Model UML - GC 5

Andrea Sgobbi, Roberto Scardia, Jonatan Sciaky, Luca Simei

March 2023

1 Main model classes

1.1 GameModel

GameModel is the model entry point.

It's initialized with the number of players. This cannot be changed dynamically, since it will be the controller's job to skip disconnected players, and the board should not change with disconnections.

Common goals are saved only by their unique ID, to minimize serialization overhead. The scores for each goal reside on two stacks.

Players are added dynamically after model initialization, and nickname uniqueness must be guaranteed externally.

A standard move, consisting of removing tiles from the board and placing them in the desired order on the shelf, is done through removeSelection() and shelveSelection(). Move legality is supposed to be checked externally by the controller, hence utility methods are provided to do all legality checks.

1.2 Player

Players are initialized by their game-unique nickname and by the ID of their personal goal. Each player keeps a reference to their shelf and a running count of their score.

1.3 Shelf

Shelves are internally represented as a list of stacks of Tiles, but can also provide a matrix representation. It provides various utility methods to check

the space left in each column, check whether it's full and add an ordered list of tiles to a given column.

1.4 Board

Boards are internally represented as a map between Coordinates and Tiles. The keyset (coordinates) is never modified: this means we can create a board for a given number of players with only the allowed coordinates, and any coordinate reserved for higher player counts will not be considered. To do this, the board must contain empty tiles, which are simply Tile.NOTILE.

Board state can be accessed through getTile() and modified by either inserting a tile or removing (setting to notile) a selection of coordinates.

The board also provides a way to refill it given a TileBag. Refill is implemented as a breadth-first traversal starting at a random coordinate, drawing a random tile from the bag. The implementation stops executing when the bag is empty, allowing different behaviour in case refill was not successful due to a lack of tiles (for example, the controller can just check whether there are empty tiles and stop the game, or continue)

1.5 TileBag

TileBag is used to represent the finite amount of tiles in the game. It counts tiles "in play", which means all those not on the shelves (hence including the board). Tiles can never be added, but they are removed after calls to shelveSelection().

2 Utility classes

2.1 Coordinate

Coordinate is a record used to index both Board and Shelf. Both use (0, 0) as the index of the bottom-left corner.

2.2 Tile

Tile is a record keeping track of both the kind of tile and the sprite on it. This allows for consistent display of each tile's sprite across players, on both the board and each player's shelf.

2.2.1 Type

Type is either one of the 6 base tile types or NOTILE.

2.2.2 Sprite

Sprite is either one of the 3 base sprites or NOSPRITE.

2.3 PersonalGoal

Personal Goals are implemented as a singleton-like class. The constructor is private, and only the 12 built-in goals are initialized and stored in an array. To access each goal, a static method is provided which returns the goal corresponding to a given ID.

Goals are immutable and simply offer a way to check their completion on a given Shelf, returning the score obtained.

2.4 CommonGoal

Common Goals are implemented in a similar fashion, though different goals are implemented as different instances of CommonGoalStrategy which implement the correct checkShelf() method. Each goal also provides a short description.