Assignment 5: Software Design

In this assignment, you will design a [cryptogram]() game which uses a [simple substitution cipher](), with the requirements listed below. To do so, you should follow the same approach that we present in the P3L2 lesson. That is, analyze the requirements to identify and refine **(1) classes, (2) attributes, (3) operations, and (4) relationships in your design**. Just to be completely clear, **your task is to design the system, not to implement it**. The requirements for the application are listed below, in the 'Requirements' section. Please note that not every requirement will be fully and directly represented in your design. For instance, you do not have to show any purely GUI specific classes, if they are *only* doing user display and input.

Your design should be expressed using a UML class diagram, and the level of detail of the design should be analogous to the level of detail we used throughout the whole P3L2 lesson. (Do not limit your design to only the elements focused on in the final video.) You must provide enough details for the design to be **self contained** and to allow us to assess whether the design suitably **realizes all system requirements**.

To help with this, you must also provide a "design information" document in which you concisely describe, **for each of the requirements listed below**, how that requirement is either realized in your design, or why it does not directly affect the design and is not shown. Copy the list of requirements, and add your explanation for each. For example:

…
2. The application must contain a list of items in object 'X'. Object 'Y' must use the list for action 'A'.
To realize this requirement, I added to the design a class 'X' with list 'Z' of items. Class 'Y' has a relationship to class 'X' and uses it in method 'A' to….
...
16. The User Interface (UI) must be intuitive and responsive.
This is not represented in my design, as it will be handled entirely within the GUI implementation.

This explanation should be clear enough to allow us to follow the rationale of your design and **how it will fulfill each specified requirement**, including any that are not directly visible in your class diagram. You can also provide in the document additional information about your design, such as assumptions or rationale for some design decisions.

You can use any UML tool to create your design. If you are not familiar with any specific tool, we recommend that you ask on Piazza for suggestions. In fact, on Piazza there is already some discussion about that.

Requirements

1. A user will be able to choose to log in as a specific *player* or create a new player when starting the application. For simplicity, any authentication is optional, and you may assume there is a single system running the application.
2. The application will allow *players* to (1) create a cryptogram, (2) choose a cryptogram to solve, (3) solve cryptograms, (4) view their *list of completed cryptograms*, and (5) view the *list of cryptogram statistics*.
3. A cryptogram will have an encoded phrase (encoded with a simple substitution cipher), a solution, and a maximum number of allowed solution attempts. A cryptogram will only encode alphabetic characters, but may include other characters (such as punctuation or white space) in the puzzle, which will remain unaltered. Capitalization is preserved when encoded.
4. To add a player, the user will enter the following player information:
    a. A first name
    b. A last name
    c. A unique username
    d. An email
5. To add a new cryptogram, a player will:
    a. Enter a unique cryptogram puzzle name.
    b. Enter a solution (unencoded) phrase.
    c. Choose different letters to pair with every letter present in the phrase.
    d. View the encoded phrase.
    e. Enter a maximum number of allowed solution attempts.
    f. Edit any of the above steps as necessary.
    g. Save the complete cryptogram.
    h. View a confirmation that the name assigned to the cryptogram is unique and return to the main menu.
6. To choose and solve a cryptogram, a player will:
    a. Choose a cryptogram from a list of all unsolved cryptograms.

    XH: Cryptomanager manage
    viewYourSolution: show your solution entire string, and match that

    b. View the chosen cryptogram and number of incorrect solution submissions (starting at 0 for a cryptogram that has not previously been attempted).
    c. Match the replacement and encrypted letters together, and view the resulting potential solution.
    d. When all letters in the puzzle are replaced and they are satisfied with the potential solution, submit their answer.
    e. Get a result indicating that the solution was successful, or incrementing the number of incorrect solution submissions if it was unsuccessful.

    f. At any point, the player may return to the list of unsolved cryptograms to try another.
    g. If the number of incorrect solution attempts increases to the maximum allowed number of solution attempts, they will get a result that the cryptogram game was lost, and this cryptogram will be moved to the completed list.

7. The list of unsolved cryptograms will show the cryptogram's name and the number of incorrect solution attempts (if any) for that player.
8. The list of completed cryptograms for that player will show the cryptogram's name, whether it was successfully solved, and the date it was completed.
9. The list of cryptogram statistics will display a list of cryptograms in order of creation. Choosing each cryptogram will display the date it was created, the number of players who have solved the cryptogram, and the username of the first three players to solve the cryptogram.
10. The User Interface (UI) must be intuitive and responsive.
11. The performance of the game should be such that players does not experience any considerable lag between their actions and the response of the game.

Submission Instructions

To submit your assignment, you should do the following:

- Create a directory called `Assignment5` in the usual **personal GitHub repository we assigned to you**. This is an **individual assignment**; do not use your new team repositories.
- Save your UML class diagram in the `Assignment5` directory as a PDF file named "`design.pdf`". **Important:** Make sure to open your PDF after generating it and double check it, as we had a number of cases of students not realizing that the conversion to PDF had not worked as expected.
- Save the "design information" document in the same directory, in markdown format, and name it "**`design-information.md`**".
- Commit and push your file(s) to your remote repository.
- Submit the commit ID for your solution on Canvas.