SWEN30006 Software Modelling and Design

Workshop 12, Thursday 12:00, Group 10
Jinrun Ji :1394227
Alexei Cherstnov: 1080039
Vincent Khuat: 1081402

P1 Refactoring

Going through the current design of the Tetris game system, something that is noted was that many of the responsibilities were assigned to the **Tetris** class causing low cohesion in the system. In addition to this we noticed that the tetris shapes contained almost identical functions and variables. This may be an issue, as including additional tetris shapes would similarly require identical code taking up unnecessary disk space, decreasing readability and reducing the overall cohesiveness of the system.
These identified issues may pose problems when implementing additional shapes and difficulty settings in the game system. The **Tetris** class will continually become less cohesive and increase the responsibilities of said class.

P2 Proposed Design Choices

In order to improve onto the current system, initial improvements on the system were to be directed at decreasing the coupling between the tetris shapes and the **Tetris** Class. It was decided to use Polymorphism for this case. In order to accomplish this it was proposed that another class be delegated to handle functions and variables relating to the creation of tetris shapes. This class was named **TetrisBlock** and handles all identical functions and variables found in the previous tetris shapes classes, which included movement actions and displays. Essentially employing polymorphism, the tetris shape classes e.g: **I,J..** contained only necessary and unique information pertaining to their classes. The addition of this class makes it more simple to add new tetris shapes effectively and easily with higher cohesion.

In addition to the **TetrisBlock** class, in order to handle the creation of multiple tetris block objects, and to decrease coupling, another class was introduced through the use of the **Pure Fabrication** design pattern. The **TetrisBlockGenerator** class handles the responsibility of routing which block gets generated in the game. This allows for future game extensions, such as, for example, introduction of logic in the creation of tetris blocks instead of random generation. This too, would be handled inside this class without affecting any of the other code. This follows the patterns of higher cohesion and decoupling.

P3 Implementation of Tetris Madness

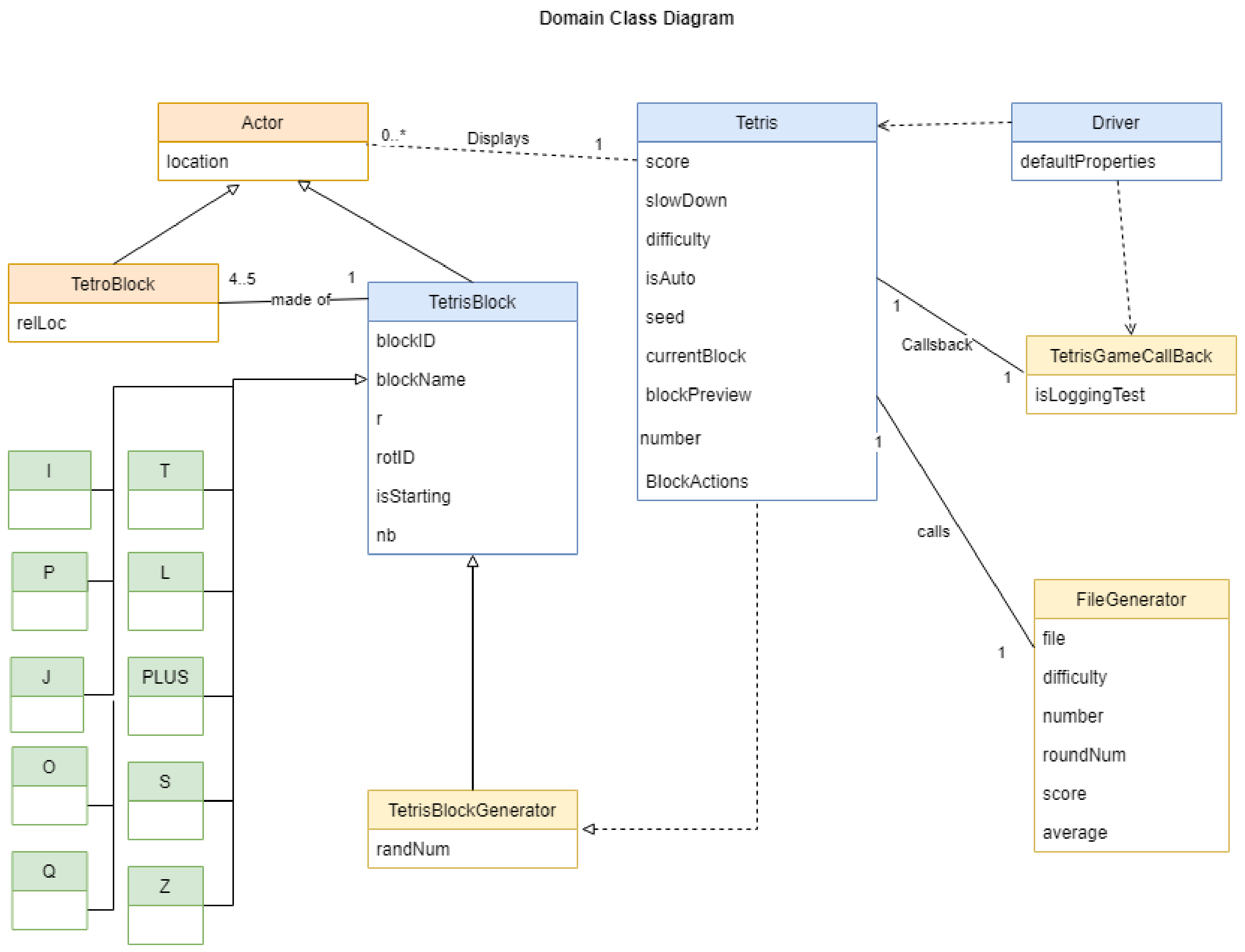
Once we were satisfied with the refactoring of the simple tetris system, implementation of the new Tetris required the addition of new tetris shape types: ‘+’, ‘q’ and ‘p’. Henceforth the classes: ‘plus’, ‘q’ and ‘p’ were added as an extension of the new **TetrisBlock class** (figure 1.1). This was relatively straightforward as the only the **TetroBlock** locations were to be added into these classes in order to be functional

The difficulty modifiers were added as small adjustments to the current basic Tetris game speed functionality. Further possible extensions include the creation of a separate class “TetrisRuleset” through the use of the Pure Fabrication design pattern, allowing to further extend the game rule algorithms all in a single class. However, for the above requirements, the addition of the speed modifiers just in the base Tetris class seemed justified by the Expert pattern. In addition, the implementation of the madness difficulty setting included disabling the rotation mechanic. This was added to the moveBlock() function within Tetris class.

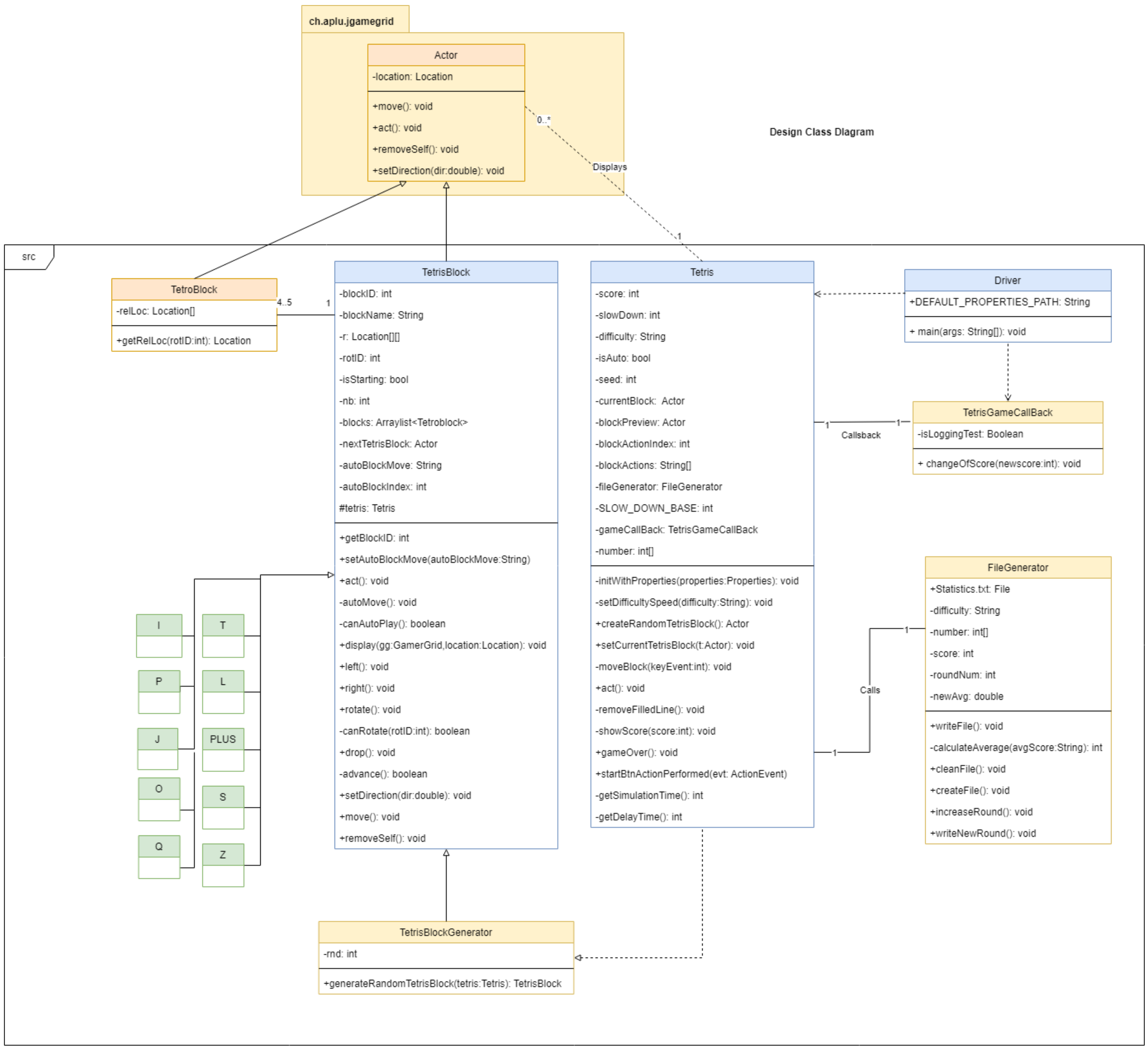
In order to implement the feature of recording statistics in the system the class **FileGenerator** by implementation of Pure Fabrication. Reducing the dependency and coupling on the Tetris class. **FileGenerator** may be called by Tetris class and is solely responsible for recording statistics. This class allows for other properties or statistics that may need to be recorded in future implementations.

.

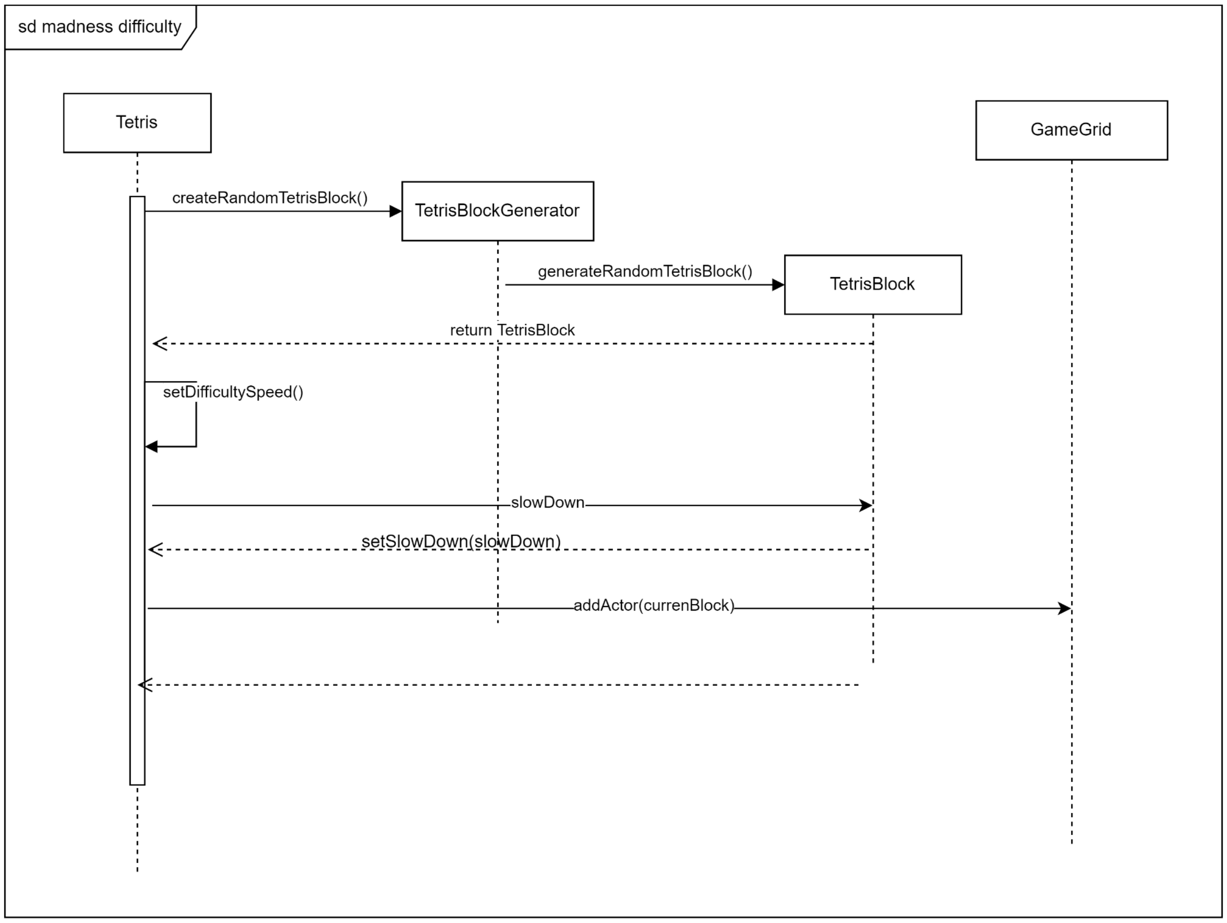
1.1 Domain Class of Proposed System



1.2 Design Class of Proposed System



System Sequence


sd madness difficulty
Tetris
GameGrid
createRandomTetrisBlock()
TetrisBlockGenerator
generateRandomTetrisBlock()
TetrisBlock
return TetrisBlock
setDifficultySpeed()
slowDown
setSlowDown(slowDown)
addActor(currenBlock)