Lab 5

[valid 2020-2021]

Starting from this week...:

- Pay attention to exception handling (otherwise: -0.5 points)
- Create your own types of exceptions to report abnormal events related to application execution.
- Organize your classes and interfaces in packages.

Media Catalog
Write an application that can manage a catalog of multimedia items. An entry in this catalog might be *a song*, *a movie*, *a book*, *an image* or any item that has at least *a name* and *a path* in the local file system. (We may also consider specifying *a release year*, *a rating* and other additional data, for example the author of the book, etc.)

The main specifications of the application are:

Compulsory (1p)

- Create an object-oriented model of the problem. You should have at least the following classes: *Catalog* and two item classes at your choice. Consider using an interface or an abstract class in order to describe the items in a catalog.
- Implement the following methods representing *commands* that will manage the content of the catalog:
  - *add*: adds a new entry into the catalog;
  - *list*: prints the content of the catalog on the screen;
  - *play*: playback using the native operating system application (see the Desktop class);
  - *save*: saves the catalog to an external file (either as a text or binary, using *object serialization*);
  - *load*: loads the catalog from an external file.
- The application will signal invalid data (year, path, etc.) using a *custom exception*.

Optional (2p)

- Create a *shell* that allows reading commands from the keyboard, together with their arguments.
- Represent the commands using **classes instead of methods**. Use an interface or an abstract class in order to desribe a generic command.
  Implement the commands *add*, *list*, *save*, *load*, *play* (create the classes *AddCommand*, *ListCommand*, etc.).
- Implement the command *report*: create (and open) an HTML report representing the content of the catalog.
  Use a template engine such as FreeMarker or Velocity, in order to create the HTML report.
- The application will signal the commands that are not valid using a *custom exception*.
- The final form of the application will be an executable JAR archive. Identify the generated archive and launch the application from the console, using the JAR.

Bonus (2p)

- Use Apache Tika in order to extract metadata from your catalog items and implement the command *info* in order to display them.
- Create a graph G in which the catalog items represent the nodes, two nodes being connected if they share some common feature (for example, two songs that have the same artist, etc).
- Suppose that you want to play all the items in the catalog in as few days as possible, so that in one day you will not play two items that are connected in G. Create an algorithm that generates the *playlists* for each day.
- Create large graphs and test your algorithm.

Resources

- Basic I/O
- Exceptions
- Regular Expressions
- Setting the class path
- Creating, Importing, and Configuring Java Projects in Netbeans
- Packaging Programs in JAR Files

Objectives

- Use basic I/O streams.
- Understand the difference between byte and character streams, primitive and decorator streams.
- Use File I/O to works with files and directories.
- Handle exceptions using *try-catch-finally* blocks.
- Handle resources using *try-with-resources* statements.
- Create custom exceptions and chained exceptions.
- Use regular expressions and glob patterns
- Understand the notion of CLASSPATH.
- Use external JAR libraries.
- Package all project classes as an executable JAR file.