



ANDROID ▾ CORE JAVA ▾ DESKTOP JAVA ▾ ENTERPRISE JAVA ▾ JAVA BASICS ▾ JVM LANGUAGES ▾ SOFTWARE DEVELOPMENT ▾ DEVOPS ▾

[Home](#) » [Core Java](#) » [io](#) » Java write to File Example

ABOUT NIKOS MARAVITSAS



Nikos has graduated from the Department of Informatics and Telecommunications of The National and Kapodistrian University of Athens. During his studies he discovered his interests about software development and he has successfully completed numerous assignments in a variety of fields. Currently, his main interests are system's security, parallel systems, artificial intelligence, operating systems, system programming, telecommunications, web applications, human – machine interaction and mobile development.



Java write to File Example

 Posted by: Nikos Maravitsas in [io](#) April 8th, 2014

In this example we are going to investigate several methods to write a File in Java. We are going to list some of the older, before Java 7 and NIO, methods to write to a text or a binary file as well as some of the modern ones. You are also going to notice that we use some methods to write to larger files and other methods to write to smaller files. Efficiently writing large amounts of data to files usually requires some buffering, which is not necessary for smaller files.

Ok, let's start with the older, before NIO methods.

1. Using classic IO libraries

1.1. Write files using FileWriter and FileOutputStream

Let's see the code and then analyze it:

[WriteToFileExample.java](#):

```
001 package com.javacodegeeks.core.writeToFile;
002
003 import java.io.BufferedWriter;
004 import java.io.File;
005 import java.io.FileNotFoundException;
006 import java.io.FileOutputStream;
007 import java.io.FileWriter;
008 import java.io.IOException;
009 import java.io.OutputStream;
010 import java.io.OutputStreamWriter;
011 import java.io.Writer;
012 import java.util.ArrayList;
013 import java.util.List;
014
015 public class WriteToFileExample {
016
017     private static final String FILEPATH = "C:\\Users\\nikos\\Desktop\\TestFiles\\testFile.txt";
018
019     public static void main(String[] args) throws IOException {
020
021         String str1 = "abc";
022         String str2 = "asdafasfasfa";
023
024         List<String> list = new ArrayList<String>();
025         list.add(str1);
026         list.add(str2);
027
028         //useBufferedFileOutPutStream(list, FILEPATH);
029         useFileOutPutStream(str1, FILEPATH);
030     }
031 }
```

Get paid for doing what y

LE

NEWSLETTER

170,992 insiders are already e
weekly updates and complimentary
whitepapers!

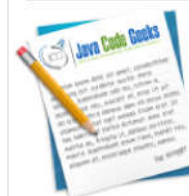
Join them now to gain **EX**
ACCESS to the latest news in the :
as well as insights about Android, S
Groovy and other related technolog

Email address:

Your email address

Sign up

JOIN US



With **1,240,1**
unique visitor:
500 authors
placed among
related sites a
Constantly be
lookout for pe
encourage yo
So If you hav
unique and interesting content then y
check out our **JCG** partners program.
be a **guest writer** for Java Code Gee
your writing skills!

```

031 }
032
033 /**
034  * Write a small string to a File - Use a FileWriter
035  */
036 public static void useFileWriter(String content, String filePath) {
037     Writer writer = null;
042     writer.write(content);
043
044     } catch (IOException e) {
045         System.err.println("Error writing the file : ");
046         e.printStackTrace();
047
048     } finally {
049
050         if (writer != null) {
051             try {
052                 writer.close();
053             } catch (IOException e) {
054
055                 System.err.println("Error closing the file : ");
056                 e.printStackTrace();
057             }
058         }
059     }
060 }
061
062 }
063
064 /**
065  * Write a big list of Strings to a file - Use a BufferedWriter
066  */
067 public static void useByfferedFileWriter(List<String> content,
068     String filePath) {
069
070     File file = new File(filePath);
071     Writer fileWriter = null;
072     BufferedWriter bufferedWriter = null;
073
074     try {
075
076         fileWriter = new FileWriter(file);
077         bufferedWriter = new BufferedWriter(fileWriter);
078
079         // Write the lines one by one
080         for (String line : content) {
081             line += System.getProperty("line.separator");
082             bufferedWriter.write(line);
083
084             // alternatively add bufferedWriter.newLine() to change line
085         }
086
087     } catch (IOException e) {
088         System.err.println("Error writing the file : ");
089         e.printStackTrace();
090     } finally {
091
092         if (bufferedWriter != null && fileWriter != null) {
093             try {
094                 bufferedWriter.close();
095                 fileWriter.close();
096             } catch (IOException e) {
097                 e.printStackTrace();
098             }
099         }
100     }
101
102 }
103
104 /**
105  * Write raw data to a small file - use FileOutputStream
106  */
107 public static void useFileOutPutStream(String content, String filePath) {
108
109     OutputStream outputStream = null;
110
111     try {
112
113         outputStream = new FileOutputStream(new File(filePath));
114         outputStream.write(content.getBytes(), 0, content.length());
115
116     } catch (FileNotFoundException e) {
117         System.err.println("Error Opening the file : ");
118         e.printStackTrace();
119     } catch (IOException e) {
120         System.err.println("Error writing the file : ");
121         e.printStackTrace();
122     } finally {
123
124         if (outputStream != null) {
125             try {
126                 outputStream.close();
127             } catch (IOException e) {
128                 e.printStackTrace();
129             }
130         }
131     }
132
133 }
134

```

GET THE JAVA SKILLS YOU NEED IN 2016.

Start Learning N

Viêm Amidar

Biến Chứng Gây N
Khuẩn Huyết Ngực
Khi Ngủ Gấp Chuy
Tư Vấn

tuvan.dakhoathaibinhduong.vn



CAREER OPPORTUNITIES

Junior Software Developer- Cloud for !
JobSAP
Palo Alto, CA
Oct, 26
Engineer-JavaAmerican Express
Arizona
Oct, 26
Junior Developer- Core Mobile SDK Joi
Dublin, CA
Oct, 25
Database Developer - Server-Side Javi
Stanley
New York, NY
Sep, 02
JAVA DEVELOPERBirlaSoft
New Jersey
Oct, 19
Back-End DeveloperGE Digital
Michigan
Oct, 19
Software Engineer - Back EndGE Digi
Michigan
Oct, 21
Entry Level Software EngineerFord M
Company
Dearborn, MI
Oct, 08
Senior Java DeveloperBarclays
New York, NY
Oct, 17
Java DeveloperConfiguration Mana
Englewood Cliffs, NJ
Oct, 25

1 2 ... 5955 »

Filter Results

jobs by indeed

```

135
136 /**
137  * Write character data to a big file - use BufferedWriter
138  */
139 public static void useBufferedFileOutputStream(List<String> content, String filePath) {
140     Writer writer = null;
141

```

GET THE JAVA SKILLS YOU NEED IN 2016.

[Start Learning Now](#)

```

146         new FileOutputStream(filePath, "utf-8"));
147
148         for (String line : content) {
149             line += System.getProperty("line.separator");
150             writer.write(line);
151         }
152     } catch (IOException e) {
153     } finally {
154         if (writer != null) {
155             try {
156                 writer.close();
157             } catch (Exception e) {
158             }
159         }
160     }
161 }
162
163 /**
164  * Write raw data to a big file - use BufferedOutputStream
165  */
166 public static void useBufferedOutPutStream(List<String> content, String filePath) {
167     BufferedOutputStream bout = null;
168     try {
169
170         bout = new BufferedOutputStream( new FileOutputStream(filePath) );
171
172         for (String line : content) {
173             line += System.getProperty("line.separator");
174             bout.write(line.getBytes());
175         }
176     } catch (IOException e) {
177     } finally {
178         if (bout != null) {
179             try {
180                 bout.close();
181             } catch (Exception e) {
182             }
183         }
184     }
185 }
186
187 }
188
189 }
190
191 }
192
193 }
194
195 }
196 }

```

In the above example we basically use two different methods:

FileWriter

to write to character/text files.

FileOutputStream

to write raw data.

In order to buffer the writes of the above classes we use a

BufferedWriter

for character streams and

BufferedOutputStream

for raw data streams. With

BufferedWriter

, we simply use an internal buffer to pack the data we want to write and reduce real IO operations, which essentially saves time. So as you can imagine, this is very useful when creating a write-intensive application that writes large amounts of data to files.

BufferedWriter

is generally created with the default buffer size, which is sufficient for most applications. If you want you can specify the size of the internal buffer using this constructor instead :

BufferedWriter(Writer out, int sz)

where

```
sz
```

is the size of the buffer in bytes.

GET THE JAVA SKILLS YOU NEED IN 2016.

Start Learning N

to wrap

```
FileOutputStream
```

. Furthermore when writing characters to a file without buffering, on every

```
write
```

invocation the system will perform a conversion from characters to bytes. Buffering will also reduce the amount of conversions performed.

1.2. Using RandomAccessFile

```
RandomAccessFile
```

was introduced to support random accessed to files. As the javadoc says "a random access file behaves like a large array of bytes stored in the file system". It is a very convenient class to write and also read files in arbitrary positions.

Let's see how :

WriteToFileNIO.java:

```
01 public static void writeWithRandomAccessFile( String content, String filePath) {
02
03     try (RandomAccessFile randomAccessFile = new RandomAccessFile(new File(filePath), "rw")) {
04
05         // move the cursor to the end of the file
06         // you can move the cursor to any position inside the file to write at random positions
07         randomAccessFile.seek(randomAccessFile.length());
08
09         randomAccessFile.write(content.getBytes());
10
11         // alternatively you can use randomAccessFile.writeChars(content)
12         // or randomAccessFile.writeUTF(content);
13     } catch (IOException e) {
14         e.printStackTrace();
15     }
16 }
```

As you can see we open a

```
RandomAccessFile
```

in read-write mode. Then, we simply move the cursor to the end of the file to append the new content (using

```
seek
```

method) and we simply write some bytes to it with its

```
write
```

method. Note that

```
RandomAccessFile
```

was introduced in JDK1.0.

That's it. Now let's take a see how can write to files in Java using modern NIO classes.

2. Write files using NIO

NIO introduced several classes that made File manipulation easier and highly efficient. We are also going to introduce here the try-with resources syntax.

Let's see the code :

2.1 Using java.nio.file.Files class

WriteToFileNIO.java:

```
001 package com.javacodegeeks.core.writeToFile;
002
003 import java.io.BufferedOutputStream;
```

```

004 import java.io.BufferedWriter;
005 import java.io.IOException;
006 import java.io.OutputStream;
007 import java.io.UnsupportedEncodingException;
008 import java.nio.charset.Charset;
009 import java.nio.file.Files;
010 import java.nio.file.Path;

```

GET THE JAVA SKILLS YOU NEED IN 2016.

[Start Learning N](#)

```

011 public class WriteToFileNIO {
012
013     private static final String FILEPATH = "C:\\Users\\nikos\\Desktop\\TestFiles\\testFile.txt";
014
015     public static void main(String[] args) throws IOException {
016
017         String str1 = "abc";
018         String str2 = "aipcipasinainainsovsdvweviasbdoviuabsudviuadv";
019
020         List<String> list = new ArrayList<String>();
021         list.add(str1);
022         list.add(str2);
023
024         bufferedWrite(list, FILEPATH);
025     }
026
027     /**
028      * Write a small string to a File - Use a FileWriter
029      */
030     public static void simpleWrite(String content, String filePath) {
031         Path fileP = Paths.get(filePath);
032         try {
033             Files.write(fileP, content.getBytes("utf-8"));
034         } catch (UnsupportedEncodingException e) {
035             e.printStackTrace();
036         } catch (IOException e) {
037             e.printStackTrace();
038         }
039     }
040
041     /**
042      * Write a big list of Strings to a file - Use a BufferedWriter
043      */
044     public static void bufferedWrite(List<String> content, String filePath) {
045         Path fileP = Paths.get(filePath);
046         Charset charset = Charset.forName("utf-8");
047         try (BufferedWriter writer = Files.newBufferedWriter(fileP, charset)) {
048             for (String line : content) {
049                 writer.write(line, 0, line.length());
050                 writer.newLine();
051             }
052         } catch (IOException e) {
053             e.printStackTrace();
054         }
055     }
056
057     /**
058      * Write raw data to file - use OutputStream
059      */
060     public static void writeWithOutputStream(String content, String filePath) {
061         Path fileP = Paths.get(filePath);
062         try (OutputStream outputStream = Files.newOutputStream(fileP)) {
063             outputStream.write(content.getBytes());
064         } catch (IOException e) {
065             e.printStackTrace();
066         }
067     }
068
069     /**
070      * Write raw data to file using BufferedOutputStream
071      */
072     public static void writeWithBufferedOutputStream(List<String> content, String filePath) {
073         Path fileP = Paths.get(filePath);
074         try (BufferedOutputStream outputStream = new BufferedOutputStream(Files.newOutputStream(fileP))) {
075             for (String line : content) {
076                 outputStream.write(line.getBytes());
077             }
078         } catch (IOException e) {
079             e.printStackTrace();
080         }
081     }
082
083     /**
084      * Write a string list to a File
085      */
086     public static void simpleWriteListOfString(List<String> content, String filePath) {
087         Path fileP = Paths.get(filePath);
088         Charset charset = Charset.forName("utf-8");
089     }
090
091
092
093
094
095
096
097
098
099
100
101
102
103
104
105
106
107

```

```

108     try {
109         Files.write(fileP, content, charset);
110     } catch (UnsupportedEncodingException e) {
111         e.printStackTrace();
112     } catch (IOException e) {
113     }
114 }

```

GET THE JAVA SKILLS YOU NEED IN 2016.

Start Learning N

As you can see things are much simpler because of the new NIO

Files

class and the new try-with resource syntax. In try-with resource syntax, when you open a resource in the

try()

clause, the resource will be automatically closed when the flow of the program exits the

try

region. You don't have to create the

finally

block to release the resources as we did previously.

2.2. Using FileChannel

Now let's see how you can use

FileChannel

, which is a very interesting class of the NIO package. It basically connects a channel of bytes to a file and enables both reading and writing from/to files. You can view it as an alternative to

FileOutputStream

. A major difference is that a

FileChannel

connects an allocated byte buffer to the file and it holds the current position of the cursor in the file.

WriteToFileNIO.java:

```

01 public static void writeWithFileChannel(String content, String filePath) {
02     try (RandomAccessFile randomAccessFile = new RandomAccessFile(new File(filePath), "rw")) {
03         // move the cursor to the end of the file
04         randomAccessFile.seek(randomAccessFile.length());
05         // obtain the a file channel from the RandomAccessFile
06         try (FileChannel fileChannel = randomAccessFile.getChannel()) {
07             ByteBuffer buf = ByteBuffer.allocate(512);
08             buf.clear();
09             buf.put(content.getBytes());
10             buf.flip();
11             while (buf.hasRemaining()) {
12                 fileChannel.write(buf);
13             }
14         } catch (IOException e) {
15             e.printStackTrace();
16         }
17     } catch (IOException e) {
18         e.printStackTrace();
19     }
20 }
21 }
22 }
23 }
24 }
25 }
26 }
27 }
28 }
29 }

```

As you can see we first create a

RandomAccessFile

and obtain a

FileChannel

from it. Then, we allocate a

ByteBuffer

of 512 bytes. Finally we write the contents of the byte buffer to the file. For performance reasons, it's not guaranteed that the buffer will be written in its entirety to the file in a single

```
write
```

operation. That's why we've used the while loop, so as long as the buffer has remaining bytes in it, we simply append them to the file.

GET THE JAVA SKILLS YOU NEED IN 2016.

Start Learning N

2.3. Using FileChannel and Memory Mapped File

In the following methods we are also going to use a

```
MappedByteBuffer
```

. This is a direct byte buffer that maps a memory region to a file region.

Let's see how you do this :

WriteToFileNIO.java:

```
01 public static void writeWithMemMappedBuffer(String content, String filePath) {
02
03     try (RandomAccessFile randomAccessFile = new RandomAccessFile(new File(filePath), "rw")) {
04
05         // move the cursor to the end of the file
06         randomAccessFile.seek(randomAccessFile.length());
07
08         // obtain the a file channel from the RandomAccessFile
09         try (FileChannel fileChannel = randomAccessFile.getChannel()) {
10
11             // Map a content.getBytes().length byte region of the file to this memory buffer
12             MappedByteBuffer memoryMappedbuffer = fileChannel.map(FileChannel.MapMode.READ_WRITE,
fileChannel.position(), content.getBytes().length);
13
14             memoryMappedbuffer.put(content.getBytes());
15
16         } catch (IOException e) {
17             e.printStackTrace();
18         }
19
20     } catch (IOException e) {
21         e.printStackTrace();
22     }
23 }
```

You can use this for performance sensitive applications. As you can see, there is no need to explicitly write the buffer to the file, that is something that the underlying systems does when it deems its necessary. You only have to manipulate the buffer, and the changes will be reflected to the file. Of course, reads are very efficient using this technique as well.

2.4. FileChannel and Direct Memory Access

There is an excellent article from IBM that describes Efficient data transfer through zero copy. In this case we are going to use

```
transferTo()/transferFrom()
```

method of

```
FileChannel
```

class. It's basic characteristics is that the it relies on the underlying system to access its DMA (Direct Memory Access) infrastructure. It might not work in all operating systems, but most modern ones offer such capabilities. What happens is that data are transferred directly from/to disc to the bus, avoiding CPU copies.

It's recommended to use that technique to transfer data from one source channel to another destination channel, e.g from a file to another file, from a file to a socket, from a database to a file and so on. But we are going to show you how to transfer data from a

```
String
```

to a

```
FileChannel
```

. We are going to consider the

```
String
```

as an

```
InputStream
```

from a data source (think hundreds of MB or GB long strings).

WriteToFileNIO.java:

```
01 package com.javacodegeeks.core.writeToFile;
02
03 import java.io.ByteArrayInputStream;
04 import java.io.File;
```



```

05 import java.io.IOException;
06 import java.io.InputStream;
07 import java.io.RandomAccessFile;
08 import java.nio.channels.Channels;
09 import java.nio.channels.FileChannel;
10 import java.nio.channels.ReadableByteChannel;
11 import java.nio.charset.Charset;

```

GET THE JAVA SKILLS YOU NEED IN 2016.

[Start Learning N](#)

```

16
17 public static void main(String[] args) throws IOException {
18
19     String str2 = "aipcipasinainainsovsdvweiasbdoviuabsudviuadv";
20
21     long numBytes = str2.getBytes().length;
22
23     // obtain an inputstream from the string
24     InputStream inputStream = new ByteArrayInputStream(str2.getBytes(Charset.forName("UTF-8")));
25
26     writeWithFileChannerDMA(inputStream, FILEPATH, numBytes);
27
28 }
29
30 /**
31  * Write to a file using a FileChanel and DMA
32  */
33 public static void writeWithFileChannerDMA(InputStream inputStream, String outputFile, long count) {
34
35     try (RandomAccessFile randomAccessFile = new RandomAccessFile(new File(
36         outputFile, "rw")) {
37
38         // move the cursor to the end of the file
39         randomAccessFile.seek(randomAccessFile.length());
40
41         // obtain the a file channel from the RandomAccessFile
42         try {
43             FileChannel fileChannel = randomAccessFile.getChannel();
44             ReadableByteChannel inputChannel = Channels.newChannel(inputStream);
45
46         } {
47             fileChannel.transferFrom(inputChannel, 0, count);
48
49         } catch (IOException e) {
50             e.printStackTrace();
51         }
52     } catch (IOException e) {
53         e.printStackTrace();
54     }
55 }
56
57 }
58

```

So, in the above example you've seen 4 major things:

1. In

main

, we've used a

ByteArrayInputStream

to obtain an input stream from a

String

.

2. We've used

Channels.newChannel

to obtain an

ReadableByteChannel

from the

InputStream

.

3. You can also see how to open multiple resources in one

try

clause.

4. We've used

transferFrom

to transfer data from one channel to the other. Hoping that the underlying system will offer DMA infrastructure, data can be transferred directly to the file system without any copying in between.

That was it! A lot of solutions there to suit every need. I hope you find this guide useful.

Download the Source Code

GET THE JAVA SKILLS YOU NEED IN 2016.

Start Learning N

Tagged with:

BUFFEREDOUTPUTSTREAM

BUFFEREDWRITER

BYTEBUFFER

DMA

FILECHANNEL

FILEOUTPUTSTREAM

FILEWRITER

MAPPEDBYTEBUFFER

RANDOMACCESSFILE

Do you want to know how to develop your skillset to become a **Java Rockstar**?

Subscribe to our newsletter to start Rocking right now!

To get you started we give you our best selling eBooks for **FREE!**

1. JPA Mini Book
2. JVM Troubleshooting Guide
3. JUnit Tutorial for Unit Testing
4. Java Annotations Tutorial
5. Java Interview Questions
6. Spring Interview Questions
7. Android UI Design

and many more

Email address:

Your email address

Sign up

KNOWLEDGE BASE

Courses

News

Resources

Tutorials

Whitepapers

THE CODE GEEKS NETWORK

.NET Code Geeks

Java Code Geeks

System Code Geeks

Web Code Geeks

HALL OF FAME

Android Alert Dialog Example

Android OnClickListener Example

How to convert Character to String and a String to Character Array in Java

Java Inheritance example

Java write to File Example

java.io.FileNotFoundException – How to solve File Not Found Exception

java.lang.arrayindexoutofboundsexception – How to handle Array Index Out Of Bounds Exception

java.lang.NoClassDefFoundError – How to solve No Class Def Found Error

JSON Example With Jersey + Jackson

ABOUT JAVA CODE GEEKS

JCGs (Java Code Geeks) is an independent online community focused on creating the ultimate Java to Java developers resource center; targeted at the technical a technical team lead (senior developer), project manager and junior developer. JCGs serve the Java, SOA, Agile and Telecom communities with daily news w domain experts, articles, tutorials, reviews, announcements, code snippets and source projects.

DISCLAIMER

All trademarks and registered trademarks appearing on Java Code Geeks are property of their respective owners. Java is a trademark or registered trademark of Oracle Corporation in the United States and other countries. Examples Java Code Geeks is not connected to Oracle Corporation and is not sponsored by Oracle Corporation.

GET THE JAVA SKILLS YOU NEED IN 2016.

[Start Learning N](#)

