



Πανεπιστήμιο Κρήτης –Τμήμα Επιστήμης Υπολογιστών

ΗΥ252– Αντικειμενοστρεφής Προγραμματισμός

Διδάσκων: Ι. Τζιτζικας

Χειμερινό Εξάμηνο 2018-2019

## Project – Ticket to Ride(Phase 2)

*Νίκος Γουνάκης*

*ΑΜ : 3932*

*4/1/2018*

### *Εισαγωγή*

Η εργασία χωρίζεται σε 3 main packages. Με το μοντέλο MVC έχουμε τα τρία κύρια πακέτα Model, View , Controller τα οποία είναι υπεύθυνα για τις λειτουργίες του προγράμματος. Επίσης το Model χωρίζεται σε Card, Deck και ένα ακόμα που περιέχει exception. Στην συνέχεια της αναφοράς θα αναλύσουμε το κάθε package.

## Περιεχόμενα

1. Εισαγωγή.....	Error! Bookmark not defined.
2. Η Σχεδίαση και οι Κλάσεις του Πακέτου <b>Model</b> .....	Error! Bookmark not defined.
3. Η Σχεδίαση και οι Κλάσεις του Πακέτου <b>Controller</b> .....	Error! Bookmark not defined.
4. <b>View</b> .....	215
5. Η Αλληλεπίδραση μεταξύ των κλάσεων – Διαγράμματα UML.....	24
6. ....	Error! Bookmark not defined.
7. Συμπεράσματα .....	Error! Bookmark not defined.

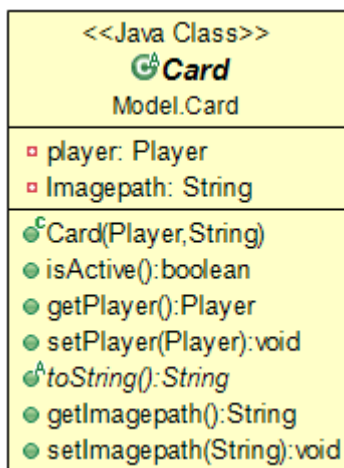
# 1. Η Σχεδίαση και οι Κλάσεις του Πακέτου Model

Εδώ περιέχονται οι κλάσεις του πακέτου Model :

Card , BigCitiesCard , CardColor , DestinationCard , TrainCard , Player, PointsCard ,Deck , OnTheTrack , RailYard , InvalidMoveException

## 1.1 Abstract Class Card

Η αφαιρεμένη κλάση αυτή υλοποιεί μια οποιαδήποτε κάρτα του παιχνιδιού, όπως θα δούμε παρακάτω οι υπόλοιπες κλάσεις καρτών είναι υποκλάσεις αυτής.



Έχει δύο γνωρίσματα :

`Player player;` // ο παίκτης στον οποίο ανήνκει η κάρτα

`String Imagepath;` // το μονοπάτι που οδηγεί στην φωτογραφία της κάρτας

Επίσης οι μέθοδοι :

`Card(Player,String)` // ο Constructor

`IsActive()` : boolean // για να δούμε αν είναι ενεργή

`getPlayer()` : Player // επιστρέφει τον παίχτη στον οποίο ανήκει

`SetPlayer(Player)` : void // θέτει το γνώρισμα του παίχτη

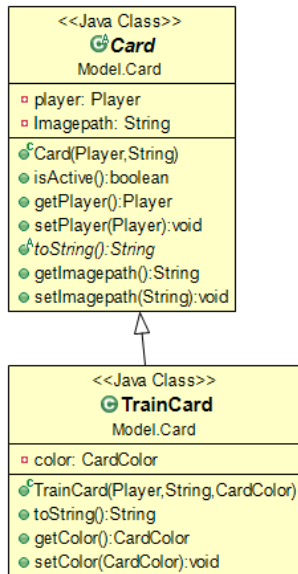
`ToString()` : String // επιστρέφει μια συμβολοσειρά που περιγράφει την κάρτα

`setImagepath()` : String // επιστρέφει μια συμβολοσειρά με την διαδρομή που οδηγεί στην εικόνα της κάρτας

setImagepath(String) : void // θέτει το γνώρισμα της συμβολοσειράς της διαδρομής προς την εικόνα

### 1.1.2 Class TrainCard extends Card

Ως ειδική περίπτωση κάρτας , είναι υποκλάση της Card. Η κλάση αυτή περιγράφει μια κάρτα τρένων όπου μπορεί να έχει 1 από τα 8 χρώματα.



**Έχει ένα γνώρισμα :**

`CardColor color` // το χρώμα της κάρτας τρένου

**Οι μέθοδοι :**

`TrainCard(Player, String, CardColor)` // ο Constructor

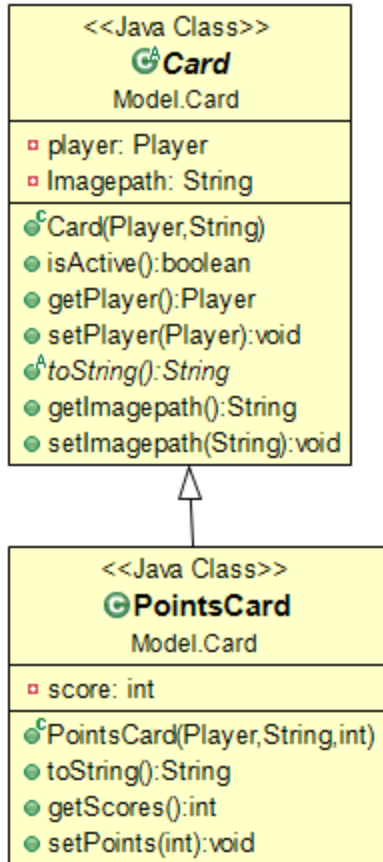
`ToString() : String` // η συμβολοσειρά που περιγραφει την κάρτα

`getColor() : CardColor` // επιστρέφει το χρώμα της κάρτας

`setColor() : void` // θέτει το χρώμα της κάρτας

### 1.1.3 Class PointsCard extends Card

Ως ειδική περίπτωση κάρτας , είναι υποκλάση της Card και είναι υπερκλάση των `DestinationCard` , `BigCitiesCard` .



Έχει ένα γνώρισμα :

`int score` // το ποσό των πόντων που αποφέρει στον παίκτη

Μεθόδοι :

`PointsCard(Player, String, int)` // ο Constructor

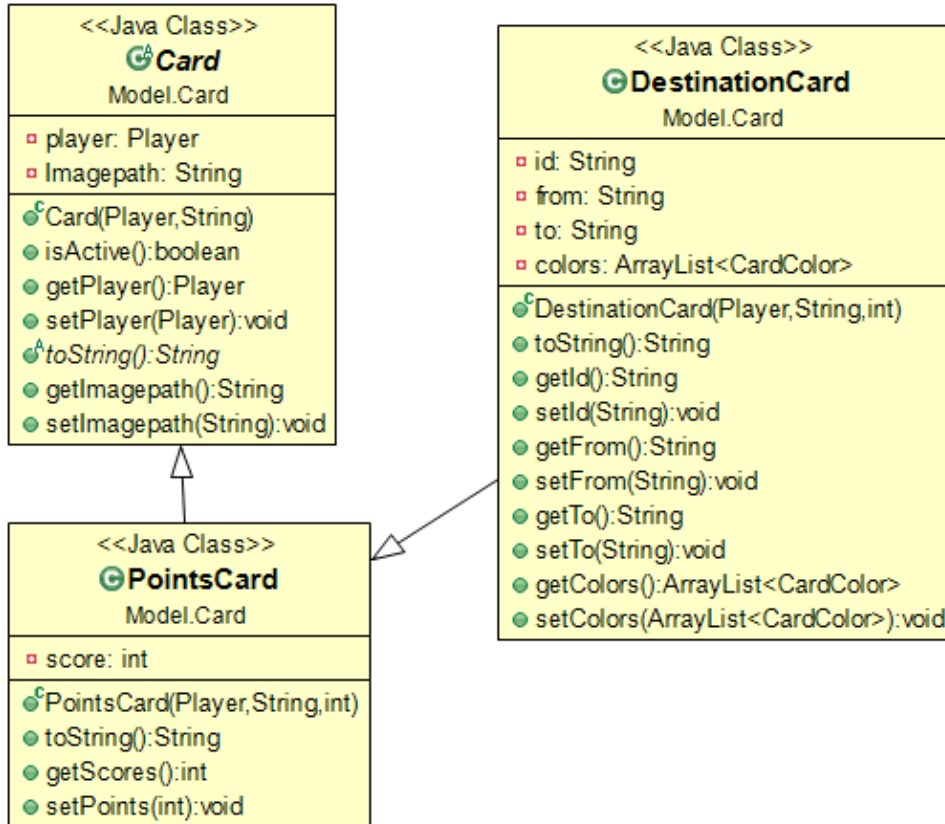
`ToString() : String` // περιγραφή της κάρτας

`getScores() : int` // επιστρέφει τους πόντους της κάρτας

`setScore(int) : void` // σετάρει τους πόντους

### 1.1.4 Class DestinationCard extends PointsCard

Ως ειδική περίπτωση κάρτας με πόντους είναι υποκλάση της `PointsCard`. Η κλάση αυτή περιγράφει μια κάρτα προορισμού από τις 46 του παιχνιδιού. Κάθε κάρτα προορισμού έχει μια πόλη από την οποία ξεκινά το τρένο και μια άλλη στην οποία καταφθάνει. Επίσης έχει τα χρώματα τα οποία απαιτούνται για να εξαχρασθεί και ένα id που την ξεχωρίζει από τις άλλες.



### Γνωρίσματα :

Int id// το id της κάρτας , η κάθε μια είναι μοναδική (changed from string id λόγω μετατροπής από String σε int από άλλη συνάρτηση )

String from // η πόλη από την οποία ξεκινάει το τρένο

String to // η πόλη προς την οποία καταφθάνει

ArrayList<CardColor> // μια λίστα με τα χρώματα που απαιτούνται για την εξαργύρωση της κάρτας

### Μεθόδους :

DestinationCard(Player ,String ,int) // ο Constructor

ToString() : String // η περιγραφή της κάρτας

GetId() : int // επιστρέφει το id

SetId(int) : void // σετάρει το id

GetTo() : String // επιστρέφει την πόλη προορισμού

SetTo(String) : void // σετάρει την πόλη προορισμού

GetFrom() : String // επιστρέφει την πόλη εκίνησης

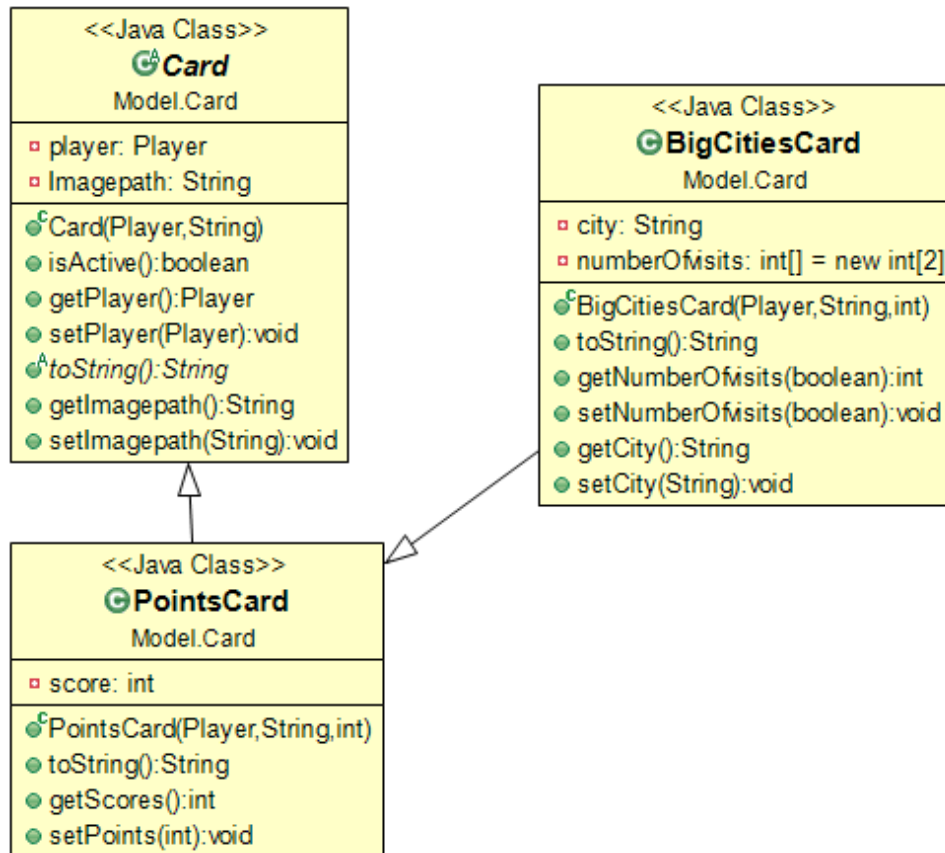
SetFrom(String) : void // σετάρει την πόλη εκίνησης

GetColors() : ArrayList<CardColors> // επιστρέφει την λίστα με τα χρώματα

SetColors(ArrayList<CardColors>) : void // σετάρει την λίστα με τα χρώματα

### 1.1.5 Class BigCitiesCard extends PointsCard

Ως ειδική περίπτωση κάρτας με πόντους είναι υποκλάση της PointsCard. Η κλάση αυτή περιγράφει μια κάρτα μεγάλης πόλης bonus.



#### Γνωρίσματα :

String city // η πόλη που αναπαριστά η κάρτα

Int[2] numberOfVisits // ένας πίνακας όπου το πρώτο κελί αναπαριστά τον αριθμό των επισκέψεων του παίχτη1 και το δεύτερο κελί του παίχτη2 αντίστοιχα

#### Μεθόδοι :

BigCitiesCard(Player, String, int) // ο Constructor

ToString() : String // η περιγραφή της κάρτας

GetNumberOfVisits(boolean) : int // επιστρέφει τον αριθμό των επισκέψεων του κάθε παίχτη

SetNumberOfVisits(boolean) : void // θέτει τον αριθμό των επισκέψεων κάθε παίχτη

GetCity() : String // επιστρέφει την πόλη της κάρτας

SetCity(String) : void // θέτει την πόλη του


## 1.1.6 Enum CardColor

Μια απαρίθμηση από χρώματα αντι να χρησιμοποιήσουμε String.

Περιλαμβάνει : **red, black, blue, green, purple, white, yellow, orange, locomotive,**

## 1.2 Class Player

Η κλάση αυτή αναπαριστά έναν παίχτη του παιχνιδιού . Κάθε παίχτης έχει στην κατοχή του κάρτες τρένων , κάρτες προορισμού και κάρτες μεγάλων πόλεων, εξισου και τα γνωρίσματα TrainCardsOnHand , DestinationCardsOnHand , BigCitiesCardsOnHand. Επίσης έχει μια περιοχή Railyard , OnTheTrack και ένα score. Τέλος ένα boolean turn που καθορίζει αν είναι ο γύρος του παίχτη.

<<Java Class>>	
 <b>Player</b> Model.Card	
<ul style="list-style-type: none"> <li>▣ TrainCardsOnHand: ArrayList&lt;TrainCard&gt;</li> <li>▣ DestinationCardsOnHand: ArrayList&lt;DestinationCard&gt;</li> <li>▣ MyBigCitiesCards: ArrayList&lt;BigCitiesCard&gt;</li> <li>▣ MyDestinationCards: ArrayList&lt;DestinationCard&gt;</li> <li>▣ railyard: RailYard</li> <li>▣ onthetrack: OnTheTrack</li> <li>▣ score: int</li> <li>▣ turn: boolean</li> </ul>	
<ul style="list-style-type: none"> <li>⚙️ Player()</li> <li>● getTrainCardsOnHand(): ArrayList&lt;TrainCard&gt;</li> <li>● setTrainCardsOnHand(ArrayList&lt;TrainCard&gt;): void</li> <li>● getDestinationCardsOnHand(): ArrayList&lt;DestinationCard&gt;</li> <li>● setDestinationCardsOnHand(ArrayList&lt;DestinationCard&gt;): void</li> <li>● getMyDestinationCards(): ArrayList&lt;DestinationCard&gt;</li> <li>● setMyDestinationCardsOnHand(ArrayList&lt;DestinationCard&gt;): void</li> <li>● getRailyard(): RailYard</li> <li>● setRailyard(RailYard): void</li> <li>● getOnthetrack(): OnTheTrack</li> <li>● setOnthetrack(OnTheTrack): void</li> <li>● getScore(): int</li> <li>● getMyBigCitiesCards(): ArrayList&lt;BigCitiesCard&gt;</li> <li>● setMyBigCitiesCardsOnHand(ArrayList&lt;BigCitiesCard&gt;): void</li> <li>● setTurn(boolean): void</li> <li>● getTurn(): boolean</li> <li>● updateScore(): void</li> </ul>	



**Γνωρίσματα :**

`ArrayList<TrainCard> TrainCardsOnHand` // οι κάρτες τρένων στην κατοχή του παίχτη

`ArrayList<DestinationCard> DestinationCardsOnHands` // οι κάρτες προορισμού στην κατοχή του παίχτη

`ArrayList<BigCitiesCard> BigCitiesCardsOnHands` // οι κάρτες μεγάλων πόλεων στην κατοχή του παίχτη

`Arraylist<DestinationCard> MyDestinationCards` // οι εξαργυρωμένες κάρτες προορισμού του παίκτη (new)

`RailYard railyard` // η περιοχή RailYard του παίχτη

`OnTheTrack onthetrack` // η περιοχή OnTheTrack του παίχτη

`Int score` // το score του παίχτη

`Boolean Turn` // καθορίζει αν είναι ο γύρος του παίχτη

**Μεθόδους :**













Όλοι οι μέθοδοι εκτός από μία είναι setter , getter και Constructor επειδή είναι πολλές δεν χρειάζονται επεξήγηση.

`UpdateScore( ) : void` // ανανεώνει και υπολογίζει το score του παίχτη

(new) : πλέον η `UpdateScore` δεν παίρνει κανένα ορίσμα , απλά διασχίζει τις λίστες καρτών προορισμού και bonus και υπολογίζει το score

## 1.3 Class Deck

Η κλάση αυτή αναπαριστά το ταμπλό στο κέντρο του παιχνιδιού, έχει δυο στοιβές μια από `TrainCards` και μια από `DestinationCards`. Επίσης δύο στοιβές, μία που αναπαριστά τις 5 κάρτες τρένων και μία που αναπαριστά τις κάρτες μεγάλων πόλεων.

<<Java Class>>  <b>Deck</b> Model.Deck
■ trainCards: Stack<TrainCard> ■ destinationCards: Stack<DestinationCard> ■ onDeckTrainCards: ArrayList<TrainCard> ■ onDeckBigCitiesCards: ArrayList<BigCitiesCard>
 Deck()  getNewTrainCard(Player):void  getNewDestinationCard(DestinationCard,Player):void  getTrainCards():Stack<TrainCard>  setTrainCards(Stack<TrainCard>):void  getDestinationCards():Stack<DestinationCard>  setDestinationCards(Stack<DestinationCard>):void  getOnDeckTrainCards():ArrayList<TrainCard>  setOnDeckTrainCards(ArrayList<TrainCard>):void  getOnDeckBigCitiesCards():ArrayList<BigCitiesCard>  setOnDeckBigCitiesCards(ArrayList<BigCitiesCard>):void

### Γνωρίσματα :

Stack<TrainCard> trainCards // στοίβα όπου οι παίκτες τραβάνε κάρτες τρένων

Stack<DestinationCard> destinationCards // στοίβα όπου οι παίκτες τραβάνε κάρτες προορισμού

ArrayList<TrainCard> onDeckTrainCards // οι 5 κάρτες τρένων όπου οι παίκτες μπορούν να επιλέξουν από το ταμπλό

ArrayList<BigCitiesCard> onDeckBigCitiesCards // οι κάρτες μεγάλων πόλεων που είναι στο ταμπλό

### Μεθόδους :

Κυρίως getter, setter και ένας Constructor.

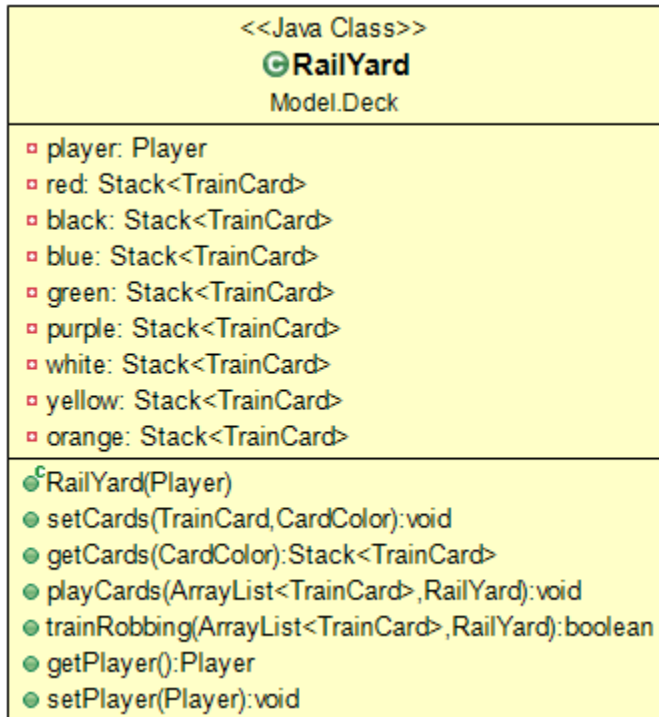
Επίσης :

getNewTrainCard(Player p) : void ,throws InvalidMoveException // δίνει στον παίκτη p μια κάρτα τρένων , αν η στοίβα είναι άδεια τότε πετάει InvalidMoveException

getNewDestinationCard(DestinationCard card,Player p) void ,throws InvalidMoveException // δίνει την κάρτα προορισμού card στον παίκτη p , πετάει InvalidMoveException αν η στοίβα είναι άδεια

## 1.5 Class RailYard

Η κλάση αυτή αναπαριστά την περιοχή RailYard του παίχτη



### Γνωρίσματα :

Player player // ο παίχτης στον οποίο ανήκει το RailYard

Και μια στοιβα από κάρτες τραίνων για κάθε χρώμα εκτος μπαλαντέρ :

Stack<TrainCard> red

Stack<TrainCard> black

Stack<TrainCard> blue

Stack<TrainCard> green

Stack<TrainCard> purple

Stack<TrainCard> white

Stack<TrainCard> yellow

Stack<TrainCard> orange

**Μεθόδοι :**

RailYard(Player) // Constructor

SetCards(TrainCard,CardColor) : void // προσθέτει κάρτες στην στοίβα με το χρώμα που παίρνει ως είσοδο

GetCards(CardColor) : Stack<TrainCard> // επιστρέφει την ανάλογη στοίβα ανάλογα το χρώμα

SetPlayer(Player) : void // θέτει τον παίχτη που ανήκει η περιοχή

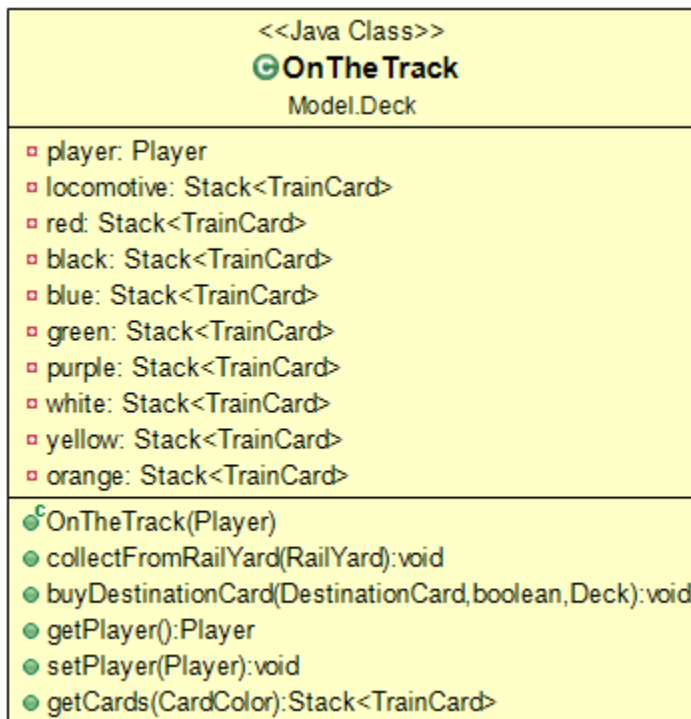
GetPlayer() : Player // επιστρέφει τον παίχτη στον οποίο ανήκει η περιοχή

playCards(ArrayList<TrainCard>,RailYard) : void // ελέγχει αν μπορεί να μεταφερθούν οι κάρτες στο RailYard και αν γίνεται τις μετακινεί

trainRobbing(ArrayList<TrainCard> ,RailYard ) : boolean // ελέγχει αν γίνεται train robbing και επιστρέφει true αν γίνεται αλλιώς false.

## 1.6 Class OnTheTrack

Η κλάση αυτή αναπαριστά την περιοχή OnTheTrack δεν είναι πλέον υποκλάση της railyard για να φανεί η διαφορετικότητα των δυό αντικειμένων αν και μοιάζουν στην δομή τους.



**Γνωρίσματα :**

Stack<TrainCard> // μια για κάθε χρώμα συμπεριλαμβανομένου του μπαλαντέρ

Player player // ο παίκτης στον οποίο ανήκει η περιοχή

**Μεθόδους :**

OnTheTrack(Player) // Constructor

CollectFromRailYard(RailYard) : void // παίρνει μία κάρτα από κάθε χρώμα από την RailYard του παίχτη

BuyDestinationCard(DestinationCard,boolean,Deck) : void // αν ο παίχτης έχει τις κάρτες που απαιτούνται τότε εξαγοράζει την κάρτα προορισμού που δέχεται ως είσοδο , ανάλογα με το ποιος από τους δύο παίχτες χρησιμοποιεί την συναρτηση παίρνει τιμή η boolean μεταβλητή για να αυξήσει το αντίστοιχο κελί του πίνακα που κρατά τις επισκέψεις στις μεγάλες πόλεις. Αν οι επισκέψεις είναι 3 τότε ο αντίστοιχος παίκτης εξαργυρώνει την αντίστοιχη κάρτα bonus. Το Deck είναι η περιοχή στο κέντρο του ταμπλό.

Getter - setter για τον παίκτη

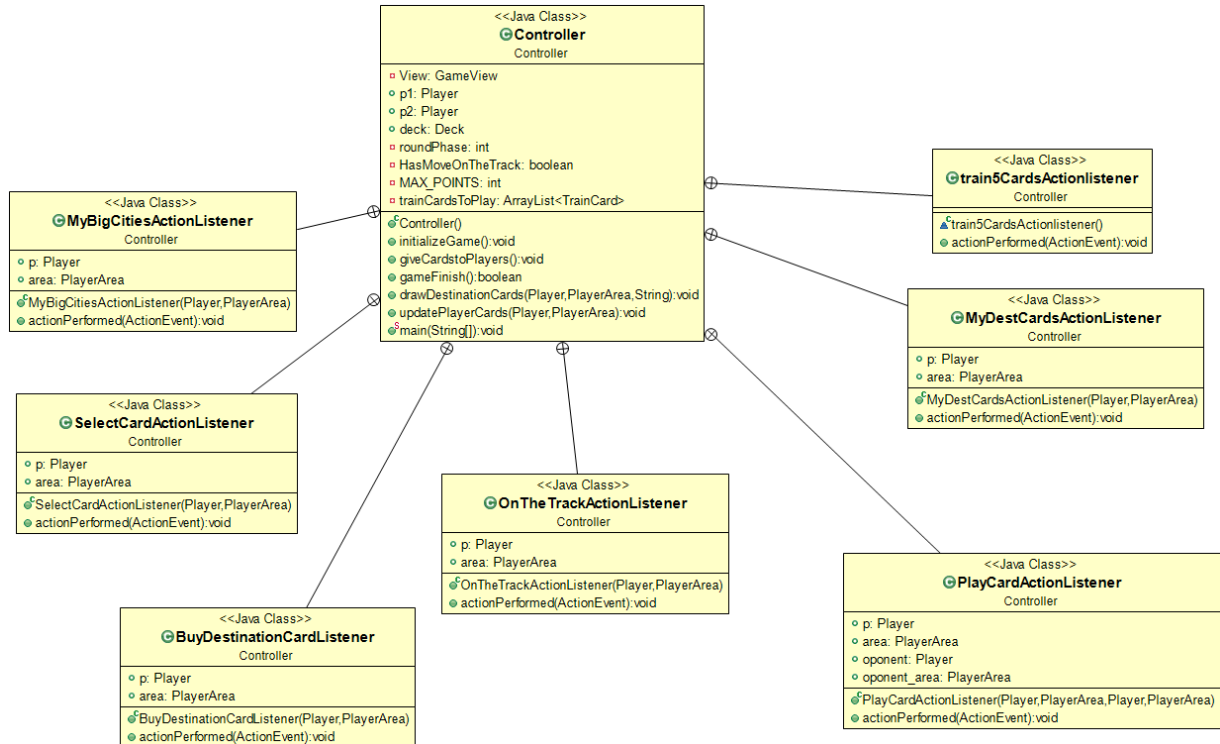
GetCards(CardColor) Stack<TrainCard> // επιστρέφει την στοίβα με το αντίστοιχο χρώμα

## 1.7 Class InvalidMoveException

Μια κλάση η οποία αναπαριστά ένα exception όταν γίνεται μη επιτρεπτή κίνηση από τον παίχτη

## 2. Η Σχεδίαση και οι Κλάσεις του Πακέτου Controller

Η κλάση αυτή είναι ο εγκέφαλος του παιχνιδιού. Γενικά χειρίζεται τις κλάσεις των πακέτων Model και View, αρχικοποιεί το παιχνίδι και είναι υπεύθυνο για την σωστή λειτουργία του παιχνιδιού.



### Γνωρίσματα :

GameView View // στιγμότυπο της κλάσης που υλοποιεί τα γραφικά

Player p1 // ο παίχτης 1

Player p2 // ο παίχτης 2

Deck deck // το ταμπλό στη μέση του παιχνιδιού

Int RoundPhase // 0 δεν έχει κάνει κίνηση , 1 έχει τραβήξει μια κάρτα , 2 έχει τραβήξει δύο κάρτες

Boolean HasMoveOnTheTrack // true αν έχει πατήσει το κουμπί μετακίνησης καρτών στο OnTheTrack , false αλλιώς

Int MAX\_POINTS // οι πόντοι που απαιτούνται για να νικήσει ένας παίκτης

ArrayList<TrainCard> trainCardsToPlay // εκεί τοποθετούνται οι κάρτες που είναι να ριχτούν στην Railway του παίχτη όταν “πατάει” κάρτες

### Μεθόδους :

Controller(GameView,Player,Player,Deck) // Constructor

InitializeGame() : void // αρχικοποιεί το παιχνίδι

GiveCardstoPlayers() : void // μοιράζει κάρτες στους παίκτες

GameFinish() : boolean // επιστρέφει true αν το παιχνίδι έχει τελειώσει και το αντίστοιχο μήνυμα, αλλιώς false

drawDestinationCards(Player ,PlayerArea ,String) : void // ανοίγει ένα παράθυρο ώστε ο παίκτης να διαλέξει κάρτες προορισμού , από 0 έως 4, ως ορίσματα παίρνει τον παίκτη , την περιοχή του στα γραφικά και το όνομα του (player 1 , player 2)

updatePlayerCards(Player ,PlayerArea) : void // ανανεώνει τα γραφικά του παίκτη και του ταμπλού με βάση τις κάρτες που έχει ο παίκτης στο χέρι του και σε όλες τις περιοχές του.

Τέλος... μια main για να τρέξει το πρόγραμμα

**ActionListeners** : Επίσης περιέχει κλάσεις που υλοποιούν το interface ActionListener όπου χειρίζονται τα Jbuttons της κλάσης View για την αλληλεπίδραση.

Όλες οι κλάσεις αυτές έχουν ένα πεδίο player , και ένα player area για να κάνουν τις αντίστοιχες λειτουργίες.

Η Train5CardActionListener δεν έχει κανένα όρισμα

Η PlayCardActionListener έχει έξι δυο ορίσματα opponent και opponent area για να χειρίζεται τον αντίπαλο σε περιπτώσεις train robbing κτλ.

MyDestinationCardsActionListener // όταν πατηθεί αυτό το κουμπί ανοίγει ένα παράθυρο με τις κάρτες προορισμού που έχει εξαργυρώσει ο παίκτης , αν δεν έχει εξαργυρώσει καμία εμφανίζεται ένα μήνυμα λάθους.

MyBigCitiesCardsActionListener // όταν πατηθεί αυτό το κουμπί ανοίγει ένα παράθυρο με όλες τις κάρτες bonus . Αν η κάρτα είναι γκρι σημαίνει πως δεν τη έχει εξαργυρώσει ο παίκτης επίσης κάτω από κάθε κάρτα υπάρχει ο αριθμός των επισκέψεων.

PlayCardActionListener // όταν πατηθεί αυτό το κουμπί εκτελείται η συνάρτηση playCards() του παίκτη , πριν από αυτό στο arraylist trainCardsToPlay αν υπάρχουν κάρτες μπαλαντέρ μπαίνουν στο τέλος της λίστας ώστε να προστεθούν τελευταίες στο railyard . Έπειτα σε ένα block try και ένα catch δοκιμάζουμε αν γίνεται να μετακινήσουμε της κάρτες διαφορετικά παίρνουμε αντίστοιχο μήνυμα λάθους. Αν δεν είναι η σειρά του παίχτη και πατηθεί το κουμπί τότε δεν γίνεται

τίποτα , διαφορετικά αν έχει τραβήξει κάρτες η έχει κάνει κάτι άλλο που δεν επιτρέπεται να ρίξει κάρτες εμφανίζεται μήνυμα λάθους.

`SelectCardActionListener` // αν είναι σειρά του παίχτη , τότε κάθε κάρτα τραίνων που “πατάει” αυτή μετακινείται μερικά pixel προς τα πάνω δηλώνοντας ότι έχει επιλεχτεί για να ριχτεί στο railyard του παίχτη. Κάθε φορά που καλείται η `updatePlayerCards()` σε όλες τις κάρτες γίνεται `reset position` (για την ακρίβεια διαγράφονται τα γραφικά των παλαιών και με βάση τις κάρτες του παίχτη δημιουργούνται καινούρια γραφικά για τις κάρτες).

`OnTheTrackActionListener` // αυτό το κουμπί πρέπει να το πατάει κάθε παίχτης στη αρχή του γύρου του , δεν μπορεί να κάνει άλλη κίνηση αν δεν το πατήσει. Η δική μου υλοποίηση επιβάλλει στον παίκτη να το πατάει ακόμα και αν δεν έχει κάρτες στο railyard για να τηρείται και να φαίνεται πάντα η σειρά των κινήσεων. Τέλος αν έχει πατηθεί το κουμπί και πατιέται για δεύτερη η περισσότερη φορά τότε εμφανίζεται μήνυμα λάθους ότι δεν μπορεί να πατηθεί για περισσότερο από μια φορές.

`Tain5cardsActionListener` // αυτό το κουμπί προσομοιώνει τις μια από τις 5 κάρτες του ταμπλό , όταν πατηθεί η κάρτα μετακινείται στο χέρι του παίκτη ανάλογα τον παίκτη που θα το πατήσει.

`BuyDestinationCardListener` // αυτό το κουμπί ανοίγει το ίδιο παράθυρο που ανοίγει στην αρχή του παιχνιδιού για επιλεγούν κάρτες προορισμού.

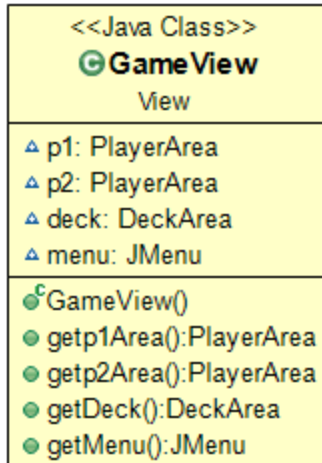
### 3. Η Σχεδίαση και οι Κλάσεις του Πακέτου View

Το πακέτο View περιέχει τις κλάσεις που κατασκευάζουν τα γραφικά του παιχνιδιού και αλληλεπιδρούν με το Controller μεσο `JButtons` , `ActionListener` όπου αυτό με την σειρά του ανανεώνει το Model. Όλες οι κλάσεις του View δεν έχουν αξιοσημείωτες μεθόδους απλά έχουν `getters` ,`setter` και `Constructor` οι οποίες δεν υπάρχει λόγος να αναφερθούν, περιγράφονται στα UML. Τέλος οι κλάσεις που έχουν κουμπιά έχουν μια συνάρτηση(η και περισσότερες) η οποία θέτει στα κουμπιά `ActionListener`.

#### 3.1 Class `GameView` extends `Jframe`

Η κλάση αυτή είναι υπεύθνη για τα γραφικά του παιχνιδιού. Συγκεκριμένα είναι το “μεγάλο παράθυρο” στο οποίο περιέχονται άλλες κλάσεις του πακέτου View όπου υλοποιούν την περιοχή του κάθε παίχτη και του ταμπλό.



**Γνωρίσματα :**

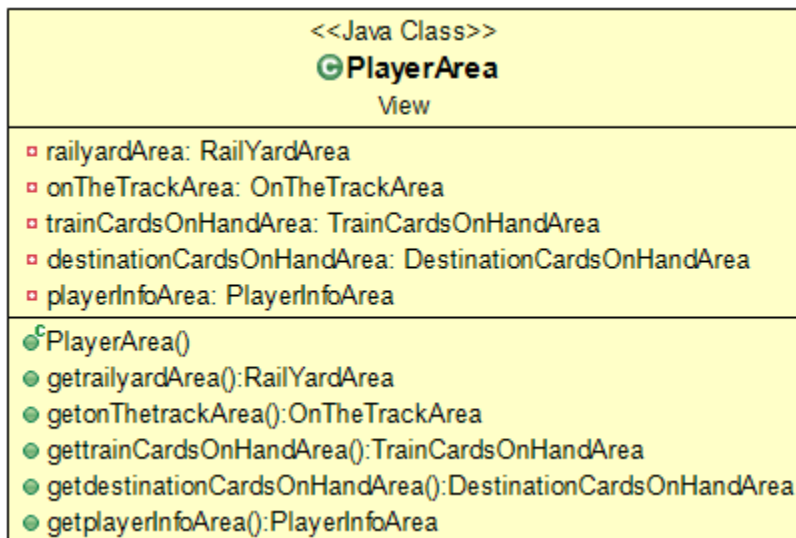
PlayerArea p1,p2 // η περιοχή του παίχτη 1,2 αντίστοιχα

DeckArea deck // η περιοχή του μεσαίου ταμπλού

JMenu menu // ένα πειραματικό μενού που τοποθετείται στο πάνω μέρος (δεν χρησιμοποιήθηκε)

### 3.2 Class PlayerArea extends JLayeredPane

Η κλάση αυτή αναπαριστά την περιοχή του παίχτη σε γραφικά.

**Γνωρίσματα :**

RailYardArea railyardArea // τα γραφικά της RailYard του παίχτη

OnTheTrackArea onTheTrackArea // τα γραφικά της OnTheTrack του παίχτη

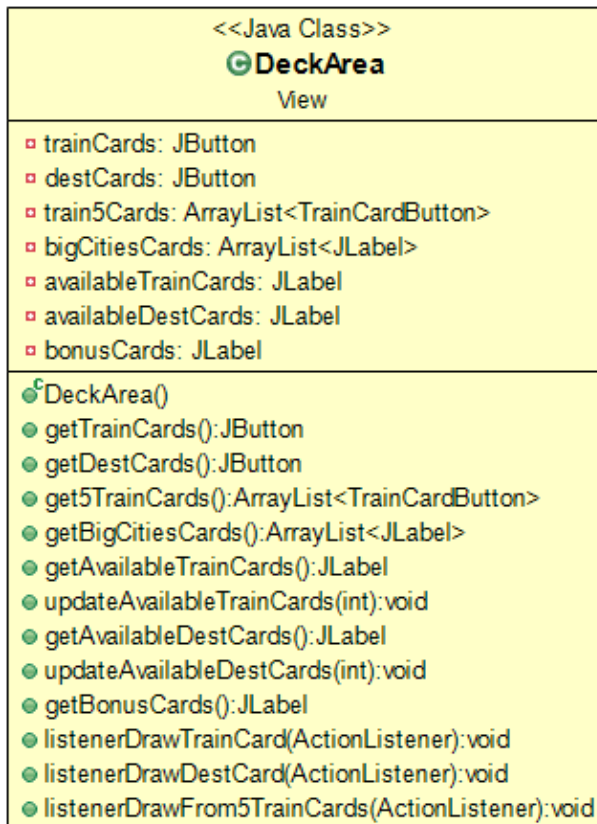
TtrainCardsOnHandArea // τα γραφικά της περιοχής των καρτών τραίνων που κατέχει ο παίχτης

DestinationCardsOnHandArea destinationCardsOnHandArea // τα γραφικά της περιοχής των καρτών προορισμού που κατέχει ο παίχτης

PlayerInfoArea playerInfoArea // τα γραφικά της περιοχής που προβάλλονται οι πληροφορίες του παίχτη

### 3.3 Class DeckArea extends JLayeredPane

Η κλάση αυτή είναι υπεύθυνη για τα γραφικά του ταμπλό στην μέση του παιχνιδιού.



#### Γνωρίσματα :

JButton trainCards // ένα κουμπί για να τραβάμε κάρτες τρένων από το ταμπλό

JButton destCards // ένα κουμπί για να τραβάμε κάρτες προορισμού από το ταμπλό

`ArrayList<TrainCardButton> train5Cards` // μια λίστα κουμπιών που αναπαριστούν τις 5 κάρτες τρένων στο ταμπλό (changed from JButton to TrainCardButton)

`ArrayList<JLabel> bigCitiesCards` // μια λίστα από JLabels που αναπαριστούν τις διαθέσιμες κάρτες μεγάλων πόλεων

`JLabel availableTrainCards` // ένα JLabel που προβάλλει τον αριθμό των διαθέσιμων καρτών τρένων στην στοίβα

`JLabel availableDestCards` // ένα JLabel που προβάλλει τον αριθμό των διαθέσιμων καρτών προορισμού στην στοίβα

`JLabel bonusCards` // ένα JLabel που προβάλλει τον αριθμό των διαθέσιμων καρτών μεγάλων πόλεων

### Μεθόδους :


`ListenerDrawTrainCard` // προσθέτει action listener στο κουμπί που του ταμπλό που τραβάμε κάρτες τρένων.

`ListenerDrawDestCard` // προσθέτει action listener στο κουμπί του ταμπλό που τραβάμε κάρτες προορισμού.

`ListenerDrawFrom5TrainCards` // προσθέτει action listener στα κουμπιά των 5 τρένων καρτών του ταμπλό

## 3.4 Class RailYardArea extends JLayeredPane

Η κλάση αυτή είναι υπεύθυνη για τα γραφικά της RailYard του παίχτη

<<Java Class>>	
 <b>RailYardArea</b> View	
<ul style="list-style-type: none"> <li>▣ red: ArrayList&lt;JLabel&gt; = new ArrayList&lt;&gt;()</li> <li>▣ black: ArrayList&lt;JLabel&gt; = new ArrayList&lt;&gt;()</li> <li>▣ blue: ArrayList&lt;JLabel&gt; = new ArrayList&lt;&gt;()</li> <li>▣ green: ArrayList&lt;JLabel&gt; = new ArrayList&lt;&gt;()</li> <li>▣ purple: ArrayList&lt;JLabel&gt; = new ArrayList&lt;&gt;()</li> <li>▣ white: ArrayList&lt;JLabel&gt; = new ArrayList&lt;&gt;()</li> <li>▣ yellow: ArrayList&lt;JLabel&gt; = new ArrayList&lt;&gt;()</li> <li>▣ orange: ArrayList&lt;JLabel&gt; = new ArrayList&lt;&gt;()</li> <li>▣ colornames: JLabel</li> <li>▣ areaname: JLabel</li> <li>▣ movetoOnTheTrack: JButton</li> </ul>	
<ul style="list-style-type: none"> <li>☛ RailYardArea()</li> <li>☛ getCardList(CardColor):ArrayList&lt;JLabel&gt;</li> <li>☛ getmovetoOnTheTrackButton():JButton</li> <li>☛ getColorNames():JLabel</li> <li>☛ getAreaName():JLabel</li> <li>☛ listeMoveToOnTheTrack(ActionListener):void</li> </ul>	

**Γνωρίσματα :**

`ArrayList<JLabel>` // μία λίστα από `JLabels` για κάθε χρώμα για τις  
στοιβαγμένες κάρτες τρενών

`JLabel colornames` // ένα `JLabel` για τα ονόματα των χρωμάτων των καρτών

`JLabel areaname` // ένα `JLabel` που προβάλει το όνομα του πεδίου (RailYard)

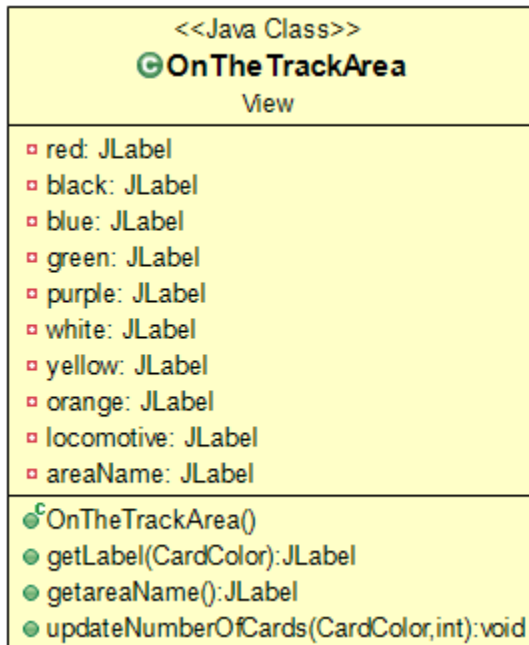
`JButton movetoOnTheTrack` // ένα κουμπί για την μετακίνηση των καρτών στην  
`OnTheTrack`

**Μεθόδους :**

`ListeMoveToOntheTrack` // προσθέτει `action listener` στο κουμπί του  
`OnTheTrack`

### 3.5 Class `OnTheTrackArea` extends `JLayeredPane`

Η κλάση αυτή είναι υπεύθυνη για τα γραφικά της `OnTheTrack` του παίχτη

**Γνωρίσματα :**

`JLabel areaName` // ένα `JLabel` που προβάλει το όνομα του πεδίου  
(`OnTheTrack`)

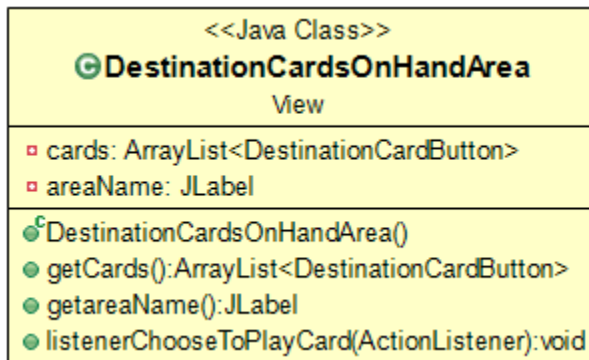
Και Ένα JLabel για κάθε είδος TrainCard (σύνολο 8) που αναπαριστούν τις εικόνες των καρτών

**Μεθόδους :**

updateNumberOfCards(CardColor ,int ) // ανανεώνει τον αριθμό των καρτών στο χρώμα που αντιστοιχεί το CardColor.

### 3.6 Class DestinationCardsOnHandArea extends JLayeredPane

Η κλάση αυτή αναπαριστά τα γραφικά της περιοχής του παίχτη στην οποία υπάρχουν οι κάρτες προορισμού του.



**Γνωρίσματα :**

**ArrayList<DestinationCardButton> cards** // μια λίστα κουμπιών που αναπαριστούν τις κάρτες (changed from JButton to DestinationCardButton)

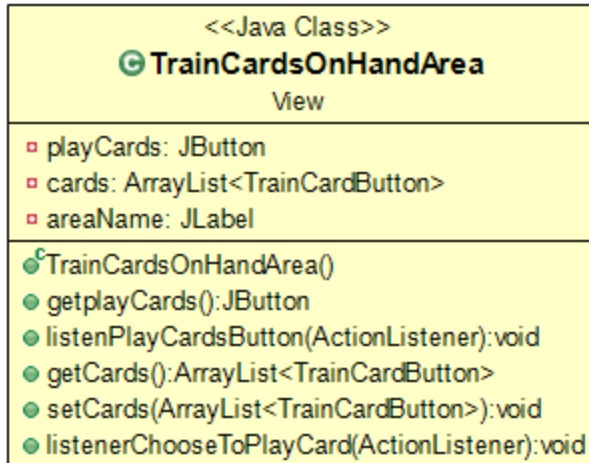
**JLabel areaName** // ένα JLabel που προβάλει το όνομα του πεδίου (DestinationCardsOnHand)

**Μεθόδους :**

**ListenerChooseToPlayCard** // προσθέτει action listener στο κουμπι της καρτας

### 3.7 Class TrainCardsOnHandArea extends JLayeredPane

Η κλάση αυτή αναπαριστά τα γραφικά της περιοχής του παίχτη στην οποία υπάρχουν οι κάρτες τρένων του. Δεν είναι υποκλάση της DestinationCardsOnHandArea πλέον για να φαίνονται τα γνωρίσματα καλύτερα.



### Γνωρίσματα :

JButton playCards // το κουμπί που “παιζει” τις κάρτες (μεταφορά σε RailYard)

ArrayList<TrainCardButton> cards // μια λίστα κουμπιών που αναπαριστούν τις κάρτες)

JLabel areaName // ένα JLabel που προβάλει το όνομα του πεδίου

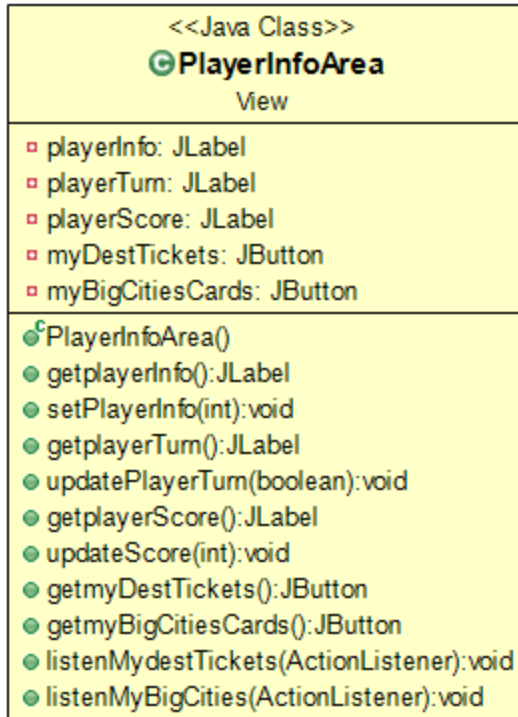
### Μεθόδοι :

ListenPlayCardsButton // προσθέτει action listener στα κουμπί play cards

ListenerChooseToPlayCard // προσθέτει action listener στα κουμπιά κάρτες στο χέρι του παίχτη

## 3.8 Class PlayerInfoArea extends JLayeredPane

Η κλάση αυτή αναπαριστά τα γραφικά του πεδίου πληροφοριών του παίχτη

**Γνωρίσματα :**

JLabel playerInfo // το όνομα του παίχτη

JLabel playerTurn // προβάλλει αν είναι η σειρά του παίχτη (player's turn : -)

JLabel playerScore // προβάλλει το score του παίχτη

JButton myDestTickets // ένα κουμπί που προβάλλει (όταν πατηθεί) τις κάρτες προορισμού του παίχτη

JButton myBigCitiesCards // ένα κουμπί που προβάλλει (όταν πατηθεί) τις κάρτες μεγάλων πόλεων του παίχτη

**Μεθόδοι :**

updatePlayerTurn(boolean) // ανανεώνει τον γύρο του παίκτη στα γραφικά

setPlayerInfo(int) // ανανεώνει το όνομα του παίκτη (Player 1,2)

updateScore(int)) // ανανεώνει το σκορ του παίκτη στα γραφικά

ListenMydestTickets // προσθέτει action listener στο κουμπί  
MyDestinationTickets

ListenMyBigCities // προσθέτει action listener στο κουμπί MyBigCitiesCards

NEW CLASSES :

### 3.9 Class TrainCardButton extends JButton

Αυτή η κλάση αναπαριστά ένα κουμπι κάρτας τρενών. Ως ορίσματα έχει ένα χρώμα CardColor και ένα boolean για να ξέρουμε αν είναι σηκωμένη η όχι.

<<Java Class>> <b>TrainCardButton</b> View
▲ color: CardColor ▲ Up: boolean
● TrainCardButton(CardColor) ● getColor(): CardColor ● setImageAndColor(CardColor): void ● isUp(): boolean ● setUp(boolean): void

**Γνωρίσματα :**

CardColor : color // το χρώμα της κάρτας

Boolean : Up // καθορίζει αν η κάρτα είναι σηκωμένη με τιμή true αλλιώς είναι κάτω (false)

**Μεθόδοι :**

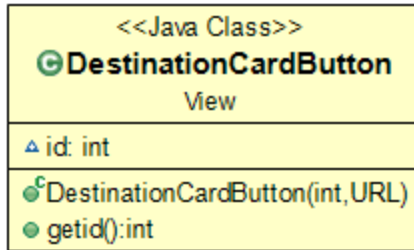
(setter and getter)

IsUp() : boolean // επιστρέφει true αν η κάρτα είναι σηκωμένη false αλλιώς

### 3.9 Class DestinationCardButton extends JButton

Αυτή η κλάση αναπαριστά ένα κουμπι κάρτας προορισμού. Ως ορίσματα έχει ένα int id γιατί κάθε κάρτα προορισμού είναι μοναδική, με αυτόν τον τρόπο θα μπορούμε να την εντοπίζουμε.





Γνωρίσματα : int id (αναφερθηκε)

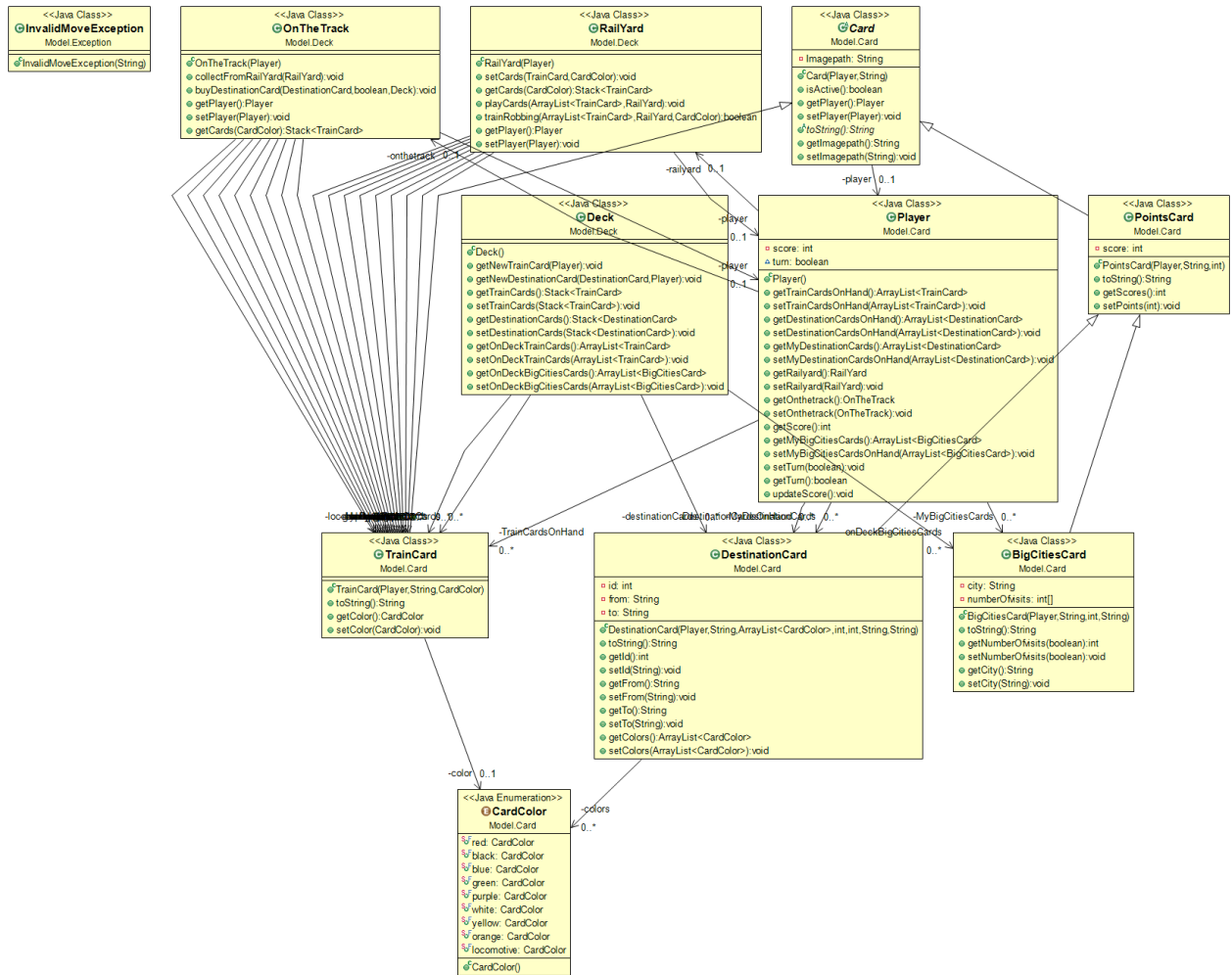
Μεθόδοι : Constructor και Getter

## 4. Η Αλληλεπίδραση μεταξύ των κλάσεων – Διαγράμματα UML

Γενικά σε κάθε κλάση έχουν περιγραφθεί οι υποκλάσεις και ο λόγος ο οποίος καθιστά την σχεδίαση όπως έχει προκύψει. Εδώ θα δούμε τα διαγράμματα UML κάθε πακέτου και υποπακέτου.

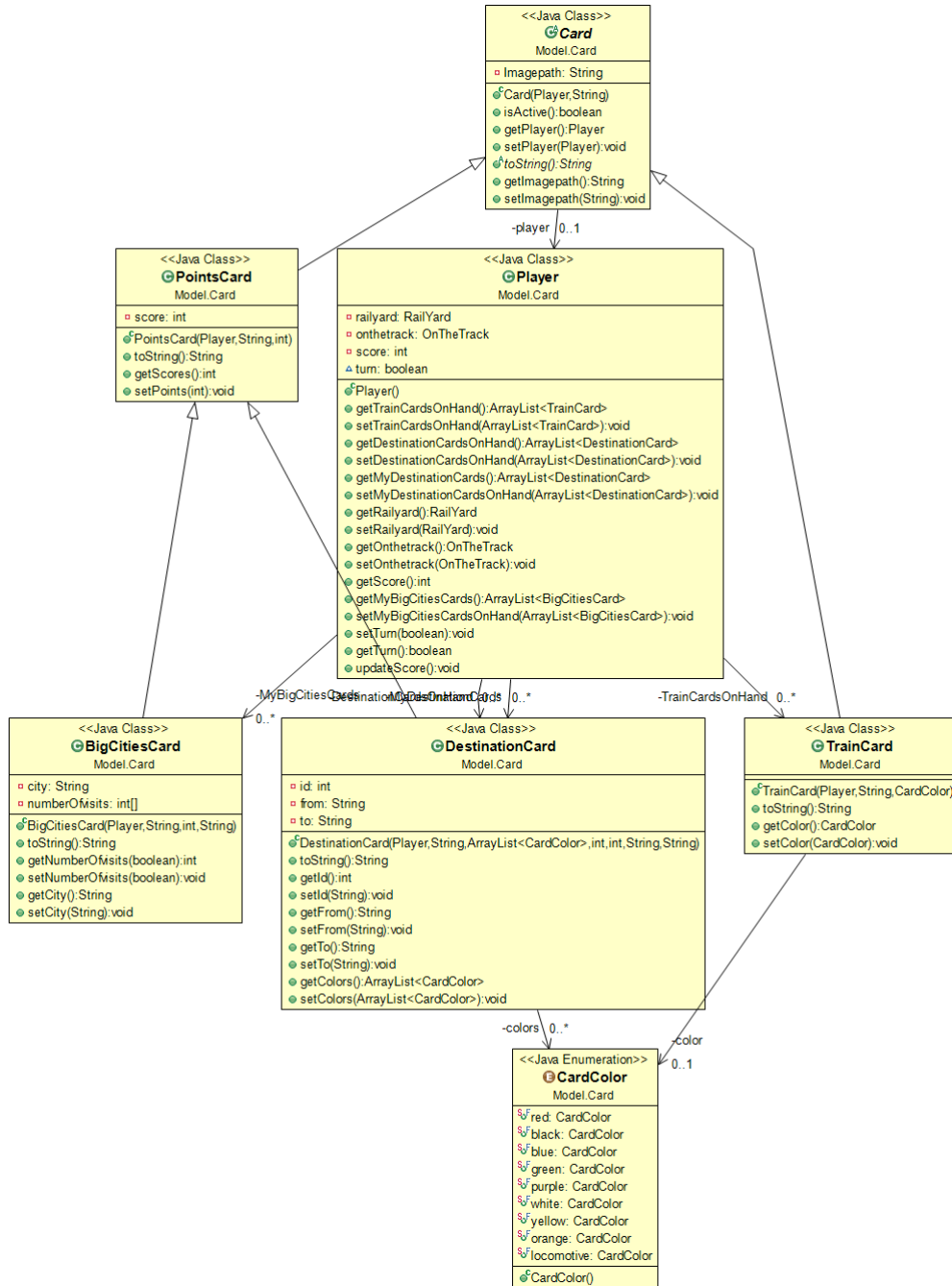
### 4.1 Model UML

Εδώ βλέπουμε ολόκληρο το Model



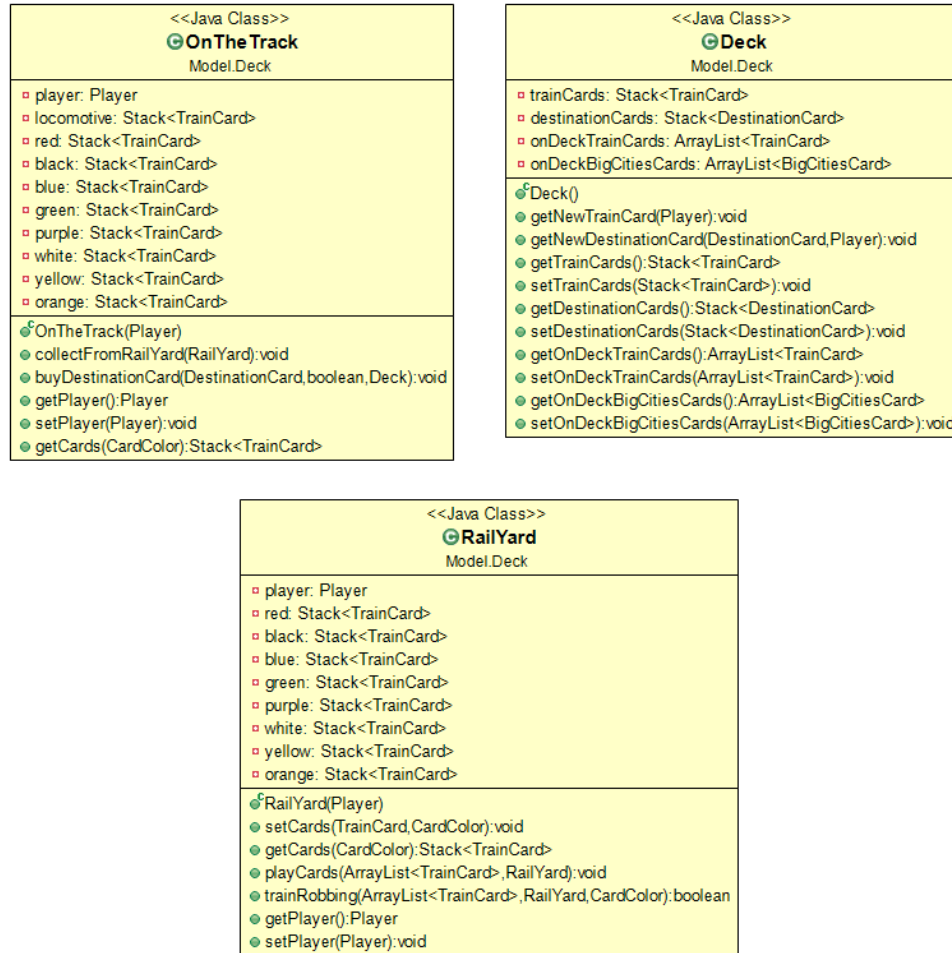
### 4.1.1 Model.Card UML

Εδώ βλέπουμε όλες τις κλάσεις καρτών ,επίσης έχει προστεθεί η κλάση Player στο πακέτο card για λόγους ευκολίας



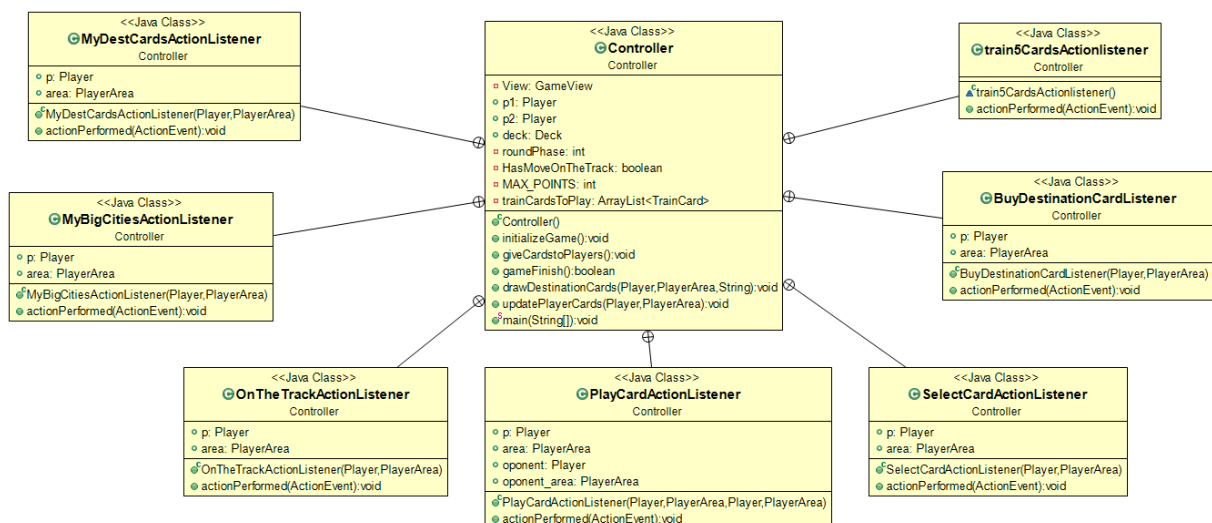
## 4.1.2 Model.Deck UML

Εδώ βλέπουμε την κλάση του ταμπλό την Railyard και την OnTheTrack



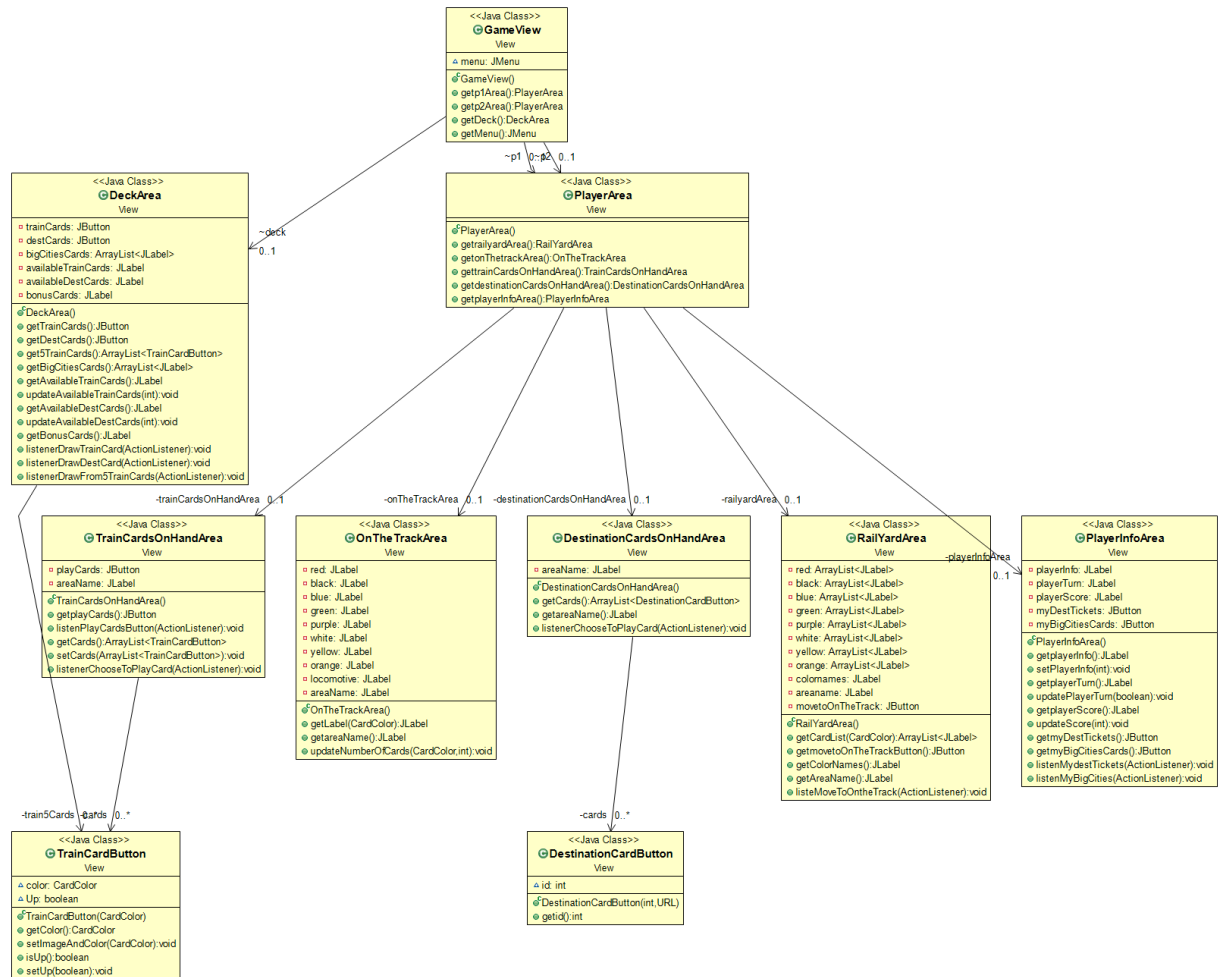
## 4.2 Controller UML

Υπάρχει στην περιγραφή του Controller παραπάνω



## 4.3 View UML

Εδώ βλέπουμε τις κλάσεις του View



## 5. Λειτουργικότητα (Β Φάση)

Όλα τα ερωτήματα υλοποιήθηκαν επιτυχώς . Επίσης χρησιμοποιήθηκαν 8 test για την λειτουργικότητα του παιχνιδιού .

testToStringCards() // τεστάρει αν εμφανίζονται σωστά οι toString μέθοδοι των καρτών

testRailYardValidMoves() // τεστάρει μια σειρά από επιτρεπτές κινήσεις στην RailYard

`testRailYardInvalidMoves()` // τεστάρει μια σειρά από λάθος κινήσεις στην RailYard και “πιάνει” τα exception

`testCollectFromRailYard()` // τεστάρει αν μετακινούνται σωστά οι κάρτες από την RailYard στο OnTheTrack

`testBuySuccessfullyDestinationCard()` // τεστάρει αν αγοράζετε σωστά μια κάρτα προορισμού

`testBuyUnsuccessfullyDestinationCard()` // τεστάρει την περίπτωση όπου δεν μπορεί ο παίκτης να αγοράσει την αντίστοιχη κάρτα προορισμού και “πιάνει” το αντίστοιχο exception

`testgameStart()` // τεστάρει αν γίνεται σωστά η αρχικοποίηση του παιχνιδιού

`testFinishGame()` // τεστάρει αν είναι σωστή η συνάρτηση που ελέγχει για τον τερματισμό του παιχνιδιού

## 6. Συμπεράσματα

Η υλοποίηση στηρίχθηκε απόλυτα στην α φάση , έγιναν μικροαλλαγές που φαίνονται παραπάνω , αν και πιστεύω ότι θα μπορούσαν να βελτιωθούν λίγο οι αλγόριθμοι που χρησιμοποιήθηκαν διότι μερικοί είναι μεγάλοι σε κώδικα. Επίσης το κουμπί `MoveToOnTheTrack` δεν ήταν απαραίτητο θα μπορούσαν να γίνονται αυτόματα οι μετακινήσεις εφόσον υπάρχουν κάρτες στην RailYard. Επιπλέον η υλοποίηση μου προβάλλει τα exceptions και τις `toString` μεθόδους για την καλύτερη κατανόηση του πως δουλεύει ο κώδικας και την εύκολη κατανόηση του προγράμματος από τους βοηθούς. Τέλος , υπάρχει ένα `jmenu` στα γραφικά που δεν χρησιμοποιήθηκε για το bonus.