
ΕΥΡΩΠΗ

Πανεπιστήμιο Κρήτης –Τμήμα Επιστήμης Υπολογιστών

HY252– Αντικειμενοστρεφής Προγραμματισμός

Διδάσκων: Ι. Τζίτζικας

Χειμερινό Εξάμηνο 2020-2021

SORRY!
PROJECT 2023-2024

Εισαγωγή

Σινάνης Ανδρέας

csd5150

2023-2024

Περιεχόμενα

1. Εισαγωγή

Το μοντέλο MVC που θα ηλοποιήσω θα βοηθείσει στην οργανμένη δομή του project. Ακολουθούν περιγραφές για το κάθε ένα ξεχωριστά.

2. Η Σχεδίαση και οι Κλάσεις του Πακέτου Model

Το πακέτο model περιέχει τα class που χρειάζονται για την λειτουργία του παιχνιδιού, πιο συγκεκριμένα το πακέτο περιλαμβάνει:

1) Τα card classes: *Που υλοποιούν τις κάρτες του παιχνιδιού*

~Abstract class card

Attributes:
private player owner, ποιος παίκτης θα χρησιμοποιήσει την κάρτα
private cardName, δηλώνει τι κάρτα ειναι
boolean active, αν η συγγκεκριμένη κάρτα μπορέι να παιχτεί
public string image, η εικόνα που θα αντιπροσωπεύει την κάρτα στο view
Constructor: card(String image), ορίζει την εικόνα
Methods:
public String getImage(), transformer ορίζει την εικόνα της κάρτας
public boolean isActive(), accessor επιστρέγει αν ειναι η κάρτα ενεργή
public String toString(), override την toString
public void movePawn(pawn Pawn, board Board), η οποια μετακινει το πιόνι αναλογα

~deck

Attributes:
private int capacity = 44, ο αριθμός των καρτών που μπορούν να παιχτούν
public Stack<card> Deck, η στοίβα που περιέχει τις κάρτες
Methods:
public card foldCard(), accessor το οποίος επιστρέφει την κάρτα που θα παίχτει
public void decreaseCapacity(), transformer το οποίο μειώνει το capacity
public int getCapacity(), accessor το οποίο επιστρέφει το capacity

~numberCard extends card

Attributes:
private int number, δινει τον αριθμό της κάρτας
private boolean interactive, δινεται η επιλογή στον παίκτη να κανει 2 πραγματα τα οποιά πρέπει να επιλέξει τι απο αυτα θα κανει
Methods:
public boolean getInteractive(), accessor το οποιο επιστρεφει αν χρειάζεται interaction με τον παικτη
Οι υπόλοιπες κάρτες δεν έχουν κάποιες παραπάνω άξιες για αναφορά μεθόδους καθώς το μόνο διαφορετικό μεταξύ τους είναι πώς εξελίσσουν το παιχνίδι μέσω της movePawn:

~sorryCard extends card

~numberCard extends card

~simpleNumberCard extends numberCard

~numberOneCard extends numberCard

~numberTwoCard extends numberCard
~numberFourCard extends numberCard
~numberSevenCard extends numberCard
~numberTenCard extends numberCard
~numberElevenCard extends numberCard

2) Τα square classes: *Που υλοποιούν τα τετράγωνα του παιχνιδιού*

~square

Attributes:

private boolean state, καθορίζει αν βρισκεται κάποιο πιόνι πανω στο τετράγωνο
private Color color, δηλώνει την περιοχή στην οποία ανήκει ανάλογα το χρώμα
private int x ,y δηλώνει την θέση του τετραγώνου
private pawn Pawn, το πιόνι που μπορει να βρίσκεται πανω στο τετράγωνο
enum typeSquare το είδος του τετραγώνου

Methods:

public boolean getState(),accessor επιστρέφει αν βρίσκεται πίονι στο τετράγωνο
public void setState(boolean state), transformer αλλάζει την κατάσταση του τετραγώνου αναλόγως
public Color getColor(), accessor επιστρέφει το χρώμα
public int getX(), getY(), accessor επιστρέφουν το x και ψ ανάλογα
public pawn getPawn(), accessor επιστρεφει το πίονι που βρίσκεται το τετράγωνο
public void setPawn(pawn Pawn), transformer ορίζει το πιόνι που βρίσκεται πάνω την δεδομένη στιγμή
public typeSquare getSquareType(),accessor επιστρεφει το είδος του τετραγώνου
public void setSquareType(typeSquare), ορίζει το είδος του τετραγώνου
public changeLocation(pawn Pawn), αλλαζει το πεδίο τοποθεσίας στο δεδομένο πιόνι σε enum.pawnLocation.regular (μονο τα homeSquare, safetyZoneSquare, startSquare αλλαζουν το *enum* αναλογα διαφορετικα απο regular)

~startSquare extends square

Methods:

public changeLocation(pawn Pawn), αλλαζει το πεδίο τοποθεσίας στο δεδομένο πιόνι σε enum.pawnLocation.home

~simpleSquare extends square

~slideSquare extends square

~safetyZoneSquare extends square

Methods:

public void setSafe(pawn Pawn), transformer αλλάζει το setSafe του δεδομένου πιονίου σε true καθως βρίσκεται σε περιοχή ασφαλίας
public void changeLocation(pawn Pawn), transformer κανει override την μεθοδο

της υπερκλάσης και αλλάζει την τοποθεσία σε safe, επίσης καλεί την setSafe()

~homeSquare extends safetyZoneSquare

Methods:

void setImmovable(pawn Pawn, player Player), transformer μετα την κλίση αυτης της μεθόδου το πιόνι δεν μπορεί να κουνιθεί αφου έχει φτάσει στο τελικό τετράγωνο (επίσης καλεί για το πιόνι την μέθοδο setImmovable(), και για το Player την decreaseMovablePawns())

~startSlideSquare extends slideSquare

~internalSlideSquare extends slideSquare

~endSlideSquare extends slideSquare

3) To board class

To board class κυρίως πορετοιμάζει το ταμπλο του παιχνιδιού με τα κατάλληλα τράγωνα, συγκεκριμένα ορίζει:

square[][] Board, τις γραμμές με 16*4 τετράγωνα (περιέχοναι: regular squares και slideSquares)

square [][] safe zone, το οποιο αντιπροσοπεύει το safeZone (περιέχει και το homeSquare)

4) To pawn class

Attributes:

private int color, το χρώμα του πιονιόν

private int y, x, δηλώνει την θέση του πιονιού

private boolean isSafe, δηλωνει αν το πιονι ειναι ασφαλης (αν μπορει να το στειλει ο αντιπαλος πισω στο start)

private boolean movable, αν εχει φτασει στο πεδιο Home

private pawnLocation location, δηλώνει σε τι τετράγωνο βρισκεται το πιόνι

Methods:

public int getY, getX, setX, setY, μεθώδοι για την τοποθεσία

public void setColor(Color color), Transformer οριζει το χρώμα

public Color get Color, Accessor επιστρέφει το χρώμα

public void setSafe(boolean isSafe), transformer οριζει αν ειναι ασφαής το πιόνι

public boolean getSafe(), accessor επιστρέφει αν ειναι ασφαής το πιόνι

pubic void setLocation(pawnLocation location), transformer ορίζει την τοποθεσία του

public pawnLocation getLocation(),accessor επιστρέφει την τοποθεσία του

public void setImmovable(),transformer χρησιμοποιείται οταν το πιόνι φτάσει στο Home

5) To player class

Attributes:

private int movablePawns, ο αριθμος απο τα πίονια
private boolean winner, αν εχει νικήσει
private pawn Pawn1, Pawn2 τα πιόνια που έχει ο παίχτης στην κατοχή του
private Color color, το χρώμα του παίκτη
Methods:
public Color getColor(), Accessor επιστρέφει το χρώμα του παίχτη
public void decreaseMovablePawns(), transformer κρατάει πληροφορίες αν τα πιόνα ειναι στο Home (στην περίπτωση που ειναι και τα δύο τοτε αλλάζει τον κηρυσσει νικητη)
public boolean isWinner(), accessor επιστρέφει αν εχει νικήσει
public pawn selectPawn(), accessor δίνει ποιό πιόνι θα κουνηθεί.

3. Η Σχεδίαση και οι Κλάσεις του Πακέτου Controller

UML για το model

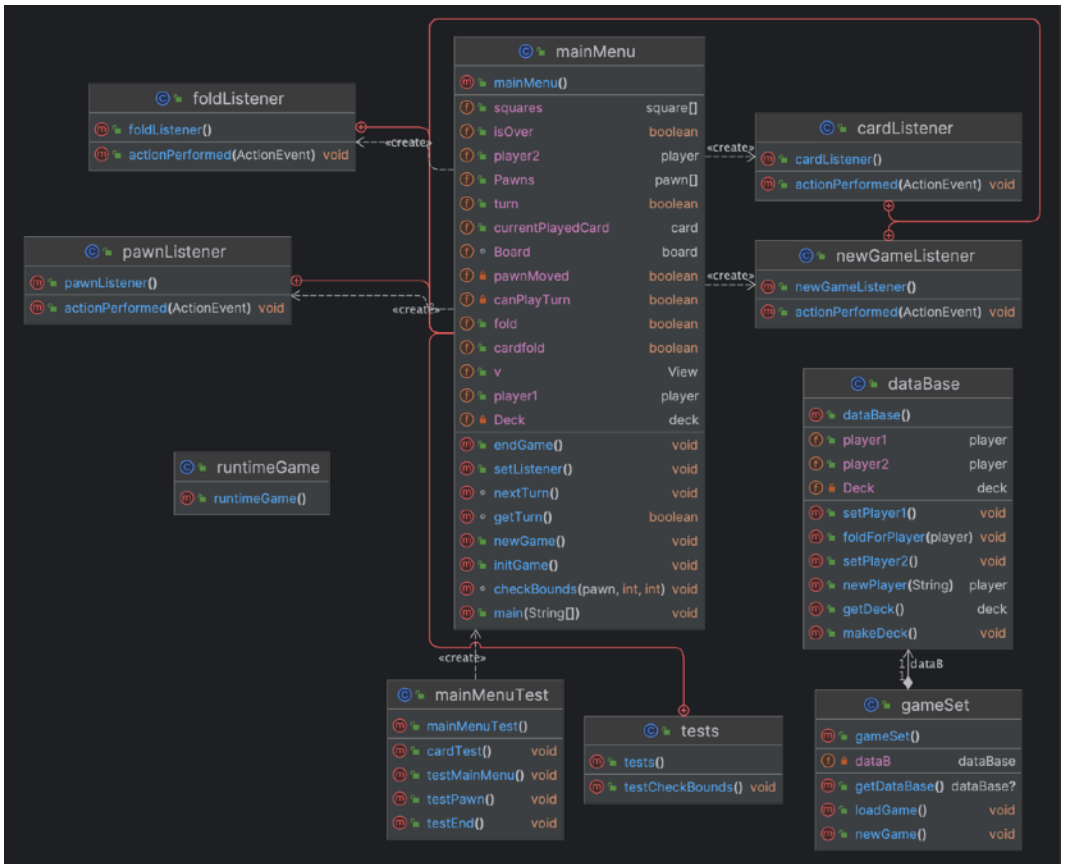
Η κλάση controller είναι υπεύθυνη για την λειτουργία του παιχνιδιού καθώς παίρνει τις επιλογές από το το view και κάνει την σύνδεση μεταξύ του model, αυτο πραγματοποιείται με τις εξής βοηθητικές κλάσεις

Το mainMenu class το οποιο περιέχει την main απο όπου ξεκινάει και το παιχνίδι.

Επίσης η mainMenu έχει τις εξής λειτουργίες.

~Το endGame, που μετά αποκάθε κίνηση που έγινε ελέγχει αν και τα δύο πιόνια βρήσκονται στην περιοχή "Σπίτι" και αναλόγως τερματίζει το παιχνίδι.

~Το SetListeners που διαμορφώνει τα παρακάτω κουμπία κουμπιά:



UML controller

1)fold button μέσω του foldListener, το κουμπί που δηλώνει πως ο παίκτης τερμάτησε την σειρά του

2)receiveCard button μέσω του cardListener, κουμπί που τραβάει κάρτα ο παίκτης και παίζει με ανάλογες κινήσεις

3)pawn button μέσω του pawnListener, εδώ γίνονται οι περισσότερες αλλαγές στο παιχνίδι, μέσω μίας switch case σε διαμορφώνονται στο παιχνίδι οι ανάλογες κινήσεις και στην περίπτωση που χρειαστέι να παρθεί απόφαση απο τον χρήστη αυτό γίνεται μέσω διαλόγου με δύο επιλογές ως απάντηση

4)menu bar μέσω του newGameListener που ξεκινάει νεό παιχνίδι

~Το nextTurn αλλάζει την σειρά του παιχνδίου αλλάζοντας την τιμή turn σε turn = !turn

~Το getTurn επιστρέφει την σειρά

~Το newGame ξεκινάει νέο παιχνίδι

~Το initGame κάνει initialize τους παίκτες και τα πίονια τους

~Το checkBounds ελέγχει αν τα πιόνια μετά την κίνηση τους βγαίνουν εκτός των ορίων και την περίπτωση που γινει κάνει τις ανάλογες αλλαγές. Στην παρούσα συνάρτηση επίησς έγινε προσπάθεια υλοποίησης για τον έλεγχο που χρειάζεται για να φτάσει ένα πιόνι στο safeZone (έχει μείνει με σχόλια).

Οι μεταβλητές boolean που χρεισημοποιήθηκαν για την σωστή λειρουργία του παιχνιδίου είναι:

1)isOver, αν έχει τελείωει το παιχνίδι

2)turn, σειρά παικτών

3)pawnMoved, αν κουνήθηκε πίονι με την κάρτα

4)canPlayTurn, αν είναι ώρα για τον παίκτη να παίξει

5)fold, για το fold να μη μπορεί να πατηθεί πολλές φορές και να αλλάζει συνέχεια η σειρά

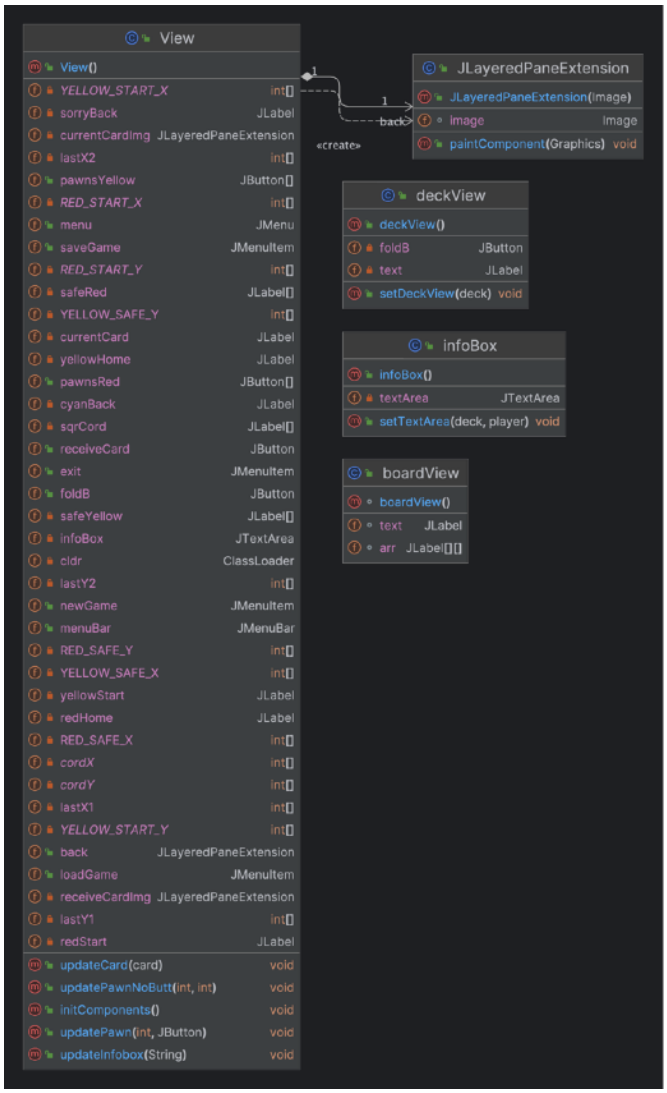
6)cardFold αν έχει τραβήχτηκε κάρτα

Τέλος η κλάση mainMenuTest που περιέχει τα JUnit tests

Για λόγους χρόνου και ευκολίας δεν υλοποιήθηκαν οι κλάσεις: dataBase gameSet και runtime game

4. Η Σχεδίαση και οι Κλάσεις του Πακέτου View

4.

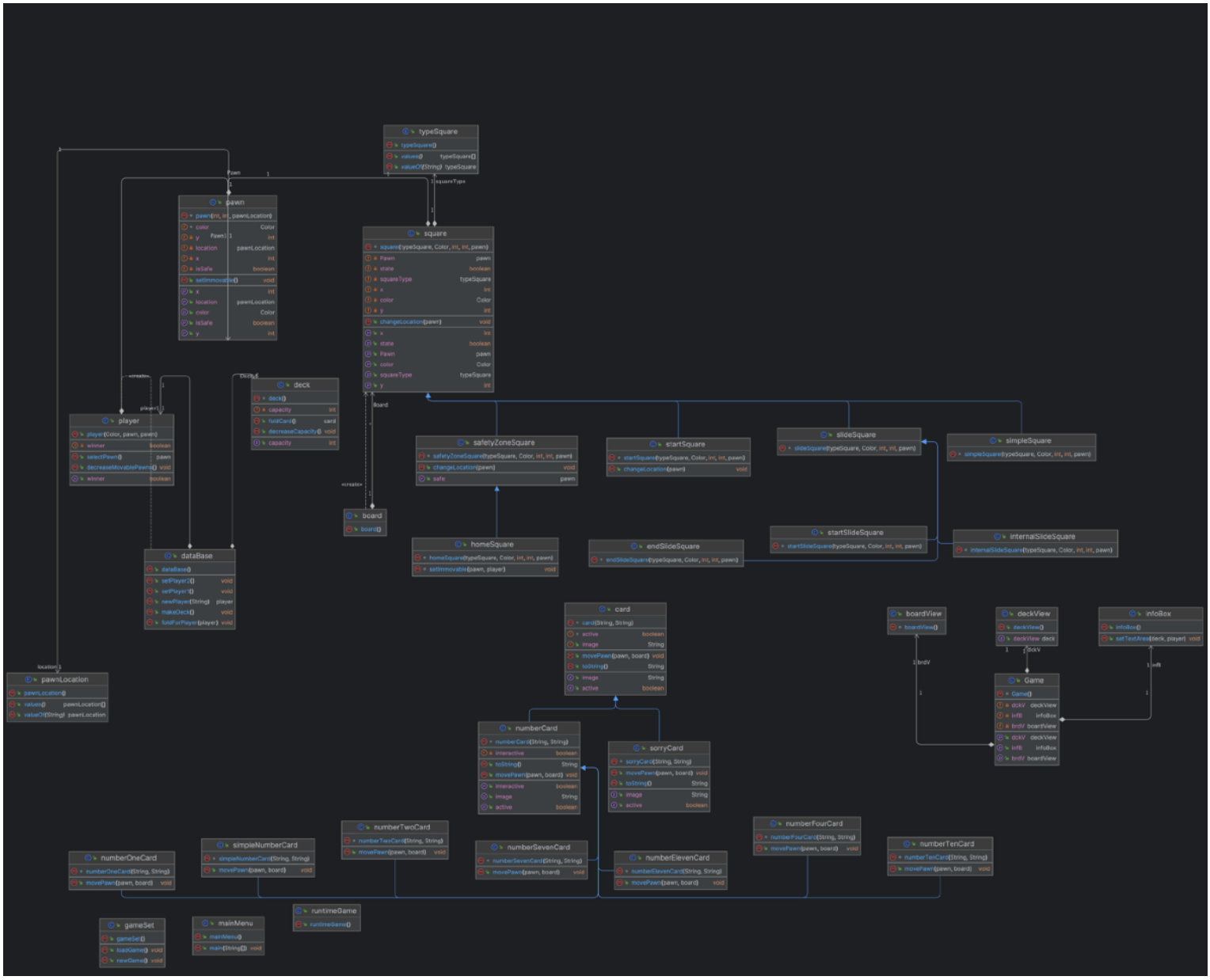


Το Πακέτο view είναι υπεύθυνο για το τι βλέπει ο χρήστης και δίνει στις τον controller τις αντίστοιχες πληροφορίες για να ανανεώσει την εικόνα και να συνεχίσει κανονικά η ροή του παιχνιδίου.
Περιλαμβάνει τις κλάσεις που φαίνονται στο UML δίπλα, για τους ίδιους λόγους όπως και πάνω χρησημοποίηθηκαν μόνο οι κλάσεις View kai JLayerdPaneExtension (η οποία ειναι απο τα έτοιμα παραδείγματα που μας δώθηκαν)

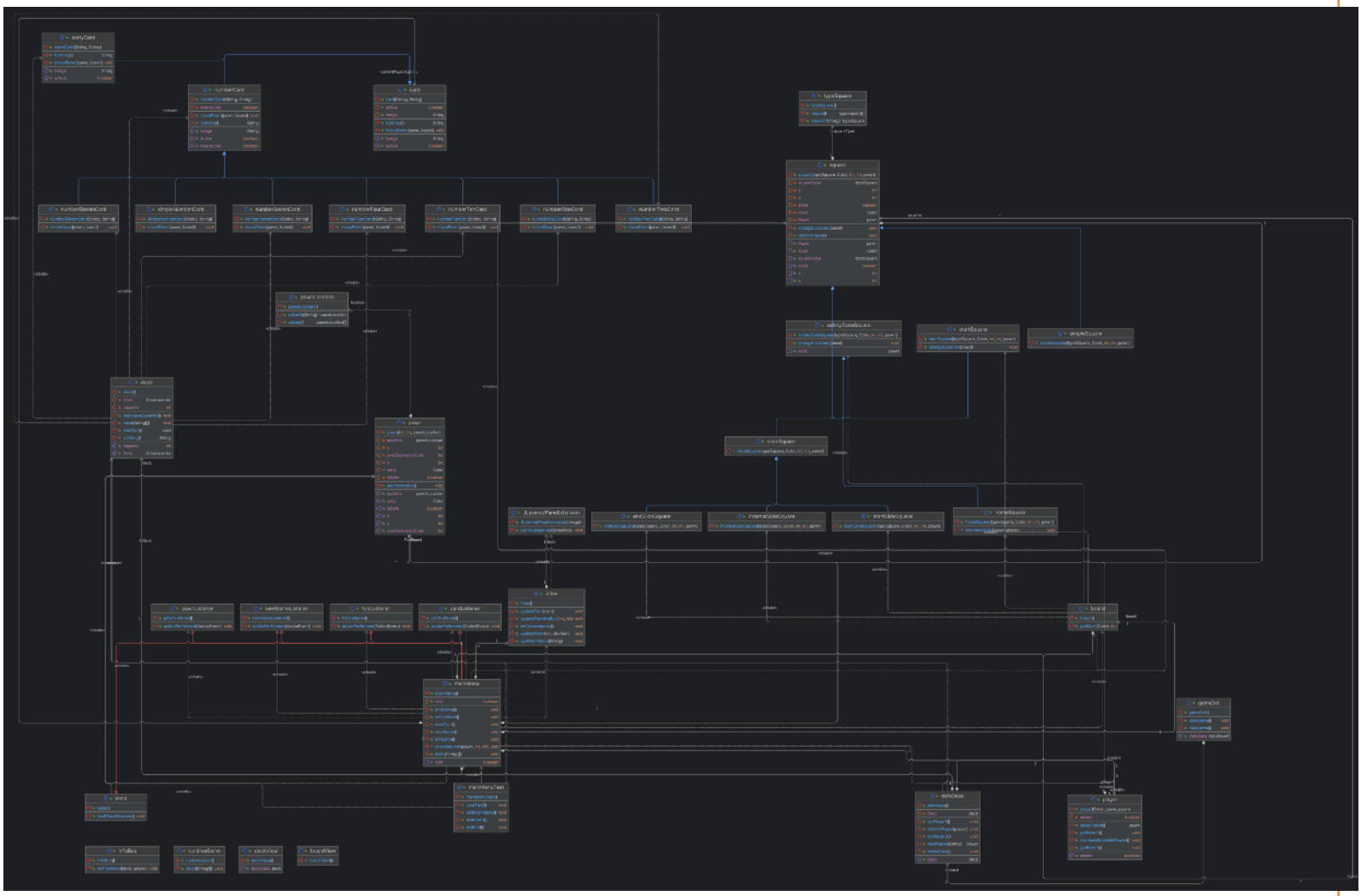
Η κλάση view περιλαμβάνει τις εξής λειτουργίες:
1) update card, η οπόια δείχνει την κάρτα που έχει τραβηχτεί στην οθόνη
2)updatePawnNoButt, αλλάζει θέση σε πίονι χωρίς να έχει πατηθεί (για τις κάρτες που κουνάνε δύο πίονια ταυτόχρονα)
3)initComponents, διαμορφόνει όλα τα components και τα ενώνει στην οθόνη
4)updatePawn, αφού πατηθεί ένα κουμπί-πιόνι το αλλάζει στην ανάλογη θέση
5)updateInfobox, ενημερώνει το infoBox με τις πληροφορίες που χρειάζεται

5. Η Αλληλεπίδραση μεταξύ των κλάσεων – Διαγράμματα UML

Ακολουθεί το UML για όλο το project



UML (before phaseB)



UML (after phaseB)

6. Λειτουργικότητα (Β Φάση)

Τα μέρη του project που δεν υλοποιήθηκαν σωστά πιστεύω όπως ήταν σωστή χρήση slide, σωστή χρήση Home και κανόνες για πίονια στην ίδια θέση στο μέρος του controller

7. Συμπεράσματα

Η εργασία με βοήθησε να κατανοήσω την λειτουργία της κληρονομικότητας αλλά, κυρίως με βοήθησε να μάθω πως να επικοινωνώ με τον χρήστη χωρίς να χρησιμοποιώ μόνο το console.