N-in-a-Row – The Desktop Application

Engine Module

**Interface GameLogic:**

* boolean play(int col, boolean popout);
* String timeFromBegining();
* int getCols();
* int getRows();
* PlayerTypes getTypeOfCurrentPlayer();
* int getIdOfCurrentPlayer();
* boolean getHasWinner();
* Set<Integer> getWinners();
* boolean getIsBoardFull();
* void setRoundFromSettings(boolean restartPlayers);
* int getNumberOfPlayersToInitialized();
* boolean isPopout();
* List<Player> getPlayers();
* Move getLastMove();
* void resignPlayer();
* GameVariant getGameVariant();
* int getNumberOfRoundsPlayed();

void increaseRoundPlayed();

**Class Game** implements GameLogic:

* final int maxNumberOfPlayers
* boolean hasWinner, isBoardFull
* Board board
* Date startingTime
* GameSettings gameSettings
* List<Player> players
* Player currentPlayer
* List<Move> playedMoves

**Enum class Directions**: LEFT, LEFTUP, UP, UPRIGHT, RIGHT, RIGHTDOWN, DOWN, LEFTDOWN

**Class Player**:

* int numOfTurnsPlayed, id
* PlayerTypes playerType
* String name

**Class Board**:

* int rows, cols, winningPlayer, emptySapces
* Col[] board
* boolean hasWinner

**Class GameSettings**:

* int target, boardNumRows, boardNumCols, numOfPlayers
* GameVariant gameVariant
* GameType gameType
* String settingFilesPath

**Class Move**:

* Static int movesCount
* int moveIndex, playerID, col

**Class Col**:

* int colNumber, freeSpaces, lastRowInserted
* Disc[] discs

**Class Disc**:

* Position position
* int discOfPlayer
* Map<Directions, Disc> discsArround

**Class ComputerTurnTask** extends Task<Void>

**Class** **App** extends Application

**Class ColumnButton** extends Button

Desktop Application Module

**Class** **AbstractToggleTableItem**

**Class DesktopAppController**

**Class TogglePlayerDisplayActive** extends AbstractToggleTableItem

**Class MoveDisplay**

**Class loadXMLTask** extends Task<Boolean>

**Class ExitConfirmController**

**Class ToggleTableRow**<T extends AbstractToggleTableItem> extends TableRow<T>

**Class xmlLoadingController**

**Enum class PlayersTypes**: HUMAN, ROBOT

**Enum class GameType**: BASIC, MULTIPLAYER, DYNAMIC\_MULTIPLAYERboolean play(int col, boolean popout);

String timeFromBegining();

int getCols();

int getRows();

PlayerTypes getTypeOfCurrentPlayer();

int getIdOfCurrentPlayer();

boolean getHasWinner();

Set<Integer> getWinners();

boolean getIsBoardFull();

void setRoundFromSettings(boolean restartPlayers);

int getNumberOfPlayersToInitialized();

boolean isPopout();

List<Player> getPlayers();

Move getLastMove();

void resignPlayer();

GameVariant getGameVariant();

int getNumberOfRoundsPlayed();

void increaseRoundPlayed();

**Enum class GameVariant**: REGULAR, CIRCULAR, POPOUT

Common Module

**Class** **GameSettings** + **Class** **SettingsFileException**

**Class** **Lock**

**Class** **PlayerSettings**

**Enum class** **MoveType**: INSERT, POPOUT

**General structure**

* The game has 2 main modules: Engine, DesktopApplication.
* The game has 1 common module: Common. This module is being used by both the Engine module and DesktopApplication module.
* The central class in the Engine module is Game, which implements the GameLogic interface:
  + It holds information about the players, the board (and its state) and the moves that were made in the game.
  + It is responsible for enforcing and validating the rules of the game.
  + The Board consists of a set of columns, each of which contains Discs, in number matching the number of rows.
  + Each Disc has its own position in the board (represented as a matrix) and may or may not have a player’s ID (depends on whether a player dropped a disc that ended up in the Disc’s position).
* The DesktopApplication module holds a resources folder, with all the resources used for the game’s display, mostly including .css and .png files.  
  The central classes in the DesktopApplication are:
  + App – this class sets the JAVAFX scene and defines and initializes the JAVAFX controller. This class also includes the PSVM function.
  + DesktopAppController –
    - This class holds an instance of GameFactory, which produces an object of GameLogic per each recently loaded GameSettings. The controller refers to the GameLogic object for every logic related need, during a game round.
    - This class consists of all of the visual components displayed to the user. It also includes implementations of the interactions between themselves and between them and the user’s input.
* The visual components of the app can be grouped into the following main panels:
  + Game Flow panel – this section of the screen includes buttons with the main game related options that the users can select from: Load XML, Play Round, End Round, Player Resign, Themes selections and Exit Game. It also includes a display of central game related information: settings file used for configurating the game, the Game Variant that is being played and the number of game rounds played.
  + Players table – displays the players defined for the game, including each player’s disc color, name, id and number of moves performed. The table also includes a visual indication of the player that should play in the current turn and of the player’s Active status (a resigned player is considered inactive).
  + Moves table – displays all the moves that were played in the round. For each move, there’s a specification if the player making the move, the column number of the move, the type of move made and the move’s timestamp. This table updates dynamically during the round, to accumulate all moves as the game progresses. This table is also being used in the Replay, so that the uses can stroll back & forth through the moves that were played.
  + Board panel – this panel holds the main game board, where each round is actually being played. It consists of a grid, where discs are being placed on top of each other. Above each board column there’s a button on which players click in order to drop their disc. In case of a Popout game variant, below each column there’s also a button giving the players the option of removing their disc from the bottom of the column.
* The Common module includes ENUM classes with static values. Both the DesktopApplication module and Engine module should be familiar with the Common module.

**Main Choices that were made**

* Computer’s moves are executed with a slight delay, to give a clear separation between the moves that are played.
* When a round of game is completed, either by a win or a tie, the user can choose between (1) starting a new game, by loading an XML file with different configuration and new players, (2) restarting the most recent game that was played
* When in Replay mode, there’s no option of making any new moves or of resigning a player. In order to make these options available, the user must end Replay mode.
* At any point of the game, the user/s can choose to exit the game all together.
* Bonuses that were implemented:
  + Changing the Skin of the game (this can be performed under Themes, in the top Game Flow panel)
  + A computer player turn is being executed from a separate designated thread
  + A single player can leave the game at any of her turns (this is accomplished by clicking on the Player Resign button, in the top Game Flow panel)
  + The User/s can replay the moves that were made during the game up until the point of entering the Replay mode (Replay mode starts and ends by clicking on the button under the moves table, and browsing moves is performed by the two arrows next to it)

**Students’ details**

* Alon Boxiner (203485248), [alonboxiner@gmail.com](mailto:alonboxiner@gmail.com)
* Miri Kogan (066719279), [mirikog@gmail.com](mailto:mirikog@gmail.com)