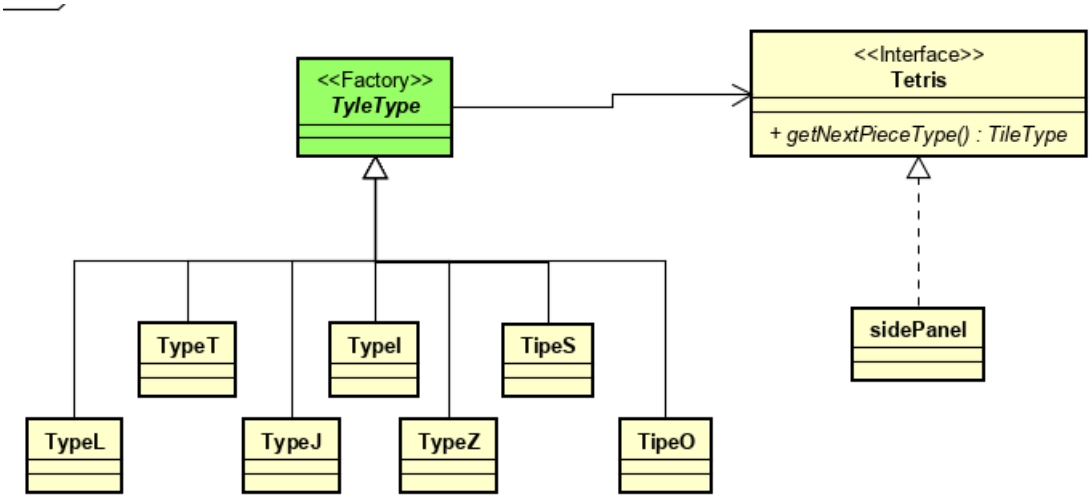
Name: Gustavo Hammerschmidt.

1. (30 Points) We already know that we can create a lunar lander application of the pipe-and-filter architecture style from three independent Java classes: GetBurnRate, CalcNewValues, and DisplayValues. As mentioned in class, they can be run as: java GetBurnRate | java CalcNewValues | java DisplayValues. Re-write Class CalcNewValues above using C++ (e.g., CalcNewValues.cpp), and run your application as: java GetBurnRate | CalcNewValues | java DisplayValues. Note that the second filter (i.e., CalcNewValues) is a C++ application. Your lunar lander application must behave in the same way as the original application. Submit (1) your CalcNewValues.cpp file; (2) a document that includes a screenshot of running the application (from the command line) and briefly explains why this would work and what framework(s) is involved.

> This would work because the values passed and received through the filters by the class CalcNewValues.cpp remain the same, therefore, even if the class involved or if the procedures in a class change, if the programmer does not change the input and output, the program will work, because the C++ framework stdio would interpret the kernel output and flush the input on the kernel for the next class, respecting the pipe-and-filter architecture style.

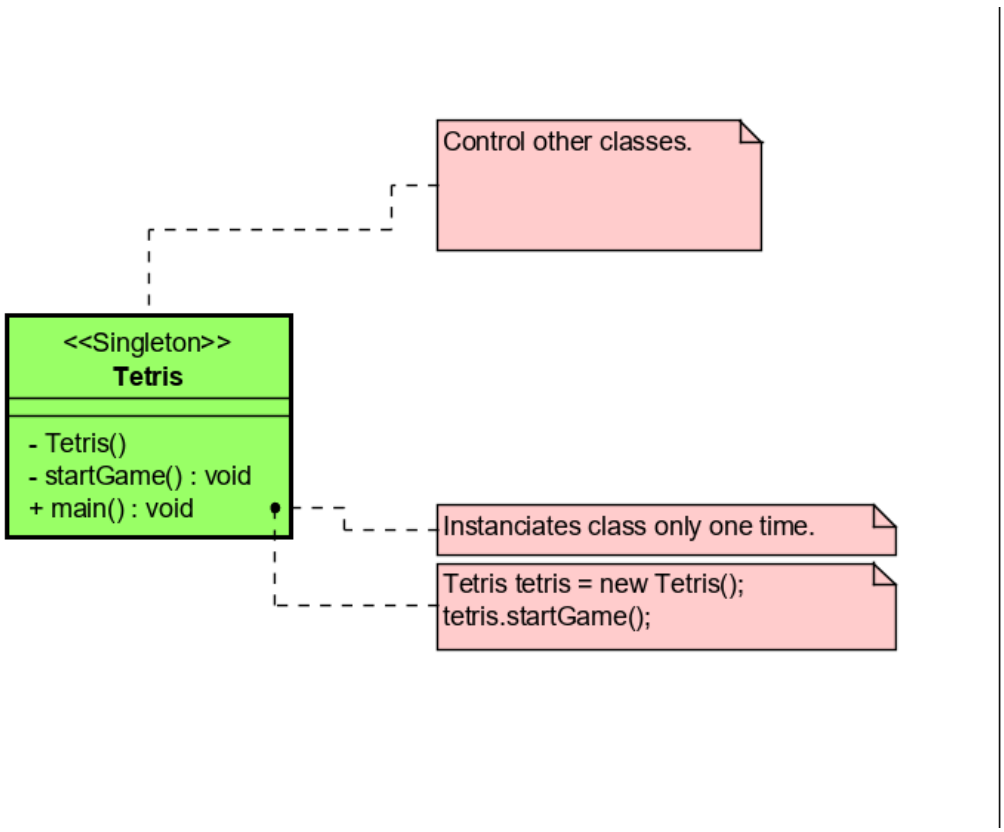
2. (20 Points) Download Tetris-Testing.zip (attached to this assignment), unzip it, and import the project into your Eclipse workspace. This is a Tetris video game application. Run the application in Eclipse, go through the code, and understand the implementation. Identify and write down TWO architecture styles, architecture patterns, or design patterns used in the implementation of the game. For each pattern or style that you identified, draw a diagram that illustrates the involved classes and their interactions.

> One design pattern used on the application is the Factory pattern. It is specifically used to return new piece types for displaying them in the Tetris game. The classes involved are SidePanel, Tetris and TileType java classes. The diagram of the pattern is as follows:



The *SidePanel* Java Class implements a Tetris Constructor that calls the function *getNextPieceType()*, this function returns an TyleType inherited class type.

The second pattern would be Singleton, used by the class Tetris. Obviously, singleton is used in a broader context, where generally concurrency is involved; but, in this program, the class constructor is defined as private and only accessed once by the its main function, working similarly as a Singleton class. If it were a multi-player online game, some modifications to it would be needed, such as defining a getinstance synchronized method: which is not a necessity in this scenario, because is a single-player game. Diagram of it is as follows:



3. (50 Points) Create a JUnit testing class for Class Tetris and Class BoardPanel respectively. You can find these two classes in the Tetris-Testing.zip file. For each class, test the following methods. - Tetris: -public void updateGame() - BoardPanel. -public boolean isValidAndEmpty(TileType type, int x, int y, int rotation) -public void addPiece(TileType type, int x, int y, int rotation) -public int checkLines() Create a JUnit test suite, and add both test classes into the test suite. Put all your test cases and test suite into a different folder (e.g., test). After you finish your assignment, add your project into a zip file. In addition, you need to write a report. The report should include the following content. (1) What problem(s) did you find in the code? For each problem, further explain how you found it (e.g. using which test case). (2) Specifically explain the test case that you have created for the updateGame method of Class Tetris. What is your input, and what is your expected output? What is your logic of testing this method? (3) Include a screenshot of the result of running your test suite.

1- The program overwrites the tiles, even when they have the same rotation; the program does not clear sometimes checked lines.

2- I have created a function in the Tetris class to access the private function updateGame(), then I saved the number of checked lines and the current score of the program's state. I've called the public function I've made, named update_p(), to alter the game current state; and, even though there were lines to be cleared and points to be add, the program did not cleared them.

3-