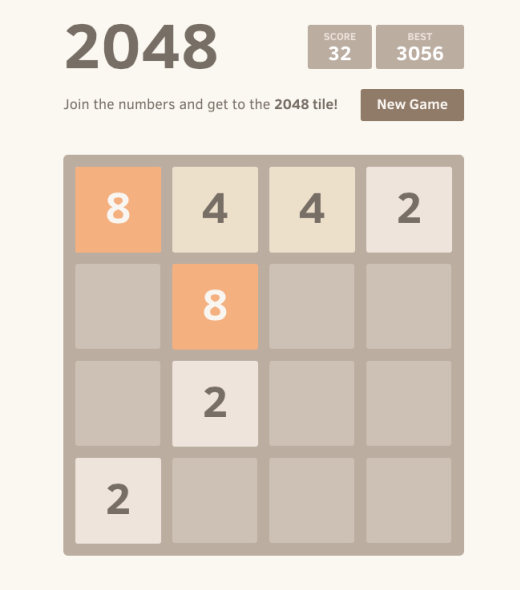
Assignment 4, Design Specification

SFWRENG 2AA4

April 13, 2021

This Module Interface Specification (MIS) document contains modules, types and methods used to support the popular game 2048. At the start of the game, the player will be provided with a sixteen tile game board where there will be two tiles occupied at a random position within the game board with the number 2. The goal of the game is to merge tiles with the same number as many times as possible in order to reach a tile numbered 2048. The player will be able to merge tiles through using the usage of the controller. The controller will allow for the player to move left, right, up, down. Each time the player uses the controller to make move left, right, up, down; the game will spawn a new numbered tile which is either numbered 2 or 4 at any position within the game board. Each player will start with a score of 0. Each time two tiles are combined together, the mathematical sum of the two tiles will be added to the score of the player.



The above board visualization is from
https://www.omgubuntu.co.uk/2014/03/install-2048-on-ubuntu-download

1 Overview of the design

This design has three modules which are able to maintain the state of the game as well as the view of the game board. The *Board.java* module is used to store information about the game, get the score,the grid as well as the control. The *Direction.java* module is a module which stores information regarding the way the user can prompt movement on the game board as well get the x and y components of the grid. These movements include $up, down, left, right$ by the user.

Likely Changes my design considers:

- Changing the way the tiles look within the game board as the value of the tile increases
- The way every move the players makes within the game is stored
- Changing minor details regarding how a new tile is spawned
- Making sure the consistency of the way the tiles are spawned or the way the tiles are merged together regardless of whether the player finishes the game or not.

Board Module

Template Module

Board

Uses

None

Syntax

Exported Constants

None

Exported Types

Board = ?

Semantics

State Variables

$NUMSTARTTILES : \mathbb{N}$
$TWOPROBABILITY : \mathbb{N}$
$GRIDSIZE : \mathbb{N}$
$RANGE : \mathbb{N}$
$TILETWO : \mathbb{N}$
$TILEFOUR : \mathbb{N}$

State Invariant

None

Assumptions

None

Direction Module

Module

Direction

Uses

None

Syntax

Exported Constants

None

Exported Types

None

Semantics

State Variables

y: $\mathbb{N}$
x: $\mathbb{N}$

State Invariant

None

Assumptions

None

Critique of Design

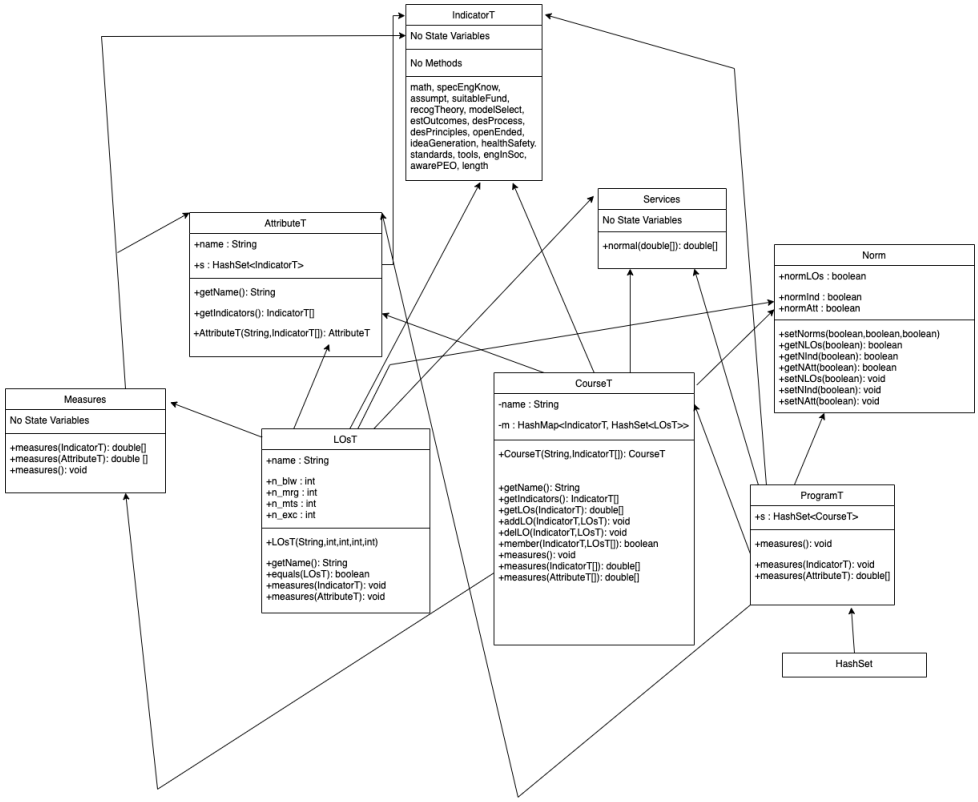
- The design is very consistent in terms of the usage of the language and terminology being used in each module. Within each specification, the names of state variables or methods do not change at all. The advantage of having a consistent specification is that a person reading the specification for the first time will be able to understand

how the module is being constructed without any confusion. The specifications have no homonyms or synonyms which prevents any further issues. In addition to the same naming conventions, the order of which each program is accessed maintains a level of consistency.

- The design maintains a level of essentiality as well because it does not have any unnecessary programs. Whatever is specified within the design specification is what exists within the modules implemented for the state of the game and the view of the gameboard.
- The design maintains generality resulting in no one module being limited to one particular case and tend to work together to allow for the user to have the best experience while playing the game. All modules (unless specifically limited due to the rules of the game) are free from any restriction or limitation and allows for each module to have a long lifespan.
-

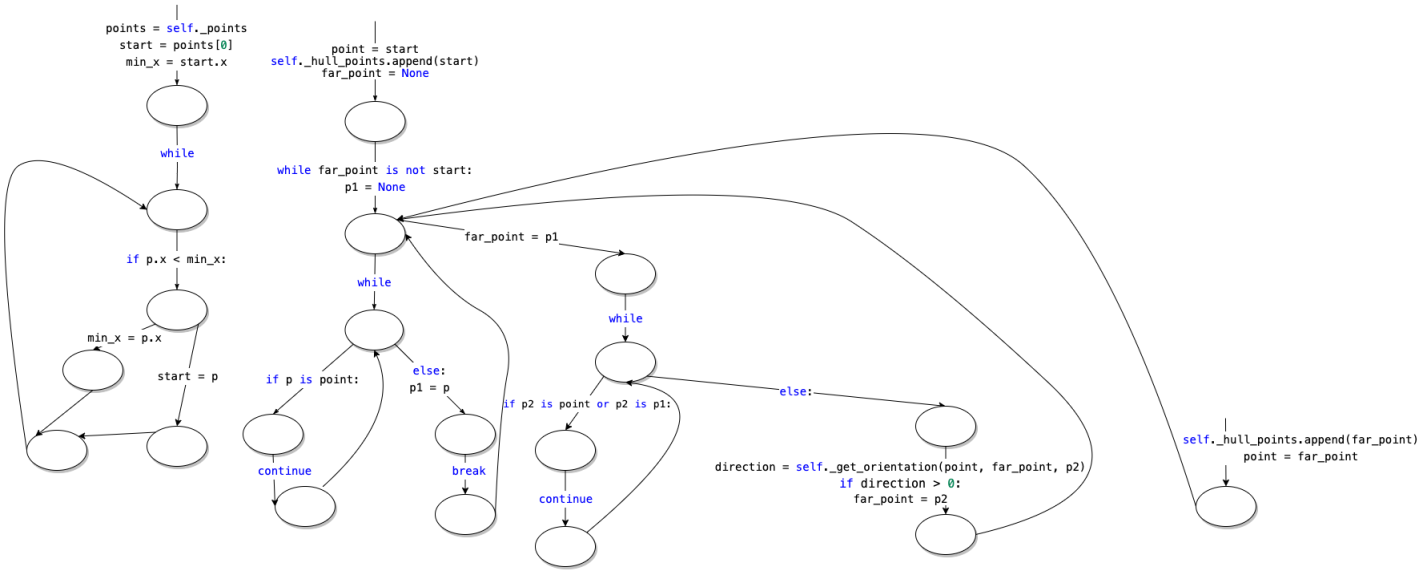
Answers to Questions:

Q1: Draw a UML diagram for the modules in A3.



The UML is constructed using https://app.diagrams.net/

Q2: Draw a control flow graph for the convex hull algorithm. The graph should follow the approach used by the Ghezzi et al. textbook. In particular, the code statements should be edges of the graph, not nodes. Code for the convex hull algorithm can be found at: https://startupnextdoor.com/computing-convex-hull-in-python/. To match the diagrams available from Ghezzi, replace the for loop in the code with a while loop.



The control flow graph is constructed using https://app.diagrams.net/