**Как создать временный каталог / папку в Java**

Неправильный пример:

“Если вы используете JDK 7, используйте новый класс [Files.createTempDirectory](http://docs.oracle.com/javase/7/docs/api/java/nio/file/Files.html" \l "createTempDirectory%28java.nio.file.Path,%20java.lang.String,%20java.nio.file.attribute.FileAttribute...%29%23createTempDirectory(java.nio.file.Path,) для создания временного каталога.

Path tempDirWithPrefix = Files.createTempDirectory(prefix);

Перед JDK 7 это должно сделать это:

public static File createTempDirectory()

throws IOException

{

final File temp;

temp = File.createTempFile("temp", Long.toString(System.nanoTime()));

if(!(temp.delete()))

{throw new IOException("Could not delete temp file: " + temp.getAbsolutePath());}

if(!(temp.mkdir()))

{throw new IOException("Could not create temp directory:"+temp.getAbsolutePath());}

return (temp);

}

“

Наивно написанный код для решения этой проблемы страдает от условий гонки. Вы могли бы использовать стороннюю библиотеку, такую ​​как Google Guava (как предлагал ответ Spina):

“В Google Guava есть множество полезных утилит. Здесь следует отметить [класс файлов](http://google.github.io/guava/releases/19.0/api/docs/com/google/common/io/Files.html). Он содержит множество полезных методов, включая:

File myTempDir = Files.createTempDir();

Это делает именно то, что вы просили в одной строке. Если вы прочтете документацию [здесь](http://google.github.io/guava/releases/19.0/api/docs/com/google/common/io/Files.html#createTempDir()), вы увидите, что предлагаемая адаптация File.createTempFile("install", "dir") обычно вводит уязвимости безопасности.”

Или вы можете написать багги-код.(ошибочный код)

Но с JDK 7 есть хорошие новости! Сама стандартная библиотека Java теперь обеспечивает надлежащим образом работающее (нерациональное) решение этой проблемы. Вы хотите [java.nio.file.Files # createTempDirectory()](http://download.oracle.com/javase/7/docs/api/java/nio/file/Files.html" \l "createTempDirectory(java.nio.file.Path,%20java.lang.String,%20java.nio.file.attribute.FileAttribute...)). Из документа :

public static Path createTempDirectory(Path dir,

String prefix,

FileAttribute<?>... attrs)

throws IOException

Создает новый каталог в указанном каталоге, используя префикс для генерации его имени. Полученный Путь связан с той же FileSystem, что и указанный каталог.

Информация о том, как создается имя каталога, зависит от реализации и поэтому не указана. По возможности префикс используется для создания имен кандидатов.

Это эффективно устраняет [неловко древний отчет об ошибках](http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=4735419) в отслеживателе ошибок Sun, который попросил именно такую ​​функцию.

Так же не используйте **deleteOnExit()**, даже если вы явно удалите его позже.

[Google "deleteonexit is evil"](http://www.google.com/search?q=deleteonexit+is+evil)для получения дополнительной информации, но суть проблемы такова:

* deleteOnExit() удаляется только при нормальных отключениях JVM, а не при сбоях или уничтожении процесса JVM.
* deleteOnExit() удаляется только при отключении JVM - не подходит для длительных серверных процессов, потому что:
* Самое злое из всех - deleteOnExit() потребляет память для каждой записи временного файла. Если ваш процесс работает в течение нескольких месяцев или создает много временных файлов за короткое время, вы потребляете память и никогда не отпускаете ее до тех пор, пока JVM не выключится.

Если вам нужен временный каталог для тестирования и вы используете **jUnit 4**, @Rule вместе с TemporaryFolder решит вашу проблему:

@Rule

public TemporaryFolder folder = new TemporaryFolder();

Из [документации](http://junit.org/apidocs/org/junit/rules/TemporaryFolder.html):

Правило TemporaryFolder позволяет создавать файлы и папки, которые гарантированно будут удалены после завершения метода тестирования (независимо от того, пройден он или нет)

Если вы используете **JUnit Jupiter** **(версия** **5.1.1 или выше**), у вас есть возможность использовать JUnit Pioneer, который является пакетом расширений JUnit 5.

Скопировано из [проектной документации](https://junit-pioneer.org/docs/temp-directory/rel=):

Например, следующий тест регистрирует расширение для одного метода теста, создает и записывает файл во временный каталог и проверяет его содержимое.

@Test

@ExtendWith(TempDirectory.class)

void test(@TempDir Path tempDir) {

Path file = tempDir.resolve("test.txt");

writeFile(file);

assertExpectedFileContent(file);

}

Maven:

<dependency>

<groupId>org.junit-pioneer</groupId>

<artifactId>junit-pioneer</artifactId>

<version>0.1.2</version>

<scope>test</scope>

</dependency>