**Files: extensions and MIME[[1]](#footnote-1) type**

**Object Oriented Programming & Java project**

### Goals

The OOP - Java module project makes it possible to implement the main elements of the content of the module as part of the design of a small application. The work is to be done in pairs (exceptionally alone, for special situations[[2]](#footnote-2)).

### Context of the project

A common belief is that the extension of a file determines its type: as long as the effective type of a file and its extension are consistent, this statement is true, for against as soon as the extension is no longer consistent with the content, it is necessary to rely on the MIME type to evaluate a file (this is generally the case for content exchanges multimedia via mail or the way files are processed by browsers when visit a web page). Under Linux, the command "file" in command line allows to know the MIME type of a file.

The objective of the project is to create a small application allowing to detect anomalies on data files[[3]](#footnote-3) of the user, so either to check a specific file (detection of empty files, detection of inconsistency between extension and MIME type, in-depth analysis of content of a file), or to explore and analyze a set of files from a folder.

The main actions of your software are:

* checking a machine file (based on the extension and MIME type of the file)
* the complete exploration (a folder and all of its sub-folders) of a directory with verification of each file.
* The validation of the extension and of the MIME type on the basis of a list of signatures to create (example: a file with extension ".html" and MIME type "text / html" should to start[[4]](#footnote-4) with "<! DOCTYPE html>"; a script file with extension ".sh" and type MIME "application / x-shellscript" or "application / x-shell" should start[[5]](#footnote-5) per "#! / Bin /" (for example #! / Bin / bash).
* For certain types of files, checking their integrity: principle: the program can start the decompression of a compressed file (zip type for example) in order to ensure that it is indeed a zip file, similarly, the files of office suites (docx, xlsx, pptx / odt, ods, odp) are actually zip files that can be checked from the same way. For images, we can retrieve the dimensions to ensure that it is indeed an image, etc.

Files of size 0 (zero) bytes should be reported to the user, likewise, your program will have to detect if the extension of a file does not correspond to its MIME type; finally Yes the content of a file does not correspond to the information detected a priori ("extension and type MIME ”), the program should detect it and inform the user.

In console (terminal) mode "CLI[[6]](#footnote-6) ": the parameters expected on the command line are: the name of the file to be processed or the name of the folder to explore: you will use a parameter additional to specify the type of entry ("-f" for a file, "-d" in the case of a directory). The program must display the result of its analysis directly in the console. If no parameter is specified (or option "-h" for "help"), the program displays the help and the possible options. An additional option "-s" allows to save the result of an analysis in a file to be specified by the user.

In GUI graphic mode[[7]](#footnote-7) ": exploring any tree structure[[8]](#footnote-8) files will list in the graphical interface all the files (but not the directories and sub-directories) and their locations. For each file, the level of compliance between the extension, MIME type and content will be indicated (problematic files will be visually identifiable).

The signature database, ie: extensions, associated MIME types and identification content if there

if applicable, will be stored as a CSV file.

Some execution scenarios (fictitious examples of launching your 2 programs[[9]](#footnote-9)):

java -jar cli.jar

java -jar cli.jar -d.

java -jar cli.jar -f test.html

java -jar cli.jar -d. -s analysis

java -jar gui.jar

***Explanation:*** the first 4 commands concern console mode (also called mode

terminal or command window); the last command launches the graphical interface.

* the first line displays the modes of use of your software in console mode (ie possible options and their role);
* the second line lists and analyzes all the files **from** the specified folder ( *“-d” = directory* ) [here from the current folder (".")] by browsing the whole tree structure sub-folders, in console mode;
* the third line takes as input the file “test.html” ( *“-f” = file* ) and displays on the screen the result of its verification, in console mode;
* the fourth line takes as input the current folder ". "And save the result of the verification in an “analysis” file ( *“-s” = save* ) in console mode;
* the last line corresponds to the launch of the graphical interface.

***NB:*** the format for saving the results of an analysis is left to your initiative, but it is

necessary to be able to exploit this result outside of your software: special attention is

therefore, to bring to this choice.

In graphical mode, it is requested to be able to keep a serialized version of the analysis between 2 program executions in order to be able to dissociate the analysis (possibly long), from display of identified problems. When the software is launched in graphic mode, it is therefore the last result obtained which is displayed, with the possibility of launching a new analysis.

Possible extensions: you can plan improvements to your solution but **only** if everything else is complete. You will therefore favor the quality of the realization over the quantity of features.

### Planning and advice for the presentation

***Planning***

* Identify the main sub-tasks of the project to be carried out, their level of priority, the distribution of roles within the pair as well as the corresponding schedule for each task (period and duration). You can create a small summary table or (better), create a GANTT diagram[[10]](#footnote-10) of your project. In all cases, the table or diagram of Gantt must be returned in png format (capture or export) before **(Date & Time to be determined)**: **1 point**
* progress points in December (compulsory presence): **3 points** (for information, one level of achievement of approximately 50% is expected at the last progress point)
* Defense (5 min) and demonstration (5 min): **6 points** : **(Date & Time to be determined).**

***Defense: 5 slides maximum, 5 minutes maximum***

* the title slide will present the binomial, the context, the subject. (ie: the cover page will be compact),
* the other slides must present the specific features of the project team’s work, therefore no "obvious" information (ex. details of the subject, personal progress, ...) should be mentioned,
* the concluding slide will highlight the level of completion of the project (points treated and untreated specifications and extensions if any),
* you will need to provide a pdf version of your slideshow in case.

The following information should be present: the distribution of tasks, the main design elements.

***NB:*** to be avoided ABSOLUTELY: unreadable UML class diagrams (too loaded, ...), programs (Java code), screenshots (since there is also a demonstration), the list of tools (eg Eclipse, etc.), ...

***Important:*** you must have your portable machine turned on, ready with all of the software already launched BEFORE entering the room. Your laptop should have a VGA port or you will need to provide a VGA adapter that matches your situation.

***Important:*** you will ensure an equitable distribution of your speaking time within the pair for both presentation and demonstration.

***Demonstration: 5 minutes maximum***

1. console mode,

2. graphic mode.

You will make sure to plan a scenario for the demonstration.

### Expected results and evaluation criteria

Completeness and quality of the project: **7 points** ("readme.txt" file, java code, javadoc, files

jar). Deliverables to be deposited on the educational platform before **(Date/Time to be determined)**.

* "readme.txt" file containing the names / first names / TD group / of the members of the project as well as specific useful information,
* project report (minimum 5 pages, maximum 10 pages): **3 points** (content and form will be evaluated). The following 2 files are to be returned: 1) the word processing document ( **docx or odt** ), 2) the **pdf** version of your report
* sets of project source files (.java),
* javadoc,
* The 2 jar files in java 1.8 compatible version.

***Important :*** All files and sub-folders to be submitted must be placed in a directory unique with the 2 names of the binomial (in the form NOM1\_NOM2) to compress into one file in zip format which will be uploaded to the educational course platform *(moodle)*.

***Some indications for the realization***

The objective of the project is to allow you to implement, as part of an achievement concrete, the concepts of OOP and Java discussed during the module. There is no need to want be exhaustive in dealing with the many situations present in the files handled. Of even, it is possible to use external libraries corresponding to your needs.

First part

To start the project, it is recommended to deal with a single case first file by relying first only on the extension, then on the MIME type and finally on a signature to be defined. Checking the integrity of certain files should only be covered in a second time.

Second part

Then, you can generalize your solution to a set of files placed in a directory and its subdirectories.

***Some indications for the report***

This document written using a word processor (MS Word or free office for example) must

allow to provide a complete report and an assessment of your work and its outcome. In particular, you will place the UML class diagram for your application there. Information planning and distribution of tasks are expected. Explanations on specific aspects of your solution. You can add some representative screenshots but don't overdo it not. You will detail the level of achievement of your achievement with also a look criticize the strengths and weaknesses that you have identified.

### Resources

* <https://www.iana.org/assignments/media-types/media-types.xhtml>
* <https://developer.mozilla.org/fr/docs/Web/HTTP/Basics_of_HTTP/MIME_Types>
* <https://developer.mozilla.org/fr/docs/Web/HTTP/Basics_of_HTTP/MIME_types/Common_types>
* <https://tools.ietf.org/html/rfc2045>

1. MIME: Multipurpose Internet Mail Extensions [↑](#footnote-ref-1)
2. Case of students in Terminal control, AJAC or odd number of students on the whole promotion by example [↑](#footnote-ref-2)
3. We will not consider the analysis of programs (of type .exe, .jar, .class, etc.). [↑](#footnote-ref-3)
4. After possibly comments of type “<! -… ->” [↑](#footnote-ref-4)
5. After possibly lines of comments preceded by the character “#” [↑](#footnote-ref-5)
6. CLI: Command Line Interface [↑](#footnote-ref-6)
7. GUI: Graphical User Interface [↑](#footnote-ref-7)
8. With exploration of all the sub-directories [↑](#footnote-ref-8)
9. cli: command line interface / gui: graphical user interface [↑](#footnote-ref-9)
10. cf. <https://www.ganttproject.biz/download/free> [↑](#footnote-ref-10)