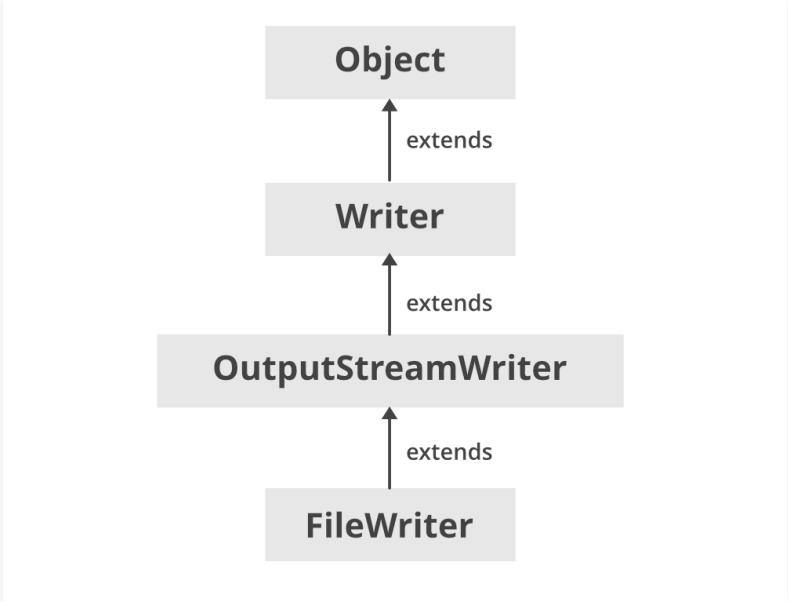
FileWriter Class in Java

Difficulty Level : Easy Last Updated : 10 Feb, 2022

Java FileWriter class of java.io package is used to write **data in character** form to file. Java FileWriter class is used to write character-oriented data to a file. It is a character-oriented class that is used for file handling in java.

- This class inherits from OutputStreamWriter class which in turn inherits from the Writer class.
- The constructors of this class assume that the default character encoding and the default byte-buffer size are acceptable. To specify these values yourself, construct an OutputStreamWriter on a FileOutputStream.
- FileWriter is meant for writing streams of characters. For writing streams of raw bytes, consider using a FileOutputStream.
- FileWriter creates the output file if it is not present already.

Hierarchy of Java FileWriter Class



FileWriter extends OutputStreamWriter and Writer classes. It implements Closeable, Flushable, Appendable, AutoCloseable interfaces.

Java FileWriter Class Declaration

```
public class FileWriter extends OutputStreamWriter
```

Constructors of FileWriter Class

1. FileWriter(File file): It constructs a FileWriter object given a File object. It throws an **IOException** if the file exists but is a directory rather than a regular file does not exist but cannot be created, or cannot be opened for any other reason.

```
FileWriter fw = new FileWriter(File file);
```

2. FileWriter(File file, boolean append): It constructs a FileWriter object given a File object. If the second argument is true, then bytes will be written to the end of the file rather than the beginning. It throws an **IOException** if the file exists but is a directory rather than a regular file or does not exist but cannot be created, or cannot be opened for any other reason.

```
FileWriter fw = new FileWriter(File file, boolean append);
```

3. FileWriter(FileDescriptor fd): It constructs a FileWriter object associated with a file descriptor.

```
FileWriter fw = new FileWriter(FileDescriptor fd);
```

4. FileWriter(File file, Charset charset): It constructs the fileWriter when file and charset is given.

```
FileWriter fw = new FileWriter(File file, Charset charset);
```

5. FileWriter(File file, Charset charset, boolean append): It constructs the fileWriter when file and charset is given and a boolean indicating whether to append the data written or not.

```
FileWriter fw = new FileWriter(File file, Charset charset, boolean append);
```

6. FileWriter(String fileName): It constructs a FileWriter object given a file name.

```
FileWriter fw = new FileWriter(String fileName);
```

7. FileWriter(String fileName, Boolean append): It constructs a FileWriter object given a file name with a Boolean indicating whether or not to append the data written.

```
FileWriter fw = new FileWriter(String fileName, Boolean append);
```

8. FileWriter(String fileName, Charset charset): It constructs a FileWriter when a fileName and charset is given.

```
FileWriter fw = new FileWriter(String fileName, Charset charset);
```

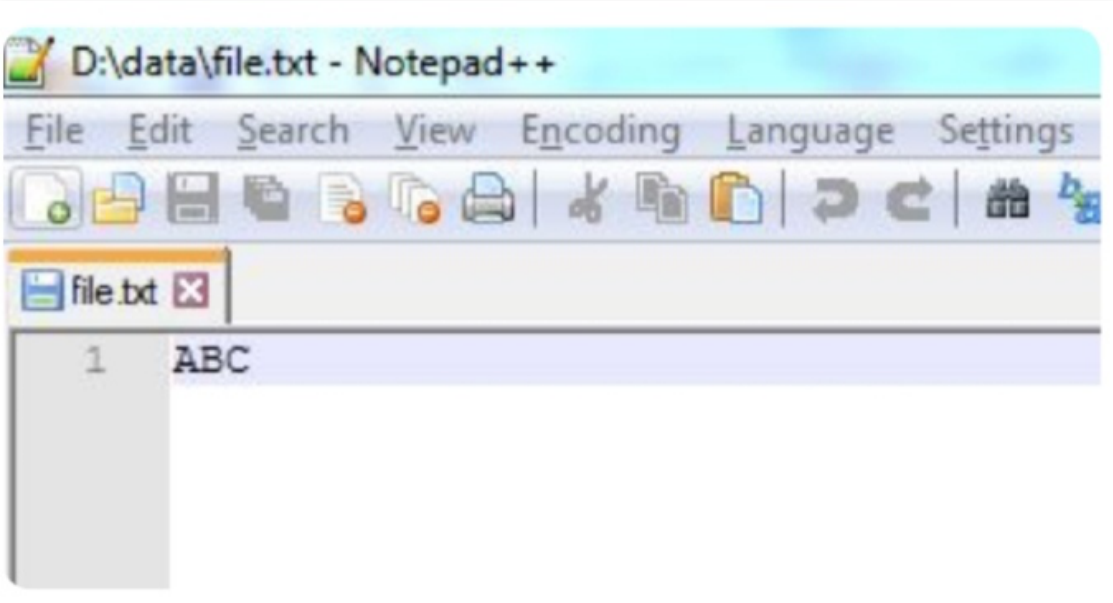
9. FileWriter(String fileName, Charset charset, boolean append): It constructs a fileWriter when a fileName and a charset are given and a boolean variable indicating whether to append data or not.

```
FileWriter fw = new FileWriter(String fileName, Charset charset, boolean appe
```

Example:

```
// Java program to create a text File using FileWriter

import java.io.FileWriter;
import java.io.IOException;
import java.util.*;
class GFG {
    public static void main(String[] args)
        throws IOException
    {
        // initialize a string
        String str = "ABC";
        try {

            // attach a file to FileWriter
            FileWriter fw
                = new FileWriter("D:/data/file.txt");

            // read each character from string and write
            // into FileWriter
            for (int i = 0; i < str.length(); i++)
                fw.write(str.charAt(i));

            System.out.println("Successfully written");

            // close the file
            fw.close();
        }
        catch (Exception e) {
            e.getStackTrace();
        }
    }
}
```

Output:



Overwriting vs Appending the File

While creating a Java FileWriter, we can decide whether we want to append the file to an existing file, or we want to overwrite any existing file. This can be decided by choosing the appropriate constructor. The constructor for **overwriting** any existing file takes only **one parameter that is a file name.**

```
Writer fileWriter = new FileWriter("c:\\data\\output.txt");
```

The constructor for **appending the file** or overwriting the file, takes **two parameters, the file name and a boolean variable** which decides whether to append or overwrite the file

```
Writer fileWriter = new FileWriter("c:\\data\\output.txt", true);  // appends
Writer fileWriter = new FileWriter("c:\\data\\output.txt", false); // overwri
```

Basic Methods of FileWriter Class

1. Write()

- **write(int a):** This method writes a single character specified by int a.
- **write(String str, int pos, int length):** This method writes a portion of the string from position **pos** until the **length** number of characters.
- **write(char ch[], int pos, int length):** This method writes the position of characters from array ch[] from position **pos** till **length** number of characters

ch[] from position **pos** till **length** number of characters.

- **write(char ch[]):** This method writes an array of characters specified by ch[].
- **write(String st):** This method writes a string value specified by 'st' into the file.

```
// Java program to write text to file

import java.io.FileWriter;

public class GFG {

    public static void main(String args[])
    {

        String data = "Welcome to gfg";

        try {
            // Creates a FileWriter
            FileWriter output
                = new FileWriter("output.txt");

            // Writes the string to the file
            output.write(data);

            // Closes the writer
            output.close();
        }

        catch (Exception e) {
            e.getStackTrace();
        }
    }
}
```

Output:

```
The file output.txt will contain "Welcome to gfg" text.
```

2. getEncoding()

This method is used to get the type of encoding that is used for writing the data.

```
// java program to show the usage
// of getEncoding() function

import java.io.FileWriter;
import java.nio.charset.Charset;

class Main {
    public static void main(String[] args)
    {

        String file = "output.txt";

        try {
            // Creates a FileReader with default encoding
            FileWriter o1 = new FileWriter(file);

            // Creates a FileReader specifying the encoding
            FileWriter o2 = new FileWriter(
                file, Charset.forName("UTF11"));

            // Returns the character encoding of the reader
            System.out.println("Character encoding of o1: "
                               + o1.getEncoding());
            System.out.println("Character encoding of o2: "
                               + o2.getEncoding());

            // Closes the reader
            o1.close();
            o2.close();
        }

        catch (Exception e) {
            e.getStackTrace();
        }
    }
}
```

Output:

```
The character encoding of output1: Cp1253
The character encoding of output2: UTF11
```

In the above example, we have created 2 file writer named output1 and output2.

- **output1:** does not specify the character encoding. Hence, the getEncoding() method returns the default character encoding.
- **output2:** specifies the character encoding, **UTF11**. Hence, the getEncoding() method returns

the specified character encoding.

3. close() method:

After finishing writing characters to a FileWriter, we should close it. And this is done by calling the close() method.

```
try {
    // Creates a FileReader with default encoding
    FileWriter o1 = new FileWriter(file);

    // Creates a FileReader specifying the encoding
    FileWriter o2 = new FileWriter(file, Charset.forName("UTF11"));

    // Returns the character encoding of the reader
    System.out.println("Character encoding of o1: " + o1.getEncoding());
    System.out.println("Character encoding of o2: " + o2.getEncoding());

    // Closes the FileWriter
    o1.close();
    o2.close();
  }
```

FileWriter vs FileOutputStream

- FileWriter writes streams of characters while FileOutputStream is meant for writing streams of raw bytes.
- FileWriter deals with the character of 16 bits while on the other hand, FileOutputStream deals with 8-bit bytes.
- FileWriter handles Unicode strings while FileOutputStream writes bytes to a file and it does not accept characters or strings and therefore for accepting strings, it needs to be wrapped up with OutputStreamWriter.

Methods of FileWriter Class

S. No.	Method	Description
1.	void write(String text)	It is used to write the string into FileWriter.
2.	void write(char c)	It is used to write the char into FileWriter.

S. No.	Method	Description
3.	void write(char[] c)	It is used to write a char array into FileWriter.
4.	void flush()	It is used to flushes the data of FileWriter.
5.	void close()	It is used to close the FileWriter.

Methods of OutputStreamWriter Class

S. No.	Method	Description
1.	flush()	Flushes the stream.
2.	getEncoding()	Returns the name of the character encoding being used by this stream.
3.	write(char[] cbuf, int off, int len)	Writes a portion of an array of characters.
4.	write(int c)	Writes a single character.
5.	write(String str, int off, int len)	Writes a portion of a string.

Methods of Writer Class

S. No.	Method	Description
1.	append(char c)	Appends the specified character to this writer.
2.	append(CharSequence csq)	Appends the specified character sequence to this writer.
3.	append(CharSequence csq, int start, int end)	Appends a subsequence of the specified character sequence to this writer.
4.	close()	Closes the stream, flushing it first.

S. No.	Method	Description
5.	nullWriter()	Returns a new Writer which discards all characters.
6.	write(char[] cbuf)	Writes an array of characters.
7.	write(String str)	Writes a string.