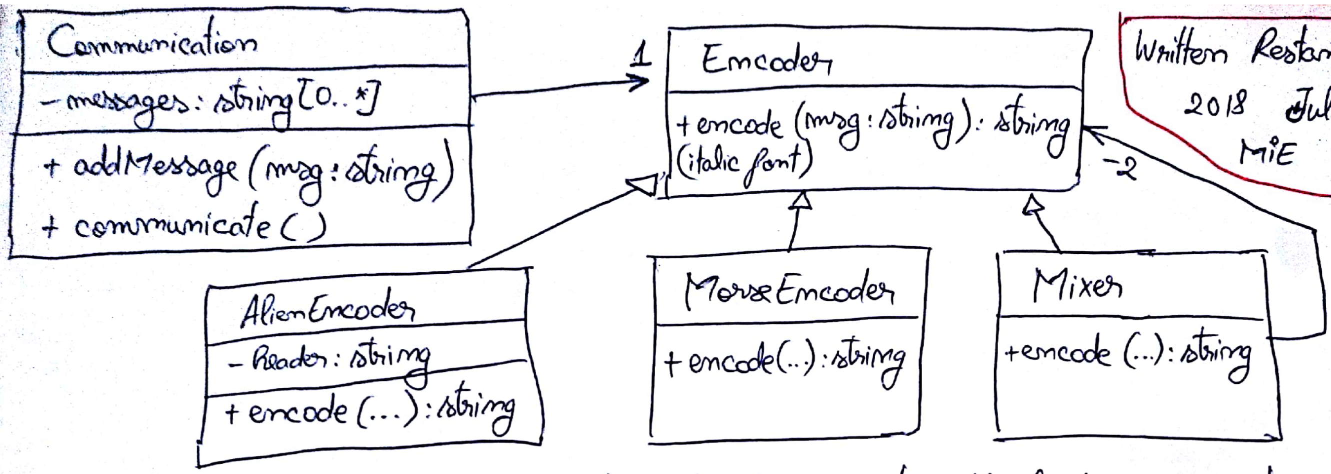
Written Restanță OOP
2018 Iuliana Bocicor
MIE



–2 = contains two private Encoders*

AlienEncoder – encode( ): appends at end of parameter the header and returns result

MorseEncoder – [illegible] : replaces each character with '.' and returns result

Mixer –||– : alternates between two encoded messages (~ ~~gt~~ encode parameter with first encoder in s1, –||– with second encoder in s2, ~~one character~~ one character from s1, one from s2, repeat... until new string is formed

"hello" → "helloET" $\xrightarrow{Mixer}$ h.e.l.l.o.ET
→ .....

Two 'Communication': 1) with MorseEncoder
2) with Mixer composed of MorseEncoder and AlienEncoder{"PS:D"}

- each has two messages
- call communicate for each
- manage memory

Restanță 2018 OOP
MiE

Screenwriting

Write an application which simulates the writing of a television episode by professional screenwriters, as follows:

1. The information about the screenwriters working on the episode is in a text file. Each writer has a **name** (string) and his/her expertize – can be *Assistant, Junior* or *Senior* (string). This file is manually created and it is read when the application starts.
2. Another file contains information about the ideas that were proposed by the writers. Each **Idea** has **a description, a status** (can be *proposed* or *accepted), **the creator*** – the name of the writer who created it and **the act** – the act it belongs to in the narrative (it can only be 1, 2 or 3). These are read when the application starts.
3. When the application is launched, a new window is created for each writer, having as title the writer's name. The window will show all the ideas launched so far (description, status, creator, and act), sorted by act and creator. **(1.5p)** 1.25
4. Any writer can add a new idea, by inputting the idea's description and act and pressing a button "Add". The idea's creator will automatically be the name of the writer who added it and the status will be *proposed*. This operation fails if the description is empty, if the act is not 1, 2 or 3 or if there is another idea with the same description in the same act. The user will be informed if the operation fails. **(1.5p)**
5. An idea can be removed only if it is **not** accepted. Only *senior writers* can remove ideas. **(1.25p)**
6. Senior writers can revise ideas and accept them, by selecting the idea and pressing a button "Accept". This button is activated only if the status of the selected idea is *proposed*. When an idea is accepted, its status changes to *accepted*. **(1p)**
7. When a modification is made by any writer, all the other writers will see the modified list of ideas. **(2p)**
8. Each writer has a button "Save plot", which will save to a file the entire plot of the episode: this will contain only the accepted ideas (description), their writers (in brackets) and will be created by acts (E.g. Act 1 – all ideas from act 1, on separate lines; Act 2 – the same; Act 3 – the same). **(0.75p)**

Observations

9.75

1. **1p - of**
2. Specify and test the following functions (repository or controller):
   a. Function which adds an idea. **(0.5p)**
   b. Function which removes an idea. **(0.5p)**
3. Use a layered architecture. If you do not use a layered architecture, you will receive 50% of each functionality.
4. If you do not read the data from file, you will receive 50% of functionalities 3, 4, 5 and 6.

Non-functional requirements

1. Use STL to represent you data structures.
2. Use objects *Idea* and *Screenwriter* to represent the necessary data.
3. Use a class *ScreenwritingRepository* to manage your screenwriters and their ideas.
4. Create a custom defined exception class to handle the constraints for the requirements 4 and 5 and use objects of this class.

You are allowed to use Qt Designer.
You are allowed to use the following sites for documentation, **but nothing else:**
- http://doc.qt.io/qt-5/
- http://en.cppreference.com/w/
- http://www.cplusplus.com/