

File Input and Output

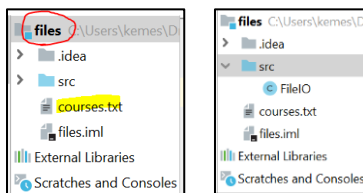
To follow along, create a new Java project **files** with a class **FileIO**.

On the desktop, create a text file (with Notepad, for example—make sure it has the extension .txt) named **courses** with the following text:

```
course credits score
cs121 4 98
cs124 3 95
eng200 3 89
```

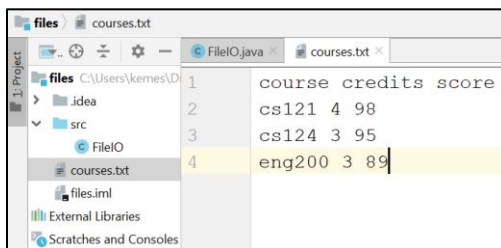
Copy **courses.txt** and paste it into the IntelliJ **PROJECT** folder—**NOT INSIDE OF THE SRC FOLDER!** (You can also drag and drop a file into the project folder, but this will move the file into the project folder and not leave a copy of it on the desktop).

The project folder is circled in red. Even though the highlighted file looks like it is in the src folder, it is not. Expand the src folder and you will see the FileIO class inside src is indented.



If **courses.txt** is inside **src**, simply drag and drop it into the **files** project folder above.

The file can be opened in IntelliJ by double clicking on it.



To access a text file, we import **Java.io.File** and create a new instance of the **File** class by passing in either the path to a file if it's on the desktop, or, if it's in the **project** folder (*outside* of the **src** folder), simply by its *name* ("courses.txt" in this example).

For now, read **inputFile1** from your desktop. You can change the code to reference **inputFile2** later and see it will give the same results with a copy of the text file in the project folder.

```
File inputFile1 = new File("C:/Users/Karl/Desktop/courses.txt");
// or
File inputFile2 = new File("courses.txt");
```

To read the file we use **Scanner** and pass in the variable name of the file.

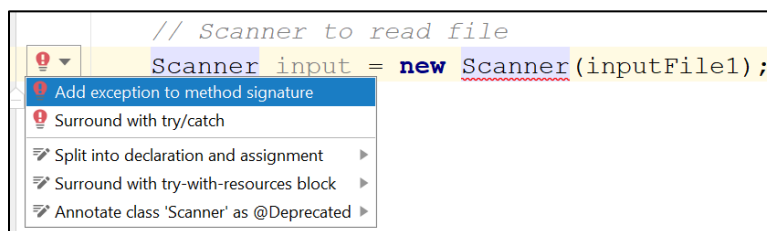
Notice in the screenshot below, Scanner is underlined in red. If the cursor is hovered over it, we see an unhandled **exception** message.

An **exception** is an event that occurs during the execution of a program that disrupts the normal flow of instructions. The **possible** exception it refers to here (*FileNotFoundException*) is Java not being able to find the file.

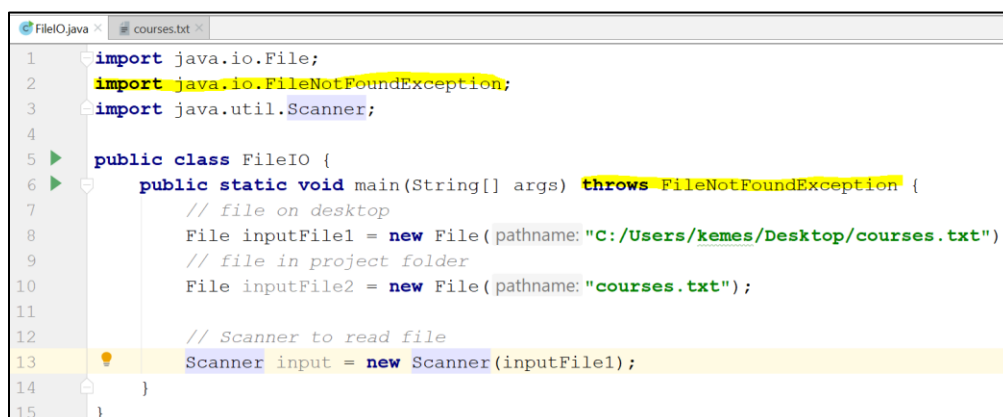


```
1 import java.io.File;
2 import java.util.Scanner;
3
4 public class FileIO {
5     public static void main(String[] args) {
6         // file on desktop
7         File inputFile1 = new File(pathname: "C:/Users/kemes/Desktop/courses.txt");
8         // file in project folder
9         File inputFile2 = new File(pathname: "courses.txt");
10
11         // Scanner to read file
12         Scanner input = new Scanner(inputFile1);
13     }
14 }
```

If an exception occurs, we need to be able to handle it in a way that doesn't stop the program from running. If you click on the underlined Scanner, a red lightbulb icon should appear to the left of the line of code. Hover over it with the cursor and a dropdown arrow should appear. Click the arrow and some suggestions on how to handle the exception should appear.



Select the first option, “Add exception to method signature” and code will be added to the end of the main method header.



```
1 import java.io.File;
2 import java.io.FileNotFoundException;
3 import java.util.Scanner;
4
5 public class FileIO {
6     public static void main(String[] args) throws FileNotFoundException {
7         // file on desktop
8         File inputFile1 = new File(pathname: "C:/Users/kemes/Desktop/courses.txt");
9         // file in project folder
10        File inputFile2 = new File(pathname: "courses.txt");
11
12        // Scanner to read file
13        Scanner input = new Scanner(inputFile1);
14    }
15 }
```

Throws is used to declare exceptions that are not handled by a method and is an instruction to the callers to either handle these explicitly or rethrow them up the call hierarchy. The **throws** keyword indicates a method can throw the exception. (We'll discuss methods next session.)

To read the file, we'll first use a while loop and add the condition that as long as the file has a next line to read, we will assign that next line to the variable *line* and print the line:

```
FileIO.java × courses.txt ×
1  import java.io.File;
2  import java.io.FileNotFoundException;
3  import java.util.Scanner;
4
5  public class FileIO {
6  public static void main(String[] args) throws FileNotFoundException {
7      // file on desktop
8      File inputFile1 = new File( pathname: "C:/Users/kemes/Desktop/courses.txt");
9      // file in project folder
10     File inputFile2 = new File( pathname: "courses.txt");
11
12     // Scanner to read file
13     Scanner input = new Scanner(inputFile1);
14
15     while (input.hasNextLine()) {
16         String line = input.nextLine();
17         System.out.println(line);
18     }
19     input.close();
20 }
21 }
```

Now we can try using the other exception handling method, putting our code inside **try/catch**. Java will “try” running the code inside the try block, if there is an exception, it will “catch” it and execute the code inside of the catch block:

```
5  public class FileIO {
6  public static void main(String[] args) throws FileNotFoundException {
7      // file on desktop
8      File inputFile1 = new File( pathname: "C:/Users/kemes/Desktop/courses.txt");
9      // file in project folder
10     File inputFile2 = new File( pathname: "courses.txt");
11
12     try {
13         Scanner input = new Scanner(inputFile1);
14
15         while (input.hasNextLine()) {
16             String line = input.nextLine();
17             System.out.println(line);
18         }
19         input.close();
20     } catch (FileNotFoundException e) {
21         e.printStackTrace();
22     }
23 }
24 }
```

Running the program, we see the content of the *courses.txt* file is read and printed in console.

```
Run: FileIO x
" C:\Program Files\Java
course credits score
cs121 4 98
cs124 3 95
eng200 3 89
```

Change the ending of the pathname of **inputFile1** from *courses.txt* to *course.txt*—a file that doesn't exist—and run the program; the **e.printStackTrace** statement in the catch block will print the **stack trace**, which is a list of the method calls that the application was in the middle of when an Exception was thrown.

```
Run: FileIO x
"C:\Program Files\Java\jdk-11.0.4\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Communit
java.io.FileNotFoundException: C:\Users\kemes\Desktop\course.txt (The system cannot find the file specified)
at java.base/java.io.FileInputStream.open0(Native Method)
at java.base/java.io.FileInputStream.open(FileInputStream.java:219)
at java.base/java.io.FileInputStream.<init>(FileInputStream.java:157)
at java.base/java.util.Scanner.<init>(Scanner.java:639)
at FileIO.main(FileIO.java:13)
```

Change the catch block to a message instead and it will print that when an exception occurs.

```
20      } catch (FileNotFoundException e) {
21          System.err.println("File not found.");
22      }
```

*Notice it's **System.err.println()**, not **System.out**.

.err will print the message in **red**.

Running the program now gives this output:

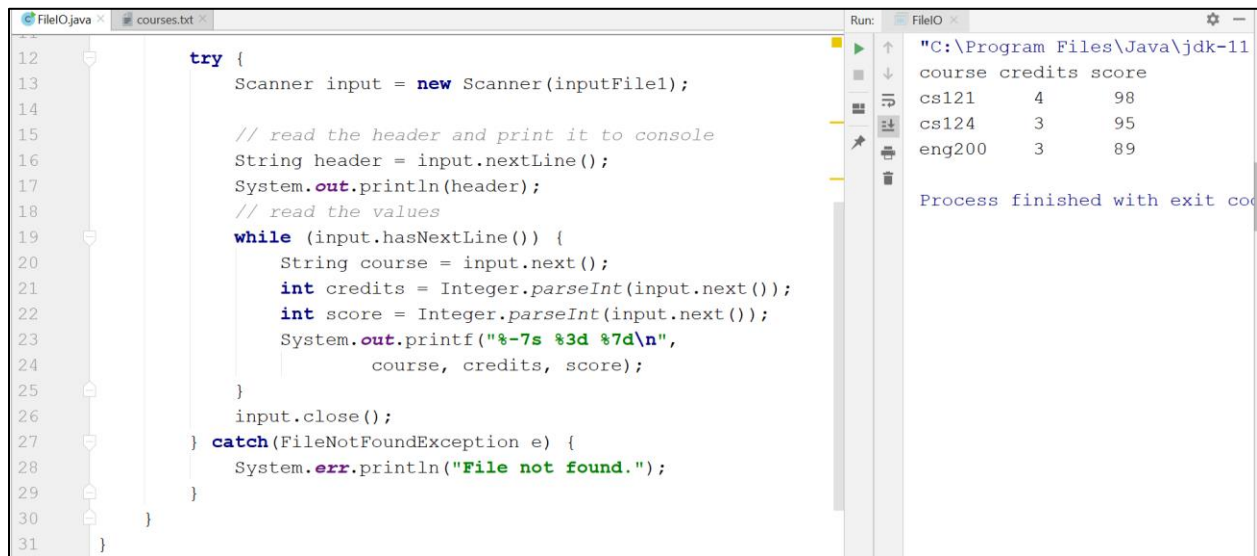
```
Run: FileIO x
"C:\Program Files\Java\jdk-11.0.4\k
File not found.
Process finished with exit code 0
```

*Change filename path back to *courses.txt*.

Instead of getting a line of the file with **nextLine()**, we can use **.next()**, which doesn't get the whole line, it gets the characters **between the white spaces** in the text.

We'll get each string between the white spaces and assign it to a variable.

First we assign the header line to a variable, then we assign each string separated by white space to a variable.



The screenshot shows an IDE with two windows. The left window, titled 'FileIO.java', contains the following Java code:

```
12     try {
13         Scanner input = new Scanner(inputFile1);
14
15         // read the header and print it to console
16         String header = input.nextLine();
17         System.out.println(header);
18         // read the values
19         while (input.hasNextLine()) {
20             String course = input.next();
21             int credits = Integer.parseInt(input.next());
22             int score = Integer.parseInt(input.next());
23             System.out.printf("%-7s %3d %7d\n",
24                             course, credits, score);
25         }
26         input.close();
27     } catch (FileNotFoundException e) {
28         System.err.println("File not found.");
29     }
30 }
31 }
```

The right window, titled 'Run: FileIO', shows the output of the program:

```
"C:\Program Files\Java\jdk-11
course credits score
cs121      4      98
cs124      3      95
eng200     3      89

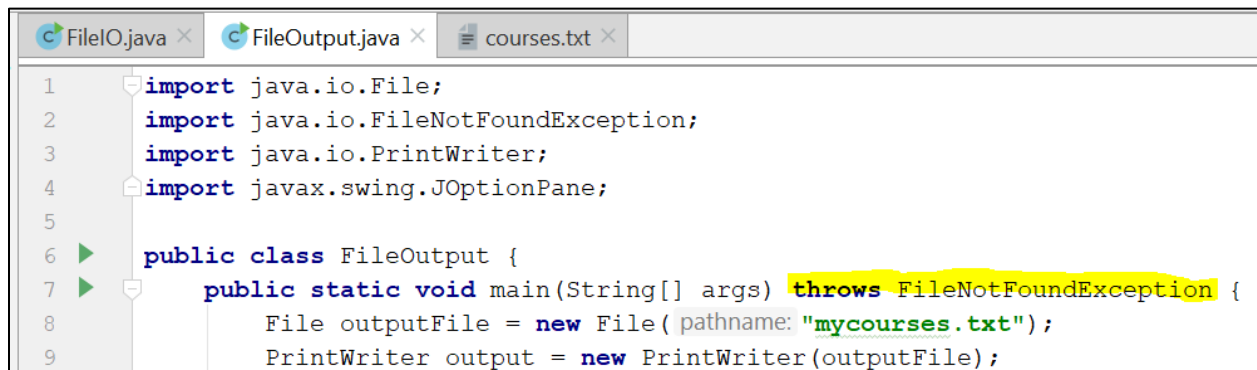
Process finished with exit co
```

Writing to a file

Create a new class **FileOutput** with a main method.

The **PrintWriter** class allows us to write to a file.

To print to a text file, we import *java.io.PrintWriter*; we create a new File object to write to as we did to read to; we create a new *PrintWriter* object (we'll get an error until we add the **throws exception** to the main method), passing the file name variable into the parameter as we similarly did with Scanner.



The screenshot shows an IDE with three windows: 'FileIO.java', 'FileOutput.java', and 'courses.txt'. The 'FileOutput.java' window is active and contains the following Java code:

```
1 import java.io.File;
2 import java.io.FileNotFoundException;
3 import java.io.PrintWriter;
4 import javax.swing.JOptionPane;
5
6 public class FileOutput {
7     public static void main(String[] args) throws FileNotFoundException {
8         File outputFile = new File(pathname: "mycourses.txt");
9         PrintWriter output = new PrintWriter(outputFile);
```

***If you don't specify a path for the output file, it will be created in your project folder. Look inside the project folder for it. Double click to open it.**

To write to the file, we use *println()*, *print()*, or *printf()* to write to a file as we would to the console.

```
FileO.java x FileOutput.java x courses.txt x
1  import java.io.File;
2  import java.io.FileNotFoundException;
3  import java.io.PrintWriter;
4  import javax.swing.JOptionPane;
5
6  public class FileOutput {
7      public static void main(String[] args) throws FileNotFoundException {
8          File outputFile = new File("mycourses.txt");
9          PrintWriter output = new PrintWriter(outputFile);
10         String course;
11         String credits;
12         String score;
13
14         // write a header for the file with spaces between each
15         output.printf("%s %s %s\n", "course", "credits", "score");
16         // ask user for info for 3 courses
17         for (int i = 1; i <= 3; i++) {
18             course = JOptionPane.showInputDialog(String.format("Enter name of course %d", i));
19             credits = JOptionPane.showInputDialog("Enter the course credits");
20             score = JOptionPane.showInputDialog("Enter the course score");
21             // print info on one line with spaces between
22             output.printf("%s %s %s\n", course, credits, score);
23         }
24         output.close();
25     }
26 }
```

Appending Data to a File

Create a new class **FileAppend** with a main method.

When you pass the name of a file to **PrintWriter** and the file already exists, it will be erased and a new empty file with the same name will be created. You may want to keep an existing file and append new data to its current contents. Appending to a file means writing new data to the end of the data that already exists in the file. To append data to an existing file, you first create an instance of the **FileWriter** class. You pass two arguments to the **FileWriter**: a string containing the name of the file, and the boolean value **true**.

```
FileO.java x FileOutput.java x FileAppend.java x mycourses.txt x courses.txt x
1  import javax.swing.JOptionPane;
2  import java.io.FileNotFoundException;
3  import java.io.FileWriter;
4  import java.io.IOException;
5  import java.io.PrintWriter;
6
7  public class FileAppend {
8      public static void main(String[] args) throws IOException {
9          try {
10             FileWriter fileWriter = new FileWriter("mycourses.txt", append: true);
11             PrintWriter output = new PrintWriter(fileWriter);
12
13             String course = JOptionPane.showInputDialog("Enter course name");
14             String credits = JOptionPane.showInputDialog("Enter course credits");
15             String score = JOptionPane.showInputDialog("Enter course score");
16
17             output.printf("%s %s %s", course, credits, score);
18             fileWriter.close();
19             output.close();
20         } catch (FileNotFoundException e) {
21             System.err.println("File not found");
22         }
23     }
24 }
```

After running, you should find the new line appended to the existing file.