CS252 - Object Oriented Programming

PROJECT PHASE 2
Γραπτή αναφορά

Γεώργιος Μάριος Κοκιάδης < csd3962@csd.uoc.gr >
Χειμερινό Εξάμηνο Περιόδου 2018-2019

Υλοποίηση του επιτραπέζιου παιχνιδιού

Ticket to Ride

Φάση Β' : Υλοποίηση.

Περιεχόμενα:

Εισαγωγή

Σε αυτό το Project καλούμαστε να υλοποιήσουμε το επιτραπέζιο παιχνίδι “Ticket to Ride”, το οποίο δημοσιεύτηκε το 2004 απο την εταιρία Days Of Wonder. Κέρδισε πολλά βραβεία και είχε γενικότερα μια μεγάλη επιτυχία στο κοινό του. Περισσότερες πληροφορίες για το παιχνίδι μπορείτε να βρείτε εδώ.

Στην υλοποίηση μας, τόσο για την διευκόλυνση την δική μας όσο και για την γρηγορότερη εκπόνηση του Project, χρησιμοποιούμε το προγραμματιστικό μοντέλο MVC (Model - View - Controller). Κατα αυτο το μοντελο, το ζήτημα του να φτιάξουμε όλο αυτό το πρόγραμμα, χωρίζεται σε 3 ευκολότερα υλοποιήσιμες κατηγορίες:

- Το “Model”, στο οποίο υλοποιούμε την λειτουργικότητα του παιχνιδιού, στην προκειμένη περίπτωση, το πως δουλεύουν οι κάρτες, οι διαφορετικές περιοχές του παιχνιδιού και η διαχείριση των καρτών.
- Το “View” στο οποίο υλοποιείται το κομμάτι που βλέπει ο χρήστης, δηλαδη το UI (User Interface) ή GUI (Graphical UI)
- Τέλος το “Controller” που βρίσκονται κομμάτια κώδικα που ενορχηστρώνουν την επικοινωνία μεταξύ “View” και “Model”.

Στις σελίδες αυτής της αναφοράς θα παρουσιαστεί ο τρόπος υλοποίησης κάθε ενός από αυτά τα κομμάτια του μοντέλου MVC, όπως και διαγράμματα UML και κομμάτια κώδικα, συνοδευμένα από βοηθητικά σχόλια τύπου Javadoc, ωστε να βοηθήσουν στην κατανόηση του σκεπτικού που ακολουθηθηκε.

Για την δεύτερη φάση, ανανεώθηκε η αναφορά από την πρώτη φάση. Προστέθηκαν εξηγήσεις για όλες τις αλλαγές, οι οποίες σηματοδοτούνται με: **+**

Model

Γεώργιος Μάριος Κοκιάδης < csd3962@csd.uoc.gr >

Υλοποίηση πακέτου Model

Σε γενικές γραμμές, το πακέτο Model αποτελείται από αρκετες κλάσεις, οι οποίες όμως επιτελούν μικρές λειτουργίες, οι οποίες αλληλοσυνδεόμενες (και με σωστή οργάνωση από το πακέτο Controller) αποτελούν τον πυρήνα της λειτουργικότητας του παιχνιδιού, σε επίπεδο μνήμης.

Ας δούμε λοιπόν τι είναι και τι κάνει κάθε κλάση.

Classes that define cards

Card Class

UML αναπαράσταση της κλάσης Card

Η Κλάση Card αποτελεί την βάση για κάθε κάρτα στο παιχνίδι.

Περιέχει 3 πεδία για τα δεδομένα της:

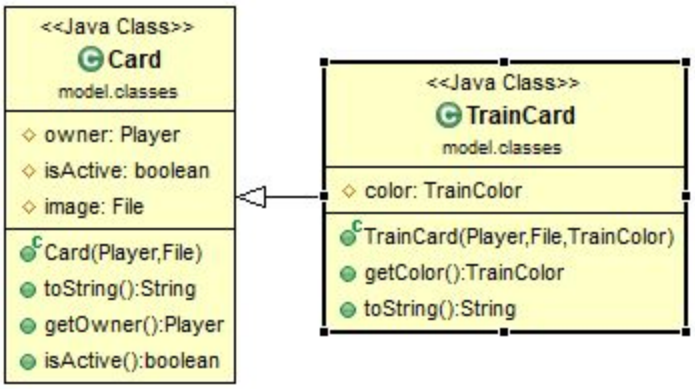
- `protected Player Owner` : Ο ιδιοκτήτης της κάρτας
- `protected boolean isActive` : αν η κάρτα είναι ενεργή ή όχι. Εξηγείται παρακάτω τι σημαίνει αυτό.
- `protected File Image` : η εικόνα που χρησιμοποιεί η καρτα για την αναπαράσταση της γραφικά

Περιέχει επίσης 3 μεθόδους:

- `public String toString()` : Επιστρέφει πληροφορίες για την κάρτα σε ένα String.
- `public Player getOwner()` : Observer, επιστρέφει τον ιδιοκτήτη της κάρτας
- `public boolean getActive()` : Observer, επιστρέφει την κατάσταση της κάρτας.

Ο Constructor απλά θέτει τις κατάλληλες τιμές εισαγόμενες από τις παραμέτρους που δέχεται.

TrainCard Class extends Card



UML αναπαράσταση της κλάσης TrainCard

Η Κλάση TrainCard αναπαριστά τις κάρτες βαγονιών, οι οποίες χρησιμοποιούνται για την εξαγορά DestinationCard έχουν ένα χαρακτηριστικό, το χρώμα τους. Καθώς πρόκειται για κάρτα, κάνει extend την κλάση Card.

Περιέχει 1 πεδίο για τα δεδομένα της:

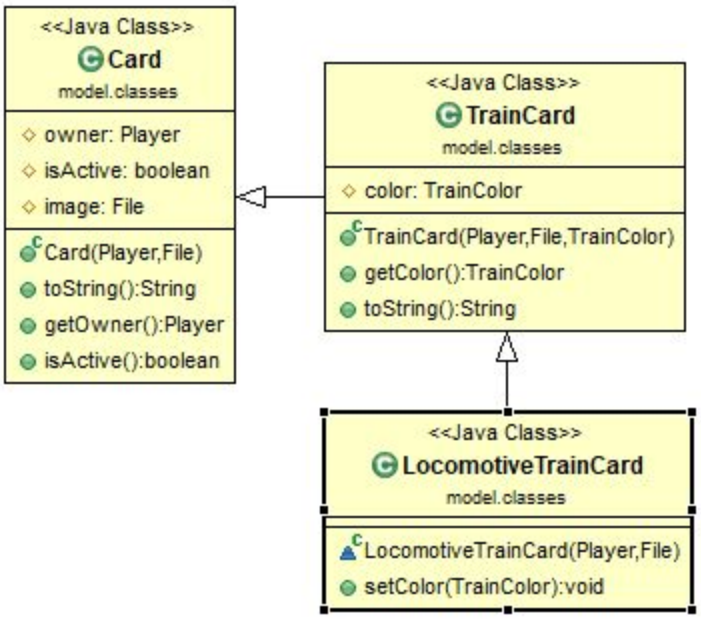
- `protected TrainColor Color` : το χρώμα που αναπαριστά η κάρτα.

Περιέχει επίσης 2 μεθόδους:

- `public TrainColor getColor()` : Observer, επιστρέφει το χρώμα της κάρτας.
- `public String toString()` : Επιστρέφει πληροφορίες για την κάρτα σε ένα String.

Ο Constructor χρησιμοποιεί τον Constructor του γονέα του και απλα θέτει το πεδίο color με βάση την εισαγόμενη παράμετρο.

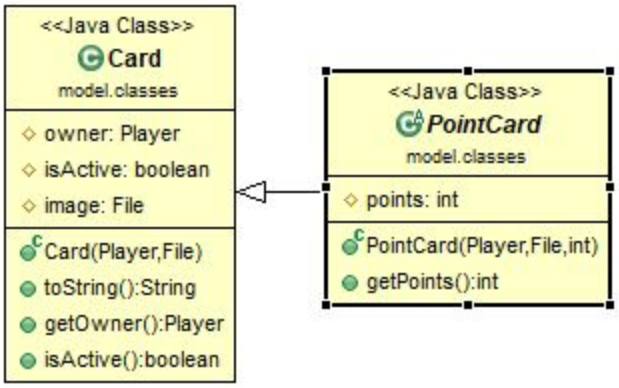
LocomotiveTrainCard Class **extends TrainCard**



UML αναπαράσταση της κλάσης LocomotiveTrainCard

- Η κλάση αυτή υπήρξε στην αρχική σχεδίαση του παιχνιδιού, όμως στην διαδικασία της υλοποίησης περισσότερο δυσκόλεψε τα πραγματα, παρά τα διευκόλυνε. Παρακάτω επεξηγείται πώς υλοποιήθηκαν εν τέλει οι Locomotive Cards.

PointCard Class **extends Card**



UML αναπαράσταση της κλάσης PointCard

Η Κλάση PointCard ειναι μια απλή προέκταση της κλάσης Card, προσθέτοντας μονάχα την ικανότητα η κάρτα να έχει πόντους.

Περιέχει 1 πεδίο για τα δεδομένα της:

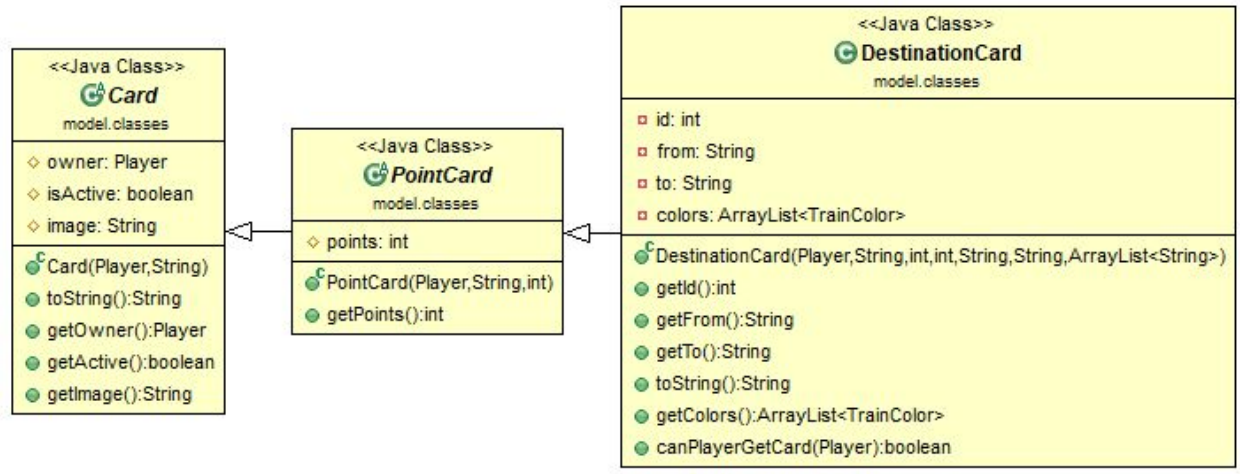
- `Protected int points` : οι πόντοι που αναπαριστά αυτή η κάρτα

Περιέχει επίσης 1 μέθοδο:

- `public int getPoints()` : Accessor, επιστρεφει τους πόντους.

Ο Constructor χρησιμοποιεί τον Constructor του γονέα του και απλα θέτει το πεδίο points με βάση την εισαγόμενη παράμετρο.

DestinationCard Class **extends PointCard**



UML αναπαράσταση της κλάσης DestinationCard

Η Κλάση DestinationCard ειναι μια προέκταση της κλάσης PointCard. Είναι η υλοποίηση των καρτών προορισμού, οι οποίες εξαγοράζονται με TrainCards.

Περιέχει 4 πεδία για τα δεδομένα της:

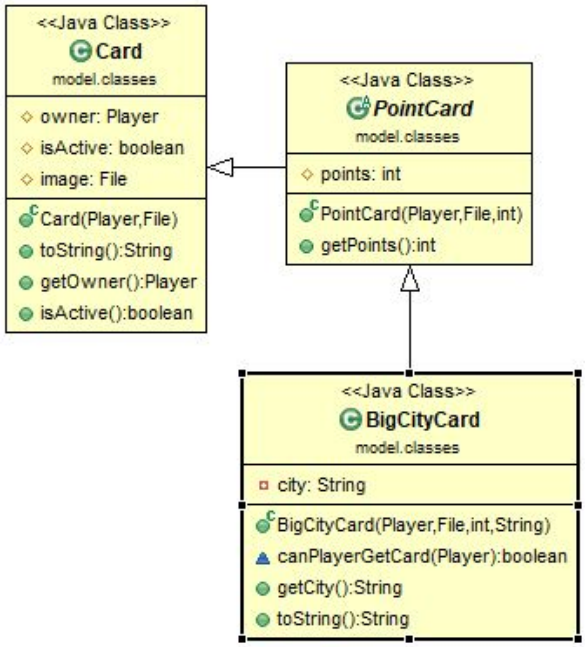
- `private int Id` : Μοναδικό id κάθε κάρτας προορισμού
- `private string From` : τοποθεσία εκκίνησης ταξιδιού (Λειτουργικότητα χαμένη λόγω απλοποίησης)
- `private string To` : τοποθεσία που καταλήγει το ταξίδι.
- `private ArrayList<TrainColor> Colors` : ένας πίνακας που περιεχει τα χρώματα που χρειάζεται αυτή η κάρτα.

Περιέχει επίσης 3 μεθόδους:

- `public int getId()` : Accessor, επιστρεφει το μοναδικό id
- `public String getFrom()` : Accessor, επιστρέφει την τοποθεσία εκκινησης
- `public String getTo()` : Accessor, επιστρεφει την τοποθεσία άφιξης
- `public String toString()` : Επιστρέφει τα στοιχεία της κάρτας σε ενα String.
- `public ArrayList<TrainColor> getColors()` : Accessor, επιστρεφει τα χρώματα που χρειάζονται για να αγοραστεί αυτή η κάρτα
- `+ public boolean canPlayerGetCard(Player p)` : Accessor, επιστρεφει εάν ο παίκτης p μπορεί να εξαγοράσει αυτήν την κάρτα

Ο Constructor χρησιμοποιεί τον Constructor του γονέα του και απλα θέτει τα πεδία id, from, to, colors με βάση τις εισαγόμενες παραμέτρους.

BigCityCard Class **extends PointCard**



UML αναπαράσταση της κλάσης BigCityCard

Η Κλάση BigCityCard πρόκειται για μια κάρτα-bonus, η οποία δίνεται σε όποιον παίκτη φτάσει 3 φορές τον προορισμό που αναγράφεται πάνω στην κάρτα.

Περιέχει 1 πεδίο για τα δεδομένα της:

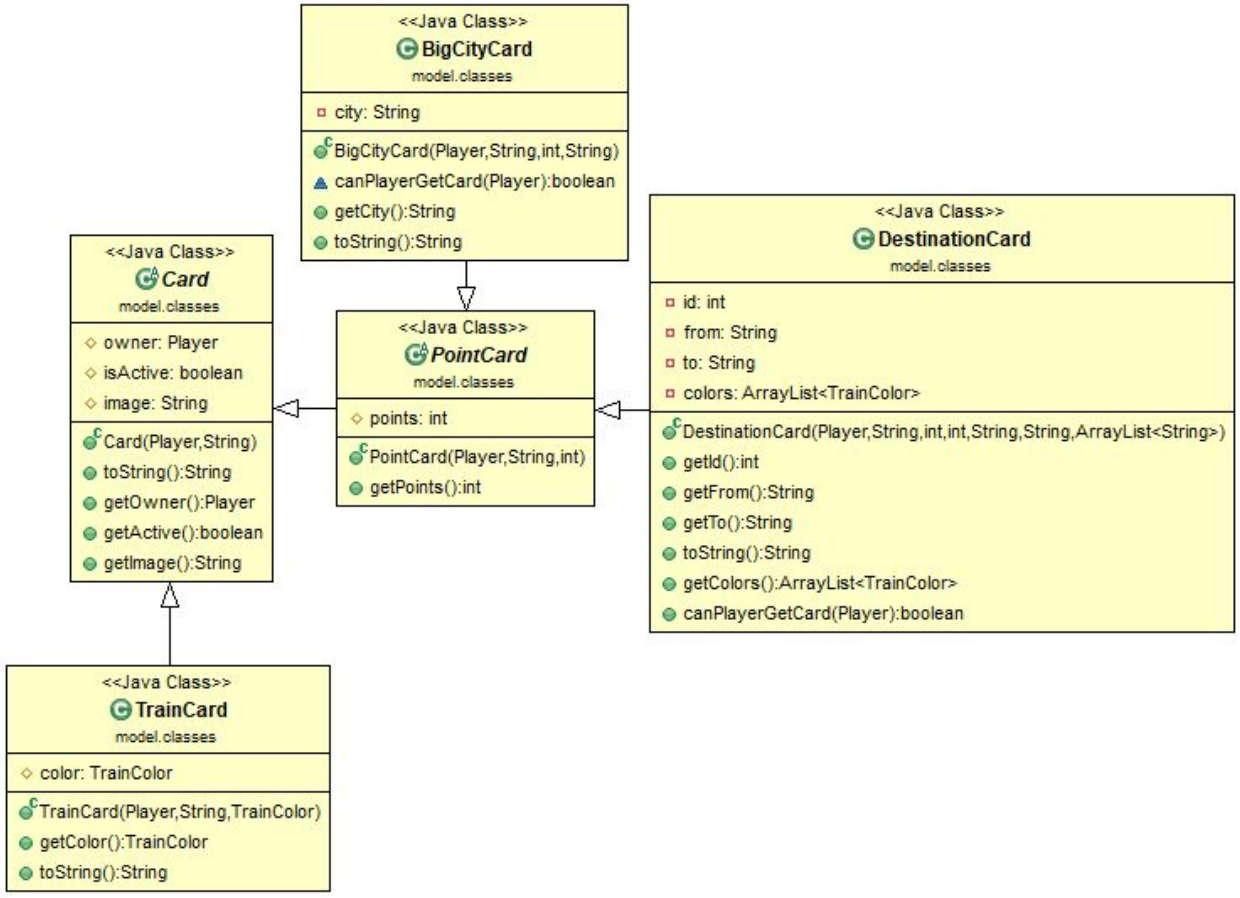
- `private string City` : η πόλη που αναπαριστά η κάρτα

Περιέχει επίσης 3 μεθόδους:

- `public boolean canPlayerGetCard(Player p)` : Observer, βλέπει αν ο παίκτης P που εισάγεται ώς παράμετρος έχει την δυνατότητα να πάρει την κάρτα.
- `public String getCity()` : Accessor, επιστρέφει την πόλη.

Ο Constructor χρησιμοποιεί τον Constructor του γονέα του και απλα θέτει το πεδίο city με βάση την εισαγόμενη παράμετρο.

Overview των υποκλάσεων της Card



UML αναπαράσταση των υποκλάσεων της Card

Οι κάρτες αυτές αποτελούν την βάση όλου του παιχνιδιού, καθώς μέσω αυτών ο παίκτης κερδίζει ή χάνει πόντους. Για την απλοποίηση των ακόλουθων σχεδίων UML, δεν θα παρουσιάζονται οι κλάσεις αυτές, διότι το σχήμα γίνεται χαοτικό και χάνει το νόημα του ως εργαλείο κατανόησης.

Classes that define the player's actions

Player Class

UML αναπαράσταση της κλάσης Player

Η Κλάση Player αναπαριστα τον παίκτη, άρα σε αυτήν την κλάση είναι αποθηκευμένα όλα τα αντικείμενα στην κατοχή του.

Περιέχει 6 πεδία για τα δεδομένα της:

- `private int Score` : Το σκορ του παίκτη
- `private String Name` : Το όνομα του παίκτη
- `private RailYard Yard`: Το RailYard του παίκτη
- `private OnTheTrack Track` : Η περιοχή OnTheTrack
- `private Hand hand` : το χέρι του κάθε παίκτη (Καρτες στην κατοχή του)
- `private int[] BigCityCollection` : Ποσες φορες εχει επισκεφθει ο παίκτης κάθε πόλη.

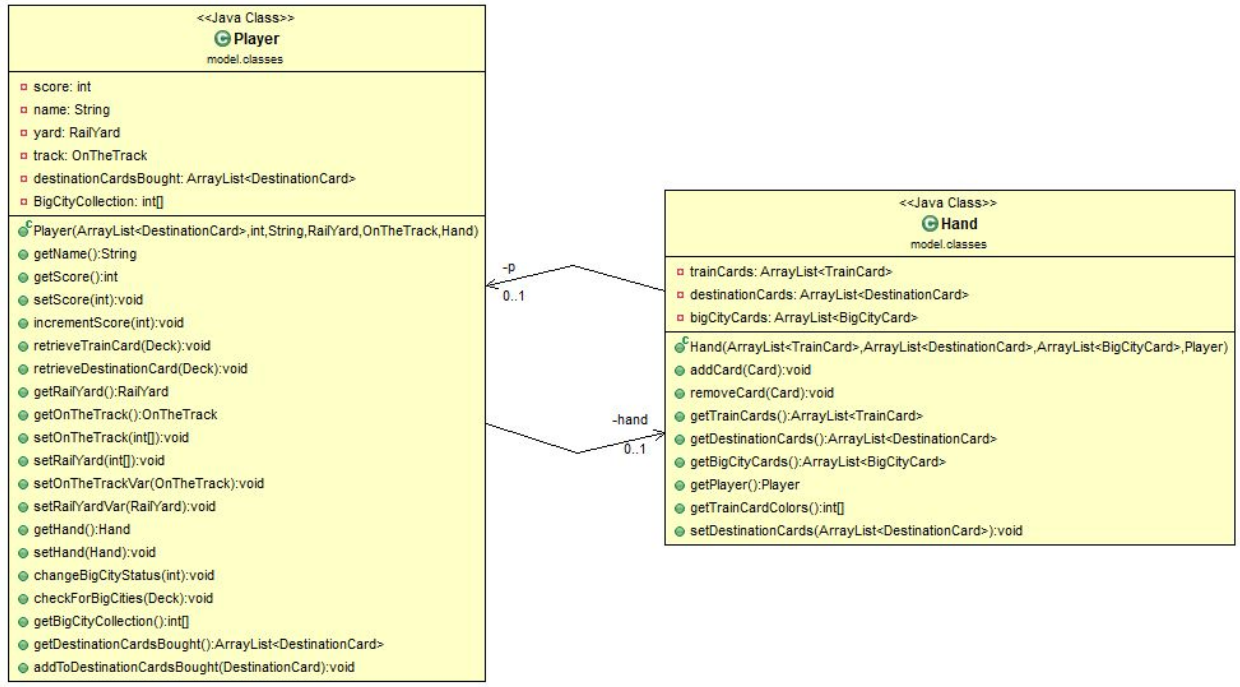
- + `private ArrayList<DestinationCard> BigCityCollection` : Κάρτες που έχει αγοράσει ο παίκτης.

Περιέχει επίσης 10 μεθόδους:

- `public String getName()` : Accessor, επιστρέφει το όνομα.
- `public int getScore()` : Accessor, επιστρέφει το σκορ
- `public void setScore(int score)` : Transformer, θέτει το σκορ
- `public void incrementScore(int increment)`: αυξάνει το σκορ κατα Increment
- `public void retrieveDestinationCard() throws CardNotFoundException`: Transformer, προσθέτει μια κάρτα προορισμού στο χέρι του παίκτη από το deck
- `public void retrieveTrainCard() throws CardNotFoundException` : Transformer, προσθέτει μια κάρτα στο χερι του παικτη απο το deck
- `public RailYard getYard()` : Accessor, επιστρέφει το Yard του παίκτη
- `public OnTheTrack getOnTheTrack()` : Accessor, επιστρέφει το OnTheTrack του παίκτη
- `public hand getHand()` : Accessor, επιστρέφει το Hand του παικτη
- + `public void setHand(Hand hand)` : Transformer,Αλλάζει την μεταβλητή Hand.
- `public void changeBigCityStatus(int city)` : Αλλάζει το περιεχόμενο του πίνακα BigCityCollection.
- + `public void checkForBigCities(Deck deck)` : Accessor, Βλέπει αν ο παίκτης έχει κερδίσει μια Big City Card
- + `public void setOnTheTrack(int[] colors)` : Transformer, Αλλάζει το περιεχόμενο της μεταβλητής OnTheTrack.
- + `public void setRailyard(int[] colors)` : Transformer, Αλλάζει το περιεχόμενο της μεταβλητής RailYard.
- + `public void setOnTheTrackVar(OnTheTrack v)` : Transformer, Αλλάζει την μεταβλητή OnTheTrack.
- + `public void setRailYardVar(RailYard v)` : Transformer, Αλλάζει την μεταβλητή RailYard.
- + `public int[] getBigCityCollection()` : Accessor, επιστρέφει το bigCityCollection του παίκτη
- + `public ArrayList<DestinationCard> getDestinationCardsBought()` : Accessor, επιστρέφει τις καρτες Destination που έχει αγοράσει ο παίκτης.
- + `public void addToDestinationCardsBought(DestinationCard c)` : Transformer, προσθέτει κάρτες στην στοίβα με κάρτες προορισμού που έχει αγοράσει ο παίκτης

Ο Constructor δέχεται ως παράμετρο το όνομα του παίκτη, όλες τις περιοχές και το αρχικό σκόρ. Δέχεται όλα τα πεδία του στον Constructor για να μπορούμε να φορτώσουμε ενα παιχνιδι απο ενα αρχείο.

Hand Class



UML αναπαράσταση της κλάσης Hand και η επικοινωνία της με την κλάση Player

Η Κλάση Hand αναπαριστά το χέρι του παίκτη, δηλαδή τις κάρτες στην κατοχή του.

Περιέχει 3 πεδία για τα δεδομένα της:

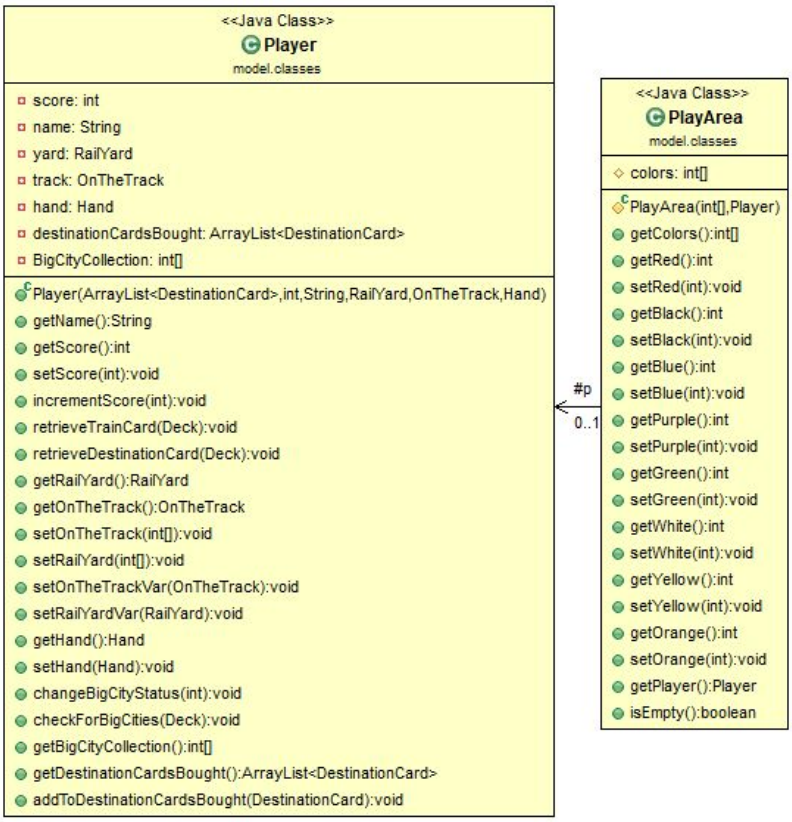
- `private ArrayList<TrainCard> trainCards` : Οι TrainCards που ανήκουν στον παίκτη
- `private ArrayList<DestinationCard> destinationCards` : Οι destinationCards που ανήκουν στον παίκτη.
- `private ArrayList<BigCityCard> bigCityCards` : Οι BigCityCards που έχει κερδίσει ο παίκτης.
- `private Player p` : Ο παίκτης στον οποίο ανήκει το χέρι.

Περιέχει επίσης 6 μεθόδους:

- `void addCard(Card c)` : Transformer, προσθέτει την c στην κατάλληλη Array
- `void removeCard(Card c) throws CardNotFoundException` : Transformer, αφαιρεί μια κάρτα από το κατάλληλο Array.
- `public ArrayList<TrainCard> getTrainCards()` : Accessor, επιστρέφει το Array με τα TrainCards.
- `public ArrayList<DestinationCard> getDestinationCards()` : Accessor, επιστρέφει το Array με τα DestinationCards.
- `public ArrayList<BigCityCard> getBigCityCards()` : Accessor, επιστρέφει το Array με τα BigCityCards.
- `public Player getPlayer()` : Accessor, επιστρέφει τον παίκτη - ιδιοκτήτη.
- `+ public int[] getTrainCardColors()` : Accessor, επιστρέφει τις Traincards σε μορφή int array
- `+ public void setDestinationCards(ArrayList<DestinationCard> c)` : Transformer, θέτει τις κάρτες προορισμού του παίκτη.

Ο Constructor δέχεται ως παράμετρο όλες τις περιοχές και τον παίκτη-ιδιοκτήτη. Δέχεται όλα τα πεδία του στον Constructor για να μπορούμε να φορτώσουμε ενα παιχνιδι απο ενα αρχείο.

PlayArea



UML αναπαράσταση της κλάσης PlayArea και η επικοινωνία της με την κλάση Player

+ Η Κλάση PlayArea περιγράφει ένα πεδίο στο παιχνίδι. Η κλάση αυτή δημιουργήθηκε λόγω πολλών ομοιοτήτων μεταξύ του OnTheTrack και του RailYard.

Περιέχει 2 πεδία για τα δεδομένα της:

- `protected int[] colors` : Η αντίστοιχη ποσότητα σε κάρτες σε κάθε χρώμα
- `private Player p` : Ο παίκτης ιδιοκτήτης του OnTheTrack.

Περιέχει επίσης 19 μεθόδους:

- `public int get"Color"()` : Accessor, αυτές οι συναρτήσεις επιστρέφουν τους αριθμούς των αντίστοιχων καρτών στις ράγες
- `public void set"Color"(int amount)` : Transformer, θέτει τους αριθμούς των αντίστοιχων καρτών στις ράγες
- `public Player getPlayer()` : Accessor, επιστρέφει τον ιδιοκτήτη.
- `public boolean isEmpty()` : Accessor, επιστρέφει true αν η περιοχή είναι άδεια

Ο Constructor δεν επιτελεί κάποια έξτρα λειτουργία περα απο ανάθεση μεταβλητών.

RailYard Class **extends PlayArea**

Η Κλάση Railyard περιγράφει το πεδίο στο παιχνίδι από το οποίο μπορεί ο παίκτης να στείλει TrainCards από το χέρι του, με κάποιους κανόνες.

Περιέχει 2 μεθόδους:

- `public void collectFromYard()` : Transformer,στέλνει την πρώτη σειρά από κάρτες στο OnTheTrack.
- `public void placeInRailYard (TrainCard[] cards,Player opponent, int numOfLocoCards) throws InvalidColourException, NotEnoughCardsException` : Τοποθετεί τις κάρτες στο Railyard (Train Robbery included)

Ο Constructor δεν επιτελεί κάποια έξτρα λειτουργία.

OnTheTrack Class **extends PlayArea**

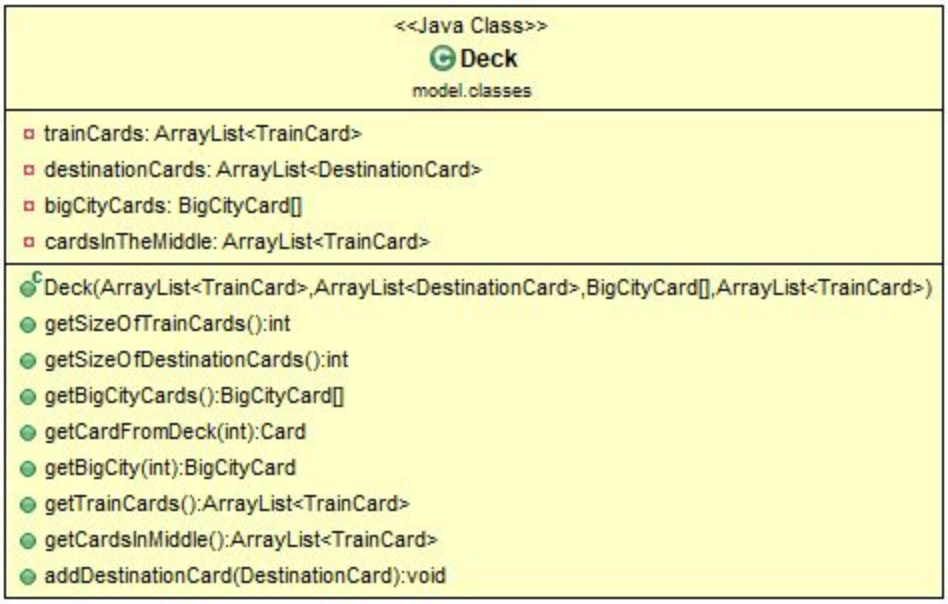
Η Κλάση OnTheTrack περιγράφει το πεδίο στο παιχνίδι από το οποίο μπορεί ο παίκτης να αγοράσει Destination Cards.

Περιέχει 1 μεθόδο:

- `public void buyDestinationCard(DestinationCard destination)` : Αγοράζει μια Destination Card για τον παίκτη - ιδιοκτήτη της περιοχής

Ο Constructor δεν επιτελεί κάποια έξτρα λειτουργία.

Deck Class



UML αναπαράσταση της κλάσης Deck και η επικοινωνία της με την κλάση Player

Η Κλάση Deck αναπαριστά τις κάρτες στις στοίβες στην μέση του τραπεζίου.

Περιέχει 3 πεδία για τα δεδομένα της:

- `private ArrayList<TrainCard> trainCards` : Οι TrainCards που ανήκουν στον παίκτη
- `private ArrayList<DestinationCard> destinationCards` : Οι destinationCards που ανήκουν στον παίκτη.
- `private ArrayList<BigCityCard> bigCityCards` : Οι BigCityCards που έχει κερδίσει ο παίκτης.
- `private ArrayList<TrainCard> cardsInMiddle` : Οι BigCityCards που έχει κερδίσει ο παίκτης.

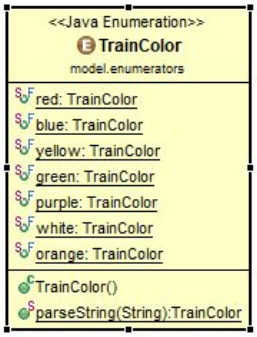
Περιέχει επίσης 8 μεθόδους:

- `Card getCardFromDeck(int deck)` : Transformer, επιστρέφει μια κάρτα από την κατάλληλη στοίβα.
- `+ public int getSizeOfTrainCards()` : Accessor, επιστρέφει το μέγεθος της στοίβας Τραίνων.
- `+ public int getSizeOfDestinationCards()` : Accessor, επιστρέφει το μέγεθος της στοίβας καρτων προορισμού.
- `+ public BigCityCard[] getBigCityCards()` : Accessor, επιστρέφει τις BigCity καρτες
- `+ public BigCityCard getBigCity(int card)` : Accessor, επιστρέφει μια απο τις BigCity καρτες
- `+ public ArrayList<TrainCard> getTrainCards()` : Accessor, επιστρέφει τις Train Cards
- `+ public ArrayList<TrainCard> getCardsInMiddle()` : Accessor, επιστρέφει τις Cards in Middle
- `+ public void addDestinationCard(DestinationCard c)` : Transformer, προσθέτει μια κάρτα στο Destination Card pack.

Ο Constructor δέχεται ως παράμετρο όλες τις περιοχές. Δέχεται όλα τα πεδία του στον Constructor για να μπορούμε να φορτώσουμε ενα παιχνιδι απο ενα αρχείο.

Enumerators

TrainColor Enum



UML αναπαράσταση της κλάσης TrainColor

Ο Enumerator TrainColor χρησιμοποιείται για διευκόλυνση, ωστε

Αποτελείται από 7 στοιχεία, τα οποία είναι τα 7 χρώματα που χρησιμοποιούνται στο παιχνίδι.

Περιέχει επίσης 1 μέθοδο:

- `public static TrainColor parseString(String s)` : Επιστρέφει ενα TrainColor με βάση το String εισαγωγής.

Exceptions

CardNotFoundException

Checked Exception, για οποιαδήποτε περίπτωση μια κάρτα δεν βρεθεί κάπου.

InvalidColourException

Checked Exception, όταν χρησιμοποιηθεί κάποιο χρώμα λάθος.

NoCardsSelectedException

Checked Exception, όταν δεν έχουν επιλεχθεί κάρτες σε παράθυρο διαλόγου (π.χ. Οταν ο παικτης επιλέγει κάρτες να βάλει στο RailYard)

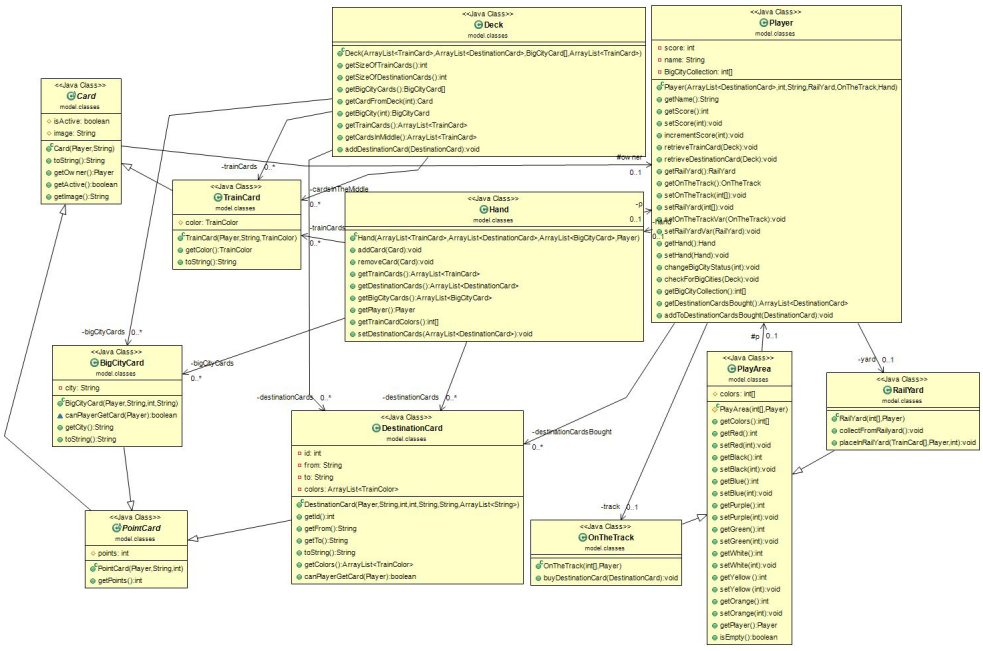
NotEnoughCardsException

Checked Exception, όταν προσπαθήσουμε να πάρουμε κάρτες από κάπου και δεν έχουν μείνει άλλες.

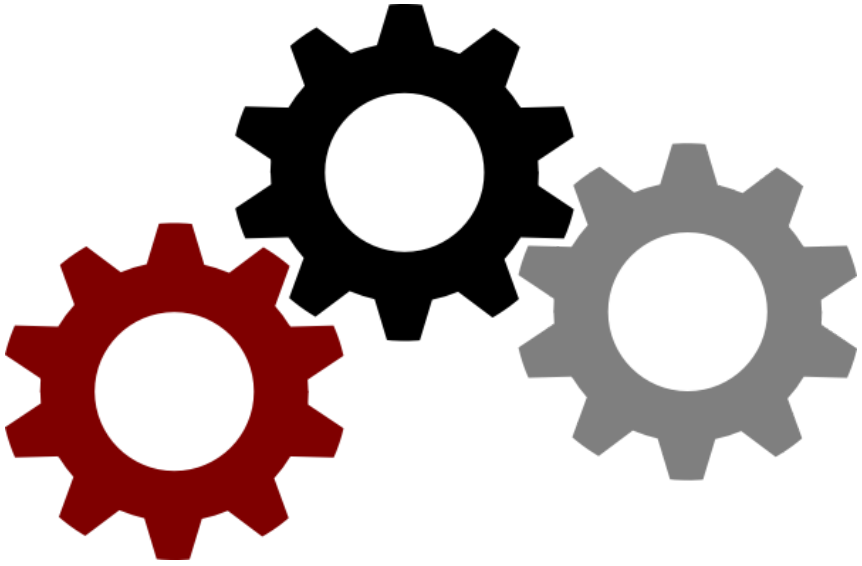
Notes & Overview

Υπήρξαν αρκετές αλλαγές σε σχέση με την φάση προσχεδιασμού, κυρίως προσθήκη Accessors και Transformers, για την διευκόλυνση της επικοινωνίας με άλλες κλάσεις και το Controller.

Ακολουθεί ένα πλήρες διάγραμμα UML του πακέτου Model:



Controller



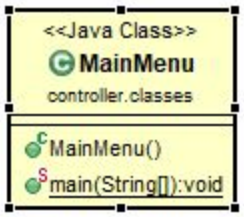
Γεώργιος Μάριος Κοκιάδης < csd3962@csd.uoc.gr >

Υλοποίηση πακέτου Controller

Σε αυτό το πακέτο περιέχονται κλάσεις που "μαζεύουν" τα δεδομένα που αποθηκεύονται με την χρήση κλάσεων απο το Model, τις επεξεργάζονται και ανανεώνουν το GUI με αυτά τα δεδομένα.

Classes in package

MainMenu Class



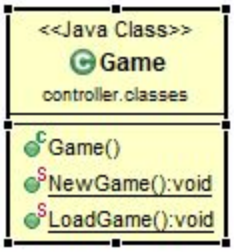
UML αναπαράσταση της κλάσης MainMenu

Η κλάση αυτή αποτελεί το σημείο εισαγωγής στο παιχνίδι. Περιέχει μονάχα την main, στην οποία εμφανίζεται ένα παράθυρο διαλόγου για την δημιουργία ενος νέου παιχνιδιου, η φόρτωσης ενος αποθηκευμένου.

Περιέχει 1 μέθοδο:

- `public static void main(String[] args)` : Σημείο αρχής εκτέλεσης παιχνιδιού

Game Class



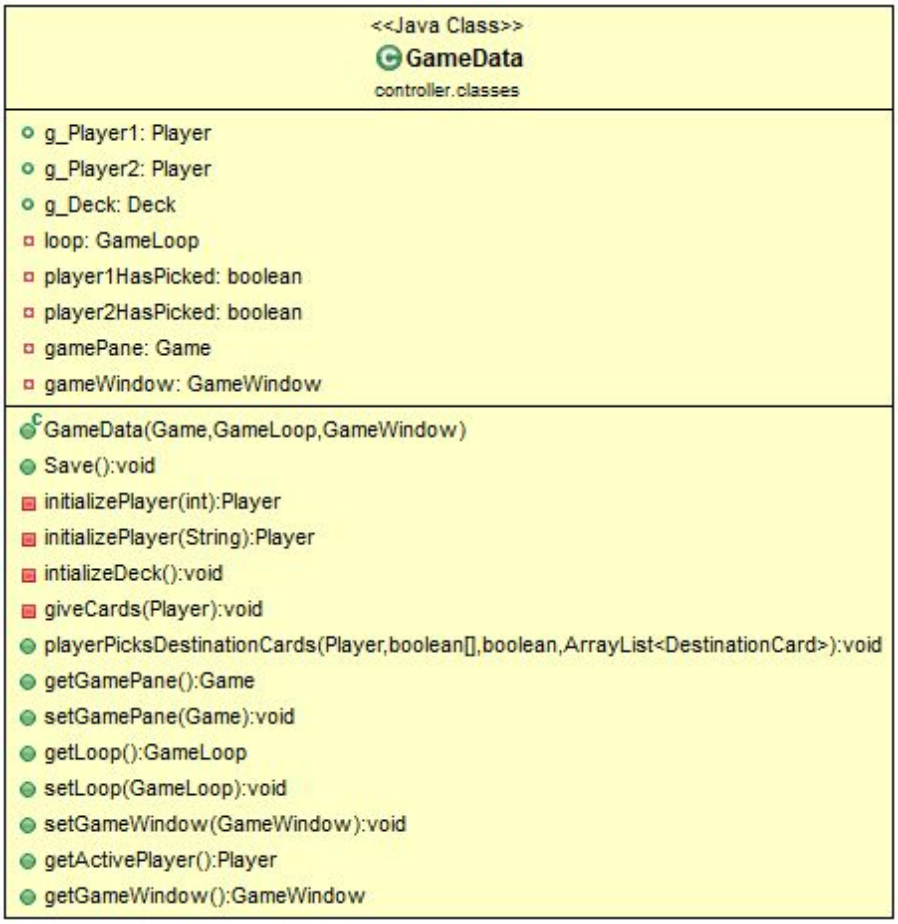
UML αναπαράσταση της κλάσης Game

Στην κλάση Game υπάρχουν 2 static συναρτήσεις, οι οποίες δημιουργούν το κατάλληλο GameData (Περιγράφεται παρακάτω), για να μπει στο GameLoop (Περιγράφεται παρακάτω)

- `public static void NewGame()` : Δημιουργεί ένα νέο παιχνίδι με Default Values.
- `public static void LoadGame()` : Δημιουργεί ένα νέο παιχνίδι με την χρήση αρχείου. Ανοίγει παράθυρο διαλόγου για την επιλογή αρχείου.

Ο Constructor δεν επιτελεί κάποια έξτρα λειτουργία.

GameData Class



UML αναπαράσταση της κλάσης GameData

Η κλάση GameData χρησιμοποιείται για να μπορούμε μέσω του GameLoop (Περιγράφεται παρακάτω) να ελέγξουμε όλα τα δεδομένα του παιχνιδιού.

Περιέχει 7 πεδία για τα δεδομένα της:

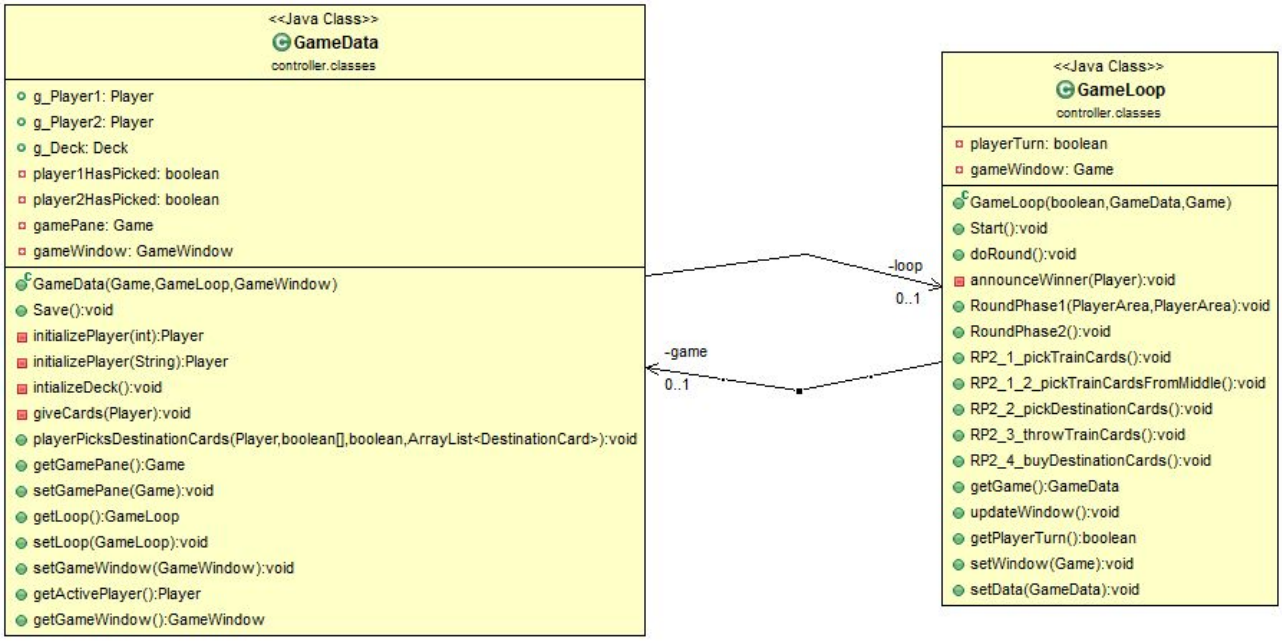
- `public Player g_Player1` : Παίκτης 1
- `public Player g_Player2` : Παίκτης 2
- `public Deck g_Deck` : Το Deck του παιχνιδιού
- `+ private GameLoop loop` : Το loop που αυτό το τρέχει αυτά τα δεδομένα
- `+ private boolean player1HasPicked, player2HasPicked` : Βοηθητικές μεταβλητές για την αρχή του παιχνιδιού, ωστε να ξέρουμε πότε θα ανοίξει το παράθυρο.
- `+ private view.classes.Game gamePane` : Το Pane που περιέχει το UI του παιχνιδιού
- `+ private view.classes.GameWindow gameWindow`: Το παράθυρο του παιχνιδιού

Περιέχει επίσης 14 μεθόδους:

- `public void Save()` : Accessor, Σώζει τα δεδομένα του παιχνιδιού σε ένα αρχείο. Ανοίγει παράθυρο διαλόγου για την επιλογή αρχείου.
- `private void Load()` : Transformer, Φορτώνει δεδομένα του παιχνιδιού από ένα αρχείο. Ανοίγει παράθυρο διαλόγου για την επιλογή αρχείου.
- `private Player initializePlayer(int i)` : Transformer, αρχικοποιεί έναν παίκτη. Με μια default τιμη.
- `private Player initializePlayer(String name)` : Transformer, αρχικοποιεί έναν παίκτη με συγκεκριμένο όνομα.
- `private void intializeDeck()` : Transformer, Αρχικοποιεί το Deck.
- `private void giveCards(Player p)` : Transformer, Μοιράζει κάρτες στον παίκτη p.
- `private void playerPicksDestinationCards(Player p, boolean[] check, ArrayList<DestinationCard> removedCards)` : Transformer, Ανοίγει παράθυρο διαλόγου για να επιλέξει ο παίκτης τις DestinationCards που θέλει να κρατήσει
- `+ public view.classes.Game getGamePane()` : Accessor, επιστρέφει το pane του παιχνιδιού.
- `+ public void setGamePane(view.classes.Game gamePane)` : Transformer, θέτει το pane του παιχνιδιού.
- `+ public GameLoop getLoop()` : Accessor, επιστρέφει το loop του παιχνιδιού.
- `+ public void setLoop(GameLoop loop)` : Transformer, θέτει το loop του παιχνιδιού.
- `+ public void setGameWindow(view.classes.GameWindow gameWindow)` : Transformer, θέτει το Window του παιχνιδιού.
- `+ public Player getActivePlayer()` : Accessor, επιστρέφει τον ενεργό παίκτη, ανάλογα τον γύρο.
- `+ public GameWindow getGameWindow()` : Accessor, επιστρέφει το Window του παιχνιδιού.

Ο Constructor δεχεται ενα gamePane (Το GUI) , ενα loop (το οποίο τρέχει τα δεδομένα αυτά), και ένα παράθυρο παιχνιδιού. Αναθέτει αυτές τις μεταβλητές και αρχικοποιεί όλα τα πεδία του παιχνιδιού. (Παίκτες, deck, GUI)

GameLoop Class



UML αναπαράσταση της κλάσης GameLoop και η σχέση της με την κλάση GameData

Η κλάση GameLoop χρησιμοποιεί τα δεδομενα απο ενα GameData, και τρέχει το παιχνίδι. Επίσης τσεκάρει αν κάποιος παίκτης κέρδισε και τερματίζει το παιχνίδι αν χρειαστεί.

Περιέχει 2 πεδία για τα δεδομένα του:

- `GameData game` : Τα δεδομένα του παιχνιδιού που τρέχουμε.
- `view.classes.Game: gameWindow` : Το παράθυρο του παιχνιδιού

Περιέχει επίσης 14 μεθόδους:

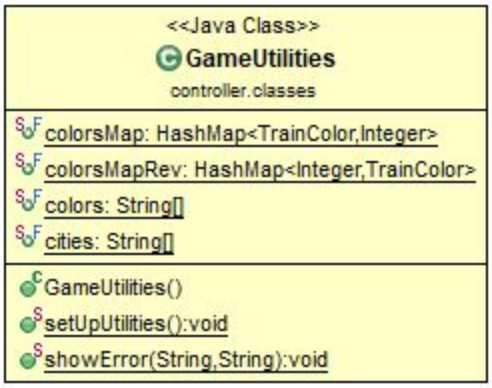
- `public void Start()` : Ξεκινά το Loop του παιχνιδιού.
- `private boolean doRound(Player p)` : Τρέχει έναν γύρο για τον παίκτη p.
- `private Player announceWinner(Player p)` : Ανακοινώνει την νίκη του παίκτη p και τερματίζει το παιχνίδι.
- `private void RoundPhase1(Player p)` : Εκτελεί την πρώτη φάση ενός γύρου για τον παίκτη p.
- `private void RoundPhase2(Player p1, Player p2)` : Εκτελεί την δεύτερη φάση ενός γύρου για τον παίκτη p1, δεδομένου μερικών πληροφοριών από τον παίκτη p2.
- `private void RP2_1_pickTrainCards(Player p)` : Επιλογή 1 της 2ης φάσης ενός γύρου για τον παίκτη p. Μπορεί να πάρει trainCards απο την στοίβα ή την κεντρική

πεντάδα.

- `private void RP2_2_pickDestinationCards(Player p)` : Επιλογή 2 της 2ης φάσης ενός γύρου για τον παίκτη p να επιλέξει κάρτες προορισμού.
- `private void RP2_3_throwTrainCards(Player p1, Player p2)` : Επιλογή 3 2ης φάσης ενός γύρου για τον παίκτη p. Τοποθέτηση καρτών στο RailYard, με την περίπτωση Train Robbery.
- `private void RP2_4_buyDestinationCards(Player p)` : Επιλογή 4 της 2ης φάσης ενός γύρου για τον παίκτη p. Αγορά DestinationCards.
- `+ public GameData getGame()` : Accessor, επιστρέφει το data που διαχειρίζεται αυτό το loop
- `+ public void updateWindow()` : Ανανεώνει το GUI
- `+ public boolean getPlayerTurn()` : Accessor, επιστρέφει true αν είναι σειρά του p1, η false για την σειρά του p2.
- `+ public void setWindow(view.classes.Game window)` : Transformer, θέτει το παράθυρο του παιχνιδιού
- `+ public void setData(GameData data)` : Transformer, θέτει το data που διαχειρίζεται αυτό το loop

Ο Constructor δέχεται τα δεδομένα που θα χρησιμοποιηθούν για αυτό το παιχνίδι, ενα Boolean που αναπαριστα τον ενεργό παίκτη και το pane του GUI.

GameUtilities Class



UML αναπαράσταση της κλάσης GameUtilities

+ Μια απλή κλάση που περιέχει πεδία που χρησιμεύουν σε όλο το πρόγραμμα.

Περιέχει 2 πεδία για τα δεδομένα του:

- `HashMap<TrainColor, Integer> colorsMap` : Αντιστοίχηση του enum TrainColor σε indexes ενος int array.
- `HashMap<Integer, TrainColor> colorsMapRev` : Η ανάποδη αντιστοίχιση απο το colorsMap.
- `String[] color` : Πίνακας που περιέχει τα χρώματα σε Strings στην σωστή σειρά.
- `String[] cities` : Big Cities σε String στην σωστή σειρά.

Περιέχει επίσης 4 μεθόδους:

- `public static void setUpUtilities()` : Αρχικοποιεί τα colorsMap, colorsMapRev, color, cities.
- `public static void ShowError(String error, String title)` : Εμφανίζει ένα error message με τις παραμέτρους που δίνονται.
- `public static ImageIcon resizeCard(String imgpath)` : Επιστρέφει την εικόνα μιας κάρτας σε μορφή ImageIcon, σε fixed μεγεθος 130x77 (HxW).
- `public static ImageIcon resizeCard(String imgpath, int width, int height)` : Επιστρέφει την εικόνα μιας κάρτας σε μορφή ImageIcon, σε fixed μεγεθος height x width.

LoadDestinationCards Class

Περιέχει μόνο μια συνάρτηση η οποία διαβάζει το αρχείο destinationCards.csv και επιστρέφει όλες τις κάρτες προορισμού σε ενα ArrayList.

Various Listener Classes

CheckBoxListener Class

+ Χρησιμοποιείται για την διαχείριση των Checkboxes στην περιοχή του χεριού του παίκτη.

CheckBoxListenerWindowPicker Class

+ Χρησιμοποιείται για την διαχείριση των Checkboxes στο παράθυρο επιλογής καρτών προορισμού.

DestinationCardBuyerListener Class

+ Χρησιμοποιείται για την διαχείριση των κουμπιών στο παράθυρο επιλογής καρτών προορισμού.

LocoColorPickerListener Class

+ Χρησιμοποιείται για την διαχείριση των κουμπιών στο παράθυρο επιλογής χρήσης καρτών Locomotive.

ViewListener Class

+ Ο σημαντικότερος Listener και μεγαλύτερος. Χρησιμοποιείται για την διαχείριση των κουμπιών στο παράθυρο του παιχνιδιού και γενικότερα συμβάλλει σε όλο το Loop του παιχνιδιού

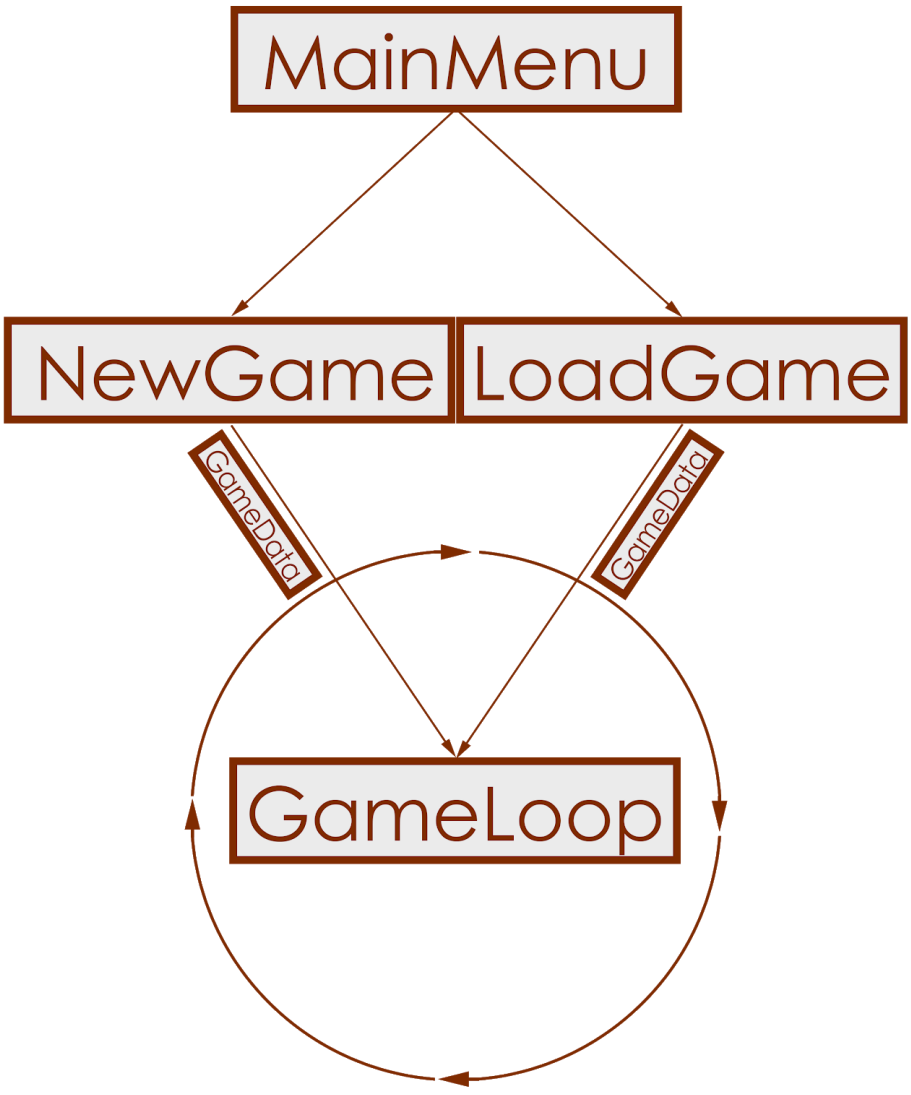
Η βαθύτερη λειτουργία των Listeners φαίνεται καλύτερα παρακάτω, στην εξήγηση του view package.

Game flow in Controller package

Η εκτέλεση, φυσικά, γίνεται μέσα στο Controller. Η main του προγράμματος βρίσκεται στην κλάση MainMenu, απο την οποία θα δούμε πως γίνεται η ροή της εκτέλεσης.

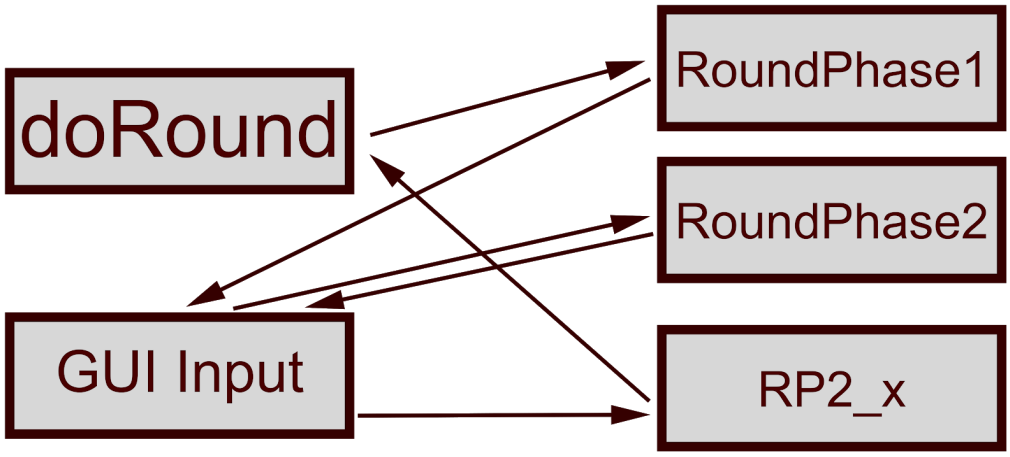
Μέσα στην MainMenu, καλούμε την NewGame ή LoadGame της κλάσης Game. Αυτές δημιουργούν ενα GameData, ανάλογα με το άν φορτώνουμε δεδομένα ή μόλις αρχίσαμε νέο παιχνίδι. Το GameData αυτο, περνάει σε ενα GameLoop το οποίο οργανώνει τους γύρους και την συνθήκη νίκης κάθε παίκτη.

Τα προαναφερθέντα σε διάγραμμα:



Flowchart λειτουργίας παιχνιδιού

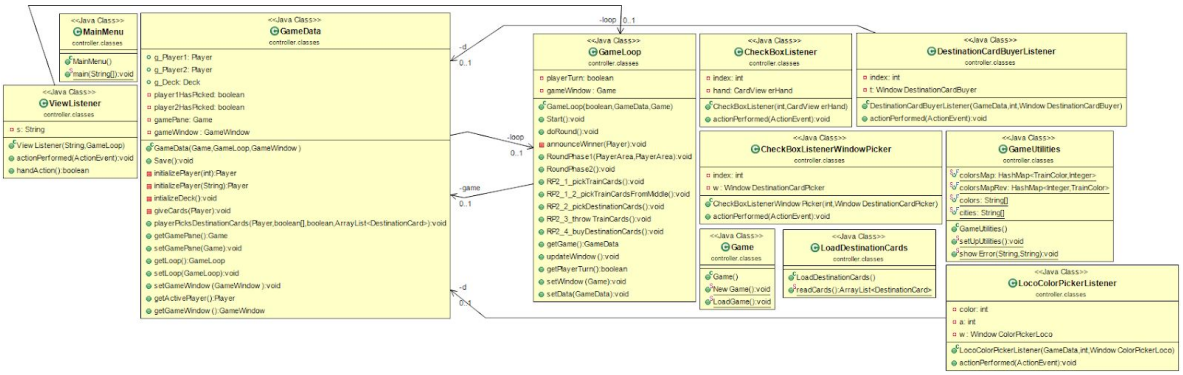
Η GameLoop καλει αρχικά την doRound, η οποία τσεκάρει αν το παιχνίδι έχει τελειώσει (και ανακοινώνει τον αντίστοιχο νικητή). Έπειτα, καλεί την RoundPhase1 για τον κατάλληλο παίκτη. Η RoundPhase1 επιτρέπει στον παίκτη μονο να μεταφέρει κάρτες στο OnTheTrack. Έπειτα, καλείται η RoundPhase2, όπου ελευθερώνονται όλες οι άλλες επιλογές. Το πρόγραμμα έπειτα περιμένει την είσοδο του χρήστη (την επιλογή αγοράς καρτων προορισμού, την μεταφορά από το χέρι στο RailYard, το να πάρει μια κάρτα προορισμού η μια κάρτα τραίνων.). Μετά από την επιλογή του χρήστη, καλείται μια απο τις 4 συναρτήσεις της δεύτερης φάσης, όπου ξεκινάμε πάλι από την αρχή.



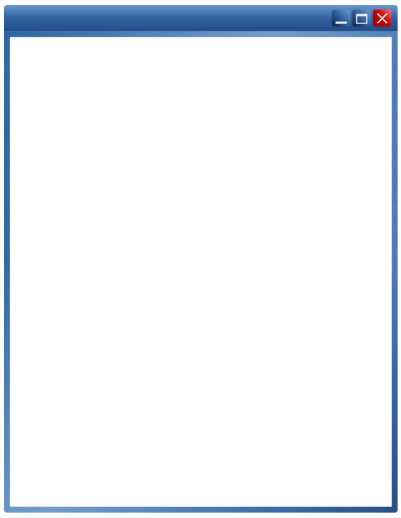
Flowchart του GameLoop

UML Overview

Ακολουθεί πλήρες διάγραμμα UML του package Controller:

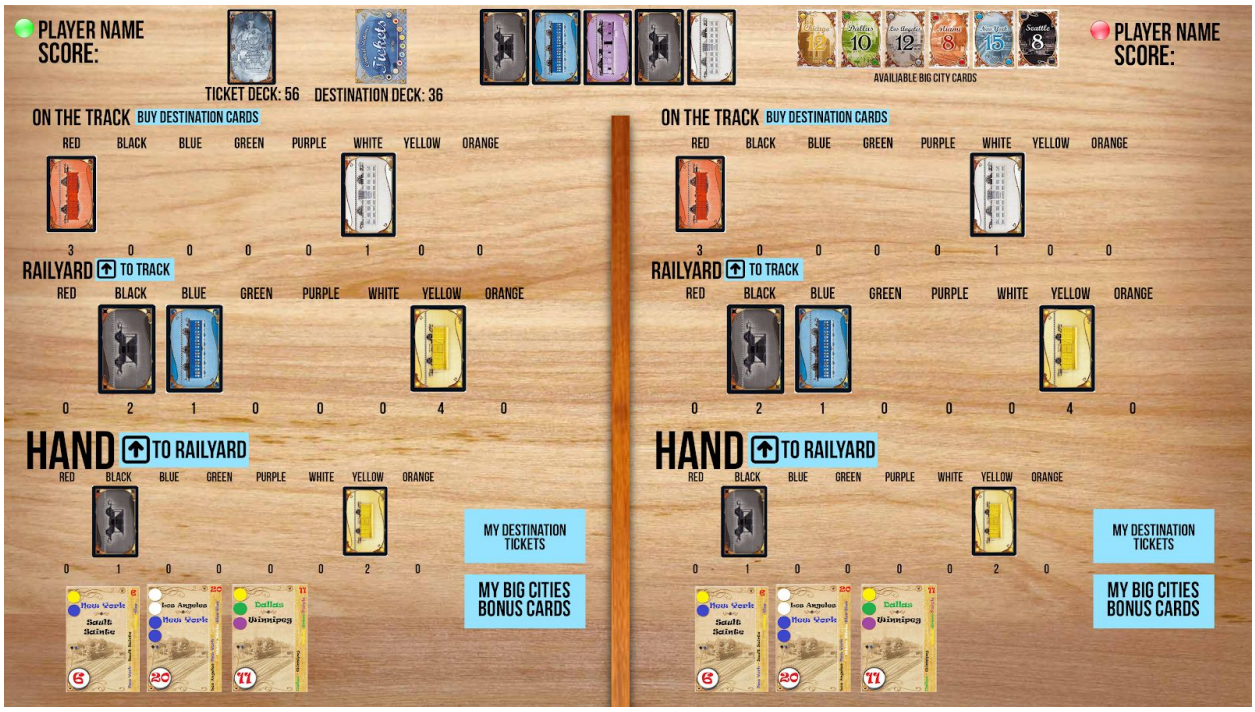


View

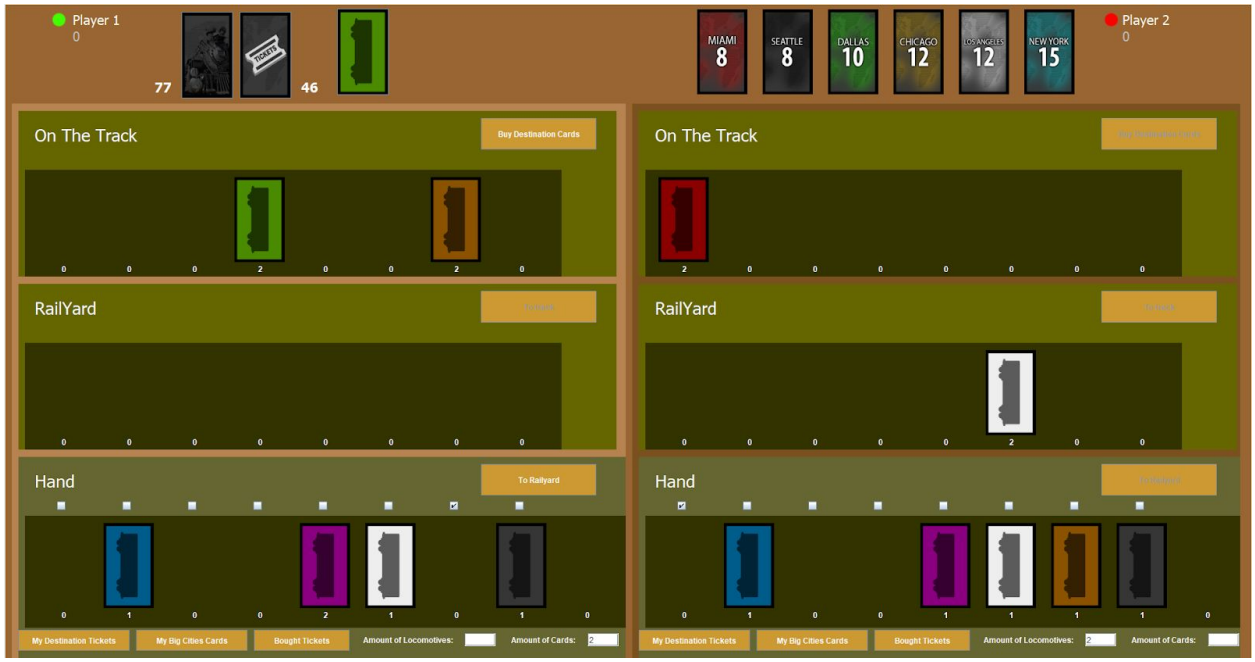


Γεώργιος Μάριος Κοκιάδης < csd3962@csd.uoc.gr >

Concept Art to Reality



Game's original Concept art



Game's Final UI

Γενικά για το GUI

Σε γενικές γραμμές, ο κύριος στόχος μου στην υλοποίηση αυτή ήταν η δημιουργία ενός GUI που επαναχρησιμοποιεί λίγο κώδικα, για πολλές χρήσεις. Αυτός ο στόχος τέθηκε κυρίως λόγω πίεσης χρόνου, αλλα προσέφερε ταυτόχρονα μια πολύ Uniform και καθαρή εμφάνιση, όπως και καλή λειτουργικότητα.

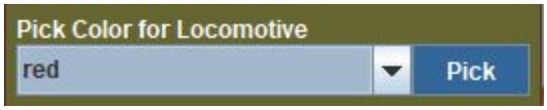
Αντικαταστάθηκαν επίσης τα Sprites των καρτών για να ταιριάξουν με τα χρώματα του υπόλοιπου GUI. Απο την προτεινόμενη υλοποίηση επίσης άλλαξε η θέση της περιοχής του κάθε παίκτη σε δεξιά και αριστερά, με το «κέντρο» του τραπεζιού να μεταφέρεται στην κορυφή. *

Υπάρχουν επίσης διάφορα Pop-ups:

Pop-up για την προβολή καρτών προορισμού που έχει ο παίκτης

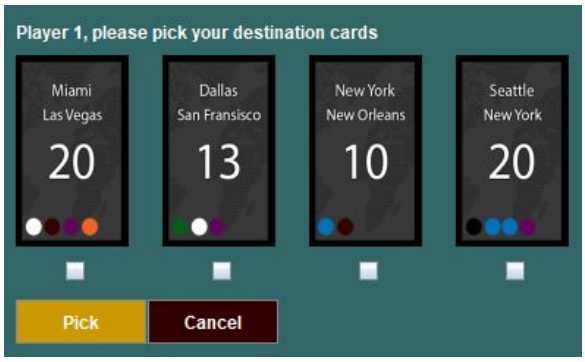
Pop-up για την προβολή καρτών προορισμού που έχει αγοράσει ο παίκτης.

Pop-up για την προβολή BigCity Cards που έχει κερδίσει ο παίκτης



Pop-up για την επιλογή του χρώματος που θα αναπαριστά μια Locomotive card αν παίζουμε μόνο Locomotive cards.

* Το GUI έχει αλλάξει ελαφρώς μετά από την συγγραφή της αναφοράς αυτής



Pop-up για την επιλογή DestinationCard απο την στοίβα

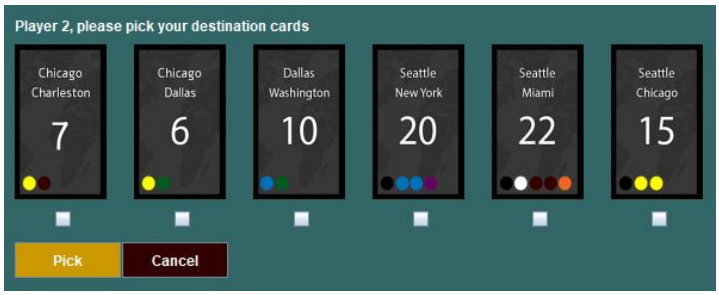
Pop-up για την αγορά DestinationCard

Ροή του παιχνιδιού (Gameplay)

Η ροή του παιχνιδιού, έχει ως εξής:

Με την εκκίνηση του παιχνιδιού, ο κάθε παίκτης επιλέγει το όνομα του:

Δεν είναι αναγκαστική η είσοδος κάποιου ονόματος, σε περίπτωση μη εισόδου δίδεται Default όνομα. Έπειτα, ζητείται από κάθε παίκτη να επιλέξει τις Destination Cards που θέλει να κρατήσει:



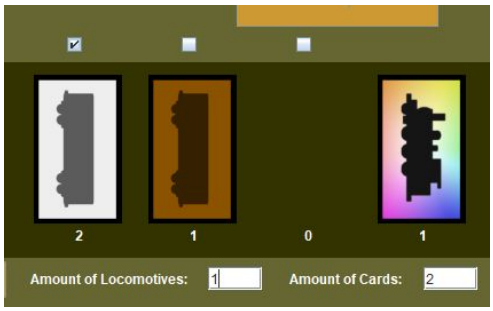
Έπειτα, με την εκκίνηση του παιχνιδιού ενας παίκτης επιλέγεται τυχαία για να ξεκινήσει πρώτος. Έτσι ξεκινάει το Loop που περιγράφηκε στο κεφάλαιο για το Controller. Ο παίκτης έχει την επιλογή μόνο να μετακινήσει κάρτες από το RailYard στο OnTheTrack.

Έπειτα, τα υπόλοιπα κουμπιά ελευθερώνονται και ο παίκτης μπορεί να επιλέξει τί θα κάνει αυτόν τον γύρο. Μπορει:

α) Να παίξει κάρτες Τραίνων.

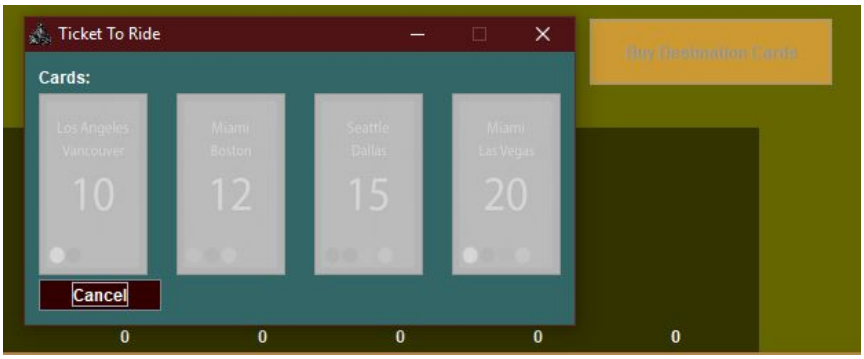
i) Να παίξει μόνο κάρτες τραίνων

ii) Να παίξει κάρτες τραίνων και κάρτες Locomotive

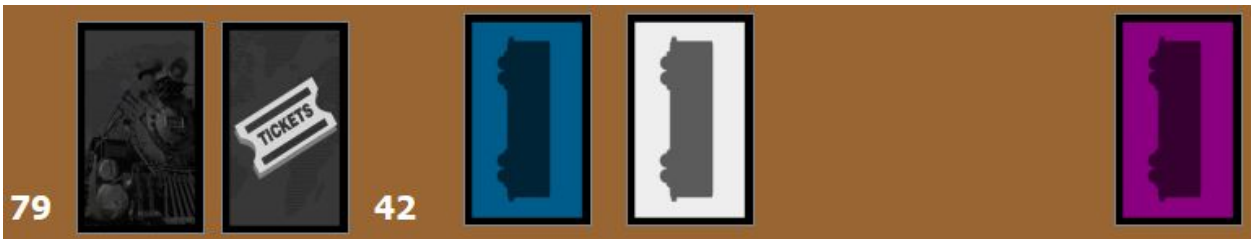


iii) Να παίξει μόνο κάρτες Locomotive

β) Να αγοράσει κάρτες προορισμού



γ) Να πάρει κάρτες προορισμού ή κάρτες τραίνων (είτε από την στοίβα ή την μεση!)



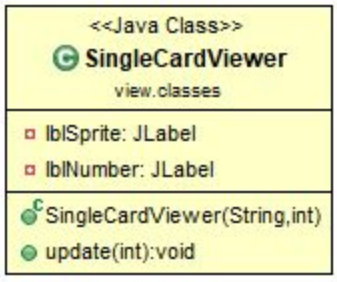
Όταν μια από αυτές τις πράξεις ολοκληρωθεί, ξεκινά ο γύρος για τον επόμενο παίκτη.

Φυσικά, ο κάθε παίκτης μπορεί να δει τις κάρτες προορισμού που εχει, που εχει αγοράσει όπως και τις κερδισμένες BigCity Cards του, μέσω των κουμπιών στον πάτο της οθόνης.

Ας δούμε λοιπόν, πέρα από το Gameplay, πώς απο μικρές κλάσεις συναρμολογείται ολόκληρο το GUI. Θα Ξεκινήσουμε απο τις κλάσεις για το κεντρικό παράθυρο του παιχνιδιού, καταλήγοντας στα Pop-ups.

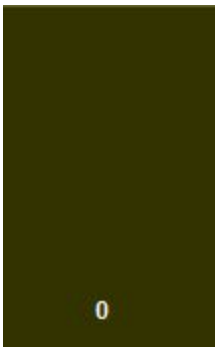
Classes in Package

SingleCardViewer Class **extends JPanel**



UML Αναπαράσταση της κλάσης SingleCardViewer

SingleCardViewer με 1 κάρτα



SingleCardViewer με καμία κάρτα

+ Η κλάση SingleCardViewer αποτελεί την βάση για την υλοποίηση της κλάσης CardViewer που περιγράφηκε και στην αναφορά της Α' Φασης αλλα και περιγράφεται παρακάτω. Περιέχει μια εικόνα της κάρτας που αντιστοιχεί στην περιοχή, οπως και τον αριθμό των καρτών που υπάρχουν. Αυτόματα εμφανίζει η εξαφανίζει το sprite αν οι κάρτες είναι 0.

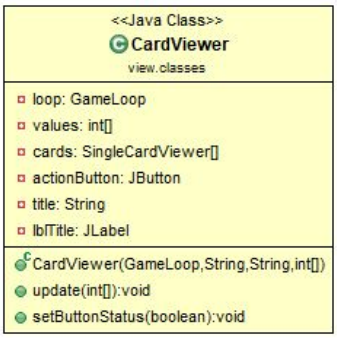
Περιέχει 2 πεδία για τα δεδομένα του:

- `JLabel lblSprite` : Η εικόνα της κάρτας
- `private JLabel lblNumber` : JLabel για τον αριθμό των καρτών.

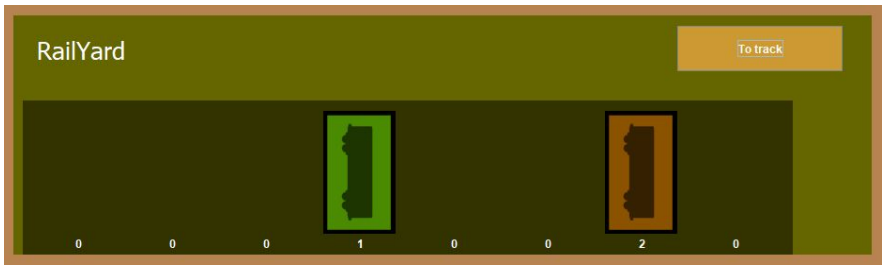
Περιέχει επίσης 1 μέθοδο:

- `public void update()` : Ανανεώνει τα περιεχόμενα.

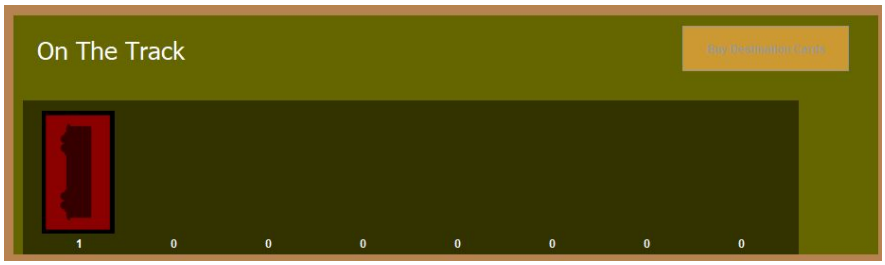
CardViewer Class **extends JPanel**



UML Αναπαράσταση της κλάσης CardViewer



CardViewer που αναπαριστά ένα Railyard, με το κουμπί να είναι Clickable.



CardViewer που αναπαριστά ένα OnTheTrack, με το κουμπί να είναι Unclickable.

+ Η κλάση CardViewer αποτελεί την βάση ολόκληρης της αναπαράστασης της περιοχής του παίκτη. Αποτελείται απο 8 SingleCardViewers, εναν τίτλο και ενα κουμπί δράσης. Τα κουμπιά όλα είναι συνδεδεμένα με τον ViewListener, ο οποίος δέχεται ως είσοδο τον τίτλο της περιοχής και κάνει την σωστή λειτουργία.

Περιέχει 6 πεδία για τα δεδομένα του:

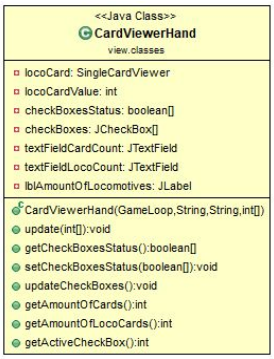
- `private GameLoop loop` : Loop που τρέχει εκείνη την στιγμή
- `private int [] values` : Τιμές που περιέχονται στα SingleCardViewers
- `private SingleCardViewer[] cards` : Τα SingleCardViewers που αποτελούν αυτό το CardViewer
- `private JButton actionButton` : Το κουμπί δράσης
- `private String title` : Ο τίτλος της περιοχής
- `private JLabel lblTitle` : Ο τίτλος της περιοχής.

Περιέχει επίσης 2 μεθόδους:

- `public void update(int[] numbers)` : Ανανεώνει τα περιεχόμενα.
- `public void setButtonStatus(boolean s)` : Transformer, Κανει το κουμπί δράσης Clickable η UnClickable.

Ο Constructor δέχεται το loop του παιχνιδιού, τον τίτλο της περιοχής, το περιεχόμενο του κουμπιού και ενα int[] που περιεχει το περιεχόμενο σε κάρτες.

CardViewerHand Class **extends CardViewer**



UML

Αναπαράσταση της κλάσης CardViewerHand

CardViewerHand με την επιλογή μιας άσπρης κάρτας και μιας καρτας Locomotive

+ Η κλάση CardViewerHand αποτελεί μια εξελιγμένη έκδοση της CardViewer, προσθέτοντας την ικανότητα ο παίκτης να επιλέξει κάρτες απο το πεδίο, σύμφωνα με τους κανόνες παιξίματος (Οι τρόποι με τους οποίους μπορεί να παίξει περιγράφηκαν παραπάνω) . Προσθέτει επιπρόσθετα ενα SingleCardViewer για την προβολή καρτών Locomotive. Τα Checkboxes επικοινωνούν με το CheckBoxListener για να σιγουρευτούμε ότι ή ένα checkbox είναι τικαρισμένο, ή κανενα.

Περιέχει 7 πεδία για τα δεδομένα του:

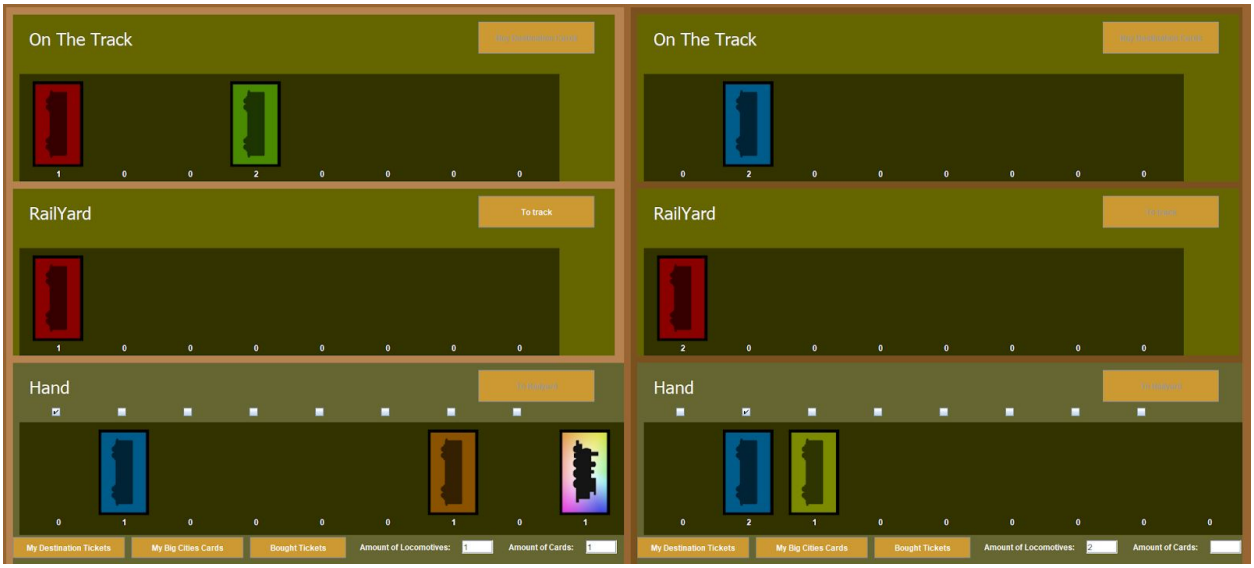
- `private SingleCardViewer locoCard` : SingleCardViewer για την Locomotive καρτα
- `private int locoCardValue` : Τιμή των Locomotive card
- `private boolean[] checkBoxesStatus` : Η κατάσταση στα οποία βρίσκονται τωρα τα checkboxes
- `private JCheckBox[] checkBoxes` : Τα Checkboxes
- `private JTextField textFieldCardCount, textFieldLocoCount` : Είσοδος απο τον χρήστη που αφορά την ποσότητα καρτών που θέλει να παίξει ο παίκτης
- `private JLabel lblAmountOfLocomotives` : Τιμή των Locomotive card

Περιέχει επίσης 7 μεθόδους:

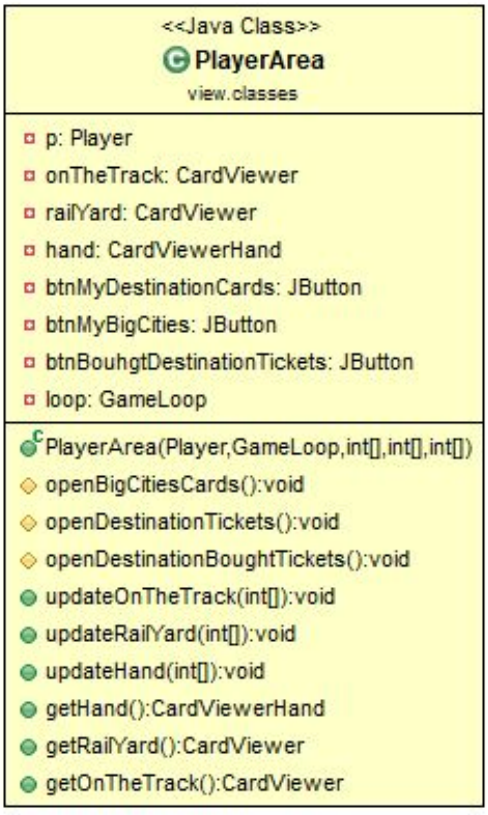
- `public void update(int[] numbers)` : Ανανεώνει τα περιεχόμενα.
- `public boolean[] getCheckBoxesStatus()` : Επιστρέφει την κατάσταση των checkboxes.
- `public void setCheckBoxesStatus(boolean[] status)` : Θέτει την κατάσταση των checkboxes.
- `public void updateCheckBoxes()` : Θέτει την κατάσταση των checkboxes στο GUI
- `public int getAmountOfCards()` : Επιστρέφει πόσες κάρτες έχει εισάγει ο χρήστης ως Input
- `public int getAmountOfLocoCards()` : Επιστρέφει πόσες κάρτες Locomotive έχει εισάγει ο χρήστης ως Input
- `public int getActiveCheckBox()` : Επιστρέφει το Index του επιλεγμένου Checkbox

Ο Constructor δεν επιτελεί κάποια έξτρα λειτουργία.

PlayerArea Class extends JPanel



PlayerArea ενεργο (αριστερά) και PlayeArea ανενεργό (δεξιά). Αν είναι ενεργό ή όχι κρίνεται απο την σειρά του παίκτη



UML Αναπαράσταση της κλάσης PlayerArea

+ Η κλάση PlayerArea "μαζεύει" τις προαναφερόμενες κλάσεις για να συναρμολογήσει μια ολόκληρη περιοχή ενός παίκτη.

Περιέχει 8 πεδία για τα δεδομένα της:

- `private Player p` : Ο παίκτης που παίζει σε αυτήν την περιοχή
- `private CardViewer onTheTrack` : Περιοχή OnTheTrack
- `private CardViewer railYard` : Περιοχή RailYard
- `private CardViewerHand hand` : Περιοχή Hand
- `private JButton btnMyDestinationCards` : Κουμπί που ανοίγει παράθυρο διαλόγου που δείχνει τις κάρτες προορισμού του παίκτη
- `private JButton btnMyBigCities` : Κουμπί που ανοίγει παράθυρο διαλόγου που δείχνει τις κερδισμένες κάρτες BigCity του παίκτη
- `private JButton btnBouhgtDestinationTickets` : Κουμπί που ανοίγει παράθυρο διαλόγου που δείχνει τις κάρτες προορισμού που εχει αγοράσει ο παίκτης

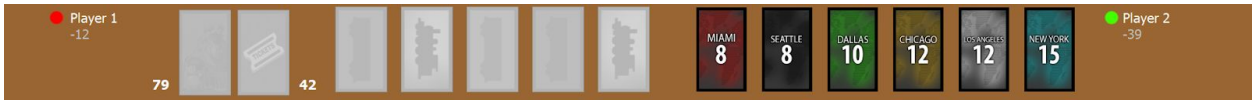
- `private GameLoop loop` : Το loop του παιχνιδιού

Περιέχει επίσης 9 μεθόδους:

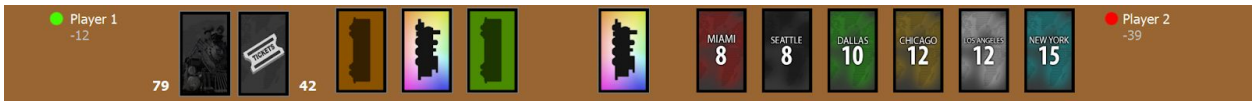
- `protected void openBigCitiesCards()` : Ανανεώνει τα περιεχόμενα.
- `protected void openDestinationTickets()` : Επιστρέφει την κατάσταση των checkboxes.
- `protected void openDestinationBoughtTickets()` : Θέτει την κατάσταση των checkboxes στο GUI
- `public void updateOnTheTrack(int[] values)` : Επιστρέφει πόσες κάρτες έχει εισάγει ο χρήστης ως Input
- `public void updateRailYard(int[] values)`: Επιστρέφει πόσες κάρτες Locomotive έχει εισάγει ο χρήστης ως Input
- `public void updateHand(int[] values)` : Επιστρέφει το Index του επιλεγμένου Checkbox
- `public CardViewerHand getHand()` : Επιστρέφει την περιοχή hand του GUI
- `public CardViewer getRailYard()` : Επιστρέφει την περιοχή Railyard του GUI
- `public CardViewer getOnTheTrack()` : Επιστρέφει την περιοχή OnTheTrack του GUI

Ο Constructor δέχεται τον παίκτη, το Gameloop και Int arrays για την αρχικοποίηση των περιοχών

Deck Class **extends JPanel**



Deck Class με ενεργό παίκτη τον παίκτη 2 και ανενεργές τις επιλογές απόκτησης καρτών



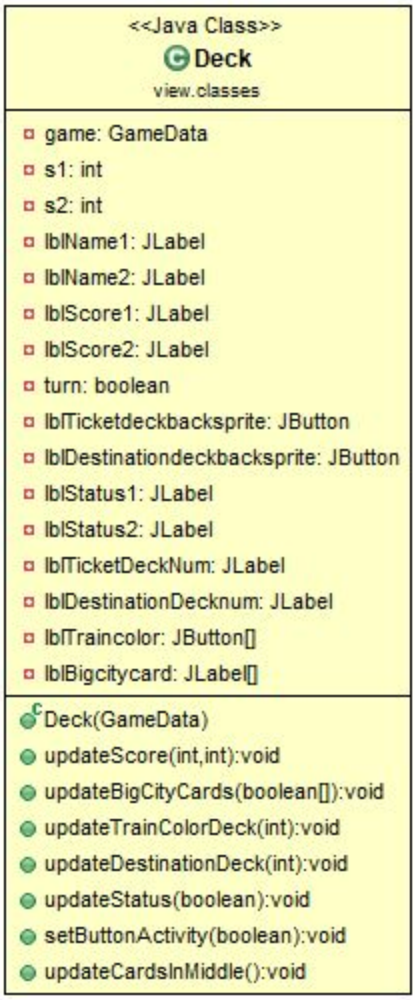
Deck Class με ενεργό παίκτη τον παίκτη 1 και ενεργές τις επιλογές απόκτησης καρτών

+ Η κλάση Deck αναπαριστά τις κάρτες που είναι κοινές στους δύο παίκτες.

Περιέχει 16 πεδία για τα δεδομένα της:

- `private GameData game` : Τα δεδομένα του παιχνιδιού
- `private int s1, s2` : Τα σκόρ των παικτών
- `private JLabel lblName1, lblName2` : Τα ονόματα των παικτών
- `private JLabel lblScore1, lblScore2` : Τα σκόρ των παικτών
- `private boolean turn` : Ο γύρος
- `private JButton lblTicketdeckbacksprite, lblDestinationdeckbacksprite` : Η στοίβες με τις κάρτες

- `private JLabel lblStatus1, lblStatus2` : Ενδείξεις για το αν είναι ο γύρος ενός παίκτη
- `private JLabel lblTicketDeckNum, lblDestinationDecknum` : Αριθμος καρτών στις στοίβες απο κάρτες
- `private JButton[] lblTraincolor` : Κάρτες TrainCard στην μέση
- `private JLabel[] lblBigcitycard` : Κάρτες BigCity



UML Αναπαράσταση της κλάσης Deck

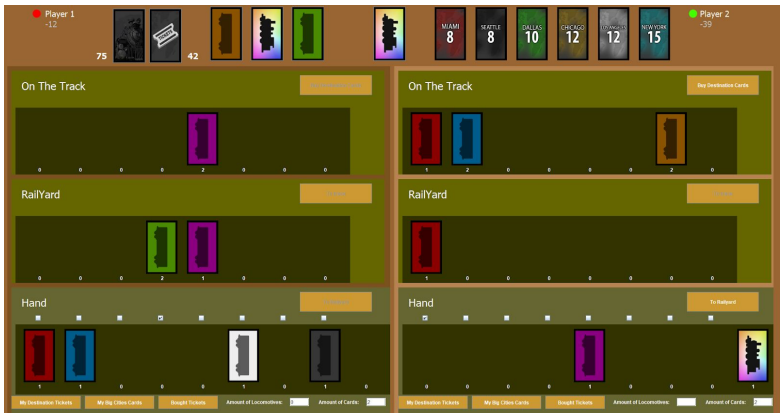
Περιέχει επίσης 7 μεθόδους:

- `public void updateScore(int s1, int s2)` : Ανανεώνει το σκόρ
- `public void updateBigCityCards(boolean[] values)` : Ανανεώνει τις ενεργές BigCityCards
- `public void updateTrainColorDeck(int v)` : Ανανεώνει τον αριθμό καρτών στην στοίβα TrainCards
- `public void updateDestinationDeck(int v)` : Ανανεώνει τον αριθμό καρτών στην στοίβα DestinationCards
- `public void updateStatus(boolean t)` : Ανανεώνει τις ενδείξεις των γύρων
- `public void setButtonActivity(boolean status)` : Θέτει την δυνατότητα απόκτησης καρτών
- `public void updateCardsInMiddle()` : Ανανεώνει τις κάρτες στην μέση

Ο Constructor δέχεται ένα GameData το οποίο χρησιμοποιεί για να αρχικοποιήσει τα πεδία του

Game Class **extends JPanel**

UML Αναπαράσταση της κλάσης Game



Η κλάση Game

+ Η κλάση Game περιέχει όλες τις προαναφερόμενες κλάσεις για να ολοκληρώσει το κεντρικό GUI.

Περιέχει 3 πεδία για τα δεδομένα της:

- `private PlayerArea player1Area, Player2Area` : Οι περιοχή του κάθε παίκτη
- `private Deck viewDeck` : Το Deck του παιχνιδιού

Περιέχει επίσης 4 μεθόδους:

- `public getPlayer1Area()` : Επιστρέφει την περιοχή του παίκτη 1
- `public getPlayer2Area()` : Επιστρέφει την περιοχή του παίκτη 2
- `public getDeck()` : Επιστρέφει το Deck
- `public updateColors(boolean turn)` : Ανανεώνει τα χρώματα των περιοχών (Η περιοχή του παίκτη που έχει σειρά ειναι πιο φωτεινή)

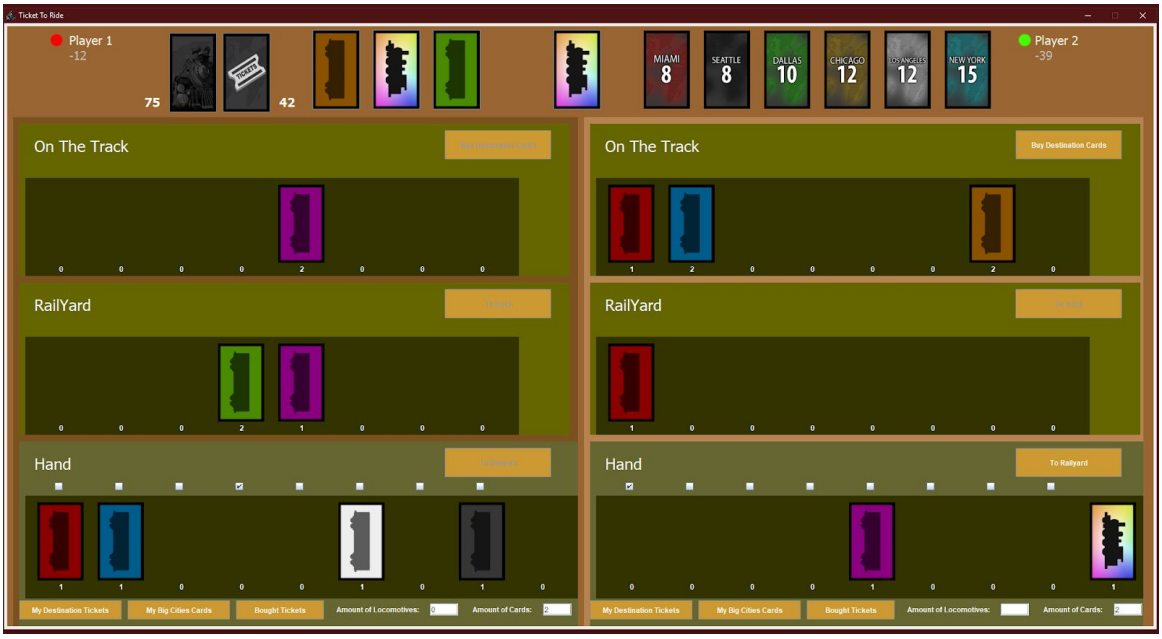
Ο Constructor Δέχεται το Loop του παιχνιδιού, 2 παίκτες και το Deck στην μνήμη

Window Classes

Αυτές οι κλάσεις περιγράφουν παράθυρα που εμφανίζονται στο παιχνίδι. Δεν έχουν αξιοσημείωτες μεθόδους , οπότε μπορούμε να τις περάσουμε συνοπτικά.

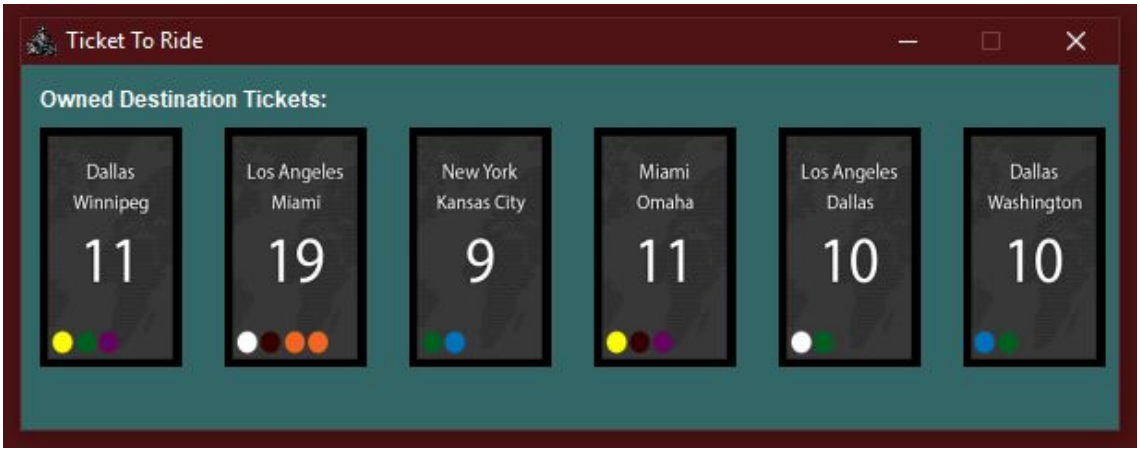
GameWindow

Το κεντρικό παράθυρο του παιχνιδιού.



WindowCardViewer

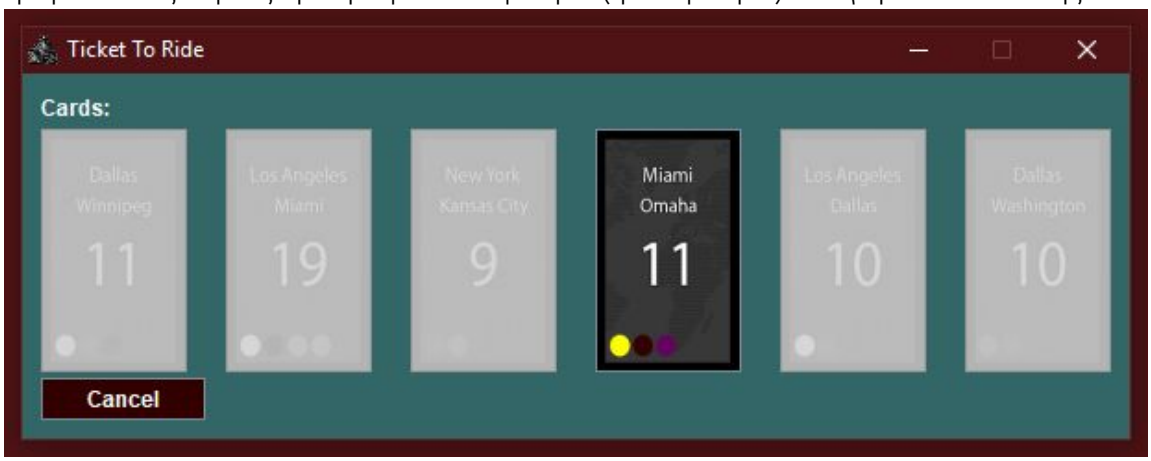
Προβάλει μια συλλογή απο κάρτες (Δέχεται ArrayList απο κάρτες)



Γεώργιος Μάριος Κοκιάδης < csd3962@csd.uoc.gr >

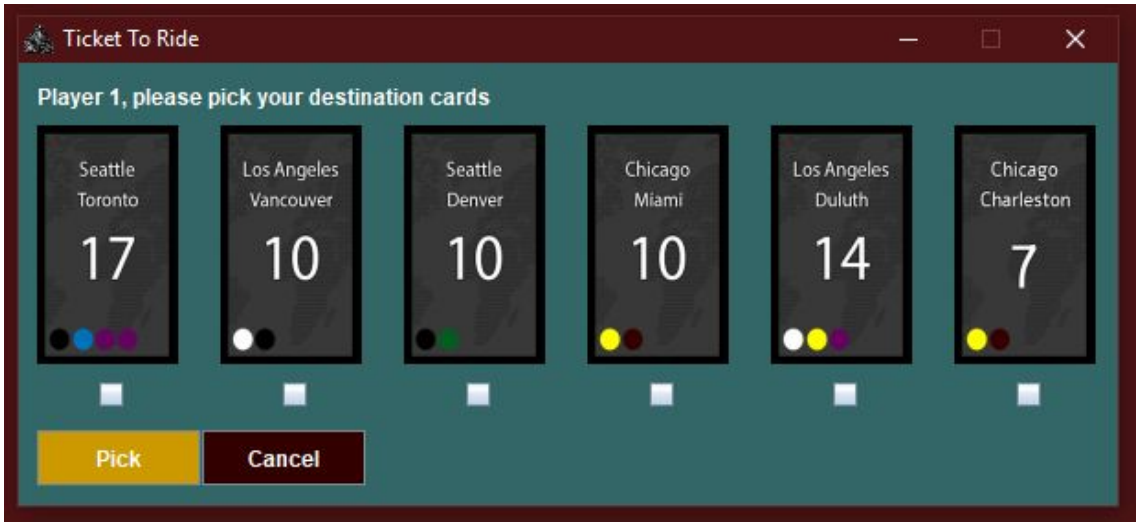
WindowDestinationCardBuyer

Προβάλλει τις κάρτες προορισμού που μπορεί (ή δεν μπορεί) να αγοράσει ο παίκτης



WindowDestinationCardPicker

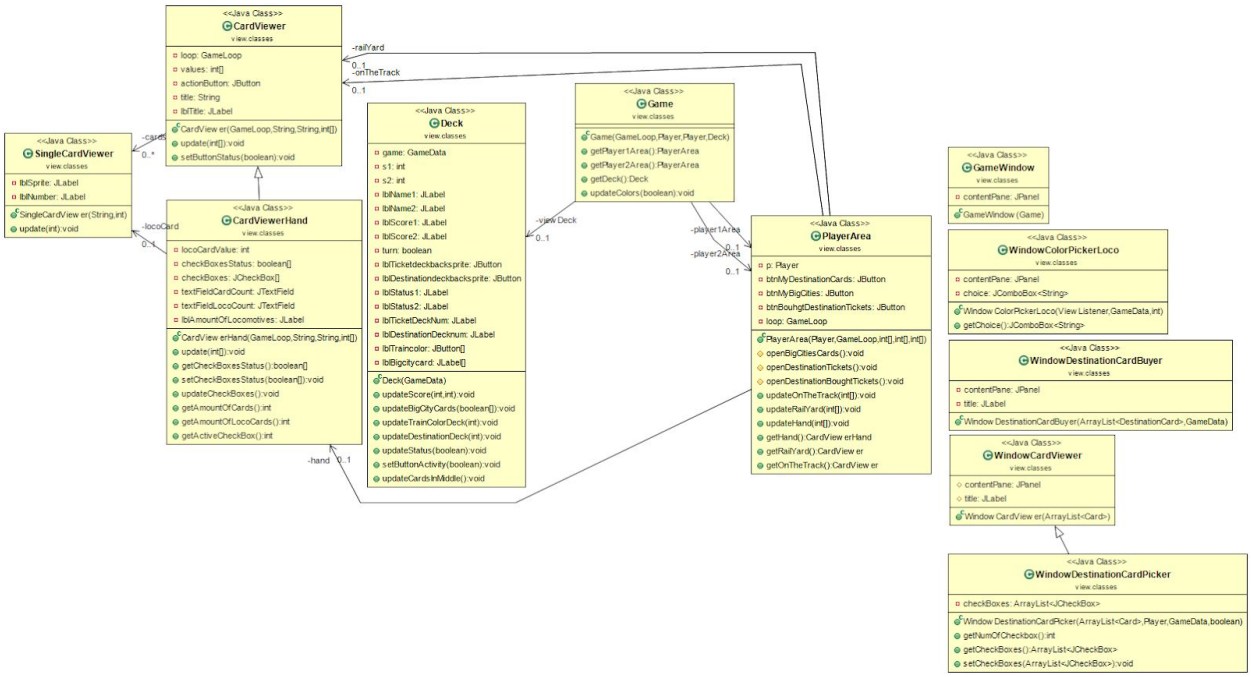
Προβάλλει κάρτες προορισμού προς επιλογή



WindowLocoCardPicker

Χρησιμοποιείται για την επιλογή του χρώματος που θα συμβολίζει μια Locomotive card

UML Overview



Save / Load Bonus

Δυστυχώς, λόγω περιορισμού χρόνου, δεν πρόλαβα να κάνω την πλήρης υλοποίηση του Save και Load. Παρόλα αυτά, κατα την διάρκεια της σχεδίασης και υλοποίησης κράτησα το σκεπτικό ότι θα το συμπεριλάβω. Έτσι, στην κλάση controller.classes.Game υπάρχει ενα function που λέγεται LoadGame, με την βοήθεια της οποίας θα διαβαζα από ένα αρχείο int[] για να γεμίσω τις περιοχές των παικτών, Strings για τα ονοματα τους, Ints για σκορ, αριθμο καρτων προορισμου και τραίνων, όπως και την κατάσταση των καρτών BigCity αλλα και τον boolean με την πληροφορία για το ποιός παίκτης έχει σειρά. Με τα Int[] θα μπορουσα να δημιουργήσω ολόκληρη την στοίβα απο Traincards. Απο ένα άλλο αρχείο, θα διάβαζα τις κάρτες προορισμού (ξανα με την βοήθεια του controller.classes.LoadDestinationCards) για τον κάθε παίκτη και την στοίβα. Θα αρχικοποιούσα (Με τους Constructors) σωστά το GameData και GameLoop και θα έτρεχα κανονικά το παιχνίδι. Save θα γινόταν μέσω ενος κουμπιού στο GUI που θα έτρεχε την Save() στο controller.classes.GameData, η οποία θα δημιουργούσε τα προαναφερθέντα αρχεία.

Full UML Diagram