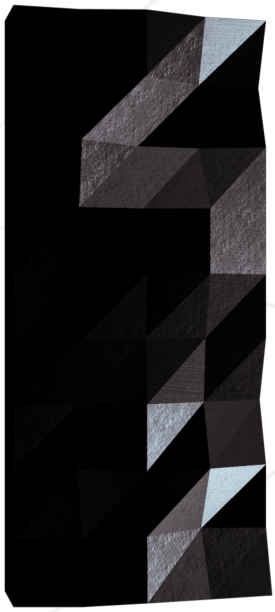


Module 04 – Piscine Java

JAR

Summary: Today you will learn to create library archives and use external libraries

Contents

Chapter I

Foreword

Any Java library or framework is a set of JAR files—archives of compiled classes and other resources.

Thus, the goal of any Java developer is to correctly organize the source code and then transfer the compiled JAR archive with implemented functionality to another programmer.

There is a range of tools to organize the project building life cycle and its structure. Nevertheless, certain skills of using standard Java infrastructure tools guarantee the correct understanding of how out-of-the-box and popular solutions work.

Today, you will manually build an application with external libraries. This is your first step in learning Maven—the most popular building system.

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 : Packages

	Exercise 00
Packages	
Turn-in directory : *ex*00/	
Files to turn in : ImagesToChar-folder (exclude target)	
Allowed functions : All	

Code can be organized on different levels. Packages are one of the code organization methods where classes are located in individual folders.

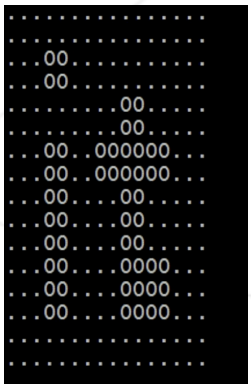
Now your task is to implement functionality that prints a two-colored image in the console.

An example of a black-and-white BMP image (this format is mandatory for the solution). Image size is 16*16 pixels.

You can find this image in the project page.

Your application shall accept input parameters corresponding to characters that should be displayed in place of white and black pixels. Another main function startup parameter is the full path to the image on your disk.

If "." character is used for white color and "0" for black, the image in the console may look as follows:



Application logic must be distributed between different packages and have the following structure:

ImagesToChar - project folder
- src - source files
  - java - files of Java source code
    - edu.school21.printer - a series of main packages
      - app - a package that contains classes for startup
      - logic - a package that contains the logic for converting an image into an array of characters
- target - compiled .class files
  - edu.school21.printer ...
- README.txt

- README.txt file must contain instructions for compiling and starting your source code from the console (non-IDE). Instruction is written for the state where the console is opened in the project's root folder.

Chapter IV

exercice 01 : First JAR

	Exercise 01
First JAR	
Turn-in directory : $ex01/$	
Files to turn in : ImagesToChar-folder (exclude target)	
Allowed functions : All	

Now you need to create a distribution package of the application—a JAR archive. It is important that the image be contained in that archive (a command-line parameter for the full path to the file is not required in this task).

The following project structure shall be adhered to:

ImagesToChar - project folder
- src - source files
  - java - files of Java source code
    - ...
  - resources - a folder with resource files
    - image.bmp - the displayed image
  - manifest.txt - a file containing the description of the initial point for archive startup
- target - compiled .class files and archive
  - edu.school21.printer ..
  - resources
  - images-to-chars-printer.jar
- README.txt

- Archive and all compiled files shall be put in target folder during assembly (without a manual file transfer; you may apply cp command to the resource folder).
- README.txt file should also contain information on the archive assembly and startup.

Chapter V

exercice 02 : JCommander & JCDP

	Exercise 02
JCommander & JCDP	
Turn-in directory : $ex02/$	
Files to turn in : ImagesToChar (excludelib and target)	
Allowed functions : All	

Now you should use external libraries:

- JCommander for the command line.
- JCDP or JColor for using colored output

Archives with these libraries shall be downloaded and included in the previous task's project.

Now application startup parameters shall be processed with JCommander tools. The image should be displayed using the "colored" output option of JCDP library.

Required project structure:

```
ImagesToChar - project folder
├── lib - external library folder
│   ├── jcommander-*.**.jar
│   └── JCDP-*.*.*.jar/JCOLOR-*.*.*.jar
├── src - source files
├── target - compiled .class files and archive
│   ├── edu.school21.printer
│   ├── com/beust ... - .class files of JCommander library
│   ├── com/diogonunes ... - .class files of JCDP library
│   ├── resources
│   └── images-to-chars-printer.jar
└── README.txt
```

README.txt file shall also contain the information about including external libraries in the final assembly.

Example of program operation:

```
$ java -jar images-to-chars-printer.jar -- white=RED --black=GREEN
```