

# COMP249 Tutorial 5

Ella Noyes

July 24, 2024

## 1 Files

Files are used to store programs and data. In this tutorial, we’re working with text files, which store sequences of characters and line breaks.

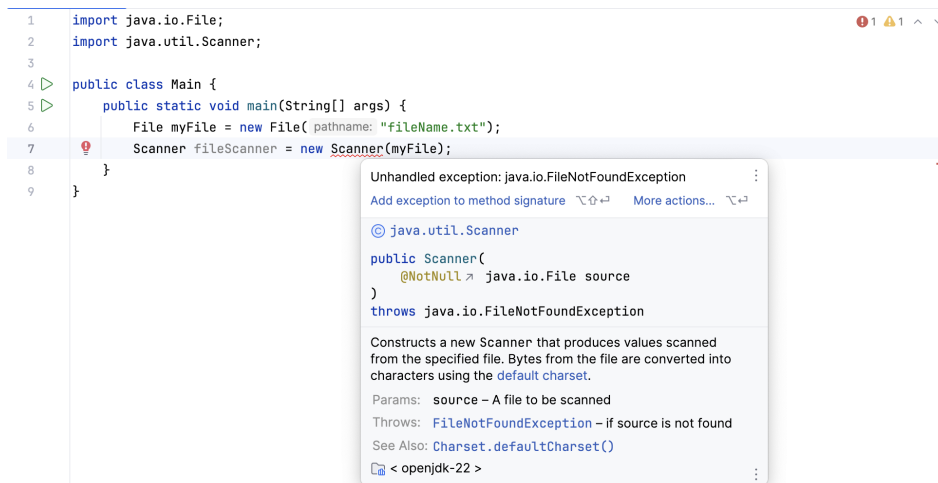
To work with a file in Java, you can manipulate it as an object of the `File` class (this requires importing `java.io.File`). A `File` object can be a file or a directory. When you create a `File` object, you can get information on the referenced file such as:

`canRead()`, `canWrite()`, `delete()`, `exists()`, `getName()`, `getAbsolutePath()`, ...

## 2 File I/O and Exception Handling

An exception is an event that interrupts a program’s flow of execution. Exceptions that you’ve likely encountered before include `ArrayIndexOutOfBoundsException`, `ClassNotFoundException`, `NullPointerException`, ...

With file I/O operations, a lot can go wrong (file not found, permission denied, ...), so many of these operations “throw” checked exceptions. You must explicitly anticipate these exceptions. If you don’t handle these using a try-catch statement, then you’ll see “unhandled exception” errors in your IDE.



```
1 import java.io.File;
2 import java.util.Scanner;
3
4 public class Main {
5     public static void main(String[] args) {
6         File myFile = new File("fileName.txt");
7         Scanner fileScanner = new Scanner(myFile);
8     }
9 }
```

Unhandled exception: java.io.FileNotFoundException  
Add exception to method signature More actions...

java.util.Scanner

```
public Scanner(
    @NotNull java.io.File source
)
throws java.io.FileNotFoundException
```

Constructs a new Scanner that produces values scanned from the specified file. Bytes from the file are converted into characters using the default charset.

Params: source – A file to be scanned

Throws: `FileNotFoundException` – If source is not found

See Also: `Charset.defaultCharset()`

< openjdk-22 >

Here, we’re warned that `Scanner`’s constructor throws a `FileNotFoundException` when called with a `File` object as its argument.

## 3 Reading from a File

There are many ways to read from a file (and to write to a file!). You can use `Scanner`, `BufferedReader`, or other options according to your program’s requirements. I’ll show you how to use `Scanner` and `BufferedReader`.

Once you’re done reading from a file, it’s essential to close that file. To do so, call `close()` on your

Scanner object (or BufferedReader, or whatever class you're using to read from that file). Now let's look at some code.

## 4 Writing to a File

There are different approaches to this, too (BufferedWriter, FileWriter, FileOutputStream,...). You could use an object of the FileWriter class and call write() to write Strings of text into your file. These classes support appending a file (adding text onto the end of a file). To append a file using a FileWriter, create an object with this overloaded constructor.

```
FileWriter fr = new FileWriter(file , true);
```

Just like when reading from a file, you must close a file when you're done writing to it by calling close() on your FileWriter object.

You can also consume from and produce to a file using streams, we'll see an example of that using FileInputStream and FileOutputStream objects.

## 5 Practice Problem : Dear Diary

This is my practice problem but I got ChatGPT to write it out clearly for me!  
Create a Java program that simulates a digital diary application, allowing users to record daily entries and store them in a text file.

### 5.1 DiaryEntry Class:

Define a DiaryEntry class with the following attributes:

- date: Represents the date of the entry.
- moodScore: An integer score out of 10 representing the user's mood on that day.
- entryText: A string containing the text content of the diary entry.
- Implement a constructor that prompts the user to input:  
The moodScore (out of 10).  
The entryText for that day.  
Automatically set the date to the current date using Java's date utilities.

### 5.2 File Handling:

- Create an empty text file named myDiary.txt where diary entries will be stored.
- Create a DiaryEntry object and write it to this file in a structured format.
- If you have time: Implement a method to read all diary entries from myDiary.txt. Display each entry on the console, formatted to include the date, mood score, and entry text.

### 5.3 Example Diary Entry:

A diary entry printed into your txt file might look something like this:

Date: 2024-07-24

Mood Score: 8

Entry: The midterm went so well and was so easy!

### 5.4 Requirements:

- Use appropriate Java file handling classes (FileWriter, BufferedReader, etc.) for writing and reading from files.
- Implement error handling to manage potential exceptions during file operations.
- Ensure the program provides a clear and user-friendly interface for recording and viewing diary entries.