The *java.io.File* class can be used to create a new file in Java. When we initialize the File object, we provide the file name, and then we can call *createNewFile()* method to create a new file in Java.

File *createNewFile()* method returns *true* if a new file is created and *false* if the file already exists. This method also throws *java.io.IOException* when it’s not able to create the file.

**Create a New File Example**

1. Create *File* class object by passing file absolute location path "C://workspace/sample.txt"
2. call createNewFile() method of file object to create new file named "sample.txt" file in directory "C://workspace"
3. File *createNewFile()* method returns true if a new file is created and false if the file already exists. This method also throws java.io.IOException when it’s not able to create the file.
4. Observe the directory whether the file is created or not.

import java.io.File;

import java.io.IOException;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

/\*\*

\* This Java program demonstrates how to create new file in Java.

\* @author javaguides.net

\*/

public class CreateFileExample {

private static final Logger LOGGER = LoggerFactory

.getLogger(CreateFileExample.class);

public static void main(String[] args) {

createFile();

}

public static void createFile() {

File file = new File("C:/workspace/sample.txt");

try {

if (file.createNewFile()) {

LOGGER.info("File is created !!");

} else {

LOGGER.info("File is already exist");

}

} catch (IOException e) {

LOGGER.error(e.getMessage());

}

}

}

**Create File using FileOutputStream**

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileOutputStream;

import java.io.FileWriter;

import java.io.IOException;

import java.io.OutputStream;

public class WriteFile {

/\*\*

\* This class shows how to write file in java

\* @param args

\* @throws IOException

\*/

public static void main(String[] args) {

String data = "I will write this String to File in Java";

int noOfLines = 10000;

// Use Streams when you are dealing with raw data

try(OutputStream os = new FileOutputStream(new File("C:/workspace/sample.txt"))){

os.write(data.getBytes(), 0, data.length());

} catch (IOException e) {

e.printStackTrace();

}

}

}

We can use FileWriter, BufferedWriter, java 7 Files, and FileOutputStream to write a file in Java.

# Using FileOutputStream

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileOutputStream;

import java.io.FileWriter;

import java.io.IOException;

import java.io.OutputStream;

public class WriteFile {

/\*\*

\* This class shows how to write file in java

\* @param args

\* @throws IOException

\*/

public static void main(String[] args) {

String data = "I will write this String to File in Java";

int noOfLines = 10000;

// Use Streams when you are dealing with raw data

try(OutputStream os = new FileOutputStream(new File("C:/workspace/sample.txt"))){

os.write(data.getBytes(), 0, data.length());

} catch (IOException e) {

e.printStackTrace();

}

}

}

# Using BufferedWriter

import java.io.BufferedWriter;

import java.io.FileWriter;

import java.io.IOException;

/\*\*

\* This Java program demonstrates how to write file in Java.

\* @author javaguides.net

\*/

public class WriteFileExample {

public static void main(String[] args) {

writeFile();

}

// Write file using BufferedWriter

public static void writeFile() {

try (BufferedWriter bw = new BufferedWriter(

new FileWriter("C:/workspace/sample.txt"))) {

String content = "This is the content to write into file\n";

bw.write(content);

} catch (IOException e) {

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

}

}

}

# Using Files

import java.io.IOException;

import java.nio.file.Files;

import java.nio.file.Paths;

public class WriteFile {

/\*\*

\* This class shows how to write file in java

\* @param args

\* @throws IOException

\*/

public static void main(String[] args) {

String data = "I will write this String to File in Java";

try {

Files.write(Paths.get("C:/workspace/sample.txt"), data.getBytes());

} catch (IOException e) {

e.printStackTrace();

}

}

}

# Using FileWriter

import java.io.FileWriter;

import java.io.File;

import java.io.IOException;

public class WriteFile {

/\*\*

\* This class shows how to write file in java

\* @param args

\* @throws IOException

\*/

public static void main(String[] args) {

String data = "I will write this String to File in Java";

File file = new File("C:/workspace/sample.txt");

try(FileWriter fr = new FileWriter(file)){

fr.write(data);

} catch (IOException e) {

e.printStackTrace();

}

}

}

## 1. Overview

In this example, we will use a **BufferedReader** class to read a file named "sample.txt". **BufferedReader** class is used to read the text from a character-based input stream. It can be used to read data line by line by *readLine()* method. It makes the performance fast.

## 2. Read File Using *BufferedReader* Examples

**BufferedReader** class offers few read methods to read a file by character by character or line by line. Let's write examples to read file character by character.

package com.javaguides.javaio.fileoperations.examples;

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

/\*\*

\* This Java program demonstrates how to read file in Java – BufferedReader.

\* @author javaguides.net

\*/

public class BufferedReaderExample {

public static void main(String[] args) {

try (FileReader fr = new FileReader("C:/workspace/java-io-guide/sample.txt");

BufferedReader br = new BufferedReader(fr);) {

int i;

while ((i = br.read()) != -1) {

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

}

} catch (IOException e) {

e.printStackTrace();

}

}

}

Let's write examples to read file line by line.

package com.javaguides.javaio.fileoperations.examples;

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

/\*\*

\* This Java program demonstrates how to to read file line by line.

\* @author javaguides.net

\*/

public class BufferedReaderExample {

public static void main(String[] args) {

try (FileReader fr = new FileReader("C:/workspace/java-io-guide/sample.txt");

BufferedReader br = new BufferedReader(fr);) {

String sCurrentLine;

while ((sCurrentLine = br.readLine()) != null) {

System.out.println(sCurrentLine);

}

} catch (IOException e) {

e.printStackTrace();

}

}

}