

Different ways of Reading a text file in Java

Difficulty Level : Medium Last Updated : 20 Dec, 2021

There are multiple ways of writing and reading a text file. this is required while dealing with many applications. There are several ways to read a plain text file in Java e.g. you can use FileReader, BufferedReader, or Scanner to read a text file. Every utility provides something special e.g. BufferedReader provides buffering of data for fast reading, and Scanner provides parsing ability.

Methods:

1. Using BufferedReader class
2. Using Scanner class
3. Using File Reader class
4. Reading the whole file in a List
5. Read a text file as String

*We can also use both BufferedReader and Scanner to read a text file line by line in Java. Then Java SE 8 introduces another Stream class **java.util.stream.Stream** which provides a lazy and more efficient way to read a file.*

Tip Note: *Practices of writing good code like flushing/closing streams, Exception-Handling etc, have been avoided for better understanding of codes by beginners as well.*

Let us discuss each of the above methods to a deeper depth and most importantly by implementing them via a clean java program.

Method 1: Using BufferedReader class

This method reads text from a character-input stream. It does buffer for efficient reading of characters, arrays, and lines. The buffer size may be specified, or the default size may be used. The default is large enough for most purposes. In general, each read request made of a Reader causes a corresponding read request to be made of the underlying character

or byte stream. It is therefore advisable to wrap a `BufferedReader` around any `Reader` whose `read()` operations may be costly, such as `FileReaders` and `InputStreamReaders` as shown below as follows:

```
BufferedReader in = new BufferedReader(Reader in, int size);
```

Example:

```
// Java Program to illustrate Reading from FileReader
// using BufferedReader Class

// Importing input output classes
import java.io.*;

// Main class
public class GFG {

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

        // File path is passed as parameter
        File file = new File(
            "C:\\Users\\pankaj\\Desktop\\test.txt");

        // Note: Double backquote is to avoid compiler
        // interpret words
        // like \\test as \\t (ie. as a escape sequence)

        // Creating an object of BufferedReader class
        BufferedReader br
            = new BufferedReader(new FileReader(file));

        // Declaring a string variable
        String st;
        // Condition holds true till
        // there is character in a string
        while ((st = br.readLine()) != null)

            // Print the string
            System.out.println(st);
    }
}
```

Output:

If you want to code refer to GeeksforGeeks

Method 2: Using FileReader class

Convenience class for reading character files. The constructors of this class assume that the default character encoding and the default byte-buffer size are appropriate.

Constructors defined in this class are as follows:

1. **FileReader(File file):** Creates a new FileReader, given the File to read from
2. **FileReader(FileDescriptor fd):** Creates a new FileReader, given the FileDescriptor to read from
3. **FileReader(String fileName):** Creates a new FileReader, given the name of the file to read from

Example:

```
// Java Program to Illustrate reading from
// FileReader using FileReader class

// Importing input output classes
import java.io.*;

// Main class
// ReadingFromFile
public class GFG {

    // Main driver method
    public static void main(String[] args) throws Exception
    {

        // Passing the path to the file as a parameter
        FileReader fr = new FileReader(
            "C:\\Users\\pankaj\\Desktop\\test.txt");

        // Declaring loop variable
        int i;
        // Holds true till there is nothing to read
        while ((i = fr.read()) != -1)

            // Print all the content of a file
            System.out.print((char)i);
    }
}
```

```
}
```

Output:

If you want to code refer to GeeksforGeeks

Method 3: Using Scanner class

A simple text scanner that can parse primitive types and strings using regular expressions. A Scanner breaks its input into tokens using a delimiter pattern, which by default matches whitespace. The resulting tokens may then be converted into values of different types using the various next methods.

Example 1: With using loops

```
// Java Program to illustrate
// reading from Text File
// using Scanner Class
import java.io.File;
import java.util.Scanner;
public class ReadFromFileUsingScanner
{
    public static void main(String[] args) throws Exception
    {
        // pass the path to the file as a parameter
        File file = new File("C:\\Users\\pankaj\\Desktop\\test.txt");
        Scanner sc = new Scanner(file);

        while (sc.hasNextLine())
            System.out.println(sc.nextLine());
    }
}
```

Output:

If you want to code refer to GeeksforGeeks

Example 2: Without using loops

```
// Java Program to illustrate reading from FileReader
// using Scanner Class reading entire File
// without using loop
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

public class ReadingEntireFileWithoutLoop
{
    public static void main(String[] args)
        throws FileNotFoundException
    {
        File file = new File("C:\\Users\\pankaj\\Desktop\\test.txt");
        Scanner sc = new Scanner(file);

        // we just need to use \\Z as delimiter
        sc.useDelimiter("\\Z");

        System.out.println(sc.next());
    }
}
```

Output:

If you want to code refer to GeeksforGeeks

Method 4: Reading the whole file in a List

Read all lines from a file. This method ensures that the file is closed when all bytes have been read or an I/O error, or other runtime exception, is thrown. Bytes from the file are decoded into characters using the specified charset.

Syntax:

```
public static List readAllLines(Path path,Charset cs)throws IOException
```

This method recognizes the following as line terminators:

\u000D followed by \u000A, CARRIAGE RETURN followed by LINE FEED
\u000A, LINE FEED
\u000D, CARRIAGE RETURN

Example

```
// Java program to illustrate reading data from file
// using nio.File
import java.util.*;
import java.nio.charset.StandardCharsets;
import java.nio.file.*;
import java.io.*;
public class ReadFileIntoList
{
    public static List<String> readFileInList(String fileName)
    {

        List<String> lines = Collections.emptyList();
        try
        {
            lines =
                Files.readAllLines(Paths.get(fileName), StandardCharsets.UTF_8);
        }

        catch (IOException e)
        {

            // do something
            e.printStackTrace();
        }
        return lines;
    }
    public static void main(String[] args)
    {
        List l = readFileInList("C:\\\\Users\\\\pankaj\\\\Desktop\\test.java");

        Iterator<String> itr = l.iterator();
        while (itr.hasNext())
            System.out.println(itr.next());
    }
}
```

Output:

If you want to code refer to GeeksforGeeks

Method 5: Read a text file as String

Example

```
// Java Program to illustrate
// reading from text file
// as string in Java
package io;

import java.nio.file.*;

public class ReadTextAsString {

    public static String readFileAsString(String fileName)throws Exception
    {
        String data = "";
        data = new String(Files.readAllBytes(Paths.get(fileName)));
        return data;
    }

    public static void main(String[] args) throws Exception
    {
        String data = readFileAsString("C:\\Users\\pankaj\\Desktop\\test.java");
        System.out.println(data);
    }
}
```

Output:

If you want to code refer to GeeksforGeeks

This article is contributed by **Pankaj Kumar**. If you like GeeksforGeeks and would like to contribute, you can also write an article using write.geeksforgeeks.org or mail your article to review-team@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

