# Lab: Streams, Files and Directories

This document defines the exercises for the "Java Advanced" course @ SoftUni.

Please submit your solutions (source code) to all below-described problems in [Judge](https://judge.softuni.org/Contests/4012/Streams-Files-And-Directories-Lab-MLC)

**For these 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.**

## 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.

### 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 "D:\".



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



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



* Select the program's output and copy it [Ctrl + C]



* Paste the output in Judge.



## Write to 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: ",", ".","!", "?".

### 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.



## 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., a **space** or a **new line**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| On January 1 , 1533 , Michael Angelo, then fifty-seven years old, writes  … | 79110 749711011797114121 49 44 49535151 44 771059910497101108 6511010310110811144 116104101110 10210510211612145115101118101110 12110197114115 11110810044 119114105116101115 |

### Hints

* Get the value of every byte as a string and then write every digit one by one.



## 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 by **white spaces**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 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().



## Write Every Third Line

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

### 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, |

### Hints

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



## Sort Lines

Read the file provided, named "**input.txt**" and sort all lines. Write text to another text file. Use Path and Files Classes.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| C  A  B  D | A  B  C  D |

### Hints

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



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



## 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**).

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
|  | input.txt: [size in bytes]  output.txt: [size in bytes] |

### Hints

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

## 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.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
|  | …  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().

## 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 deserialize the instance created. When saved to a file, the object should look like something like the example below.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| *(no input)* |  |

### Hints

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



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



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

