

**Πανεπιστήμιο Κρήτης –Τμήμα Επιστήμης Υπολογιστών**

**ΗΥ252– Αντικειμενοστρεφής Προγραμματισμός**

**Διδάσκων: Ι. Τζίτζικας**

**Χειμερινό Εξάμηνο 2020-2021**

STRATEGO

Ice vs Fire

Εισαγωγή

Σπυρίδων Χρυσοβαλάντης Ζερβός

Csd4878

21/11/2022

Think and describe what you plan to do and why it will be useful.

Περιεχόμενα

[1. Εισαγωγή 2](#_Toc530045458)

[2. Η Σχεδίαση και οι Κλάσεις του Πακέτου Model 3](#_Toc530045459)

[3. Η Σχεδίαση και οι Κλάσεις του Πακέτου Controller 1](#_Toc530045460)2

[4. Η Σχεδίαση και οι Κλάσεις του Πακέτου View 14](#_Toc530045461)

[5. Η Αλληλεπίδραση μεταξύ των κλάσεων – Διαγράμματα UML 17](#_Toc530045462)

[6. Λειτουργικότητα (Β Φάση) 18](#_Toc530045463)

[7. Συμπεράσματα 18](#_Toc530045464)

## Εισαγωγή

Η εργασία χρησιμοποιεί το μοντέλο MVC (Model-View-Controller). Το Model περιέχει τα βασικά αντικείμενα του παιχνιδιού, αυτά δηλαδή με τα οποία τρέχει το παιχνίδι αφού περιέχουν το σημαντικότερο μέρος της πληροφορίας κατάστασης. Το View περιέχει όλες τις κλάσεις που αφορούν την διεπαφή με τους χρήστες (GUI), δηλαδή τον τρόπο με τον οποίο ο χρήστης αλληλοεπιδρά με τα αντικείμενα του. Τέλος, το Controller είναι η κλάση-διαιτητής του παιχνιδιού, αυτή δηλαδή που ελέγχει την ροή του προγράμματος και συγχρονίζει το Model με το View. Στις επόμενες τρείς ενότητες της αναφοράς γίνεται η ανάλυση των μερών του MVC. Συμπεριλαμβάνονται UML διαγράμματα για τις κλάσεις που «συμμετέχουν» σε κάποια ιεραρχία (π.χ. Piece). Τα UML διαγράμματα της λειτουργικότητας και της αλληλεπίδρασης μεταξύ των κλάσεων θα παραδοθούν στην Β φάση στην ενότητα 5. Ομοίως, η λειτουργικότητα του παιχνιδιού επεξηγείται στην ενότητα 6 και, τέλος, στην ενότητα 7 δίνονται κάποια συμπεράσματα από την υλοποίηση του παιχνιδιού.

## Η Σχεδίαση και οι Κλάσεις του Πακέτου Model

Ανάλυση του Model:

* Package piece
  + Package movable:
    - Classes Beast Rider, Dragon, Elf, Knight, Lava Beast, Mage, Sorceress, Yeti

Οι κλάσεις αυτές αντιπροσωπεύουν τα πιόνια του παιχνιδιού που δεν έχουν ειδικές δυνάμεις. Περιέχουν τα inherited από την MovablePiece fields και methods.

* + - Classes Elf, Scout, Slayer

Οι κλάσεις αυτές αντιπροσωπεύουν τα πιόνια του παιχνιδιού που έχουν ειδικές δυνάμεις. Περιέχουν τα inherited από την MovablePiece fields και methods, ενώ κάνουν implement το SpecialPiece. Το interface δεν προσφέρει κάτι παραπάνω όσον αφορά την υλοποίηση: Τα διαχωρίζει μόνο από τα υπόλοιπα pieces.

* + - Abstract Class MovablePiece

Όλα τα πιόνια που μπορούν να κινηθούν κάνουν extend αυτή την κλάση.

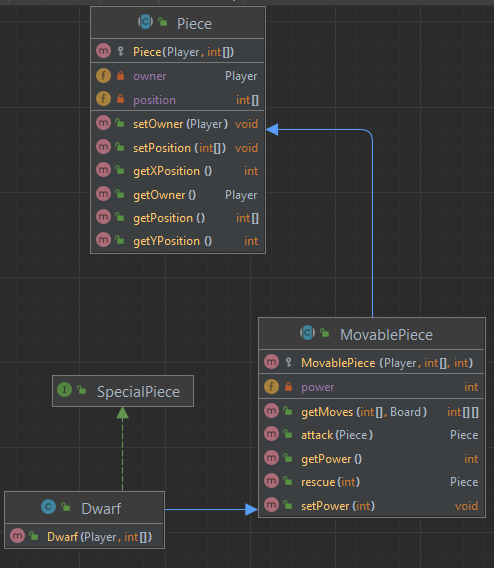
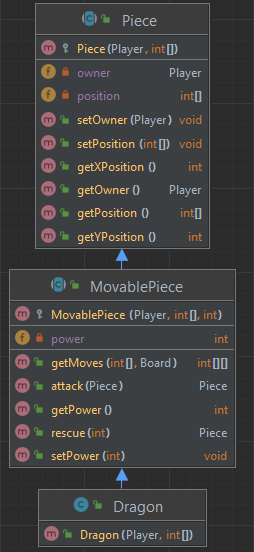
Extended by classes: Beast Rider, Dragon, Elf, Knight, Lava Beast, Mage, Sorceress, Yeti, Elf, Scout and Slayer.

|  |  |
| --- | --- |
| **Field** | **Description** |
| Private int power | Power of the piece |

|  |  |  |
| --- | --- | --- |
| **Method** | **Type** | **Description** |
| Public void setPower(int power); | Transformer (Mutative) | Sets the power of this piece |
| Public int getPower(); | Accessor | Gets the power of this piece |
| Public int[][] getMoves(int[] pos, Board board); | Accessor | Returns an array of positions the piece can move to |
| Public void move(int[] pos); | Transformer (Mutative) | Moves the piece to a certain location and triggering any needed actions (e.g. attack, rescure) |
| Public Piece attack(Piece opp); | Transformer (Mutative) | Returns the piece that wins the fight while deleting the losing one |
| Public Piece rescue(int piece); | Transformer (Applicative) | Rescues the piece of the given piece code |

* + - Interface SpecialPiece

Χρησιμοποιείται ως ”tag” των ειδικών πιονιών. Δεν προσφέρει κάποια αξιοσημείωτη λειτουργικότητα.



Τα UML διαγράμματα των Dragon και Dwarf/

* + Package stationary
    - Class Flag

Η κλάση αυτή αποτελεί τη σημαία του παίκτη. Πέραν της μεγάλης αξίας της για το παιχνίδι, περιέχει μόνον ό,τι κληρονομεί η StationaryPiece.

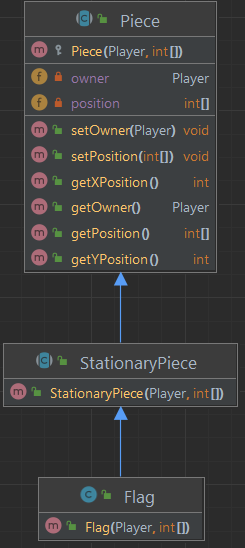
* + - Class Trap

Το πιόνι-παγίδα που σκοτώνει ό,τι πέσει πάνω του, εκτός του νάνου (dwarf). Περιέχει μόνον ό,τι κληρονομεί η StationaryPiece.

* + - Abstract Class StationaryPiece

Extended by classes: Flag, Trap.

Χρησιμοποιείται για να διαχωρίσει τα ακίνητα pieces από τα κινούμενα. Περιέχει μόνο τα fields και με methods της Piece.



Το UML διάγραμμα της Flag

* + Abstract Class Piece

Η κεντρική κλάση που κάνει extend κάθε πιόνι. Οτιδήποτε την κάνει extend, θεωρείται πιόνι, κινούμενο ή όχι.

|  |  |
| --- | --- |
| **Field** | **Description** |
| Private Player owner | Piece’s owner – to which players the piece belongs to |
| Private int[] position | Its position on the board |

|  |  |  |
| --- | --- | --- |
| **Method** | **Type** | **Description** |
| Public void setOwner(Player player); | Transformer (Mutative) | Sets the owner of the piece. |
| Public Player getOwner(); | Accessor | Gets the owner of the piece. |
| Public void setPosition(int[] position); | Transformer (Mutative) | Sets the position of the piece. |
| Public int[] getPosition(); | Accessor | Gets the position of the piece. |
| Public int getYPosition(); | Accessor | Gets the Y position of the piece (height). |
| Public int getXPosition(); | Accessor | Gets the X position of the piece (width). |

* Package Player
  + Class Army

Η κλάση αυτή περιέχει τον στρατό (δηλαδή τα πιόνια) του παίκτη και ό,τι λειτουργείες σχετίζονται με αυτόν.

|  |  |
| --- | --- |
| **Field** | **Description** |
| Public static final int PIECE\_TYPES (=11) | All different piece types, excluding the flag. |
| Public static final int[] DEFAULT\_PIECES (=[1, 4, 5, 2, 2, 2, 3, 2, 1, 1, 6]) | Default number of pieces of each type in an army with reducedArmy option set to false. |
| Private String Owner | Army’s owner |
| Private int[] activePieces | Active (alive) pieces of each type |
| Private ArrayList<ArrayList<Piece>> pieces | All pieces of the army (every piece instance) |

|  |  |  |
| --- | --- | --- |
| **Method** | **Type** | **Description** |
| Public void setOwner(Player owner); | Transformer (Mutative) | Sets the army’s owner |
| Public Player getOwner(); | Accessor | Gets the army’s owner |
| Public int getPieceCode(Piece piece); | Transformer (Applicable) | Gets the piece code of the given piece. Piece codes are used mostly to locate piece information in Army’s arrays. |
| Public void removePiece(Piece piece); | Tranformer (Mutative) | Removes the piece from the army |
| Public void addPiece(Piece piece); | Tranformer (Mutative) | Adds the piece to the army |
| Public int getNumberOfPieces(int piece); | Accessor | Gets the number of active pieces of the given piece type in the army |
| Public boolean hasMovablePieces(); | Observer | Checks if the army has active movable pieces |
| Public void createArmy(boolean mode); | Transformer (Mutative) | Creates the army based on the mode. |
| Public void resetArmy(); | Transformer (Mutative) | Resets the army: Sets all pieces to 0 / Empties army. |
| Public boolean hasAvailableRescues(); | Observer | Checks if there are pieces that can be rescued |
| Public int[] getAvailableRescues(); | Accessor | Gets a list of the number of pieces that can be rescued from each type. |

* + Class Player

Η κλάση αυτή είναι ουσιαστικά ο παίκτης και περιέχει τα στοιχεία του, χρήσιμες πληροφορίες και τις μεθόδους που σχετίζονται με αυτόν

|  |  |
| --- | --- |
| **Field** | **Description** |
| Private String name | Player’s name. Defaults to player+<Id> |
| Private int Id | Player’s id |
| Private Army army | Player’s army (their pieces) |
| Private boolean flag | The status of the player’s flag. True = alive, false = dead |
| Private int[] saves | Saves performed by each piece type |
| Private int totalSaves | Total saves performed |
| Private int[] captures | Captured opponent pieces of each type |
| Private int totalCaptures | Total captures performed |
| Private int totalAttacks | Total attacks performed |
| Private int successfulAttacks | Total successful attacks performed |
| Private int totalMoves | Total moves performed |

|  |  |  |
| --- | --- | --- |
| Public Army getArmy(); | Accessor | Gets player’s army |
| Public void setName(String name); | Transformer (Mutative) | Sets player’s name |
| Public String getName(); | Accessor | Gets player’s name |
| Public void setId(int id); | Transformer (Mutative) | Sets player’s Id |
| Public int getId(); | Accessor | Gets player’s Id |
| Public void setFlag(boolean status); | Transformer (Mutative) | Sets flag status |
| Public boolean hasFlag(); | Observer | Checks if the flag is alive |
| Public void setSavesOf(int piece, int saves); | Transformer (Mutative) | Sets saves performed by a piece type |
| Public int getSavesOf(int piece); | Accessor | Gets saves performed by a piece type |
| Public void setTotalSaves(int saves); | Transformer (Mutative) | Sets total saves |
| Public int getTotalSaves(); | Accessor | Gets total saves |
| Public void setCapturesOf(int piece, int captures); | Transformer (Mutative) | Sets captures performed by a piece type |
| Public int getCapturesOf(int piece); | Accessor | Gets captures performed by a piece type |
| Public void setTotalCaptures(int captures); | Transformer (Mutative) | Sets total captures |
| Public int getTotalCaptures(); | Accessor | Gets total captures |
| Public void setTotalAttacks(int attacks); | Transformer (Mutative) | Sets total attacks |
| Public int getTotalAttacks(); | Accessor | Gets total attacks |
| Public void setSuccessfulAttacks(int successfulAttacks); | Transformer (Mutative) | Sets successful attacks |
| Public int getSuccessfulAttacks(); | Accessor | Gets successful attacks |
| Public void setTotalMoves(int moves); | Transformer (Mutative) | Sets total moves |
| Public int getTotalMoves(); | Accessor | Gets total moves |
| Public void resetPlayer(); | Transformer (Mutative) | Resets player’s stats (including army) |

* Class Board

Η κλάση αυτή αποτελεί το ταμπλό του παιχνιδιού, δηλαδή εκεί που τοποθετούνται (και περιέχονται) τα πιόνια των δύο παικτών

|  |  |
| --- | --- |
| **Field** | **Description** |
| Public static final int BOARD\_WIDTH (=10) | Board’s width (X) |
| Public static final int BOARD\_HEIGHT (=8) | Board’s height (Y) |
| Private Piece[][] board | The actual board, containing both player’s pieces. |

|  |  |  |
| --- | --- | --- |
| **Method** | **Type** | **Description** |
| Public boolean isRestrictedArea(int y, int x); | Observer | Checks whether the given position is restricted for the pieces to move into |
| Public Pieces getPieceAt(int y, int x); | Accessor | Gets the piece located at the given position |
| Public void setPieceAt(int y, int x, Piece piece); | Transformer (Mutative) | Sets the piece located at the given position |
| Public void setArmyOnBoard(Player player; | Transformer (Mutative) | Places player’s army on the board based on their Id |

* Class Timer

Η κλάση ρολόι που μετράει τη διάρκεια του παιχνιδιού. Αποτελεί ένα thread.

|  |  |
| --- | --- |
| **Field** | **Description** |
| Private int[] time | The current time. Index 0, 1 or 2 for hours, minutes and seconds respectively |

|  |  |  |
| --- | --- | --- |
| **Method** | **Type** | **Description** |
| Public void setTime(int[] time); | Transformer (Mutative) | Sets the time |
| Public int[] getTime(); | Accessor | Gets the current time |
| Public String toString(); | Accessor | Gets a formatted/readable string of the current time |
| Public void run(); | Transformer (Mutative) | Starts the clock |
| Public void updateTime(); | Transformer (Mutative) | Updates the time every second |

Η κλάση Piece χωρίζεται αρχικά με βάση το αν το πιόνι μπορεί να κινηθεί και στη συνέχεια, για τα κινούμενα πιόνια, χρησιμοποιείται το interface SpecialPiece για να τα διαχωρίσει από τα υπόλοιπα, κάτι σαν tag. Τα Movable Pieces έχουν όλα τα fields και methods που χρειάζονται για να χρησιμοποιηθούν. Η κίνηση γίνεται μέσω της move η οποία, εν συνεχεία, θα καλεί κάποια άλλη μέθοδο αν χρειαστεί. Τα Stationary δεν χρειάζονται κάτι παραπάνω αφού θα αναγνωρίζονται εντός της attack των Movable.

Ο Player περιέχει ό,τι έχει να κάνει με τον παίκτη, δηλαδή στατιστικά, τον στρατό του, όνομα, την κατάσταση της σημαίας του κτλ. Η Army όμως διαχειρίζεται τον στρατό (δηλαδή τα πιόνια). Τα piece codes χρησιμοποιούνται για την άμεση χρήση των arrays που περιέχουν τα statuses των πιονιών και άλλες παρόμοιες πληροφορίες που σχετίζονται με τον τύπο του πιονιού και όχι με κάποιο συγκεκριμένο πιόνι. Εξάλλου, τα πιόνια, σε αντίθεση με τους παίκτες, δεν έχουν id. Έτσι, το μόνο που τα διαχωρίζει είναι ο owner και το position π.χ. αν κάποιο πιόνι γίνει rescue, θα δημιουργηθεί ένα νέο instance του και όχι το ίδιο που έγινε capture.

Η Board είναι ουσιαστικά το ταμπλό. Περιέχει τα πιόνια και μεθόδους που σχετίζονται με τις θέσεις του board. Οι θέσεις αριθμούνται από το 0 μέχρι το 7 για το ύψος και το 9 για το μήκος για να είναι πιο εύκολη η χρήση των positions εντός του κώδικα. Γι’ αυτό το λόγο, όταν περνιέται ένα position σε μια μέθοδο δίνεται πρώτα το ύψος και μετά το μήκος π.χ. pos(5, 8) == pos(ύψος=5, μήκος=8).

Το Timer είναι απλώς ένα thread που μετράει τον χρόνο κατά τη διάρκεια του παιχνιδιού.

## Η Σχεδίαση και οι Κλάσεις του Πακέτου Controller

|  |  |
| --- | --- |
| **Field** | **Description** |
| Private static Board board; | Game’s board |
| Private static Player player1 | Player 1 (blue) |
| Private static Player player2 | Player 2 (red) |
| Private static Player nowPlaying | Player currently playing |
| Private static int round | Current round |
| Private static boolean reducedArmy | Reduced Army setting (true=enabled, false=disabled) |
| Private static boolean noRetreat | No Retreat setting (true=enabled, false=disabled) |
| Private static Timer timer | Timer (clock) of the game |
| Private static GameBoard gameboard | GameBoard View instance |

|  |  |  |
| --- | --- | --- |
| **Method** | **Type** | **Description** |
| Public static Board getBoard(); | Accessor | Gets board’s instance |
| Public static Player getPlayer1(); | Accessor | Gets player 1’s instance |
| Public static Player getPlayer2(); | Accessor | Gets player 2’s instance |
| Public static Player getNowPlaying(); | Accessor | Gets player currently playing |
| Public static int getRound(); | Accessor | Gets current round number |
| Public static boolean getReducedArmy(); | Observer | Gets ReducedArmy setting |
| Public static boolean getNoRetreat(); | Observer | Gets NoRetreat setting |
| Public static Timer getTimer(); | Accessor | Gets timer’s instance |
| Public static void endGame(Player winner); | Transformer (Applicative) | Ends the game with the given winner |
| Public static boolean checkIfEnded(); | Observer | Checks if the game has ended (if one of the two players has lost) |
| Public static void setPlayerPlaying(Player player); | Transformer (Mutative) | Changes the player playing |
| Public static void nextRound(); | Transformer (Mutative) | Moves on to the next round |
| Public static void makeMove(Piece piece, int[] position); | Transformer (Mutative) | Attempts to move a piece at a certain location. Calls any necessary actions if needed (e.g. attack, rescue). |
| Public static void main(String[] args); | Transformer (Mutative) | Initiates the game. |

Η Controller περιέχει όλα τα πεδία και τις μεθόδους που χρειάζεται για να «ενορχηστρώσει» το παιχνίδι: Από πεδία στατιστικών και λειτουργικών πληροφοριών μέχρι μεθόδους που αλληλοεπιδρούν και με το Model και με το View.

Η main του Controller ξεκινά το πρόγραμμα δημιουργώντας το MainMenu του View όπου οι παίκτες βάζουν τα ονόματα τους και επιλέγουν ρυθμίσεις. Το MainMenu καλεί (μόλις πατηθεί το play) την startGame με τις παραμέτρους που έδωσαν οι παίκτες, η οποία κάνει set το παιχνίδι και μπαίνει στον πρώτο γύρο. Κάθε φορά που γίνεται μια κίνηση καλείται η makeMove με το αντίστοιχο πιόνι και την κίνηση που πήγε να κάνει. Η makeMove καλεί όποια άλλη μέθοδο χρειαστεί για την πραγματοποίηση της κίνησης (π.χ. attack, rescue) και μετά την checkIfEnded για να δει αν αυτή η κίνηση ήταν καθοριστική για το παιχνίδι (δηλαδή αν το έληξε). Αν δεν έληξε, καλεί την nextRound που προχωράει το παιχνίδι στον επόμενο γύρο, δηλαδή κάνει update το board του GameView (μέσω της GameView::updateBoard()) και δίνει την σειρά στον επόμενο παίκτη με την setPlayerPlaying. Το controller περιέχει μόνο getters καθώς οι setters δεν χρειάζονται εκτός της κλάσης.

## Η Σχεδίαση και οι Κλάσεις του Πακέτου View

Περιέχονται 4 Views:

* MainMenu

Είναι το αρχικό – πρώτο view στο οποίο οι παίκτες βάζουν τα ονόματα τους και επιλέγουν ρυθμίσεις (reducedArmy, noRetreat). Όταν είναι έτοιμοι πατούν το play για να ξεκινήσει το παιχνίδι.

|  |  |
| --- | --- |
| **Field** | **Description** |
| Private JTextField player1TextField | Input field for player 1’s name |
| Private JTextField player2TextField | Input field for player 2’s name |
| Private JCheckBox reducedArmyCheckBox | Reduced Army setting input checkbox |
| Private JCheckBox noRetreatCheckBox | No Retreat setting input checkbox |
| Private JButton playButton | Play button (starts game) |

|  |  |  |
| --- | --- | --- |
| **Method** | **Type** | **Description** |
| Public void actionPerformed(ActionEvent e); | Transformer (Mutative) | Handles the PLAY button click event (starts game) |

* GameBoard

Αποτελεί το ταμπλό, εκεί δηλαδή που στήνονται τα πιόνια και παίζουν οι παίκτες. Περιέχει μεθόδους για την επιλογή πιονιού, την εμφάνιση κινήσεων, την μετακίνηση πιονιού, την ανανέωση του board κτλ. Καθώς και μερικές άλλες βοηθητικές μεθόδους.

|  |  |
| --- | --- |
| Field | Description |
| Private PieceButton[][] boardLayout | The layout of the clickable pieces on the board. Basically, a copy of Piece[][] board but with PieceButtons. |

|  |  |  |
| --- | --- | --- |
| **Method** | **Type** | **Description** |
| Public void actionPerformed(ActionEvent e); | Transformer (Mutative) | Handles all clicks on board |
| Public PieceButton[][] convertBoard(Board board); | Transformer (Applicative) | Converts a board of Pieces to a board of PieceButtons |
| Public void updateBoardView(Board board); | Transformer (Mutative) | Updates the GameBoard to the given board |

* Statistics

Είναι το παράθυρο με τα στατιστικά και τις πληροφορίες του παιχνιδιού. Αποτελεί ξεχωριστό παράθυρο από το ταμπλό. Δεν περιέχει κανένα field και έχει μόνο μεθόδους που κάνουν update τα στατιστικά.

|  |  |  |
| --- | --- | --- |
| Public void updateStatistics(); | Transformer (Mutative) | Updates all statistics and round info |
| Public void updateTimer(); | Transformer (Mutative) | Updates the timer (clock) only |

* GameEnd

Είναι το τελευταίο παράθυρο του παιχνιδιού, αυτό δηλαδή που εμφανίζεται όταν τελειώσει το παιχνίδι δείχνοντας τον νικητή και τα τελικά statistics. Έχει ένα κουμπί exit που απλώς βγαίνει από το παιχνίδι.

|  |  |
| --- | --- |
| **Field** | **Description** |
| Private JButton exitButton(); | Exit game button |

|  |  |  |
| --- | --- | --- |
| **Method** | **Type** | **Description** |
| Public void actionPerformed(ActionEvent e); | Transformer (Mutative) | Handles the exit button click event (exits game) |

* PieceButton

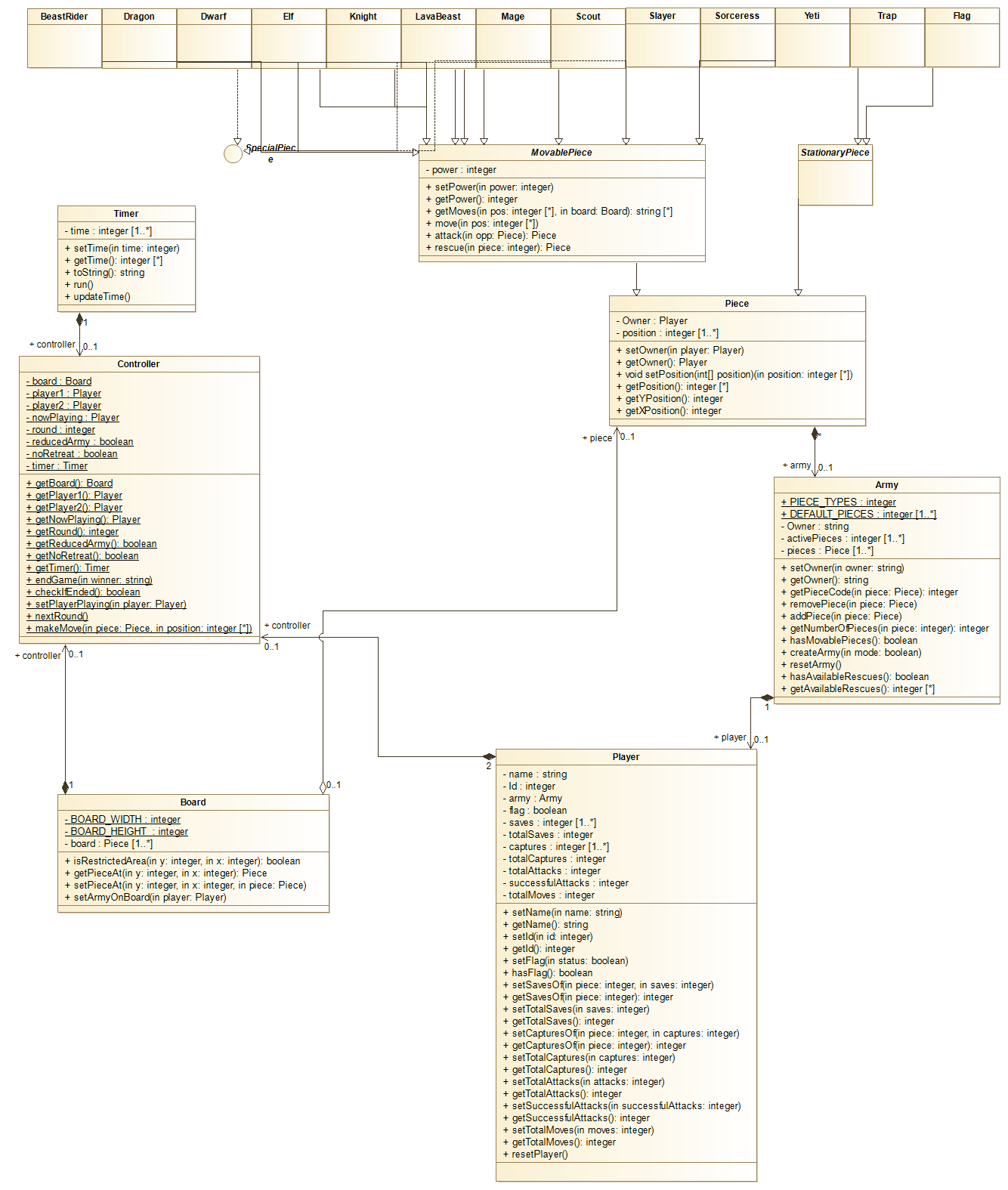
Αποτελεί μια «επέκταση» του Piece. Στην πραγματικότητα είναι το Piece αλλά περιέχουν και ένα button που τα κάνει clickable. Χρησιμοποιούνται από το GameBoard.

|  |  |
| --- | --- |
| **Field** | **Description** |
| Private Piece piece | The piece it represents |
| Private JButton button | The contained button so that it is clickable |

Κάθε κλάση παραπάνω εξηγεί τον εαυτό της και ποιος είναι ο ρόλος της. Η MainMenu είναι το αρχικό παράθυρο πριν ξεκινήσει το παιχνίδι, οι GameBoard και Statistics συνυπάρχουν με την πρώτη να είναι το ταμπλό που παίζουν οι παίκτες και η δεύτερη αναγράφει τα στατιστικά του παιχνιδιού και χρήσιμες πληροφορίες, και τέλος η GameEnd εμφανίζεται στο τέλος του παιχνιδιού αναγράφοντας τον νικητή και τα τελικά στατιστικά.

Αρκετά ιδιαίτερη κλάση είναι όμως η PieceButton η οποία είναι ο τρόπος που αποκτούν τα πιόνια την δυνατότητα να χρησιμοποιηθούν από τον παίκτη, να μπουν δηλαδή μέσα στο παιχνίδι. Η PieceButton[][] που εμφανίζεται στην GameBoard είναι ένα αντίγραφο του Piece[][] board, δηλαδή του ταμπλό που περιέχεται στον Controller. Γι’αυτό το λόγο μετά από κάθε γύρο και, γενικώς, όποτε γίνεται κάποια αλλαγή στο board (π.χ. μετακίνηση πιονιού) καλείται η updateBoardView της GameBoard, ώστε να ενημερωθεί το ταμπλό που βλέπουν οι παίκτες.

## Η Αλληλεπίδραση μεταξύ των κλάσεων – Διαγράμματα UML



## 

## Λειτουργικότητα (Β Φάση)

Σε αυτήν την ενότητα θα γράψετε στη Β φάση ποια ερωτήματα καταφέρατε να υλοποιήσετε είτε επιτυχώς είτε εν μέρει (και ενδεχομένως ποια όχι).

## Συμπεράσματα

Σε αυτήν την ενότητα θα γράψετε τα συμπεράσματα σας για την εργασία, τυχόν προβλήματα που συναντήσατε και γενικά ότι άλλο κρίνετε απαραίτητο να αναφερθεί.