**Java Writer**

It is an [abstract](https://www.javatpoint.com/abstract-class-in-java) class for writing to character streams. The methods that a subclass must implement are write(char[], int, int), flush(), and close(). Most subclasses will override some of the methods defined here to provide higher efficiency, functionality or both.

**Fields**

|  |  |  |
| --- | --- | --- |
| **Modifier and Type** | **Field** | **Description** |
| protected Object | lock | The object used to synchronize operations on this stream. |

**Constructor**

|  |  |  |
| --- | --- | --- |
| **Modifier** | **Constructor** | **Description** |
| protected | Writer() | It creates a new character-stream writer whose critical sections will synchronize on the writer itself. |
| protected | Writer(Object lock) | It creates a new character-stream writer whose critical sections will synchronize on the given [object](https://www.javatpoint.com/object-and-class-in-java). |

**Methods**

|  |  |  |
| --- | --- | --- |
| Modifier and Type | **Method** | **Description** |
| Writer | append(char c) | It appends the specified character to this writer. |
| Writer | append(CharSequence csq) | It appends the specified character sequence to this writer |
| Writer | append(CharSequence csq, int start, int end) | It appends a subsequence of the specified character sequence to this writer. |
| abstract void | close() | It closes the stream, flushing it first. |
| abstract void | flush() | It flushes the stream. |
| void | write(char[] cbuf) | It writes an [array](https://www.javatpoint.com/array-in-java) of characters. |
| abstract void | write(char[] cbuf, int off, int len) | It writes a portion of an array of characters. |
| void | write(int c) | It writes a single character. |
| void | write(String str) | It writes a [string](https://www.javatpoint.com/java-string). |
| void | write(String str, int off, int len) | It writes a portion of a string. |

**Java Writer Example**

import java.io.\*;

public class WriterExample {

    public static void main(String[] args) {

        try {

            Writer w = new FileWriter("output.txt");

            String content = "I love my country";

            w.write(content);

            w.close();

            System.out.println("Done");

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

}

Output:

Done

output.txt:

I love my country

**Java Reader**

[Java](https://www.javatpoint.com/java-tutorial) Reader is an [abstract class](https://www.javatpoint.com/abstract-class-in-java) for reading character [streams](https://www.javatpoint.com/java-8-stream). The only methods that a subclass must implement are read(char[], int, int) and close(). Most subclasses, however, will [override](https://www.javatpoint.com/method-overriding-in-java) some of the methods to provide higher efficiency, additional functionality, or both.

Some of the implementation [class](https://www.javatpoint.com/object-class) are [BufferedReader](https://www.javatpoint.com/java-bufferedreader-class), [CharArrayReader](https://www.javatpoint.com/java-chararrayreader-class), [FilterReader](https://www.javatpoint.com/java-filterreader-class), [InputStreamReader](https://www.javatpoint.com/Input-from-keyboard-by-InputStreamReader), PipedReader, [StringReader](https://www.javatpoint.com/java-stringreader-class)

**Fields**

|  |  |  |
| --- | --- | --- |
| **Modifier and Type** | **Field** | **Description** |
| protected Object | lock | The object used to synchronize operations on this stream. |

**Constructor**

|  |  |  |
| --- | --- | --- |
| [**Modifie**](https://www.javatpoint.com/access-modifiers)**r** | [**Constructor**](https://www.javatpoint.com/java-constructor) | **Description** |
| protected | Reader() | It creates a new character-stream reader whose critical sections will synchronize on the reader itself. |
| protected | Reader(Object lock) | It creates a new character-stream reader whose critical sections will synchronize on the given object. |

**Methods**

|  |  |  |
| --- | --- | --- |
| **Modifier and Type** | **Method** | **Description** |
| abstract void | close() | It closes the stream and releases any system resources associated with it. |
| void | mark(int readAheadLimit) | It marks the present position in the stream. |
| Boolean | markSupported() | It tells whether this stream supports the mark() operation. |
| int | read() | It reads a single character. |
| int | read(char[] cbuf) | It reads characters into an [array](https://www.javatpoint.com/array-in-java). |
| abstract int | read(char[] cbuf, int off, int len) | It reads characters into a portion of an array. |
| int | read(CharBuffer target) | It attempts to read characters into the specified character buffer. |
| Boolean | ready() | It tells whether this stream is ready to be read. |
| void | reset() | It resets the stream. |
| long | skip(long n) | It skips characters. |

**Example**

import java.io.\*;

public class ReaderExample {

    public static void main(String[] args) {

        try {

            Reader reader = new FileReader("file.txt");

            int data = reader.read();

            while (data != -1) {

                System.out.print((char) data);

                data = reader.read();

            }

            reader.close();

        } catch (Exception ex) {

            System.out.println(ex.getMessage());

        }

    }

}

file.txt:

I love my country

Output:

I love my country

**Java FileWriter Class**

Java FileWriter class is used to write character-oriented data to a [file](https://www.javatpoint.com/java-file-class). It is character-oriented class which is used for file handling in [java](https://www.javatpoint.com/java-tutorial).

Unlike FileOutputStream class, you don't need to convert string into byte [array](https://www.javatpoint.com/array-in-java) because it provides method to write string directly.

**Java FileWriter class declaration**

Let's see the declaration for Java.io.FileWriter class:

1. public class FileWriter extends OutputStreamWriter

**Constructors of FileWriter class**

|  |  |
| --- | --- |
| **Constructor** | **Description** |
| FileWriter(String file) | Creates a new file. It gets file name in [string](https://www.javatpoint.com/java-string). |
| FileWriter(File file) | Creates a new file. It gets file name in File [object](https://www.javatpoint.com/object-and-class-in-java). |

**Methods of FileWriter class**

|  |  |
| --- | --- |
| **Method** | **Description** |
| void write(String text) | It is used to write the string into FileWriter. |
| void write(char c) | It is used to write the char into FileWriter. |
| void write(char[] c) | It is used to write char array into FileWriter. |
| void flush() | It is used to flushes the data of FileWriter. |
| void close() | It is used to close the FileWriter. |

**Java FileWriter Example**

In this example, we are writing the data in the file testout.txt using Java FileWriter class.

package com.javatpoint;

import java.io.FileWriter;

public class FileWriterExample {

    public static void main(String args[]){

         try{

           FileWriter fw=new FileWriter("D:\\testout.txt");

           fw.write("Welcome to javaTpoint.");

           fw.close();

          }catch(Exception e){System.out.println(e);}

          System.out.println("Success...");

     }

}

Output:

Success...

testout.txt:

Welcome to javaTpoint.

**Java FileReader Class**

Java FileReader class is used to read data from the file. It returns data in byte format like [FileInputStream](https://www.javatpoint.com/java-fileinputstream-class) class.

It is character-oriented class which is used for [file](https://www.javatpoint.com/java-file-class) handling in [java](https://www.javatpoint.com/java-tutorial).

**Java FileReader class declaration**

Let's see the declaration for Java.io.FileReader class:

1. public class FileReader extends InputStreamReader

**Constructors of FileReader class**

|  |  |
| --- | --- |
| **Constructor** | **Description** |
| FileReader(String file) | It gets filename in [string](https://www.javatpoint.com/java-string). It opens the given file in read mode. If file doesn't exist, it throws FileNotFoundException. |
| FileReader(File file) | It gets filename in [file](https://www.javatpoint.com/java-file-class) instance. It opens the given file in read mode. If file doesn't exist, it throws FileNotFoundException. |

**Methods of FileReader class**

|  |  |
| --- | --- |
| **Method** | **Description** |
| int read() | It is used to return a character in ASCII form. It returns -1 at the end of file. |
| void close() | It is used to close the FileReader class. |

**Java FileReader Example**

In this example, we are reading the data from the text file **testout.txt** using Java FileReader class.

package com.javatpoint;

import java.io.FileReader;

public class FileReaderExample {

    public static void main(String args[])throws Exception{

          FileReader fr=new FileReader("D:\\testout.txt");

          int i;

          while((i=fr.read())!=-1)

          System.out.print((char)i);

          fr.close();

    }

}

Here, we are assuming that you have following data in "testout.txt" file:

Welcome to javaTpoint.

Output:

Welcome to javaTpoint.

**Java - RandomAccessFile**

This [class](https://www.javatpoint.com/object-class) is used for reading and writing to random access file. A random access file behaves like a large [array](https://www.javatpoint.com/array-in-java) of bytes. There is a cursor implied to the array called file [pointer](https://www.javatpoint.com/c-pointers), by moving the cursor we do the read write operations. If end-of-file is reached before the desired number of byte has been read than EOFException is [thrown](https://www.javatpoint.com/throw-keyword). It is a type of IOException.

**Constructor**

|  |  |
| --- | --- |
| [**Constructor**](https://www.javatpoint.com/java-constructor) | **Description** |
| RandomAccessFile(File file, [String](https://www.javatpoint.com/java-string) mode) | Creates a random access file stream to read from, and optionally to write to, the file specified by the File argument. |
| RandomAccessFile(String name, String mode) | Creates a random access file stream to read from, and optionally to write to, a file with the specified name. |

**Method**

|  |  |  |
| --- | --- | --- |
| **Modifier and Type** | **Method** | **Method** |
| void | close() | It closes this random access file stream and releases any system resources associated with the stream. |
| FileChannel | getChannel() | It returns the unique [FileChannel](https://www.javatpoint.com/data-transfer-between-channels) object associated with this file. |
| int | readInt() | It reads a signed 32-bit integer from this file. |
| String | readUTF() | It reads in a string from this file. |
| void | seek(long pos) | It sets the file-pointer offset, measured from the beginning of this file, at which the next read or write occurs. |
| void | writeDouble(double v) | It converts the double argument to a long using the doubleToLongBits method in class Double, and then writes that long value to the file as an eight-byte quantity, high byte first. |
| void | writeFloat(float v) | It converts the float argument to an int using the floatToIntBits method in class Float, and then writes that int value to the file as a four-byte quantity, high byte first. |
| void | write(int b) | It writes the specified byte to this file. |
| int | read() | It reads a byte of data from this file. |
| long | length() | It returns the length of this file. |
|  |  |  |

**Example**

import java.io.IOException;

import java.io.RandomAccessFile;

public class RandomAccessFileExample {

    static final String FILEPATH ="myFile.TXT";

    public static void main(String[] args) {

        try {

            System.out.println(new String(readFromFile(FILEPATH, 0, 18)));

            writeToFile(FILEPATH, "I love my country and my people", 31);

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

    private static byte[] readFromFile(String filePath, int position, int size)

            throws IOException {

        RandomAccessFile file = new RandomAccessFile(filePath, "r");

        file.seek(position);

        byte[] bytes = new byte[size];

        file.read(bytes);

        file.close();

        return bytes;

    }

    private static void writeToFile(String filePath, String data, int position)

            throws IOException {

        RandomAccessFile file = new RandomAccessFile(filePath, "rw");

        file.seek(position);

        file.write(data.getBytes());

        file.close();

    }

}

The myFile.TXT contains text "This class is used for reading and writing to random access file."

After running the program it will contains

This class is used for reading I love my country and my people