Πανεπιστήμιο Κρήτης –Τμήμα Επιστήμης Υπολογιστών

ΗΥ252 – Αντικειμενοστρεφής Προγραμματισμός

Διδάσκων: Ι. Τζίτζικας

Χειμερινό Εξάμηνο 2019-2020

ΔΗΜΙΟΥΡΓΙΑ ΕΠΙΤΡΑΠΕΖΙΟΥ ΠΑΙΧΝΙΔΙΟΥ ΣΕ JAVA

ΦΑΣΗ Β’

Μάνος Χατζάκης

4238

6/1/2019

ΠΕΡΙΕΧΟΜΕΝΑ

1. Εισαγωγή

Η παρόν αναφορά αποτελεί μία γενική παρουσίαση της υλοποίησης ενός επιτραπέζιου παιχνιδιού στα πλαίσια του μαθήματος «Αντικειμενοστρεφής Προγραμματισμός». Η εργασία έγινε βάσει του μοντέλου MVC (Model – View – Controller) και θα παρουσιαστούν τα πακέτα, οι κλάσεις και οι βασικότερες μέθοδοι χρησιμοποιήθηκαν. Ακολουθούν κατά σειρά: Το πακέτο model και οι κλάσεις του, το πακέτο Controller και η κλάση του, το πακέτο View καθώς και τα επιπρόσθετα πακέτα Utilities, Test, Application και Exceptions. Τέλος, υπάρχει το UML και κάποια γενικά σχόλια.

2. Η Σχεδίαση και οι Κλάσεις του Πακέτου Model

Στο πακέτο model υπάρχουν 4 υπό πακέτα, που περιέχουν τις απαραίτητες κλάσεις που αποτελούν το μοντέλο/πυρήνα του παιχνιδιού.

A. Το Πακέτο Player

Σε αυτό το πακέτο περιέχεται η **κλάση Player**, η οποία **προσομοιώνει έναν παίχτη του επιτραπέζιου παιχνιδιού**. Κάθε παίχτης, έχει συγκεκριμένο όνομα και χρώμα, κρατάει στην κατοχή του 2 πιόνια ενώ παράλληλα χαρακτηρίζεται από το αν είναι η σειρά του να παίξει η όχι.

Μέσα σε αυτήν την κλάση υπάρχουν τα αντίστοιχα πεδία **color, name, pawn1, pawn2 και turn**, με όλους τους αντίστοιχους setters/getters και τον constructor που αρχικοποιεί αυτά τα πεδία.

Μέσα στην κλάση αυτή περιέχονται και κάποιες βασικές μέθοδοι, που θα χρησιμοποιηθούν για την ομαλή ροή του παιχνιδιού. Συγκεκριμένα, η transformer μέθοδος **fold():void** η οποία είναι υπεύθυνη για την επιλογή του παίχτη να πάει «πάσο», αλλάζοντας την σειρά (turn).

Επιπροσθέτως, υπάρχει η observer μέθοδος **hasWon():Boolean** που ελέγχει αν ο παίκτης έχει νικήσει. Αυτό το κάνει βλέποντας αν και τα δύο πιόνια του παίκτη είναι στο home square.

Παράλληλα, η observer μέθοδος **isHisPawn(Pawn p):Boolean** ελέγχει αν το πιόνι p ανήκει στον παίκτη, καθώς και η **hasPawnInStart():Boolean** που ελέγχει αν ο παίκτης έχει έστω ένα πιόνι στο start. Η μέθοδος **hasBothPawnsInStart():Boolean** ελέγχει αν και τα δύο πιόνια του παίκτη είναι στο start.

Υπάρχει και η observer μέθοδος **hasPlayed():Boolean** η οποία επιστέφει true αν και μόνο αν ο παίκτης έχει παίξει τον γύρο του. Με την

βοήθεια αυτής της μεθόδου (που υπάρχει αντί της getter μεθόδου του πεδίου turn) καθορίζεται η σειρά των παικτών.

Τέλος υπάρχει η μέθοδος **toString():String** που κάνει override την toString(); της κλάσης object, για να τυπώνει πληροφορίες για τον παίκτη, στην κονσόλα η σε πλαίσιο πληροφοριών που θα υπάρχει στα γραφικά.

Β. Το πακέτο Pawn

Στο συγκεκριμένο πακέτο, υπάρχει η αντίστοιχη κλάση Pawn, η οποία προσομοιώνει ένα πιόνι του παιχνιδιού. Τα πιόνια του επιτραπέζιου έχουν χρώμα και θέση στο ταμπλό, ενώ μπορούν να είναι διαθέσιμα για τον παίκτη να τα κουνήσει(enabled) αν και μόνο αν αυτά δεν βρίσκονται στην θέση Home.

Συνεπώς, η κλάση έχει τα πεδία **color, position, homePosition** και **enabled** καθώς και τις αντίστοιχες setter/getter μεθόδους ενώ ο constructor τα αρχικοποιεί.

Υπάρχουν επίσης κι άλλες μέθοδοι που βοηθούν στην λειτουργία του παιχνιδιού. Συγκεκριμένα:

- Observer μέθοδοι σχετικά με τα positions του πιονιού:
  - **isInStart():Boolean**
  - **isInHome():Boolean**
  - **isInSafetyZone():Boolean**

  Οι παραπάνω επιστέφουν true αν το πιόνι είναι σε ένα από τα παραπάνω τετράγωνα και προσδίδουν χρήσιμη πληροφορία για την ροή του παιχνιδιού.

  Γνωρίζοντας σε ποια θέση είναι ένα πιόνι, μπορούμε να γνωρίζουμε ποιες κινήσεις επιτρέπονται βάσει της κάρτας που έχει ανοίξει ένας παίκτης, ενώ μπορούμε εύκολα να καταλάβουμε αν κάποιος από τους δύο αντιπάλους έχει κερδίσει.

- Επιπλέον observer μέθοδοι σχετικά με τα positions του πιονιού:
  - **isProtected():Boolean.** Αυτή η μέθοδος αξιοποιεί τις μεθόδους που παρουσιάστηκαν, βοηθώντας στο να καταλάβουμε αν επιθέσεις από κάρτες του αντιπάλου είναι εφικτές. (Ελέγχει αν το πιόνι είναι σε κάποιες από τις θέσεις start, home, safety zone και προσδίδει στον κώδικα περισσότερη αναγνωσιμότητα).

- **isEnabled():Boolean.** Με την συγκεκριμένη μέθοδο ελέγχεται αν ο παίκτης μπορεί να κουνήσει ένα πιόνι.

- Η transformer μέθοδος **disablePawn():void**. Αυτή η μέθοδος απενεργοποιεί το πιόνι. Μετά την απενεργοποίηση, ο παίχτης δεν μπορεί να χρησιμοποιήσει πλέον το συγκεκριμένο πιόνι.
- Η transformer μέθοδος **changePosition(int position):void** η οποία, αυξάνει την θέση του πιονιού στο ταμπλό κατά position θέσεις. Η αλλαγή του position πρέπει να συμφωνεί με τους κανόνες του παιχνιδιού. Αυτή η μέθοδος θα καλείται ύστερα από την επιλογή κάρτας του παίκτη, με σκοπό να υλοποιήσει την κίνηση που υπαγορεύει η κάρτα.

Επίσης υπάρχει και η μέθοδος **toString():String** η οποία κάνει override την toString()::String της κλάσης object, με σκοπό να τυπώνει πληροφορίες σχετικά με το εκάστοτε πιόνι (όπως πχ θέση, χρώμα κτλ).

C. Το πακέτο Square

Το παρόν πακέτο δημιουργήθηκε με σκοπό να συμπεριλάβει όλες τις κλάσεις που αναπαριστούν τα τετράγωνα που υπάρχουν στο ταμπλό.

Στο πακέτο αυτό, υπάρχουν αρκετές κλάσεις, μία για κάθε είδος τετραγώνου ενώ παράλληλα γίνεται χρήση των κανόνων κληρονομικότητας που η Java προσφέρει, με σκοπό να επιτευχθούν όλα τα χαρακτηριστικά που διέπουν κάθε τετράγωνο που υπάρχει στο ταμπλό.

Συγκεκριμένα, περιλαμβάνονται οι εξής κλάσεις και μέθοδοι:

- Η **abstract class Square.**

  Αυτή η κλάση είναι abstract αφού περιλαμβάνει όλα τα χαρακτηριστικά που διέπουν τα τετράγωνα του ταμπλό, όμως σε καμία περίπτωση τα τετράγωνα είναι τύπου Square, αφού κάθε υποείδος τετραγώνου έχει και δικά του χαρακτηριστικά.

  Τα τετράγωνα έχουν χρώμα, μπορούν να έχουν πάνω τους κάποιο πιόνι, και λόγο αυτού, μπορούν να είναι κατειλημμένα ή όχι. Για αυτό υπάρχουν τα πεδία **color**, **pawn** και **occupied** με setter/getter μεθόδους και τον constructor που τα αρχικοποιεί.

  Υπάρχει και η μέθοδος transformer μέθοδος **occupy(Pawn pawn):Void** η οποία κάνει το τετράγωνο κατειλημμένο από το πιόνι pawn(ενώ η

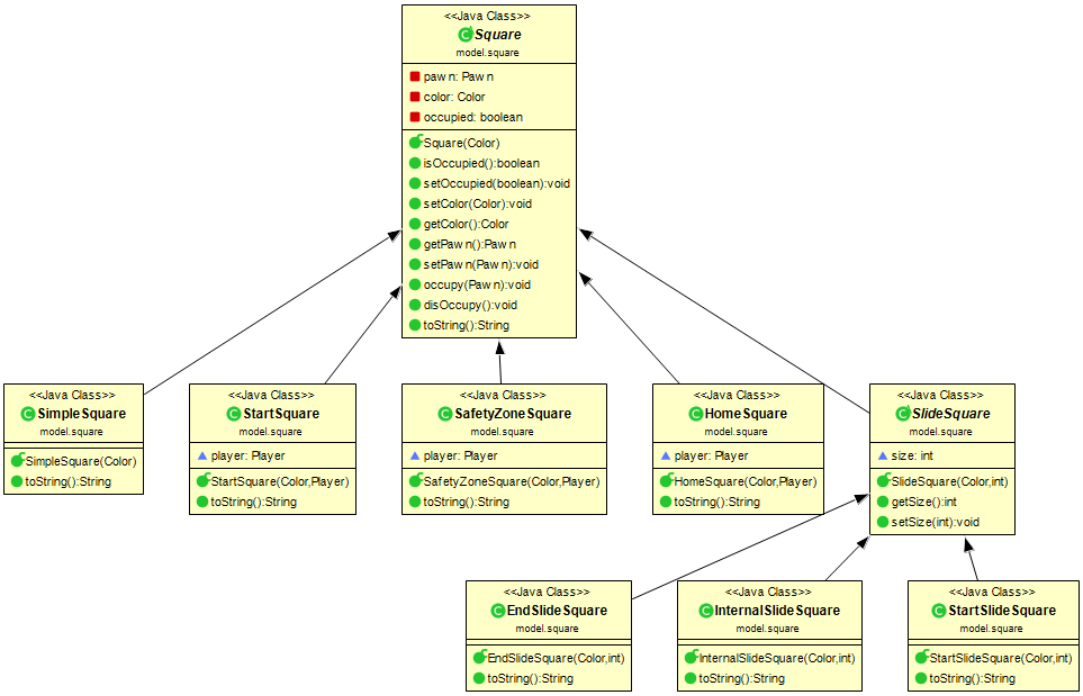
μέθοδος **disOccupy():void** το ελευθερώνει) καθώς και μία **toString()** που τυπώνει πληροφορίες για το τετράγωνο.

Για τον παραπάνω λόγο, υπάρχουν οι υποκλάσεις της Square, που ακολουθούν:

- **Κλάση StartSquare.** Η παρόν κλάση αναπαριστά τα τετράγωνα εκκίνησης κάθε παίκτη. Τα τετράγωνα αυτά αρχικά περιέχουν ένα μόνον παίκτη, για αυτό μέσα στην κλάση υπάρχει το επιπλέον **πεδίο Player**. Στην κλάση αυτή ο Constructor αρχικοποιεί και το πεδίο player.
- **Κλάση SimpleSquare.** Αυτή η κλάση αναπαριστά τα απλά λευκά τετράγωνα του ταμπλό. Ο constructor της άπλα δημιουργεί το αντικείμενο. Τα απλά τετράγωνα δεν έχουν παραπάνω συγκεκριμένα χαρακτηριστικά.
- **Κλάση HomeSquar**e. Αυτή η κλάση αναπαριστά τα τετράγωνα που ο κάθε παίκτης πρέπει να οδηγήσει τα πιόνια του. Κάθε Home αντιστοιχεί σε διαφορετικό παίκτη, για αυτό η κλάση έχει ένα επιπλέον **πεδίο Player**, το οποίο αρχικοποιείται στον constructor και θα αξιοποιηθεί στο να γίνει αντιληπτό σε ποιο Home έχει δικαίωμα ο κάθε παίκτης να πάει.
- **Κλάση SafetyZoneSquare.** Αυτή η κλάση αναπαριστά το safety zone κάθε παίκτη. Κι αυτή η κλάση έχει ως νέο πεδίο **έναν παίκτη**, αφού κάθε παίκτης έχει το δικό του safety zone. Ο παίκτης αρχικοποιείται στον constructor της κλάσης.
- **Abstract κλαση SlideSquare.** Αυτή η κλάση αναπαριστά μία τσουλήθρα συγκεκριμένου μεγέθους. Επομένως έχει ένα νέο **πεδίο size** με αντίστοιχες setter/getter μεθόδους. *Υπάρχουν 3 υποκλάσεις της slide square:*
  - **StartSlideSquare.** Αυτή η κλάση αναπαριστά το πρώτο τετράγωνο μίας τσουλήθρας. Μόνο αυτό το τετράγωνο μπορεί να κουνήσει περεταίρω ένα πιόνι. **InternalSlideSquare.** Αυτή η κλάση αναπαριστά ένα εσωτερικό τετράγωνο της τσουλήθρας. Δεν έχει παραπάνω ιδιότητες ή μεθόδους.

- **EndSlideSquare.** Αυτή η κλάση αναπαριστά το τελικό τετράγωνο μίας τσουλήθρας, δηλαδή το τετράγωνο που μεταφέρεται ένα πιόνι από την κορυφή της. Δεν περιέχει παραπάνω ιδιότητες ή μεθόδους.

Παρατίθεται το UML των τετραγώνων:



D. Το πακέτο Card

Στο πακέτο αυτό περιέχονται κλάσεις που αναπαριστούν τις διαθέσιμες κάρτες του επιτραπέζιου. Οι κάρτες αυτές είναι χωρισμένες βάσει του είδους τους και των χαρακτηριστικών τους.

Συγκεκριμένα, μέσα στο πακέτο υπάρχουν οι εξής κλάσεις και μέθοδοι:

- **Η abstract κλάση Card.** Η κλάση αυτή είναι η γενική μορφή που μπορεί να έχει μία κάρτα του παιχνιδιού. Είναι αφηρημένη, καθώς περιέχει όλα τα χαρακτηριστικά που μοιράζονται οι κάρτες, αλλά δεν υπάρχουν κάρτες τόσο απλές όσο αυτή.

  Περιέχει δύο βασικά πεδία **name και image**, με τις αντίστοιχες setter/getter μεθόδους και τον αντίστοιχο

constructor. Το name είναι το όνομα που έχει η κάρτα, ενώ το image είναι το path για την εικόνα που θα έχει η κάρτα.

Περιέχει επίσης την **abstract μέθοδο movePawn(Pawn pawn, Board board)::void** η οποία είναι υπεύθυνη για την κίνηση του πιονιού και υλοποιείται για κάθε κάρτα, βάση των ιδιοτήτων της.

Τέλος υπάρχει και η μέθοδος **toString()::String** η οποία τυπώνει πληροφορίες για την κάρτα.

Οι κλάσεις που ακολουθούν είναι υποκλάσεις της Card.

- **Η κλάση sorryCard.** Η κλάση αυτή προσομοιώνει την κάρτα "Sorry" του παιχνιδιού. Αυτή η κάρτα παίρνει ένα πιόνι από την αφετηρία, το πάει στην θέση ενός αντίπαλου πιονιού εφόσον δεν προστατεύεται λόγο της θέσης του και στέλνει το αντίπαλο πιόνι στην αφετηρία του. Επομένως, η κλάση περιέχει:

  Την transformer μέθοδο **movePawn(Pawn pawn1, Pawn pawn2, Board board)** η οποία απλά αντιμεταθέτει τα δύο πιόνια.

  Για να ελεγχθεί κατά πόσο είναι σωστή και επιτρεπτή από τους κανόνες η κίνηση αυτή υπάρχει η μέθοδος **isValidMove(Pawn pawn1,Pawn pawn2,Board board):Boolean** η οποία επιστρέφει True αν και μόνο αν η κίνηση είναι σωστή, δηλαδή αν τα πιόνια είναι σε σωστές/επιτρεπτές θέσεις στο ταμπλό.

  Παράλληλα, εφόσον κληρονομεί abstract μέθοδο από την κλάση Card, υλοποίει και την **movePawn(Pawn pawn1, Pawn pawn2, Board board)**, όμως δεν την χρησιμοποιεί.

- **Η abstract κλάση NumberCard.** Αυτή η κλάση αναπαριστά την συλλογή των καρτών του παιχνιδιού που έχουν πάνω τους αριθμό. Περιέχει το **πεδίο value**, που αφόρα τον αριθμό της κάρτας. Υπάρχουν οι απαραίτητες setter/getter μέθοδοι και ο constructor.

  Περιέχει επίσης και την transformer **μέθοδο isValidMove(Pawn pawn, Board board):Boolean**, η οποία περιέχει έναν γενικό αλγόριθμο για την κίνηση των πιονιών, ο οποίος θα εξηγηθεί στο τέλος.

Η transformer μέθοδος **isValidMove(Pawn pawn, Board board):Boolean** υλοποιείται εδώ, με βάσει τον αλγόριθμο της movePawn.
Τέλος υπάρχει και η transformer μέθοδος **followTheRules(Pawn pawn, Board board, int value):Int** η οποία προσαρμόζει ανάλογα την νέα θέση του πιονιού με βάσει το αν βρίσκεται κοντά σε κάποιο safety zone.

Οι κλάσεις που ακολουθούν είναι υποκλάσεις της NumberCard.

- **Η κλάση SimpleNumberCard.** Η κλάση αυτή αναπαριστά της κάρτες με **αριθμό 3, 4, 8, 12**. Αυτές οι κάρτες επιτρέπουν στον παίκτη κίνηση προς τα μπροστά κατά 3, 4, 8 ή 12 θέσεις αντίστοιχα.

- **Η κλάση NumberOneCard**. Η κλάση αυτή αναπαριστά κάρτες με αριθμό «1». Οι κάρτες αυτές θα αρχικοποιηθούν στο ταμπλό με **value 1.**

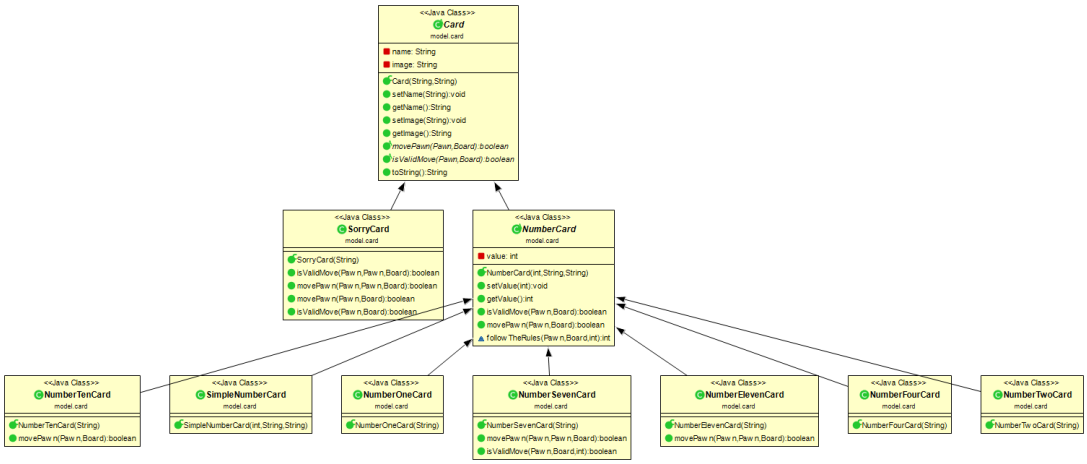
- **Η κλάση NumberTwoCard**. Αυτή η κλάση αναπαριστά την κάρτα με αριθμό 2.

- **Η κλάση NumberFourCard.** Αυτή η κλάση αναπαριστά κάρτες με αριθμό 4. Δεν έχει παραπάνω ιδιότητες, όμως η κίνηση κατά 4 θέσεις είναι προς τα πίσω.

- **Η κλάση NumberSevenCard.** Αυτή η κλάση αναπαριστά κάρτες με αριθμό 7. Με αυτήν την κάρτα ο παίκτης μπορεί είτε να κουνήσει κάποιο πιόνι του κατά 7 θέσεις μπροστά, είτε και τα δύο κατά θέσεις των οποίων το άθροισμα είναι 7. Για να αξιοποιηθούν οι κανόνες κληρονομικότητας, αφού ληφθούν dialog input τα values της κίνησης, το value της κάρτας αλλάζει για κάθε ξεχωριστή κίνηση του κάθε πιονιού, και καλείται εν συνεχεία μητρική movePawn και isValidMove.

- **Η κλάση NumberTenCard.** Αυτή η κλάση αναπαριστά κάρτες με αριθμό 10. Αυτές οι κάρτες υποχρεώνουν τον παίκτη είτε να πάει 10 θέσεις μπροστά, είτε 1 θέση πίσω. Για αυτό, ο τρόπος λειτουργίας της είναι ο ίδιος με αυτής της κάρτας 7.
- **Η κλάση NumberElevenCard.** Αυτή η κλάση αναπαριστά την κάρτα με αριθμό 11. Αυτή η κάρτα επιτρέπει στον παίκτη να πάει 11 θέσεις μπροστά, είτε να αλλάξει θέση με κάποιο πιόνι του αντιπάλου αν το επιτρέπουν οι σχετικοί κανόνες. Για αυτό, υπάρχει και η μέθοδος **movePawn(Pawn pawn1, Pawn pawn2, Board board):Boolean** που αντιμεταθέτει τα δύο πιόνια εφόσον είναι επιτρεπτό.

Παρατίθεται το UML των καρτών:



Ε. Το πακέτο Board.

Αυτό το πακέτο σχετίζεται με το ταμπλό του παιχνιδιού. **Ουσιαστικά είναι υπεύθυνο για την αρχικοποίηση, το μοίρασμα και το ανακάτεμα των καρτών, καθώς και για τα τετράγωνα και την ομαλή ροή του παιχνιδιού.**

Περιέχει την μοναδική κλάση Board η οποία περιέχει τον πίνακα **Squares(τα τετράγωνα του ταμπλό),** τις λίστες **Cards(οι κάρτες της στοίβας) και playedCards(οι κάρτες που έχουν ήδη ανοιχτεί).** Επίσης υπάρχει και το field **lastCardOpened** που αναπαριστά την κάρτα που παίζεται εκείνη την στιγμή. Η κλάση περιέχει και τις αντίστοιχες setter/getter μεθόδους και constructor.

Επίσης περιέχει μεθόδους (κυρίως transformers) για την ομαλή ροή του παιχνιδιού:

- **Μέθοδος initialize():void**. Αρχικοποιεί το ταμπλό.
- **Μέθοδος initializeSquares():void**. Υπεύθυνη για την αρχικοποίηση των τετραγώνων του ταμπλό.
- **Μέθοδος initializeCards():void.** Υπεύθυνη για την αρχικοποίηση των καρτών.
- **Μέθοδος takeCard(Player player):void.** Υπεύθυνη για να παίρνουν κάρτες οι παίκτες. Παίρνει την πρώτη κάρτα της στοίβας, και διαγράφει την κάρτα. Επίσης όταν τελειώσουν οι κάρτες της ξαναβάζει στην στοίβα και τις ανακατεύει.
- **Μέθοδος cardsLeft():int**. Επιστρέφει πόσες κάρτες υπάρχουν στην στοίβα.
- **Μέθοδος slideEffect():void.** Ελέγχει αν το πιόνι που βρίσκεται πάνω στην αρχή μίας τσουλήθρας έχει διαφορετικό χρώμα από αυτήν και το μεταφέρει στο τέλος της τσουλήθρας, μεταφέροντας όσα πιόνια βρίσκονται μέσα σε αυτή στην αφετηρία.
- **Μέθοδος areCardsLeft():Boolean.** Πληροφορεί για το αν υπάρχουν άλλες διαθέσιμες κάρτες στην στοίβα.
- **Μέθοδος shuffleCards():void.** Ανακατεύει τις κάρτες.
- **Μέθοδος reInitializeCards():void.** Ξανακάνει initialize τις κάρτες σε περίπτωση που η στοίβα αδειάσει.
- **Μέθοδος removeCard():void.** Μέτα το άνοιγμα μίας κάρτας από έναν παίκτη, την αφαιρεί από την στοίβα των διαθέσιμων καρτών και την πάει στην στοίβα των καρτών που έχουν παιχτεί.
- **Μέθοδος hasPlayerWon(Player player):Boolean.** Ελέγχει αν κάποιος παίκτης έχει κερδίσει.
- **Μέθοδος decideTurn(Player player1,Player player2):void.** Υπεύθυνη για να καθορίσει την σειρά παιξίματος.
- **Μέθοδος isOccupied(int pos):Boolean.** Επιστρέφει αν το τετράγωνο με index pos είναι πιασμένο από κάποιο πιόνι.
- **Μέθοδος getOccupationPawn(int pos):Pawn.** Επιστρέφει το πιόνι που υπάρχει πάνω στο τετράγωνο με index pos.

3. Η Σχεδίαση και οι Κλάσεις του Πακέτου Controller

Στο πακέτο αυτό περιέχεται η κλάση Controller, η οποία είναι **ο εγκέφαλος επικοινωνίας των γραφικών με τον πυρήνα του παιχνιδιού.**

Περιέχει τα πεδία **board(Board),** που είναι το ταμπλό του παιχνιδιού**,** ενώ επίσης έχει και το πεδίο **view(View),** για τα γραφικά του παιχνιδιού. Παράλληλα, υπάρχουν **2 players(Player)** και **4 pawns(Pawn).** Για την ομαλή και οργανωμένη ροή του παιχνιδιού υπάρχουν τα πεδία **currentPlayer(Player), currentCard(Card) και roundEnded(Boolean).** Υπάρχουν οι αντίστοιχοι setters.

Ο controller συντονίζει το παιχνίδι με χρήση κάποιον βασικών μεθόδων:

- **Μέθοδος initializeBoard():void.** Αρχικοποιεί το ταμπλό.
- **Μέθοδος initializeGraphics():void.** Δημιουργεί τα γραφικά.
- **Μέθοδος initializePlayers():void.** Αρχικοποιεί τους παίκτες.
- **Μέθοδος initializePawns():void.** Αρχικοποιεί τα πιόνια.
- **Μέθοδος beginNewGame():void.** Ξεκινά νέο παιχνίδι.
- **Μέθοδος continueSavedGame():void.** Συνεχίζει ένα αποθηκευμένο παιχνίδι.
- **Μέθοδος createMenuListeners():void.** Προσθέτει λειτουργικότητα στα items του μενού.
- **Μέθοδος createListeners():void.** Προσθέτει listeners στα JButtons του view με σκοπό να προστεθεί σε αυτά λειτουργικότητα.
- **Μέθοδος checkAndMovePawn(Pawn pawn):void.** Ελέγχει για την κίνηση του πιονιού και το μετακινεί στην επιθυμητή θέση.
- **Μέθοδος checkAndMovePawns(Pawn pawn1,Pawn pawn2):void.** Υπεύθυνη για την κίνηση των πιονιών για τις special cards.
- **Μέθοδος decideTurn():void.** Αποφασίζει για την σειρά που θα παίζουν οι παίχτες.
- **Μέθοδος checkWinner(Player player):void.** Ελέγχει αν κάποιος νίκησε.
- **Μέθοδος infoSaver(int [] info):void.** Αποθηκεύει το progress ενός παιχνιδιού με σκοπό την δημιουργία αρχείου txt.
- **Μέθοδος loadAndApplyInfo():void.** Εφαρμόζει το progress ενός αποθηκευμένου παιχνιδιού.

Υπάρχει επίσης η **inner class ButtonListener** που κάνει implement το mouseListener, με σκοπό να υλοποιηθούν τα events των JButtons των γραφικών. Περιέχει την βοηθητική μεταβλητή **moveMade(Boolean).**

- Override της μεθόδου **mouseClicked**. Για κάθε button, η ενέργεια που υλοποιείται είναι ανάλογη.
- **Μέθοδος isValidFold():Boolean.** Ελέγχει αν επιτρέπεται να γίνει fold.
- **Μέθοδος foldPressed():void.** Υπεύθυνη για τις λειτουργίες που θα γίνουν όταν πατηθεί το fold.
- **Μέθοδος pressedPawn(int index):void.** Υπεύθυνη για τις λειτουργίες που θα γίνουν όταν πατηθεί ένα πιόνι.

- **Μέθοδος sorryCardAlg (int index):void.** Υπεύθυνη για τις λειτουργίες που θα γίνουν όταν προκύψει sorryCard.
- **Μέθοδος elevenCardAlg(int index):void.** Υπεύθυνη για τις λειτουργίες που θα γίνουν όταν προκύψει eleven card.
- **Μέθοδος sevenCardAlg(int index):void.** Υπεύθυνη για τις λειτουργίες που θα γίνουν όταν προκύψει seven card.

4. Η Σχεδίαση και οι Κλάσεις του Πακέτου View

Το πακέτο view είναι υπεύθυνο για τα γραφικά της εφαρμογής και την ενημέρωση αυτών κατά την διάρκεια του παιχνιδιού.

Περιέχει την βασική **κλάση view**, που περιέχει τα καίρια fields: **positions(Jlabels), pawns(JButtons), stackOfCards(JButton), openedCard(JButton), fold(JButton), mainboard(JPanel), backPanel(JPanel), infoBox(JTextArea), homePositions(JLabels), startingPositions(JLabels), gameMenu(GameMenu)** .Υπάρχουν και όλες οι setter/getter μέθοδοι. Περιέχει επίσης μεθόδους που αρχικοποιούν και ενημερώνουν τα πεδία.

- **Μέθοδος createGraphics():void**. Δημιουργεί τα γραφικά.
- **Μέθοδος setWindowParameters():void**. Δημιουργεί το βασικό frame.
- **Μέθοδος createGameMenu():void**. Δημιουργεί την μπάρα με το menu.
- **Μέθοδος createSlides():void.** Δημιουργεί τις τσουλήθρες.
- **Μέθοδος createStartingPositions():void**. Δημιουργεί της αρχικές θέσεις.
- **Μέθοδος createHomePositions()**:**void**. Δημιουργεί τις τελικές θέσεις.
- **Μέθοδος createBackPanel():void.** Δημιουργεί το background πάνω στο οποίο θα προστεθούν όλα τα υπόλοιπα γραφικά.
- **Μέθοδος createMainBoard():void.** Δημιουργεί το ταμπλό.
- **Μέθοδος createPawns():void.** Δημιουργεί τα πιόνια τον παικτών.
- **Μέθοδος createStackOfCards():void.** Δημιουργεί την στοίβα των καρτών.
- **Μέθοδος createLastCardPlayed():void.** Δημιουργεί την στοίβα των καρτών που έχουν παιχτεί.
- **Μέθοδος initializePositions():void.** Αρχικοποιεί τα τετράγωνα και τα προσθέτει στο ταμπλό.
- **Μέθοδος createInfoBox(String message):void**. Δημιουργεί το παράθυρο πληροφοριών με ένα αρχικό μήνυμα.
- **Μέθοδος createFoldButton():void.** Δημιουργεί το fold button.
- **Μέθοδος updateInfoBox(String newMessage):void.** Ενημερώνει με νέο μήνυμα το παράθυρο πληροφοριών.

- **Μέθοδος clearInfoBox():void**. Αδειάζει τον πίνακα πληροφοριών των γραφικών.
- **Μέθοδος updatePawn(Pawn [] pawn):void.** Μετακινεί τα πιόνια.
- **Μέθοδος updateLastCardOpened(Card c):void.** Ενημερώνει την κάρτα που μόλις ανοίχτηκε.
- **Μέθοδος showWinningMessage(String winningMessage).** Δείχνει κάποιο μήνυμα νίκης όταν κάποιος παίκτης κερδίσει.
- **Μέθοδος refresh():void**. Ανανεώνει το frame.

Πέραν της κλάσης view υπάρχει και η κλάση GameMenu που δημιουργεί τη μπάρα των επιλογών. Περιέχει τα πεδία saveMenu και about (JMenu), καθώς και τα αντίστοιχα στοιχεία τους newGame, saveGame, continueGame και exitGame (JMenuItems) με τις αντίστοιχες setter/getter μεθόδους. Περιέχει επίσης τους transformers:

- **Μέθοδος createItems():void**. Δημιουργεί τα items του μενού.
- **Μέθοδος createSaveMenu():void.** Υλοποιεί το save μενού.
- **Μέθοδος createAboutMenu():void.** Υλοποιεί το about μενού.

Ο constructor της κλάσης αυτής προσθέτει τα στοιχεία στην μπάρα.

5. Η Σχεδίαση και οι Κλάσεις του Πακέτου Utilities

Σε αυτό το πακέτο, περιέχονται βοηθητικές κλάσεις και μέθοδοι με σκοπό την ευκολότερη υλοποίηση κάποιον λειτουργιών του προγράμματος.

Υπάρχει **η κλάση currentDate** που περιέχει μέθοδο που επιστρέφει σε μορφή string την ημερομηνία, με σκοπό την αποθήκευση σε όνομα αρχείου.

Επίσης, **η κλάση FileHandler** περιέχει static μεθόδους για την ανάγνωση και το γράψιμο σε αρχείο.

Επιπρόσθετα, **η κλάση PopMessage** περιέχει μέθοδο που κάνει την εμφάνιση των dialogs ευκολότερη.

Τέλος, **η κλάση RandomGenerator** περιέχει μία static μέθοδο που επιστρέφει τυχαίες τιμές.

6. Η Σχεδίαση και οι Κλάσεις του Πακέτου Test

Αυτό το πακέτο περιέχει **την κλάση BasicPlayTesting**, η οποία αξιοποιεί της δυνατότητες του JUnit5, για να ελέγξει κάποιες βασικές κινήσεις της movePawn χρησιμοποιώντας assertions.

Συγκεκριμένα, ελέγχονται 5 βασικές περιπτώσεις, δηλαδή:

- **Απαγόρευση** εκκίνησης από start χωρίς κάρτα 1 2, ή “sorry”.
- **Εκκίνηση** από αφετηρία με σωστή κάρτα.
- **Απαγόρευση** μετακίνησης πιονιού σε τετράγωνο που υπάρχει πιόνι ίδιου χρώματος.
- **Επίθεση** αντίπαλου πιονιού μετακινώντας το άλλο στην αφετηριακή του θέση.
- **Απενεργοποίηση** πιονιού που φτάνει στο home.

Τα περιεχόμενα JUnit tests κάνουν όλα pass.

7. Η Σχεδίαση και οι Κλάσεις του Πακέτου Application

Αυτό το πακέτο περιέχει **την κλάση Game**, την οποία την τρέχουμε για να παίξουμε το παιχνίδι.

Επίσης από αυτήν την κλάση μπορούμε να δημιουργήσουμε το runnable jar file.

8. Η Σχεδίαση και οι Κλάσεις του Πακέτου Exception

Κατά την δημιουργία του παιχνιδιού ορίστηκαν δύο ενδεικτικά exceptions.

Συγκεκριμένα τα InvalidMoveException και NotValidDisableException.

Τελικά, το error handling αποδείχθηκε ευκολότερο να γίνει μέσω κάποιον συνθηκών και dialogs, επομένως το πακέτο αυτό εν τέλει δεν αξιοποιήθηκε/χρησιμοποιήθηκε.

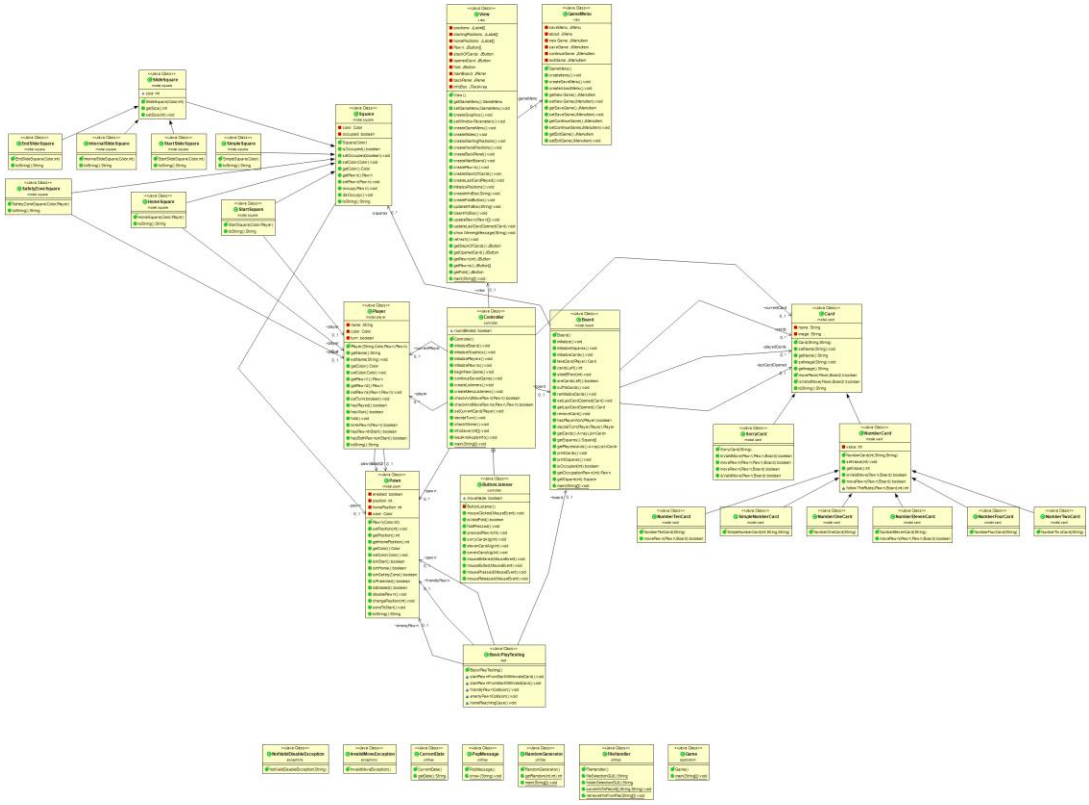
9. Η Αλληλεπίδραση μεταξύ των κλάσεων – Διαγράμματα UML

Παρακάτω παρουσιάζεται το διάγραμμα UML των κλάσεων τον πακέτων που δημιουργήθηκαν. Παρατηρούμε πως η αλληλεπίδραση του πυρήνα με τα γραφικά γίνεται μέσω του controller, και από αυτό κατανοούμε τον «συντονιστικό» ρόλο του στην υλοποίηση του παιχνιδιού.

Οι κλάσεις κάθε πακέτου, άμεσα ή έμμεσα, επικοινωνούν με τον Controller. Παράλληλα, βλέπουμε πως και η κλάση Board έχει συντονιστικό χαρακτήρα μέσα στο πακέτο model, αφού είναι υπεύθυνη για την ροή και την τήρηση των κανόνων του παιχνιδιού.

Μέσω την κλάσης αυτής, επικοινωνούν οι κάρτες με τον controller.

Τέλος μέσα από το διάγραμμα οπτικοποιούνται και οι σχέσεις κληρονομικότητας στα πακέτα square και card.



10. Λειτουργικότητα/Αλγόριθμοι/Επιπρόσθετες Λειτουργίες

- Λειτουργικότητα: Όλα τα ζητούμενα ερωτήματα έχουν υλοποιηθεί, και το παιχνίδι δουλεύει κανονικά.
- Αλγόριθμος της movePawn:
    - Έλεγχος αν η κίνηση είναι εφικτή μέσω της isValidMove.
    - Αν το πιόνι δεν είναι στην αφετηρία, απελευθερώνουμε την θέση του στο ταμπλό (disOccupy).
    - Αν είναι στην αφετηρία, ελέγχουμε το χρώμα του και την θέση που πρόκειται να πάει. Αν είναι κατειλημμένη στέλνουμε το πιόνι που τη καταλαμβάνει στην αφετηρία.
    - Αν δεν είναι στην αφετηρία, τότε το μεταφέρουμε στην θέση που υποδεικνύει η ανοιγμένη κάρτα.
    - Αν μετά την κίνηση είναι στο home, το κάνουμε disable.
    - Αν δεν είναι στο home, κάνουμε occupy την θέση που βρίσκεται.

- Πρόσθετα:
    - Υλοποίηση λειτουργίας “new game”.
    - Υλοποίηση λειτουργίας “exit game”
    - Υλοποίηση δυνατότητας αποθήκευσης και συνέχισης παιχνιδιού (5%).

11. Συμπεράσματα

Σε αυτήν την προγραμματική εργασία το MVC ακολουθείται όσο περισσότερο γίνεται.

Το παιχνίδι δοκιμάστηκε και λειτουργεί κανονικά, ενώ στον παραδοτέο φάκελο υπάρχει και ενδεικτικό README που εξηγεί το πώς τρέχει και παίζεται το παιχνίδι ώστε να αποφευχθούν λάθη και δημιουργία προβλημάτων από λάθος συμπεριφορά των παικτών.