get(filename),
                              StandardOpenOption.CREATE_NEW))) {
    // Work with out
  } catch (IOException x) {
      // File not writable...handle error
  }
}
```

Applicability

The ability to determine whether an existing file has been opened or a new file has been created provides greater assurance that only the intended file is opened or overwritten and that other files remain undisturbed.

Bibliography

[API 2013]	Class java.io.File Class java.nio.file.Files
[Oracle 2010b]	New I/O APIs