Lab: Files and Streams

Problems for exercises and homework for the [“Java Advanced” course @ SoftUni](https://softuni.bg).

You can check your solutions here: https://judge.softuni.bg/Contests/Practice/Index/403#0.

For this lab exercises you are given a zipped folder with resources, that you will need to use. For each exercise submit the output of the program, not the code.

I. Stream Basics

1. Read File

You are given a file named "input.txt". Read and print all of its contents as a sequence of bytes (the binary representation of the ASCII code for each character) separated by a single comma.

Submit in Judge only the output of the program.

Examples

Input	Output
`On January 1 , 1533 , Michael Angelo, then fifty-seven years old, writes…`	`11101111 10111011 10111111 1001111 1101110 100000 1001010 1100001 1101110 1110101…`
`Two households, both alike in dignity, In fair Verona, where we lay our scene…`	`1010100 1110111 1101111 100000 1101000 1101111 1110101 1110011 1100101 1101000…`

Hints

- Create a string variable holding the path to the file. If, for example, the file is located in "C:\"

```
        String path = "D:\\input.txt";
```

- Use try-with-resources to open the file and to be sure that it will be closed after you are done with it

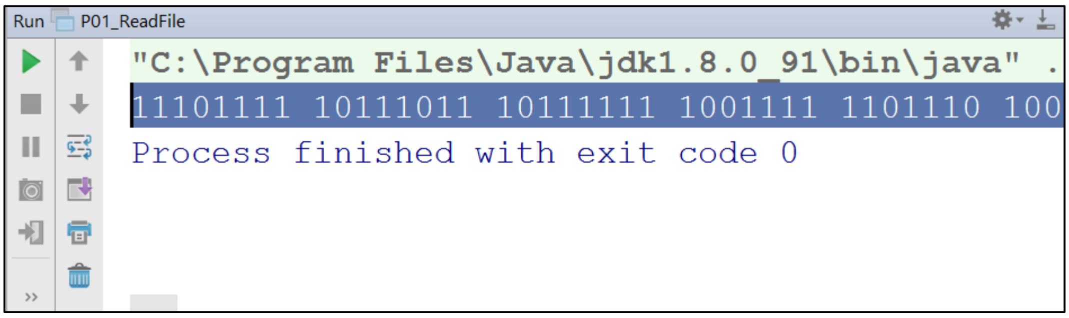
```
try (FileInputStream fileStream = new FileInputStream(path)) {

} catch (IOException e) {
    e.printStackTrace();
}
```

- Use the **read()** method to read each byte of the file until it returns -1

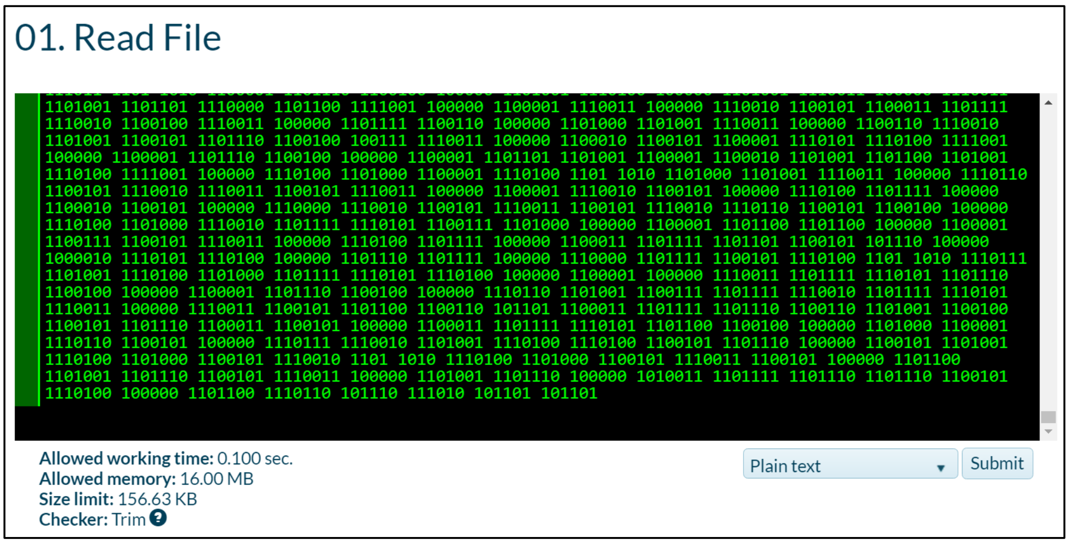
```
try (FileInputStream fileStream = new FileInputStream(path)) {
    int oneByte = fileStream.read();
    while (oneByte >= 0) {
        System.out.printf("%s ", Integer.toBinaryString(oneByte));
        oneByte = fileStream.read();
    }
} catch (IOException e) {
    e.printStackTrace();
}
```

- Select the output of the program and copy it [Ctrl + C]



```
"C:\Program Files\Java\jdk1.8.0_91\bin\java" .
11101111 10111011 10111111 1001111 1101110 100
Process finished with exit code 0
```

- Paste the output in Judge



01. Read File

```
1101001 1101101 1110000 1101100 1111001 100000 1100001 1110011 100000 1110010 1100101 1100011 1101111
1110010 1100100 1110011 100000 1101111 1100110 100000 1101000 1101001 1110011 100000 1100110 1110010
1101001 1100101 1101110 1100100 100111 1110011 100000 1100010 1100101 1100001 1110101 1110100 1111001
100000 1100001 1101110 1100100 100000 1100001 1101101 1101001 1100001 1100010 1101001 1101100 1101001
1110100 1111001 100000 1110100 1101000 1100001 1110100 1101 1010 1101000 1101001 1110011 100000 1110110
1100101 1110010 1110011 1100101 1110011 100000 1100001 1110010 1100101 100000 1110100 1101111 100000
1100010 1100101 100000 1110000 1110010 1100101 1110011 1100101 1110010 1110110 1100101 1100100 100000
1110100 1101000 1110010 1101111 1110101 1100111 1101000 100000 1100001 1101100 1101100 100000 1100001
1100111 1100101 1110011 100000 1110100 1101111 100000 1100011 1101111 1101101 1100101 101110 100000
1000010 1110101 1110100 100000 1101110 1101111 100000 1110000 1101111 1100101 1110100 1101 1010 1110111
1101001 1110100 1101000 1101111 1110101 1110100 100000 1100001 100000 1110011 1101111 1110101 1101110
1100100 100000 1100001 1101110 1100100 100000 1110110 1101001 1100111 1101111 1110010 1101111 1110101
1110011 100000 1110011 1100101 1101100 1100110 101101 1100011 1101111 1101110 1100110 1101001 1100100
1100101 1101110 1100011 1100101 100000 1100011 1101111 1110101 1101100 1100100 100000 1101000 1100001
1110110 1100101 100000 1110111 1110010 1101001 1110100 1110100 1100101 1101110 100000 1100101 1101001
1110100 1101000 1100101 1110010 1101 1010 1110100 1101000 1100101 1110011 1100101 100000 1101100
1101001 1101110 1100101 1110011 100000 1101001 1101110 100000 1010011 1101111 1101110 1101110 1100101
1110100 100000 1101100 1110110 101110 111010 101101 101101
```

Allowed working time: 0.100 sec.
Allowed memory: 16.00 MB
Size limit: 156.63 KB
Checker: Trim

2. Read a File

Read the file named "input.txt" that is provided for this exercise and write all its content to a file while skipping any punctuation. Skip the following symbols: ',', '.', '!', '?'.

Submit in Judge only the output of the program.

Examples

Input	Output

`On January 1 , 1533 , Michael Angelo,`
`then fifty-seven years old, writes`
	`On January 1  1533  Michael Angelo then`
	`fifty-seven years old writes`

`Two households, both alike in dignity.`
`In fair Verona, where we lay our scene.`
	`Two households both alike in dignity`
	`In fair Verona where we lay our scene`

Hints

- Create a **`FileInputStream`** to read the file
- Create a **`FileOutputStream`** to write to a file
- Create a list, containing all characters that you need to skip and check if the current char is contained in it

```
if (!punctuation.contains((char)oneByte)) {
    out.write(oneByte);
}
```

3. Copy Bytes

Read the file named "input.txt" and write to another file every character's ASCII representation.

Write every space or new line as it is, e.g. as a space or a new line.

Examples

Input	Output

`Two households, both alike in dignity.`
`In fair Verona, where we lay our scene.`
	`84119111 104111117115101104114…`
	`73110 10297105114 86101114111110744 1…`

Hints

- Get the value of every byte as string and then write its every digit one by one

```java
String digits = String.valueOf(oneByte);

for (int i = 0; i < digits.length(); i++) {
    out.write(digits.charAt(i));
}
```

4. Extract Integers

Read the file provided, named "input.txt" and extracts all integers that are not a part of a word in a separate file. A valid integer is surrounded with white spaces.

Submit in Judge only the output of the program.

Examples

Input	Output

`Households 2 , 2 alike in 3nity,`
`In fair Verona 4, where we lay our…`
	2
	2

`On January 1 , 1533 , Michael Angelo,`
`then fifty-seven years old, writes`
	1
	1533

Hints

- Wrap a **`FileInputStream`** in a Scanner and use the methods, **`hasNext()`**, **`hasNextInt()`** and **`nextInt()`**

```java
while (scanner.hasNext()) {
    if (scanner.hasNextInt()) {
        out.println(scanner.nextInt());
    }

    scanner.next();
}
```

5. Write Every Third Line

Read the file provided, named "input.txt" and write to another file all lines which number is divisible by 3. Line numbers start from one.

Submit in Judge only the output of the program.

Examples

Input	Output
On January 1 , 1533 , Michael Angelo, then fifty-seven years old, writes from Florence to Tommaso de' Cavalieri, a youth of noble Roman family,	then fifty-seven years old, Tommaso de' Cavalieri,
Two households, both alike in dignity, In fair Verona, where we lay our scene, From ancient grudge break to new mutiny	In fair Verona, break to new mutiny

Hints

- To get the functionality to read and write lines use **`BufferedReader`** and **`PrintWriter`**
- Wrap streams appropriately

```
BufferedReader in = new BufferedReader(new FileReader(inputPath));
PrintWriter out = new PrintWriter(new FileWriter(outputPath));
```

6. Sort Lines

Read the file provided, named "input.txt" and write to another file all lines which number is divisible by 3. Line numbers start from one.

Submit in Judge only the output of the program.

Examples

Input	Output
C A B D	A B C D

Input	Output
5 2 4 1	1 2 4 5

Hints

- To read all lines together use **`Files.readAllLines()`**

```
List<String> lines = Files.readAllLines(path);
```

- To sort the list of strings use **`Collections.sort()`**

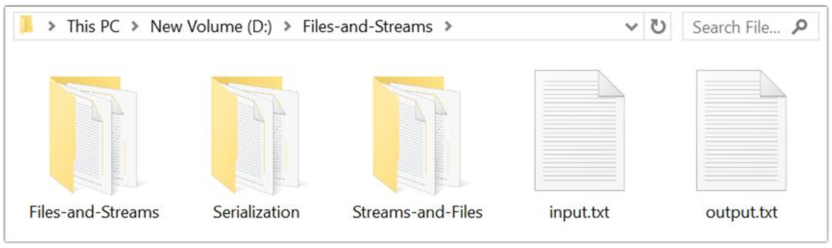
```
Collections.sort(lines);
```

7. List Files

You are provided a folder named "Files-and-Streams". Create a program that lists the names and file sizes (in bytes) of all files that are placed directly in it (do not include files in nested folders).

Submit in Judge only the output of the program.

Examples

Input	Output
 This PC > New Volume (D:) > Files-and-Streams > Search File... Files-and-Streams Serialization Streams-and-Files input.txt output.txt	input.txt: [size in bytes] output.txt: [size in bytes]

Hints

- Use the **File** class and its method **listFiles()**

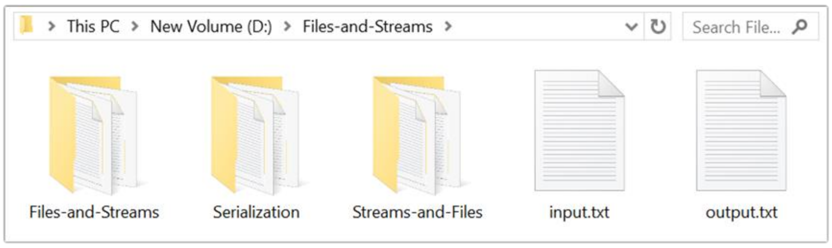
8. Nested Folders

You are provided a folder named "Files-and-Streams". Create a program that lists the names of all directories in it (including all nested directories).

On the last line, print the count of all folders, including the root folder.

Submit in Judge only the output of the program.

Examples

Input	Output
 This PC > New Volume (D:) > Files-and-Streams > Search File... Files-and-Streams Serialization Streams-and-Files input.txt output.txt	… Streams-and-Files Files-and-Streams Streams-and-Files Serialization Streams-and-Files **[folder count]** folders

Hints

- Use the **File** class and its method **listFiles()**

9. Serialize Custom Object

Create a class called "Cube". It should have properties for color, width, height and depth.

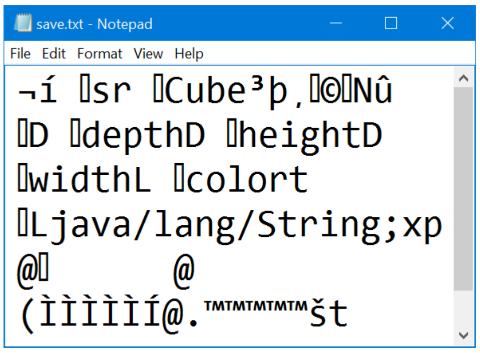
Create an instance of the class with the following values:

- Color: "green"

- Width: 15.3
- Height: 12.4
- Depth: 3.0

Serialize and deserilalize the instance created. When saved to a file the object should look like something like the example below.

Examples

Input	Output
(no input)	

save.txt - Notepad

File Edit Format View Help

```
¬í ▯sr ▯Cube³þ¸▯©▯Nû
▯D ▯depthD ▯heightD
▯widthL ▯colort
▯Ljava/lang/String;xp
@▯       @
(ÌÌÌÌÌÍ@.™™™™™št
```

Hints

- Create a class called Cube, which should implement the Serializable interface:

```java
class Cube implements Serializable {
    String color;
    double width;
    double height;
    double depth;
}
```

- Create a new instance of the Cube class and set its properties:

```java
Cube cube = new Cube();
cube.color = "green";
cube.width = 15.3d;
cube.height = 12.4d;
cube.depth = 3d;
```

- Use **`ObjectOutputStream`** to serialize the object:

```java
String path = "D:\\save.txt";

try (ObjectOutputStream oos =
             new ObjectOutputStream(new FileOutputStream(path))) {
    oos.writeObject(cube);
} catch (IOException e) {
    e.printStackTrace();
}
```