Programming Project 2018/19 Project Report
Title: Project report
Student: Francesco Piccoli
StudentID: 17117

I used the programming techniques as follows.

Technique	**Description**
1. Data IO (web)	I used Data IO (web) to read (BufferedReader, InputStreamReader, InputStream classes) the hmtl source code of the webpages (website: https://genius.com) of the songs in the list. Class: DownloadThread.
2. Data IO (local files)	I used Data IO (local files) to read(FileReader/BufferedReader classes) the csv files (songlist.csv, ignorelist.csv, emotionword.csv) and the lyrics (-clean.md), and to write (PrintWriter class) the -.html files, the files which count the words (-wordcount.csv, -.emotioncount.csv) and the -.clean.md files. Classes: Downloader; DownloadThread; HtmlCleaner; FrequencyCounter; EmotionWordCounter.
3. Multithreading	I used multithreading for downloading simultaneously files. A for loop in the Downloader class creates a Thread (an instance of the DownloadThread class which extends Thread) for every song in the songlist.csv file. In addition, I used multithreading in the Main class for the creation of a thread which makes sure that all the downloads have finished, namely, a Downloader object (Downloader class implements Runnable) is created and its method have finished to execute before executing the other instructions (clean, write…). This is because these latter instructions to be executed, need first some files to be created . Classes: Main; Downloader; DownloadThread.
4. JUnit testing	I used JUnit testing to check several things: If the number of -.clean.md files is equal to the number of -.html files. If a Thread (namely, an instance of the DownloadThread class) is created for every song in the list. If the CounterFactory class (factory design pattern) works correctly (returns the right class according to the String parameter). If the Log class creates a file (where it writes the log data) with the right name Classes: TestJUnit; TestRunner..
5. Logging using java.util.logging	I used Logging (java.util.logging, not Log4j) to write: the various steps of execution of the program (downloading, cleaning, writing (-wordcount.csv, -.emotioncount.csv), if a custom exception occurred and finally the time of execution of the program. Class: Log; Main; WrongFileNameException; NoConnectionException.

6. Exception throwing	I created two custom exception classes: WrongFileNameException in case there is no song with the given name(url) and NoConnectionException in case there is no internet connection or wrong domain name (since both the problems throw the same exception in general in java) Classes: WrongFileNameException; NoConnectionException.
7. Generics and Collections	I used Maps, Lists and Set in several classes to find the frequency of the words, to make lists of songs and files… Classes: Downloader; HtmlCleaner; FrequencyCounter; EmotionWordCounter.
8. Design patterns (use at least three design patterns)	I used the Singleton, the Factory and the Builder design patterns. Singleton is used in the Downloader class in order to have only one instance of the class downloader that performs the download. Factory is used in the classes of the Counter package, since FrequencyCounter and EmotionWordCounter classes have somehow similar functionality, it was useful to have an interface (Counter) which contains some common methods. Builder is used in the Download package, since both the Downloader class and DownloadThread class "belong" to the same "family". Classes: Downloader; Download package; Counter package.
9. Regular Expressions	I used regular expressions to find the url/link of the website from the name and artist of the song in the songlist.csv file, to clean the -html files and keep just the lyrics of the songs which is then saved in -.clean.md files and to count the frequency of the words. Classes: Downloader; HtmlCleaner; FrequencyCounter; EmotionWordCounter.
10. Advanced Inheritance	I used Advanced Inheritance in the Counter package to create the factory design pattern (to implement an interface), to extend Thread in the DownloadThread class, to implement the runnable interface in the Downloader class and in the custom exception which extends one of the many Exception class subclasses. Classes: Counter package; Downloader; DownloadThread; Exception Package;
11. Javadoc documentation	I used java documentation in every class to explain the usefulness of the methods/classes. Classes: all the classes.