

# Java Program to Create a New File

Difficulty Level : Easy Last Updated : 09 Feb, 2022

A File is an abstract path, it has no physical existence. It is only when “using” that File that the underlying physical storage is hit. When the file is getting created indirectly the abstract path is getting created. The file is one way, in which data will be stored as per requirement.

## Steps to Create a New File in Java

1. Primary, in order to create a new file, inbuilt files and functions are used which definitely will throw Exceptions here playing it safe. So in order to deal with it, we will be using Exception Handling Techniques. Here, we will use one of them known as-try-catch block techniques.

2. Secondary, additional work is simply we will be importing File Class for which we will be importing File Class.

**Syntax:** To import file library or Classes

```
import java.util.File ;
```

**Syntax:** To create a new file

```
File object_name = new File(Directory)
```

**Syntax:** To specify a directory is different in different operating systems (suppose java file is in a folder named ‘Folder’ is created on desktop)

In Linux and Mac

```
/Users/mayanksolanki/Desktop/Folder/
```

In Windows: ‘ \\ ‘ used instead of ‘ / ‘ to escape ‘ \ ‘ character. So the same directory is accessed as

```
\\Users\\mayanksolanki\\Desktop\\Folder\\
```

**There are two standards methods to create a new file** either directly with the help of File class or indirectly with the help of FileOutputStream by creating an object of the file in both the approaches.

- **By using File Class**
- **By using FileOutputStream Class**

### File Class

It is a class that is just a handle for

**Method:** File.createNewFile()

It is used for those objects which do not have physical existence

### FileOutputStreamClass

It is an output stream that can be written to [FileOutputStream JavaDoc](#)

**Method:** FileOutputStream

**Example:** echo > myFile.txt

It is used for those objects which are already existing

Both classes provide some methods which are mainly used to do operations regarding files. For example, to create, to write, to compare two path names, to check whether a specific file is present or not, and many more. To understand this topic, first, consider one example for both approaches.

1. *Terminal Command used to compile any java code on the machine*
2. *Terminal Command used to Run any java code on the machine*

- *javac class\_name.java // For Compilation*
- *java class\_name // For Execution*

*Terminal of Mac operating system will be used for implementation and providing an output of accessing the directory*

*Directory Used : /Users/mayanksolanki/Desktop/Folder/*

### Method 1: Create a new file using the File class

---

```
// Java Program to create new file using File class

// Importing new files
```

```
import java.io.File;
import java.io.BufferedReader;

// Importing as it converts bits to strings
import java.io.InputStreamReader;

public class GFG {

    // Main Driver Method
    public static void main(String args[])
    {
        // Creating New File via function
        GFG gfg = new GFG();
        gfg.newFile();
    }

    // Function To Make New File
    public void newFile()
    {
        String strPath = "", strName = "";

        // Try-catch Block
        try {

            // Creating BufferedReaded object
            BufferedReader br = new BufferedReader(
                new InputStreamReader(System.in));
            System.out.println("Enter the file name:");

            // Reading File name
            strName = br.readLine();
            System.out.println("Enter the file path:");

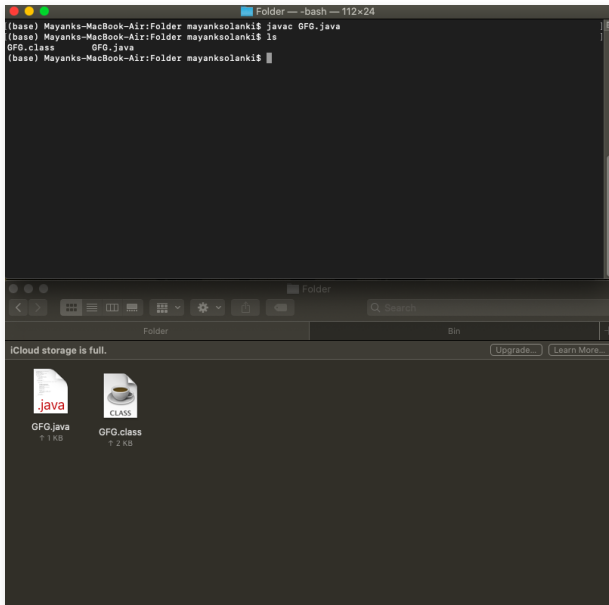
            // Reading File Path
            strPath = br.readLine();

            // Creating File Object
            File file1
                = new File(strPath + "" + strName + ".txt");

            // Method createNewFile() method creates blank
            // file.
            file1.createNewFile();
        }

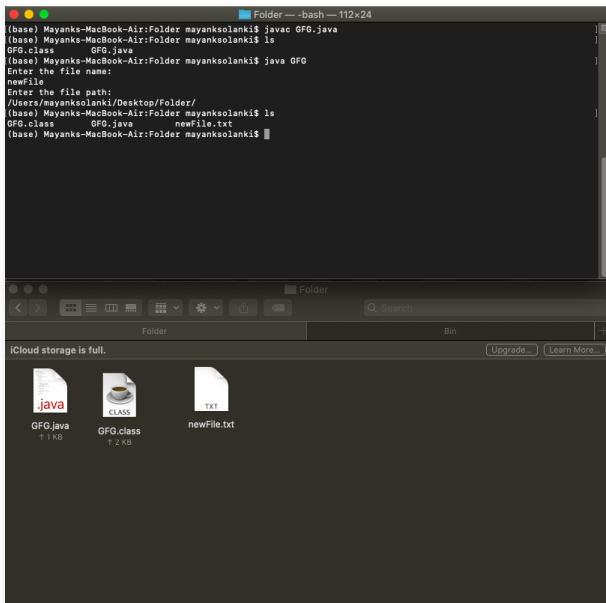
        // Try-Catch Block
        catch (Exception ex1) {
        }
    }
}
```

## Output:



Name of file to be added : newFile.txt

Directory where file is to be added : /Users/mayanksolanki/Desktop/Folder/



Added file name : newFile.txt

Added file name directory : /Users/mayanksolanki/Desktop/Folder/

## Explanation:

1. To create a new file using java language, the “File” class is used here. “BufferedReader” and “InputStreamReader” both classes are used to make file names and paths from the user as input. These classes are located inside the “java.io” package. So to make use of those classes, it is necessary to import those classes at the beginning of the program.

2. Here class is created namely, “GFG”. And inside that class “main()” method is defined from which execution will be started.
3. Inside the “*main()*” method object of the class is created. And this object is used to call the “*newFile()*” method.
4. Outside of the *main()* method, the *newFile()* method is declared which covers code for taking input from the user and create the file as per input.
5. As the file have its own name and path, it is necessary to give a name for the file (which we want to create) and a path (where to create the file) for the file. In this line, two blank strings are declared namely, strName, strPath. “strName, and strPath are used to store the name and path of the file when the user gives this information.
6. To take file name and path from the user as input, here BufferedReader class and InputStreamReader class are used. The object of BufferedReader “br” is useful to take input values given by the user.
7. This line print some text to give an indication to the user like ”Enter the file name:”. To print the text “*println()*” function is used.
8. Here “*readLine()*” method is used to take input and store it, in strName and strPath.
9. Here object of the File class is created and as a parameter, the file path and name are given to the constructor. In this line of code “.txt” is a format of the file. You can change it as per need. The object of the File class is necessary to call methods provided in its class.
10. Here “*createNewFile()*” method is called with the help of the File class object. This method creates a blank file on a given directory path.
11. Lastly, enclosed by “try{ }” block. Because, methods like *readLine()* and *createNewFile()* methods generates exception. So to handle that exception try, the catch is used.

## Method 2: Create a new file using FileOutputStream class

---

```
// Java Program to create new file
// using FileOutputStream class

// Importing File Classes
import java.io.FileOutputStream;

// Importing BufferedReader Class for taking input
import java.io.BufferedReader;

// Importing as it converts bits to strings
import java.io.InputStreamReader;
```

```
// Function Helping Create New File
public class GFG {

    // Main Driver Method
    public static void main(String args[])
    {
        // Creating File Object
        GFG gfg = new GFG();
        gfg.newFile();
    }

    // Function To Create A New File
    public void newFile()
    {
        String strFilePath = "", strFileName = "";

        // Try-Catch Block
        try {

            // Creating BufferClass Object
            BufferedReader br = new BufferedReader(
                new InputStreamReader(System.in));
            System.out.println("Enter the file name:");

            // Asking file name from User
            strFileName = br.readLine();
            System.out.println("Enter the file path:");

            // Asking file path from User
            strFilePath = br.readLine();

            // Creating Object of FileOutputStream Class
            FileOutputStream fos = new FileOutputStream(
                strFilePath + "" + strFileName + ".txt");
        }

        // Try-Catch Block
        catch (Exception ex1) {
        }
    }
}
```

## Explanation:

In the second example, the File class is not used to create a new File programmatically.

1. To create a new file using java language, “FileOutputStream” class is used here and “BufferedReader” & “InputStreamReader” both are used to take file name and path from the

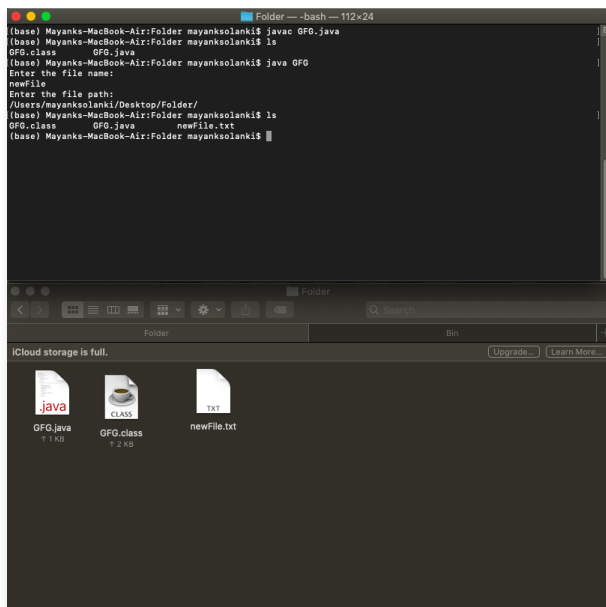
user as input. These classes are located inside of “java.io” package. So to make use of those classes, it is necessary to import them at the beginning of the program.

2. Class is created namely, “GFG”. And inside that class “main()” method is defined from which execution will be started.
3. Inside of “main()” method object of the class is created. And this object is used to call the “newFile()” method.
4. Outside of the main() method, the newFile() method is declared which is covers code for taking input from the user and create a file as per input.
5. In this line, two blank strings are declared namely, strFileName, strFilePath. “strFileName and strFilePath are used to store the name and path of the file when the user gives this information.
6. To take file name and path from the user as input, here BufferedReader class and InputStreamReader class are used. The object of BufferedReader “br” is useful to take input values given by the user.
7. This line print some text to give an indication to the user like ”Enter the file name:”. To print text “println()” function is used.
8. Here “readLine()” method is used to take input and store it, in strFileName and strFilePath.
9. Here object of FileOutputStream class is created and as a parameter, file path and name are given to the constructor. In this line of code “.txt” is a format of the file. You can change it as per need. The object of FileOutputStream class is necessary to call methods provided in its class. For example, if the user wants to store some text in a newly created file programmatically then, the write() method is helpful.
10. Lastly, enclosed by “try{ }” block. Because, readLine() method generates exception. So to handle that exception try, catch block is used.

**Output:** It will be the same as the previous one because just the approach to create a new file has changed the rest of the file name and the directory where it is added remains the same.

Added file name : newFile.txt

Added file name directory : /Users/mayanksolanki/Desktop/Folder/



The screenshot shows a macOS desktop environment. At the top, a terminal window titled "Folder — -bash — 112x24" displays the following commands and output:

```
(base) Mayanks-MacBook-Air:Folder mayanksolanki$ javac GFG.java
(base) Mayanks-MacBook-Air:Folder mayanksolanki$ ls
GFG.class      GFG.java
(base) Mayanks-MacBook-Air:Folder mayanksolanki$ java GFG
Enter the file name:
newFile
Enter the file path:
/Users/mayanksolanki/Desktop/Folder/
(base) Mayanks-MacBook-Air:Folder mayanksolanki$ ls
GFG.class      GFG.java      newFile.txt
(base) Mayanks-MacBook-Air:Folder mayanksolanki$
```

Below the terminal, a Finder window titled "Folder" is open, showing the contents of the "/Users/mayanksolanki/Desktop/Folder/" directory. A message at the top states "iCloud storage is full." with "Upgrade" and "Learn More" buttons. The directory contains three files:

- GFG.java (1 KB)
- GFG.class (2 KB)
- newFile.txt