

**Πανεπιστήμιο Κρήτης –Τμήμα Επιστήμης Υπολογιστών**

**ΗΥ252– Αντικειμενοστρεφής Προγραμματισμός**

**Διδάσκων: Ι. Τζίτζικας**

**Χειμερινό Εξάμηνο 2020-2021**



**

*Εισαγωγή*

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

**Περιεχόμενα**

[**1. Εισαγωγή**](#_gjdgxs) **1**

[**2. Η Σχεδίαση και οι Κλάσεις του Πακέτου Model**](#_30j0zll) **2**

[**3. Η Σχεδίαση και οι Κλάσεις του Πακέτου Controller**](#_1fob9te) **6**

[**4. Η Σχεδίαση και οι Κλάσεις του Πακέτου View**](#_3znysh7) **7**

[**5. Η Αλληλεπίδραση μεταξύ των κλάσεων – Διαγράμματα UML**](#_2et92p0) **8**

[**6. Λειτουργικότητα (Β Φάση)**](#_tyjcwt) **10**

[**7. Συμπεράσματα**](#_3dy6vkm) **10**

## Εισαγωγή

Για την εργασία μου, χρησιμοποίησα το μοντέλο MVC(Model-View-Controller). Σύμφωνα με αυτό το μοντέλο, χωρίζουμε την λειτουργικότητα του προγράμματος μας σε 3 υποκατηγορίες: Μία που αναλαμβάνει να κρατά τα δεδομένα του προγράμματος, μια που αναλαμβάνει την διεπαφή του προγράμματος με τον χρήστη, και μια που καθιστά εφικτή την επικοινωνία μεταξύ μοντέλου, και διεπαφής. Το καλό αυτού του τρόπου προγραμματισμού είναι, ότι μπορούμε να κάνουμε εύκολα αλλαγές σε ένα από τα τρία κομμάτια του προγράμματος, χωρίς να πειράξουμε τα υπόλοιπα 2. Στις υπόλοιπες ενότητες, θα περιγράψω πως σχεδίασα τις κλάσεις για καθένα από τα 3 κομμάτια του προγράμματος, καθώς και μια γραφική τους απεικόνιση, χρησιμοποιώντας διαγράμματα UML.

## Η Σχεδίαση και οι Κλάσεις του Πακέτου Model

Πρώτα, η κλάση Piece:

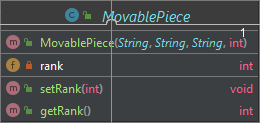


Η κλάση Piece αποτελείται από 5 private πεδία, (με τους αντίστοιχους setters και getters):

* String team: Η ομάδα του πιονιού(μπλε ή κόκκινη)
* Boolean hasRevived: true, εαν το πιόνι έχει πραγματοποιήσει κάποια διάσωση, false σε άλλη περίπτωση
* String image: Το μονοπάτι στην φωτογραφία του πιονιού
* int[][] position: Η θέση του πάνω στο ταμπλό
* String name: Το όνομα του πιονιού

Και από μια συνάρτηση String getPathOfImage(String, String) που επιστρέφει το μονοπάτι στην φωτογραφία που θα χρησιμοποιηθεί για το πιόνι.

Έπειτα, έχουμε 2 κλάσεις που κάνουν extend αυτήν την κλάση, την MovablePiece και την ImmovablePiece:

* Η ImmovablePiece δεν έχει κάποια επιπλέον γνωρίσματα.
* Η MovablePiece έχει:
* int rank(με τα αντίστοιχα setters και getters): η κατάταξη του πιονιού ανάμεσα στα άλλα πιόνια.

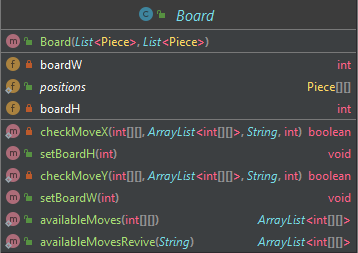
Επίσης, έχουμε μια κλάση SpecialMovablePiece, που κάνει extend την MovablePiece, η οποία δεν έχει κάποια παραπάνω γνωρίσματα.

Συγκεκριμένα, έχουμε τα πιόνια:

* Δράκος, με κατάταξη 10 και διαθεσιμότητα 1,
* Μάγος, με κατάταξη 9 και διαθεσιμότητα 1,
* Ιππότης, με κατάταξη 8 και διαθεσιμότητα 2,
* Αναβάτης, με κατάταξη 7 και διαθεσιμότητα 3,
* Μάγισσα, με κατάταξη 6 και διαθεσιμότητα 2,
* Θηρίο λάβας(κόκκινο)/ Γέτι(μπλε), με κατάταξη 5 και διαθεσιμότητα 2,
* Ξωτικό, με κατάταξη 4 και διαθεσιμότητα 2,
* Νάνος, με κατάταξη 3 και διαθεσιμότητα 5,
* Ανιχνευτής, με κατάταξη 2 και διαθεσιμότητα 4,
* Εξολοθρευτής, με κατάταξη 1 και διαθεσιμότητα 1,
* Παγίδα, με διαθεσιμότητα 6, και
* Σημαία, με διαθεσιμότητα 1.

Τα πιόνια με κατάταξη 10, 9, 8, 7, 6, 5, 4 κάνουν extend την κλάση MovablePiece. Τα πιόνια με κατάταξη 3, 2, 1 κάνουν extend την κλάση SpecialMovablePiece, και η παγίδα και η σημαία κάνουν extend την ImmovablePiece.

Στην συνέχεια, έχουμε την κλάση Board, η οποία αναλαμβάνει ότι έχει να κάνει με τα δεδομένα του ταμπλό:



Η κλάση Board έχει τα εξής πεδία:

* int boardW, boardH: οι διαστάσεις του ταμπλό(στην δική μας περίπτωση 8x10), με τα αντίστοιχα setters και getters.
* Piece[][] positions: Είναι ένας δισδιάστατος πίνακας 8χ8, όπου κάθε θέση είναι ένα αντικείμενο τύπου Piece. Για τις απαγορευμένες περιοχές, είναι ένα αντικείμενο Barrier, και για τις ελεύθερες περιοχές (άσπρες θέσεις) είναι null.

και από τις εξής μεθόδους:

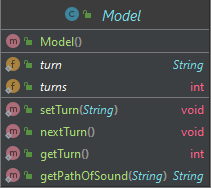
* ArrayList<int[][]> availableMoves(int[][]): Αυτή η μέθοδος παίρνει ως όρισμα την θέση ενός πιονιού, και επιστρέφει μία λίστα με όλες τις δυνατές θέσεις που μπορεί να μετακινηθεί το πιόνι. Εάν το πιόνι δεν είναι ο Scout, τότε προσθέτουμε και αφαιρούμε το 1 στην θέση του πιονιού και ελέγχουμε εάν είναι έγκυρη η θέση, και εάν είναι, την προσθέτουμε στην λίστα. Εάν είναι ο Scout, χρησιμοποιούμε τις συναρτήσεις checkMoveX, checkMoveY, οι οποίες ουσιαστικά λειτουργούν ως εξής:

Προσθέτουμε κάποιο offset, το οποίο συνεχίζουμε να το αυξάνουμε μέχρι η θέση που βρισκόμαστε να μην είναι έγκυρη, 4 φορές, 2 για κάθε άξονα, όπου η μία είναι με θετικό offset, και η άλλη με αρνητικό.

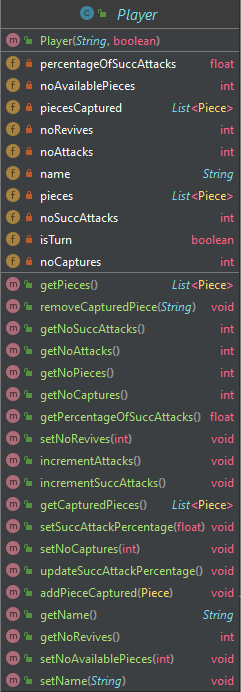
Εάν η θέση που βρισκόμαστε τώρα είναι κενή, ξανακαλούμε την συνάρτηση με αυξημένο(ή μειωμένο) το offset κατά 1. Εάν συναντήσουμε απαγορευμένη περιοχή ή κάποιο πιόνι της ίδιας ομάδας, τότε σταματάμε χωρίς να προσθέσουμε αυτήν την θέση στην λίστα. Αλλιως, δηλαδή εάν συναντήσουμε κάποιον εχθρό, προσθέτουμε την θέση στην λίστα, και σταματάμε.

* boolean checkMoveX, checkMoveY: Αυτές οι συναρτήσεις είναι υπεύθυνες για τον έλεγχο της παρούσας θέσης, και την προσθήκη της στην λίστα των διαθέσιμων θέσεων. Εάν η τωρινή θέση είναι κενή, τότε την προσθέτουμε στην λίστα, και επιστρέφουνε false, αλλιώς επιστρέφουμε true και, εάν συναντήσουμε εχθρικό πιόνι, προσθέτουμε την θέση στην λίστα.
* ArrayList<int[][]> availableMovesRevive(String): Αυτή η συνάρτηση επιστρέφει μία λίστα με τις διαθέσιμες θέσεις που μπορούμε να τοποθετήσουμε ένα πιόνι μετά από την διάσωση του. Συγκεκριμένα, ελέγχει ποιες από τις αρχικές 30 θέσεις της ομάδας του είναι ελεύθερες, και τις προσθέτει στην λίστα.

Έπειτα, η κλάση Utilities, κρατά κάποια δεδομένα για τις διαστάσεις του παραθύρου συγκεκριμένα το ύψος και το πλάτος του.

Προτελευταία, η κλάση Model:

Η οποία κρατά δεδομένα σχετικά με το ποιός παίκτης παίζει αυτόν τον γύρο και τον αριθμό των γύρων.

Τέλος, η κλάση Player, η οποία έχει τα εξής γνωρίσματα και μεθόδους:

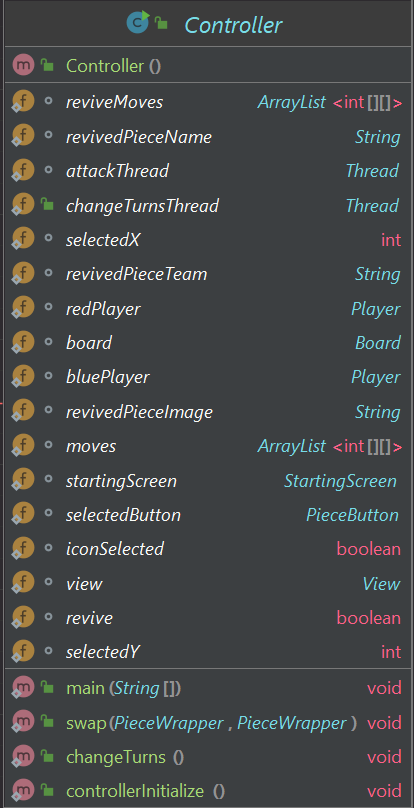
* List pieces: Τα πιόνια που έχει αυτήν την στιγμή διαθέσιμα
* int noRevives, noAttacks, noSuccAttacks, noCaptures: Ο αριθμός διασώσεων, επιθέσεων, επιτυχημένων επιθέσεων, και αιχμαλωτίσεων, αντίστοιχα.
* int noAvailablePieces: Ο αριθμός των διαθέσιμων πιονιών.
* String name: Το όνομα του παίκτη, κόκκινο ή μπλε.
* Double percentageOfSuccAttacks: το ποσοστό των επιτυχημένων επιθέσεων.
* Boolean isTurn: Εάν είναι true, τότε είναι η σειρά του παίκτη να επιτεθεί, αλλιώς να αμυνθεί

Setters και getters για τα παραπάνω γνωρίσματα, και τις εξής μεθόδους:

* void incrementAttacks(): Αυξάνει τον μετρητή των επιθέσεων κατά 1,
* void incrementSuccAttacks(): Αυξάνει τον μετρητή των επιτυχημένων επιθέσεων κατά 1,
* void addPieceCaptured(Piece piece): Προσθέτει το πιόνι piece στην λίστα των πιονιών που έχει αιχμαλωτίσεις,
* void removeCapturedPiece(String pieceName): Αφαιρεί από την λίστα των αιχμαλωτισμένων πιονιών το πιόνι με ονομα pieceName,
* void updateSuccAttackPercentage(): Ενημερώνει το ποσοστό των επιτυχημένων επιθέσεων.

## Η Σχεδίαση και οι Κλάσεις του Πακέτου Controller

Για αυτό το πακέτο, υπάρχει μία κλάση, η κλάση Controller, η οποία έχει μία εσωτερική κλάση, την PieceListener:



Κάποια από τα πιο σημαντικά γνωρίσματα αυτής της κλάσης είναι:

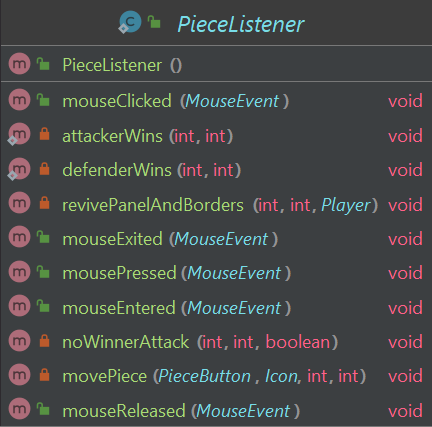
* int selectedX, selectedY: οι συντεταγμένες του πιονιού που πατήθηκε τελευταίο,
* ArrayList<int[][]> moves, reviveMoves: Λίστες με τις διαθέσιμες θέσεις για να κινηθεί το πιόνι, και για να τοποθετηθεί ένα πιόνι μετά από διάσωση, αντίστοιχα.

Στην controllerInitialize() αρχικοποιούνται οι 2 παίκτες, το ταμπλό και το view.

Η swap() είναι υπεύθυνη για την ανταλλαγή των πιονιών μετά από κάθε γύρο. Αυτό γίνεται μέσω μιας βοηθητικής κλάσης PieceWrapper, η οποία έχει ένα όρισμα τύπου Piece, και η ανταλλαγή γίνεται μέσω αυτής της κλάσης.

Επίσης, η changeTurns(), κρύβει τα πιόνια του αντίπαλου παίκτη, καθώς και αλλάζει διάφορα δεδομένα στο παράθυρο, όπως το sidepanel που βρίσκεται και στα δεξιά.

Τελος, η checkForEndGame(), ελέγχει εάν υπάρχουν διαθέσιμες κινήσεις για κάποιον παίκτη, οπότε εάν δεν υπάρχουν κινήσεις, το παιχνίδι τελειώνει.(Η μέθοδος αυτή δεν υπάρχει στην φωτογραφία.)

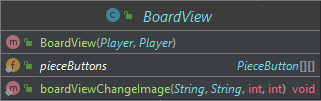
Η κλάση Piece Listener:

Κάνει extend την MouseListener, και υλοποιεί την mouseClicked(MouseEvent) μέθοδο. Ουσιαστικά, όταν πατάμε ένα κουμπί δηλαδή μία θέση στο ταμπλό, εάν είναι πιόνι του παίκτη που παίζει τώρα, αποθηκεύεται η θέση του στο ταμπλό σε βοηθητικές μεταβλητές. Επίσης, υπογραμμίζονται οι διαθέσιμες θέσεις που μπορεί να μετακινηθεί το πιόνι, και εάν πατήσει κάποιο από αυτά τα κουμπιά, τότε μετακινείται το πιόνι, κρύβονται τα πιόνια του παίκτη, και εμφανίζονται τα πιόνια του άλλου παίκτη.

## Η Σχεδίαση και οι Κλάσεις του Πακέτου View

Στην αρχή, υπάρχει ένα παράθυρο, που υλοποιείται από την κλάση StartingScreen, που περιέχει το κουμπί εκκίνησης παιχνιδιου, και τις επιλογές για τους ειδικούς κανόνες. Έπειτα, ανοίγει το βασικό παράθυρο, για το οποίο θα μιλήσω παρακάτω. Μόλις κάποιος παίκτης κατακτήσει την σημαία, το παιχνίδι λήγει και γίνεται μετάβαση σε ένα άλλο παράθυρο, το οποίο υλοποιείται από την κλάση EndScreen, η οποία περιέχει ένα label, όπου λέει “X Player Won”, όπου X ο μπλε παίκτης ή ο κόκκινος παίκτης.

Το βασικό παράθυρο αποτελείται από 2 βασικά κομμάτια: Το ταμπλό, και το sidepanel. Το ταμπλό υλοποιείται από την κλάση BoardView:



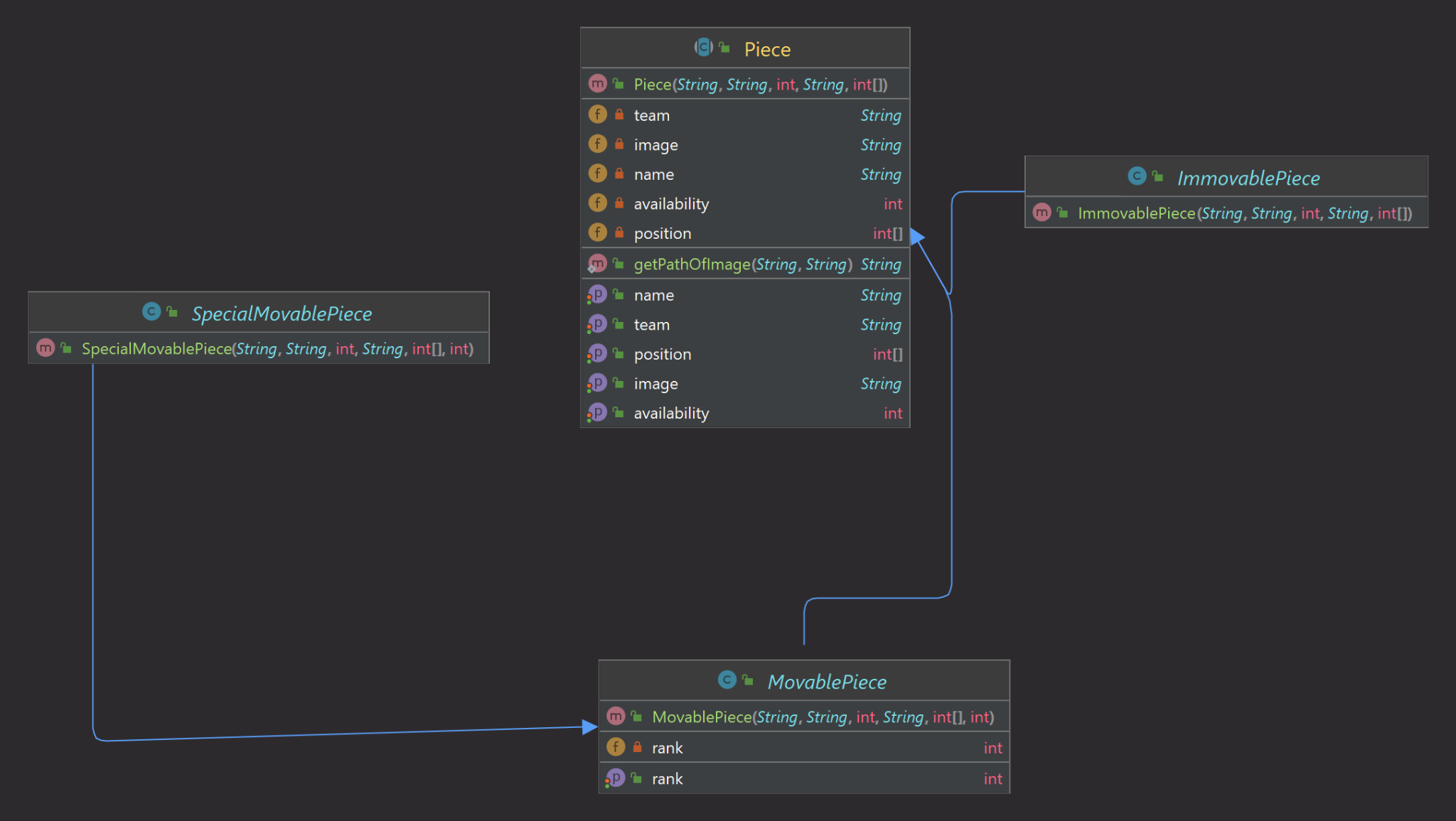
Η οποία έχει έναν πίνακα 8x8 τύπου PieceButton, όπου η κλάση αυτή είναι μία επέκταση της JButton, η οποία έχει όταν αρχικοποιείται παίρνει μία εικόνα, εάν είναι πιόνι, ή ένα χρώμα, εάν είναι κάτι άλλο.

Ουσιαστικά αυτός ο πίνακας αντιπροσωπεύει το ταμπλό του παιχνιδιού, και πάνω σε αυτόν κάνουμε αλλαγές για το view.

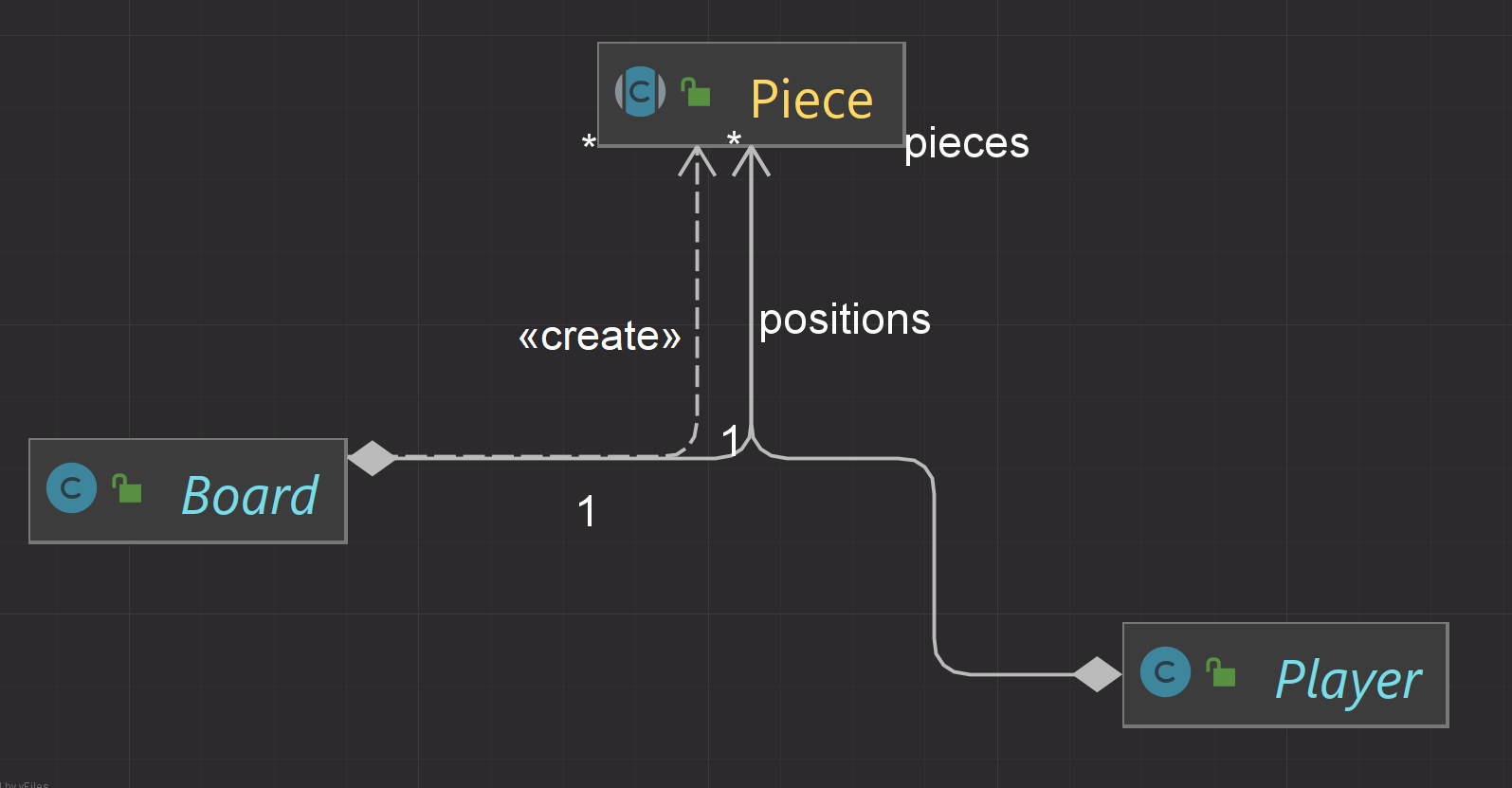
Η μέθοδος boardViewChangeImage, παίρνει σαν ορίσματα το όνομα του πιονιού που θέλουμε να είναι η εικόνα, την ομάδα του πιονιού, και τις συντεταγμένες πάνω στο ταμπλό, και θέτει σαν εικόνα την εικόνα του πιονιού που δόθηκε, με την ομάδα που δόθηκε, στο κουμπί που βρίσκεται στις συντεταγμένες που δόθηκαν.

## Η Αλληλεπίδραση μεταξύ των κλάσεων – Διαγράμματα UML

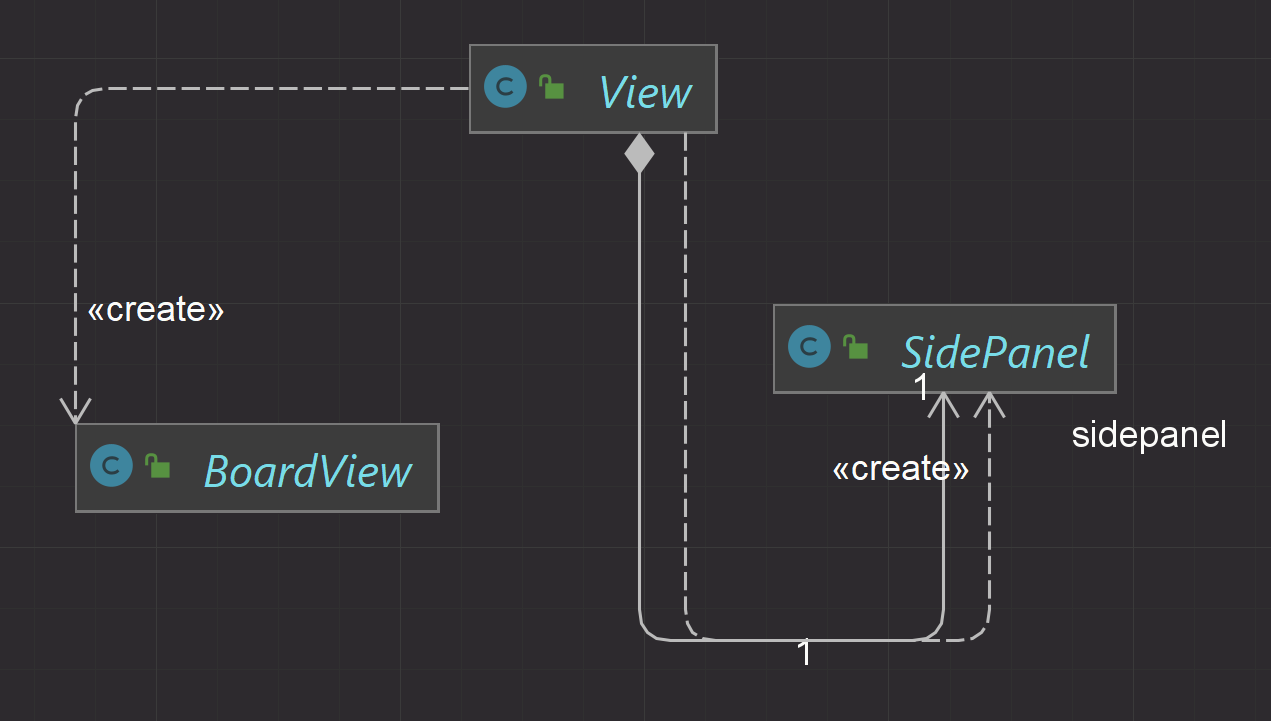
Διαγραμματικά, υπήρχαν οι εξής αλληλεπιδράσεις μεταξύ κλάσεων:



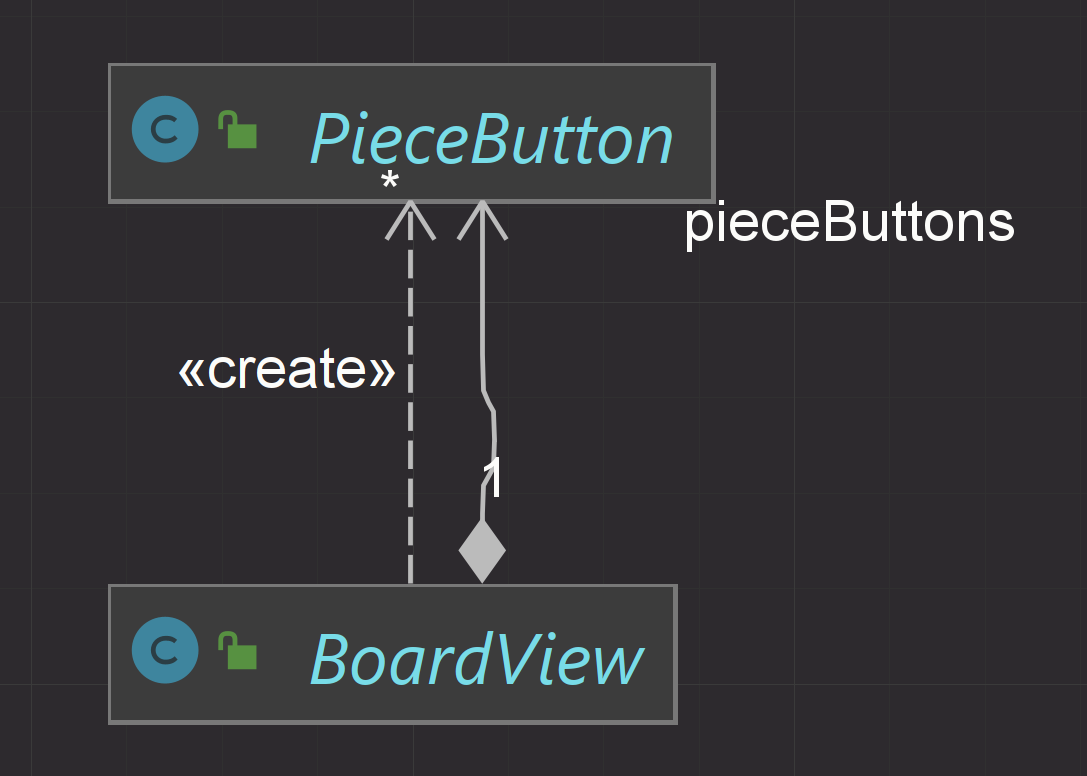
Η Piece class, είναι η super class, όπου κάνουν extend 2 κλάσεις, η MovablePiece και η ImmovablePiece. Επίσης, από την MovablePiece, κάνει extend η SpecialMovablePiece.



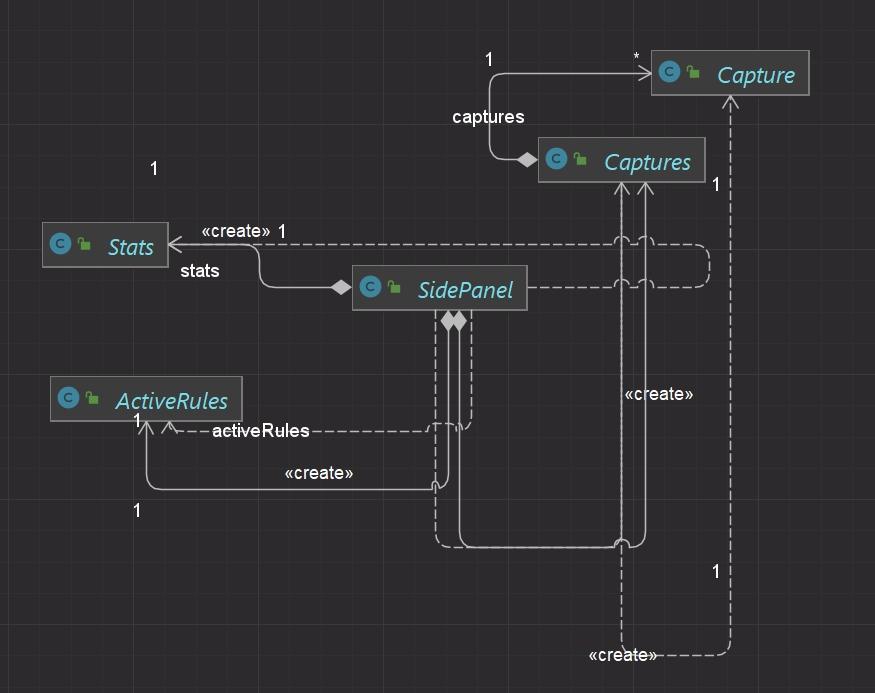
Το ταμπλό έχει πολλά πιόνια, όπως επίσης και ο κάθε παίκτης έχει πολλά πιόνια. Αυτό που δεν φαίνεται στο σχήμα, είναι ότι το ταμπλό θα έχει 2 παίκτες συνδεδεμένους με αυτό, καθώς και 2 παίκτες θα παίζουν το παιχνίδι.



Όπως βλέπουμε, το View δημιουργεί ένα BoardView, και ένα Sidepanel, τα οποία προστίθενται πάνω στο View.



Όπως παρατηρούμε από το διάγραμμα, ένα BoardView δημιουργεί πολλαπλά PieceButton, τα οποία είναι τα πιόνια στο ταμπλό.



Τέλος, στο παραπάνω διάγραμμα βλέπουμε ότι το SidePanel δημιουργεί τα 3 κομμάτια ActiveRules, Stats, Captures. Αυτά αποτελούν τα 3 μέρη του SidePanel. Επίσης είναι σημαντικό να αναφερθεί ότι για το κομμάτι Captures, η κάθε μία αιχμαλώτιση είναι μία κλάση Capture, και όπως φαίνεται από το σχήμα, μπορούν να δημιουργηθούν πολλαπλά στιγμιότυπα της κλάσης αυτής.

## Λειτουργικότητα (Β Φάση)

Τα ερωτήματα τα οποία κατάφερα να ολοκληρώσω με επιτυχία ήταν:

Την λειτουργικότητα και ορθότητα του παιχνιδιού, την σωστή μετακίνηση των πιονιών στο ταμπλό, την σωστή λειτουργικότητας της διάσωσης κάποιου πιονιού, την ορθή λειτουργικότητα της επίθεσης μεταξύ 2 πιονιών, την υλοποίηση του βοηθητικού πάνελ, την ορθότητα των ειδικών κανόνων.

Σε γενικές γραμμές το πρόγραμμα λειτουργεί άψογα, με ορθή λειτουργικότητα.

## Συμπεράσματα

Τα συμπεράσματα μου για αυτήν την εργασία είναι ότι ήταν σίγουρα μία αρκετά χρονοβόρα εργασία, αρκετά απαιτητική, αλλά όχι ακατόρθωτη. Έμαθα πολλά πράγματα για την Java κάνοντας την εργασία, και σίγουρα πέρασα πολύ ωραία. Προσωπικά, νιώθω πολύ περήφανος που κατάφερα να υλοποιήσω, αν όχι στο μέγιστο, σε πολύ μεγάλο βαθμό ένα παιχνίδι που μεγάλωσα παίζοντας.