



Module 02 – Piscine Java

IO, Files

Summary: Today you will learn how to use input/output in Java and implement programs to manipulate a file system

Version:

Contents

I	Foreword	2
II	Instructions	3
III	Exercise 00 : File Signatures	5

Chapter I

Foreword

Input/output operations play an important role in corporate system development. It is often necessary to implement functionality for loading and processing user files, sending various documents by mail, etc.

Apparently, input/output never boils down to working with a file system. Any client/server interaction between applications implies input/output operations. For example, Java Servlets technology used in web development enables to format HTML pages using `PrintWriter` class.

It is important to remember that the input/output functionality is not limited to Java IO stack. There are many libraries that greatly simplify interaction with data flows. Apache Commons IO is one of them.

Chapter II


Instructions

- Use this page as the only reference. Do not listen to any rumors and speculations about how to prepare your solution.
- Now there is only one Java version for you, 1.8. Make sure that compiler and interpreter of this version are installed on your machine.
- You can use IDE to write and debug the source code.
- The code is read more often than written. Read carefully the [document](#) where code formatting rules are given. When performing each task, make sure you follow the generally accepted [Oracle standards](#)
- Comments are not allowed in the source code of your solution. They make it difficult to read the code.
- Pay attention to the permissions of your files and directories.
- To be assessed, your solution must be in your GIT repository.
- Your solutions will be evaluated by your piscine mates.
- You should not leave in your directory any other file than those explicitly specified by the exercise instructions. It is recommended that you modify your .gitignore to avoid accidents.
- When you need to get precise output in your programs, it is forbidden to display a precalculated output instead of performing the exercise correctly.
- Have a question? Ask your neighbor on the right. Otherwise, try with your neighbor on the left.
- Your reference manual: mates / Internet / Google. And one more thing. There's an answer to any question you may have on Stackoverflow. Learn how to ask questions correctly.
- Read the examples carefully. They may require things that are not otherwise specified in the subject.
- Use "System.out" for output

- And may the Force be with you!
- Never leave that till tomorrow which you can do today ;)

Chapter III

Exercise 00 : File Signatures

	Exercise 00
File Signatures	
Turn-in directory : <i>ex00/</i>	
Files to turn in : *.java, signatures.txt	
Allowed functions : All Recommended types : Java Collections API (List<T>, Map<K, V> , etc.) InputStream, OutputStream, FileInputStream, FileOutputStream	

Input/output classes in Java are represented by a broad hierarchy. The key classes describing byte input/output behavior are abstract classes `InputStream` and `OutputStream`. They do not implement specific mechanisms for byte flows processing, rather delegate them to their subclasses, such as `FileInputStream`/`FileOutputStream`.

To understand the use of this functionality, you should implement an application for analyzing signatures of arbitrary files. This signature allows to define file content type and consists of a set of "magic numbers." These numbers are usually located in the beginning of the file. For example, a signature for PNG file type is represented by first eight bytes of a file that are equal for all PNG images:

89 50 4E 47 0D 0A 1A 0A

You need to implement an application that accepts the `signatures.txt` as an input (you should describe it on your own; the file name is explicitly stated in the program code). It contains a list of file types and their respective signatures in the HEX format. Example (specified format of this file must be adhered to):

PNG, 89 50 4E 47 0D 0A 1A 0A
GIF, 47 49 46 38 37 61

During execution, your program shall accept full paths to files on hard disk and keep the type which file signature corresponds to. The result of program execution should

be written to result.txt file. If a signature cannot be defined, the execution result is UNDEFINED (no information should be written into the file).

Example of program operation:

```
$java Program  
-> C:/Users/Admin/images.png  
PROCESSED  
-> C:/Users/Admin/Games/WoW.iso  
PROCESSED  
-> 42
```

Contents of result.txt file (there is no need to load this file as a result):

PNG

GIF

Notes:

- We can accurately determine the content type by analyzing the file signature, since the file extension contained in the name (e. g. image.jpg) can be changed by simply renaming the file.
- The signatures file shall contain at least 10 different formats for analysis.