University of Windsor
School of Computer Science
https://cs.uwindsor.ca

Master of Applied Computing
COMP-8117
Advanced Software Engineering Topics

Dr. Aznam YACOUB – https://www.y-az.one
519-253-3000 (ext. 3781) - aznam@uwindsor.ca

Lab 2 : UML and Design (1%)

Goal

In this lab, you'll learn how to create high-level and low-level design from specifications. You'll also learn to use different UML diagrams for design purpose.

- Week 1 : Training (Non-Evaluated)
- Week 2 : Mandatory Part

Scenario

ToyToy Inc. is a company which edits old-school video games on new platforms. They recently acquired the license of a well-known game : Tetris, and they would like you to develop a new version of this game for the next-gen platforms (PS5 and Xbox Series X). They provides to you the following specifications :

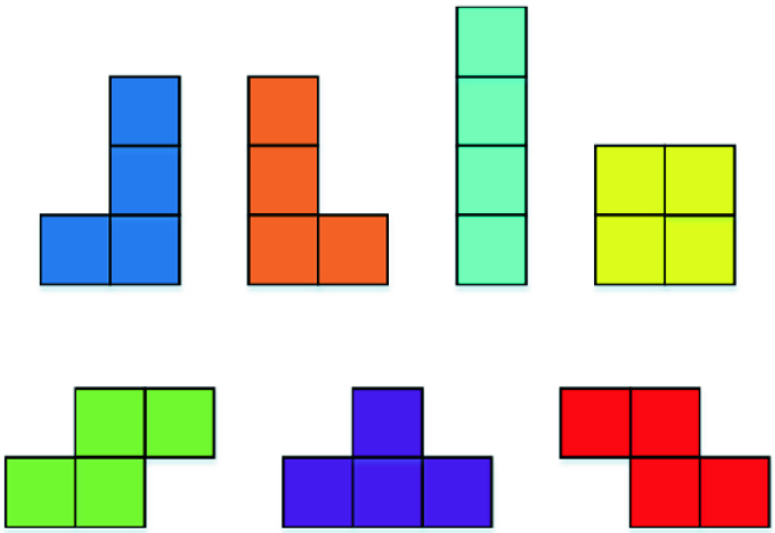
In *Tetris*, players complete lines by moving differently shaped pieces (tetrominoes), which descend onto the playing field. The completed lines disappear and grant the player points, and the player can proceed to fill the vacated spaces. The game ends when the playing field is filled.

The longer the player can delay this inevitable outcome, the higher their score will be.

The game zone is a 10x20 grid (20 lines, 10 columns), in which a tetramino falls until reaching the lowest available and possible line. The player can controls the falling tetramino using left, right, and down keys (change of the direction or instant placement), and space to rotate it (in the clockwise direction). The player can also decide at any moment (while the current tetramino is moving) to store it in a hold space by pressing the H key. If a tetramino is already hold and the player presses H, the current tetramino is swapped by the hold tetramino.

The speed of the tetramino is given in lines/sec. Each time a line is completed, the player get : 20 + 3*combo_lines points (where combo_lines represent the number of lines completed in the same time). Every 1000 points, the level changes. The speed is increased by two times, meanwhile the music changes for a more stressful one. Each music are so defined by : a stress value (1-10), and a filename.

Each tetramino has a specific shape and color :



The color is expressed using the RGB color model.

The order of tetraminos follows a specific pattern depending on the level (this pattern will be described in Appendix A).

The user interface of the gamescreen is composed by 4 zones : the gamezone (the grid), a zone on the right annoucing the next tetramino (up to 3 depending on the level), a zone on the left giving the current score and level, and one indicated the hold tetramino.

When the player launches the game, there is a splash screen with an intro music, then a main menu in which the player is asked to press the space bar and then the game.

The font used for the interface is Arial, 16pt. Colors of the UI is provided in Appendix B.

Training part

For this exercise, select one software on your computer and specify some functional requirements (eg: for a web-browser, you can specify the navigation feature, display page feature...).

1. In your opinion, what kind of computational problems (representations, computations, structures, algorithms, ...), do you have to solve when you read these specifications ? What details are not expressed in these specifications ?

2. Rewrite the specifications from the computational point of view in an informal way.

3. Represents these new specifications using Class Diagrams and Sequence Diagrams. What do you notice ?

4. Implement in JAVA these diagrams without assuming more than the informations represented in your diagrams. What do you notice ?

Mandatory Part

In this part, you'll work on the scenario mentioned above.

1. Choose a software architecture to design your application. Why do you choose this architecture ?

2. Transform the specification into a high-level design. You'll make sure that your architecture is visible.

3. Choose a programming language and a framework. Create low-level class diagrams and sequence diagrams representing the design of your software. You'll make sure that your architecture is still visible.

4. Write the corresponding code. Evalute the purcentage of work overload (how many details were not present in the diagram and that you had to add during the implementation).

Optional Part (1% extra credit)

1. In this part, you're required to do research about the GoF Design Patterns. For each GoF Design Patterns, make some research about it. For each of them, summarize the problem they solve, the solution they provide, advantages and drawbacks.