Πανεπιστήμιο Κρήτης –Τμήμα Επιστήμης Υπολογιστών

ΗΥ252– Αντικειμενοστρεφής Προγραμματισμός

Διδάσκων: Ι. Τζίτζικας

Χειμερινό Εξάμηνο 2018-2019

HY252 Project

PHASE 2

Αντώνης Ντρουμπογιάννης

ΑΜ : 4014

21 Δεκεμβρίου 2018

Περιεχόμενα

1. Εισαγωγή 2
2. Η Σχεδίαση και οι Κλάσεις του Πακέτου Model 2
3. Η Σχεδίαση και οι Κλάσεις του Πακέτου Controller 8
4. Η Σχεδίαση και οι Κλάσεις του Πακέτου View 10
5. Η Αλληλεπίδραση μεταξύ των κλάσεων – Διαγράμματα UML 12
6. Λειτουργικότητα (Β Φάση) 12
7. Συμπεράσματα 12

1. Εισαγωγή

Η υλοποιήση της εργασιάς βασίζεται πάνω στο ευρέως διαδεδομένο μοντέλο MVC ή αλλιώς Model-View-Controller. Στο μοντέλο αυτό, η παρουσίαση της πληροφορίας (View) διαχωρίζεται από αυτήν που αποθηκεύεται στο σύστημα (Model) , ενώ ο Controller αποτελεί τον συνδετικό κρίκο αυτών των δυο. Παρακάτω θα αναλύσουμε κυρίως το Model και το Controller, ενω θα αναφέρουμε συνοπτικά στη σχεδίαση του View.

2. Η Σχεδίαση και οι Κλάσεις του Πακέτου Model

Σε αυτό το πακέτο περιέχεται η κλαση Card, οι κλάσεις PointsCard και Train Card που την κληρονομούν, καθως και οι κλάσεις που κληρωνομούν την PointsCard (DestinationCard, BigCitiesCard). Επιπλέον περιέχεται μία κλάση enum CardColor, μία κλάση Player, μία κλάση Deck, μία κλάση On-the-track και τέλος μία κλαση RailYard .

2.1 | Κλάση Card και οι κλάσεις που την κληρονομούν

Card

Αυτή η κλάση περιέχει τα βασικά attributes και τις μεθόδους μια κάρτα χωρίς να χρειάζεται να γνωρίζουμε το είδος της. Πιο αναλυτικά :

Τα attributes :

1) private player owner; (Ο κάτοχος της κάρτας)
2) private Boolean isActive; (Boolean μεταβλητή για το αν μία κάρτα ειναι ενεργή ή όχι)
3) private String image; (Ένα αλφαριθμητικό που θα περιέχει το path της εικόνας της κάρτας)

Οι μέθοδοι της κλάσης :

1) public void set_owner(Player owner); (Ορίζει τον κάτοχο της κάρτας)
2) public Player get_owner(); (Επιστρέφει το κάτοχο της κάρτας)
3) public void set_image(String image); (Ορίζει το path της εικόνας της κάρτας)
4) public String get_image(); (Επιστρέφει το path της εικόνας της κάρτας)
5) public Boolean is_Active(); (Επιστρέφει αν η κάρτα ειναι ενεργή ή όχι)

PointsCard

Αυτή η κλάση κληρονομεί τη κλάση Card και την εξειδικεύει για τις κάρτες οι οποίες σχετίζονται με πόντους. Πιο αναλυτικά τα επιπλέον attributes και μεθόδοι της κλάσης εκτός αυτά της Card είναι:

Τα attributes :

1) private final int points; (Οι πόντοι της κάρτας, οι οποίοι ειναι final καθώς δεν αλλάζουν στη διάρκεια του παιχνιδιού)

Οι μέθοδοι της κλάσης :

1) public int get_points(); (Επιστρέφει τους πόντος της κάρτας)

CardColor

Είναι μία κλάση enum, η οποία περιέχει όλα τα χρώματα των καρτών τραίνου (Blue, Black, Green, Purple, Yellow, White, Orange, Red, Locomotive)

TrainCard

Η κλάση αυτή κληρονομεί τη κλάση Card και υλοποιεί μία κάρτα τραίνου η οποια περιέχει επιπρόσθετα ενα χρώμα . Πιο αναλυτικά τα επιπλέον attributes και μεθόδοι της κλάσης εκτός αυτά της Card είναι:

Τα attributes :

1) private CardColor color; (Το χρώμα της κάρτας τραίνου)

Οι μέθοδοι της κλάσης :

2) public int get_points(); (Επιστρέφει το χρώμα της κάρτας τραίνου)

DestinationCard

Η κλάση αυτή κληρονομεί τη κλάση PointsCard και υλοποιεί μια κάρτα προορισμού. Πιο αναλυτικά τα επιπλέον attributes και μεθόδοι της κλάσης εκτός αυτά της PointsCard είναι:

Τα attributes :

1) private ArrayList<String> colors; (Τα χρώματα που απαιτούνται για την αγορά της κάρτας προορισμού)
2) private String destination; (Η πόλη προορισμόυ της κάρτας)
3) private String departure; (Η πόλη αναχώρησης της κάρτας)

Οι μέθοδοι της κλάσης :

1) public String get_destination(); (Επιστρέφει τη πόλη προορισμού της κάρτας)
2) public String get_departure(); (Επιστρέφει τη πόλη αναχώρησης της κάρτας)
3) public ArrayList<String> get_colors(); (Επιστρέφει τα χρώματα που απαιτούνται για την αγορά της κάρτας)

BigCitiesCard

Η κλάση αυτή κληρονομεί τη κλάση PointsCard και υλοποιεί μια κάρτα μπόνους μεγάλης πόλης. Πιο αναλυτικά τα επιπλέον attributes και μεθόδοι της κλάσης εκτός αυτά της PointsCard είναι:

Τα attributes :

1) private final String city; (Η πόλη την οποία αναπαριστά η κάρτα)

Οι μέθοδοι της κλάσης :

1) public String get_city(); (Επιστρέφει τη πόλη την οποία αναπαριστά η κάρτα

Το παρακάτω UML διάγραμμα αναπαριστά της σχέσης των κλάσεων που υλοποιούν τις κάρτες του παιχνιδιού :

2.2| Κλάση Deck

Αυτή η κλάση ειναι υπεύθυνη για την αρχικοποίηση των καρτών του παιχνιδιού, καθώς επίσης και για το μοίρασμα τους σε κάθε παίκτη σύμφωνα με τους κανόνες του παιχνδιού. Επιπλέον αποθηκεύει τις στοίβες καρτών οι οποίες βρίσκονται στα decks των καρτών τραίνου και καρτών προορισμού, καθώς επίσης και των καρτών που βρίσκονται στο κέντρο, και είναι υπεύθυνη για το τράβηγμα αυτών. Πιο αναλυτικά τα attributes και οι μεθόδοι αυτής της κλάσης :

Τα attributes :

1) Stack<TrainCard> train_cards; (Οι στοίβα των καρτών τραίνου)
2) Stack<DestinationCard> dest_cards; (Οι στοίβα των καρτών προορισμού)
3) TrainCard center_cards[5]; (Η λίστα με τις 5 κάρτες που βρίσκονται στο κέντρο)
4) BigCitiesCard bigC_cards[6]; (Η λίστα με τις 6 κάρτες μπόνους μεγάλων πόλεων)
5) Player deck; (Δημιούργουμε έναν νέο Player ο οποίος θα είναι ο ιδιοκτήτης των καρτών που δεν ανήκουν σε κανένα παίκτη

Οι μέθοδοι της κλάσης :

1) public void initCards(); (Αρχικοποιεί όλες τις κάρτες του παιχνιδιού σύμφωνα με τους κανόνες του)
2) public void moirasma(Player p1, Player p2); (Μοιράζει τις πρώτες κάρτες στους παίκτες p1 και p2, σύμφωνα με τους κανόνες του παιχνιδιού)
3) public TrainCard[] get_center_cards(); (Επιστρέφει τις κάρτες που βρίκονται στο κέντρο)
4) public BigCitiesCard[] get_bigC_cards() (Επιστρέφει τις κάρτες μπόνους μεγάλων πόλεων)
5) public TrainCard draw_train_card(); (Επιστρέφει τη πρώτη κάρτα απο τη στοίβα καρτών τραίνου)
6) public DestinationCard[] draw_dest_card(); (Επιστρέφει ένα πίνακα απο κάρτες προορισμού για να διαλέξει ο παίκτης)

7) public int get_trainC_size(); (Επιστρέφει το μέγεθος της στοίβας των καρτών τραίνου)

8) public int get_destC_size(); (Επιστρέφει το μέγεθος της στοίβας των καρτών προορισμού)

9) public Stack<TrainCard> get_train_cards(); (Επιστρέφει τη στοίβα των καρτών τραίνου)

10) public Stack<DestinationCard> get_dest_cards(); (Επιστρέφει τη στοίβα των καρτών προορισμού)

2.3| Κλάση RailYard

Αυτή η κλάση ειναι υπεύθυνη για την αποθηκεύση των δεδομένων της περιοχής rail-yard κάθε παίχτη. Περιέχει 7 στοίβες καρτών τραίνου για τα 7 χρώματα των καρτών τραίνου. Επιπλέον περιέχει τις μεθόδους ώστε να αποκτήσουμε πρόσβαση σε αυτές τις στοίβες. (get_black_cards, get_blue_cards, get_yellow_cards, get_purple_cards, get_white_cards, get_red_cards, get_orange_cards)

2.4| Κλάση On-the-track

Αυτή η κλάση ειναι υπεύθυνη για την αποθηκεύση των δεδομένων της περιοχής on-the-track του παίκτη. Περιέχει 8 στοίβες καρτών τραίνου για τα 7 χρώματα των καρτών τραίνου και της κάρτας μπαλαντέρ, καθώς επίσης και τις μεθόδους ώστε να αποκτήσουμε πρόσβαση σε αυτές τις στοίβες. Επιπλέον περιέχει τη μέθοδο void collectFromRailYard(RailYard rail); , η οποία ειναι υπεύθυνη για να μαζεύει τις πρώτες κάρτες καθε στοίβας του rail , και να τις τοποθετεί στις στοίβες του on-the-track ανάλογα με το χρώμα τους.

2.5| Κλάση Player

Αυτή η κλάση ειναι υπεύθυνη για την αποθήκευση των δεδόμενων κάθε παίκτη και τη πρόσβαση σε αυτά, και γι' αυτό αποτελεί τις πιο βασικές κλάσεις της εργασίας. Πιο αναλυτικά τα attributes και οι μεθόδοι της κλάσης είναι :

Τα attributes :

1) private String name; (Το όνομα του παίκτη)

2) private ArrayList<TrainCard> train_cards; (Οι κάρτες τραίνου που έχει στο χέρι του ο παίκτης)

3) private ArrayList<DestinationCard> dest_cards; (Οι κάρτες προορισμού που έχει στο χέρι του ο παίκτης)

4) private ArrayList<BigCitiesCard> bigC_cards; (Οι κάρτες μπόνους μεγάλων πόλεων που εχει στα χέρια του ο πάικτης)

5) private ArrayList<DestinationCard> dest_cards_bought; (Οι κάρτες προόρισμους που έχει αγοράσει ο παίκτης)

6) private RailYard rail; (Η περιοχή rail-yard του παιίκτη)

7) private OnTheTrack track; (Η περιοχή on-the-track του παίκτη)

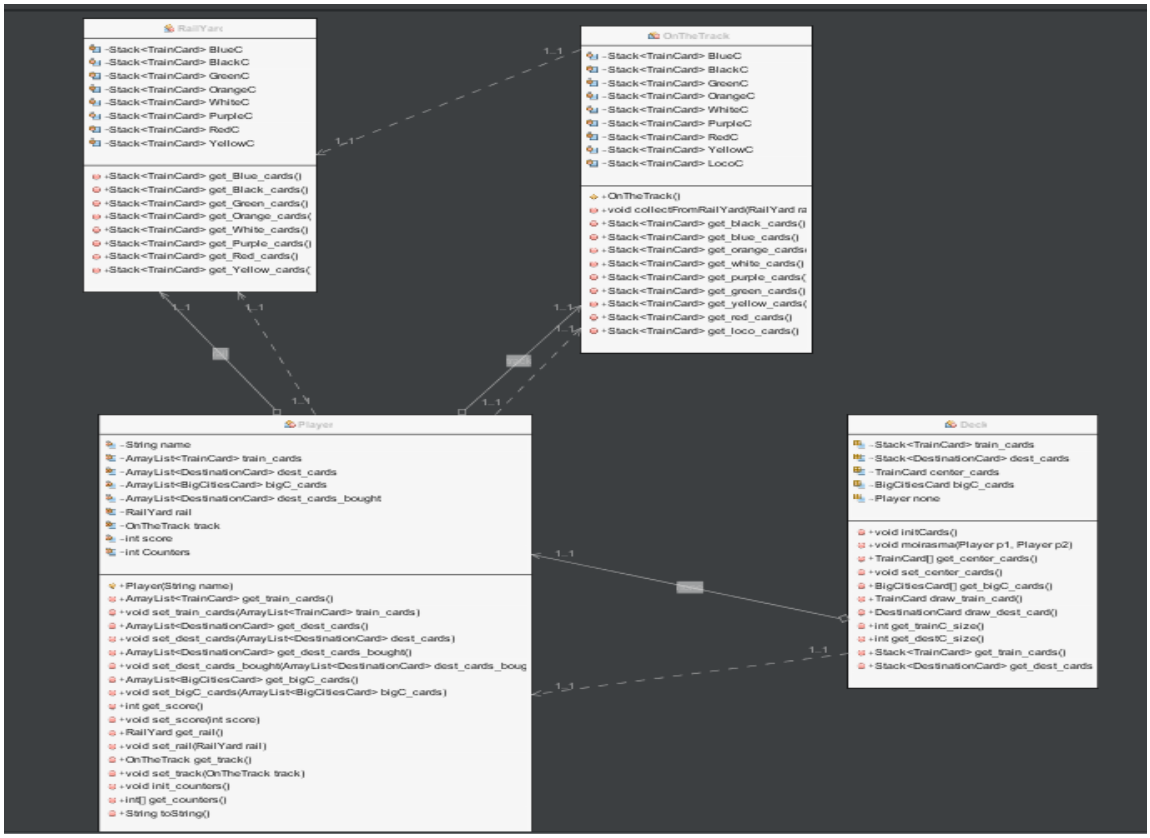
8) private int score; (Το σκορ του παίκτη)

9) private int Counters[6]; (Ο πίνακας ακεραίων με 6 θέσεις που αποθηκεύει πόσες φορές επισκέφτηκε μια μεγάλη πόλη)

Οι μέθοδοι της κλάσης :

Οι μέθοδοι της κλάσης περιλαμβάνουν όλα τα setters και τα getters των παραπάνω attributes , σύμφωνα με την αρχή της ενθυλάκωσης. Επιπλέον περιλαμβάνει μια μέθοδο public void init_counters(); , η οποία αρχικοποιεί τους counters των μεγάλεων πόλεων με 0.

Το παρακάτω UML διάγραμμα αναπαριστά τις σχέσεις μεταξύ των κλάσεων Player, RailYard, OnTheTrack και Deck :



3. Η Σχεδίαση και οι Κλάσεις του Πακέτου Controller

Κλάση Controller

Η κλάση αυτή είναι ο εγκέφαλος του παιχνιδιού και ειναι υπεύθυνη για την αρμονική συνεργασία των πακέτων View και Model. Η κλάση θα παίρνει τα δεδομένα που δίνει ο χρήστης απο το γραφικό περιβάλλον και κάνοντας τις κατάλληλες ενέργειες θα ενημερώνει τα δεδομένα που έχουν αποθηκευτεί στα αντικείμενα του πακέτου Model. Συνεπώς ειναι υπεύθυνη για την αρχικοποιήση του παιχνιδιού, τη διατήρηση της σειράς, τους ελέγχους ορθότητας των κινήσεων, την ενημέρωση του σκορ, τον τερματισμό το παιχνιδιού και την ανάδειξη του νικητή. Πιο αναλυτικά τα attributes και οι μέθοδοι αυτης της κλάσης :

Τα attributes :

1) private final Player p1; (Ο πρώτος παίκτης)
2) private final Player p2; (Ο δεύτερος παίκτης)
3) private final Deck mainDeck; (Το κύριο deck του παχινδιού)
4) private final View view; (Το αντικείμενο της κλάσης view του παιχνιδιού)
5) private Player PlayerTurn; (Μεταβλητή που κρατάει ποιός παίκτης εχει σειρά)
6) private ArrayList<TrainCard> PlayedCards; (Λίστα που κρατάει ποιές κάρτες θέλει να παίξει ο παίκτης)
7) private Boolean moved_on_track; (Boolean που ελέγχει αν ο παίκτης μετακίνησε κάρτες στο track ή όχι)
8) private Boolean start; (Μεταβλητή που ελέγχει αν βρισκόμαστε στην αρχή του παιχνιδιού ή όχι)
9) private Player FirstTurn; (Μεταβλητή που αντιστοιχεί στο παίκτη που αρχίζει πρώτος το παιχνίδι)

Οι μέθοδοι της κλάσης είναι :

1) public void init() (Αρχικοποιεί το παιχνίδι)
2) public void SetRandomFirstTurn(); (Αρχικοποιεί τυχαία το παίκτη που ξεκινάει πρώτος)
3) public void UpdateRailAndTrack(Player p); (Ενημερώνει τα πεδία rail-yard και on-the-track του παίκτη έπειτα απο μετακίνηση της πρώτης κάρτας κάθε στοίβας της rail-yard στο track)
4) public void RailContainsColor(Player p, CardColor color); (Ελέγχει αν ο παίκτης p εχει κάρτα με το χρώμα color στο rail-yard του)
5) public void PlayCards(); (Η μέθοδος αυτή ειναι υπεύθυνη για τον έλεγχο της ορθότητας των καρτών που θέλει να παίξει ο παίκτης, και αν είναι ορθές τις παίζει)

6) public Boolean BuyTicket(DestinationCard c, Player p); (Ελέγχει αν ο παίκτης μπορεί να αγοράσει τη συγκεκριμένη κάρτα προορισμού και το κάνει)
7) public void DrawTrainCard(); (Ο παίκτης τραβάει μια κάρτα τραίνου από τη στοίβα)
8) public boolean checkTrainRobbing(Player p, Player opp, CardColor color); (Ελέγχει αν υπάρχει περίπτωση Train robbing του παίκτη p στο παίκτη opp, για το χρώμα color)
9) public void setTurn(Player turn); (Αλλάζει τη σειρά των παίκτων και αρχικοποιεί τις κατάλληλες μεταβλητές (όπως PlayedCards, moved_on_track))
10) public Player getTurn(); (Επιστρέφει το παίκτη που έχει σειρά)
11) public void CalculateScores(); (Υπολογίζει τα σκορ των δύο παικτών και τα ενημερώνει κατάλληλα)
12) public void getWinner(); (Επιστρέφει το νικητή του παιχνιδιού)
13) public Boolean checkIfGameFinished(); (Ελέγχει αν το παιχνίδι πρέπει να τελειώσει)
14) public void checkBigCityCards(Player p); (Ελέγχει εαν ο παίκτης δικαιούται να πάρει μια κάρτα μπόνους μεγάλης πόλης)
15) public void UpdateBigCityCounters(Player p); (Ενημερώνει τους μετρήτες του παίκτη για τις κάρτες μπόνους μεγάλων πόλεων)
16) private void setListeners(); (Συνδέει τα κουμπιά της κλάσεις view, με τους listeners που θα περιγράψουμε ακριβώς απο κάτω)

Επιπλέον η κλάση Controller περιέχει 4 κλάσεις που υλοποιούν το interface ActionListener , και ειναι υπεύθυνες για τη κλήση των κατάλληλων μεθόδων έπειτα απο το πάτημα ενός κουμπιού απο τον χρήστη. Πιο συγκεκριμένα έχουμε :

1) CardListener που είναι υπεύθυνος για τα κουμπιά που συνδέονται με τις κάρτες τραίνου του παίκτη
2) TicketListener που είναι υπεύθυνος για τα κουμπιά που συνδέονται με τις κάρτες προορισμού του παίκτη
3) ActionButtonListener που ειναι υπεύθυνος για όλα τα κουμπιά του view πλην αυτών που είναι υπεύθυνοι οι υπόλοιποι listeners
4) DrawCardListener που ειναι υπεύθυνος για τα κουμπια που συνδέονται με το τράβημα νέων καρτών

4. Η Σχεδίαση και οι Κλάσεις του Πακέτου View

Σε αυτό το πακέτο υπάρχουν 2 κλάσεις. Η μία κλάση ειναι η JLayeredPaneExtension , η οποία μας επιτρέπτει να δημιουργούμε ενα panel το οποίο θα έχει μία εικόνα για background. Η αλλη κλάση και η πιο σημαντική, έιναι η view, η οποία είναι υπεύθυνη για όλο το γραφικό περιβάλλον της εφαρμογής. Ας πάμε να την δούμε αναλυτικότερα.

Κλάση View

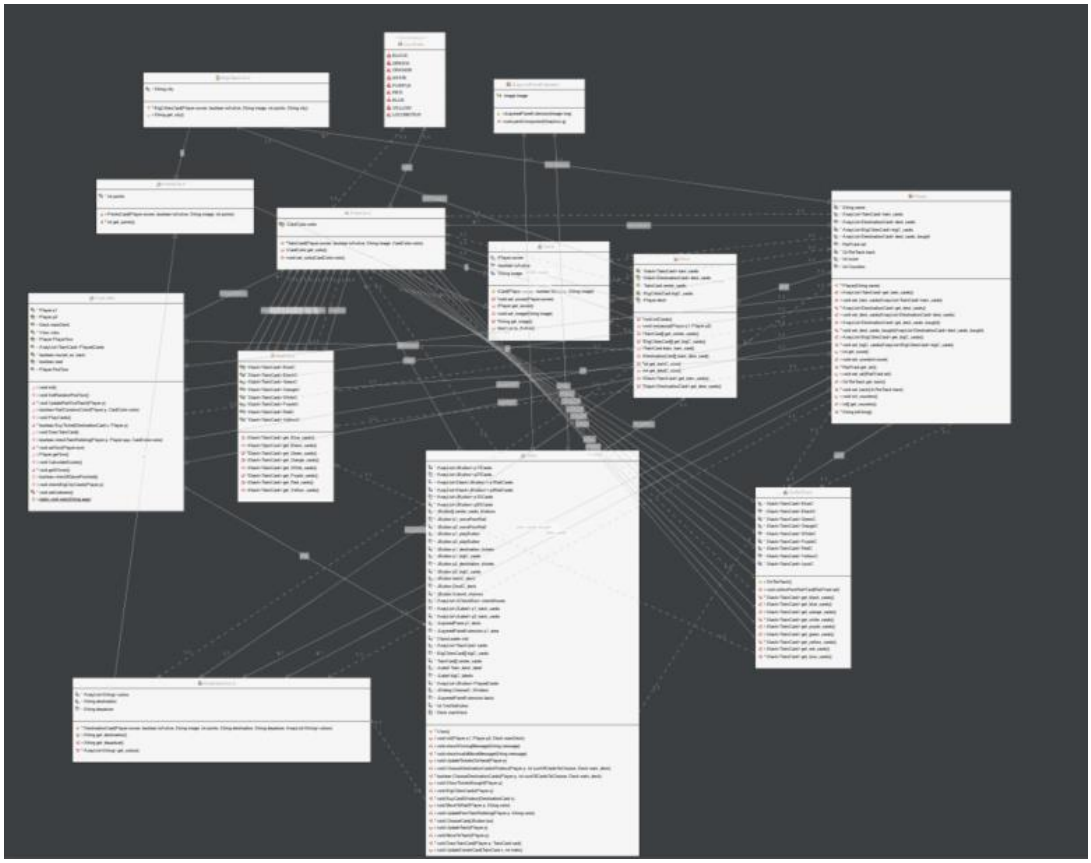
Αυτή η κλάση δημιουργεί ενα frame , όπου μέσα του θα υπάρχει το βασικό panel(basic). Μέσα σε αυτό το panel , υπάρχουν 2 μικρότερα panel, ένα για το κάθε παίκτη. Μέσα σε εκείνα υπάρχουν 5 μικρότερα panel, ένα για τις κάρτες τραίνου που έχει στο χερι του ο παίκτης, ένα για τις κάρτες προορισμού, ένα για το rail-yard, ένα για το on-the-track και τέλος ένα που θα ενημερώνει για το σκορ , τη σειρά και τις κάρτες μπόνος του παίκτη. Ανάμεσα στα δύο panel των παικτών βρίσκονται οι στοίβες των καρτών τραινου και προορισμού και θα αντιστοιχούν σε κουμπιά, 5 κεντρικές κάρτες τραίνου , και 6 εικόνες με τις κάρτες μπόνους μεγάλων πόλεων. Κάθε κάρτα του παιχνιδιού εκτός από τις κάρτες μπόνους θα αντιστοιχεί σε ενα κουμπί. Επιπλέον θα υπάρχουν 4 κουμπιά για το κάθε παίκτη που θα αντιστοιχούν για όταν θέλει να παίξει τις κάρες, όταν θελει να τις μεταφέρει στο on-the-track, όταν θέλει να δει τις κάρτες μπόνους του, και όταν θέλει να δει τις κάρτες προορισμού που έχει αγοράσει. Επιπλέον στη κλάση view περιέχονται 4 μεθόδοι , οι οποίοι ειναι υπεύθυνοι για τη δημιουργία JDialog. Μια για όταν ο παίκτης έχει να επιλέξει κάρτες προορισμού, μια για όταν ο παίκτης θέλει να αγοράσει μία καρτα, μια για να δημιουργήσει παράθυρο με τις κάρτες μπόνους του παίκτη, και μια για να δημιουργήσει παράθυρο για τις κάρτες προορισμού που έχει αγοράσει. Οι υπόλοιπες μέθοδοι που περιέχονται είναι υπεύθυνο για τη σωστή μετακίνηση των καρτών από το ένα panel στο άλλο ανάλογα τη κίνηση του παίκτη, την ενημέρωση των panel, αλλα και την ενημέρωση του scoreboard.

Αναλυτικότερα η αναπαράσταση σε UML :


View
- ArrayList<JButton> p1TCards
- ArrayList<JButton> p2TCards
- ArrayList<Stack<JButton>> p1RailCards
- ArrayList<Stack<JButton>> p2RailCards
- ArrayList<JButton> p1DCards
- ArrayList<JButton> p2DCards
- JButton[] center_cards_Buttons
- JButton p1_moveFromRail
- JButton p2_moveFromRail
- JButton p1_playButton
- JButton p2_playButton
- JButton p1_destination_tickets
- JButton p1_bigC_cards
- JButton p2_destination_tickets
- JButton p2_bigC_cards
- JButton trainC_deck
- JButton DestC_deck
- JButton Submit_choices
- ArrayList<JCheckBox> checkBoxes
- ArrayList<JLabel> p1_track_cards
- ArrayList<JLabel> p2_track_cards
- JLayeredPane p1_deck
- JLayeredPaneExtension p1_area
- ClassLoader cldr
- ArrayList<TrainCard> cards
- BigCitiesCard[] bigC_cards
- TrainCard[] center_cards
- JLabel Train_deck_label
- JLabel bigC_labels
- ArrayList<JButton> PlayedCards
- JDialog ChooseC_Window
- JLayeredPaneExtension basic
- int YesNoButton
- Deck mainDeck
+ View()
+ void init(Player p1, Player p2, Deck mainDeck)
+ void showWinningMessage(String message)
+ void showInvalidMoveMessage(String message)
+ void UpdateTicketsOnHand(Player p)
+ void ChooseDestinationCardsWindow(Player p, int numOfCardsToChoose, Deck main_deck)
+ boolean ChooseDestinationCards(Player p, int numOfCardsToChoose, Deck main_deck)
+ void ShowTicketsBought(Player p)
+ void BigCitiesCards(Player p)
+ void BuyCardWindow(DestinationCard c)
+ void MoveToRail(Player p, String color)
+ void UpdateFromTrainRobbing(Player p, String color)
+ void ChooseCard(JButton but)
+ void UpdateTrack(Player p)
+ void MoveToTrack(Player p)
+ void DrawTrainCard(Player p, TrainCard card)
+ void UpdateCenterCard(TrainCard c, int index)
+ void RemoveBigCity(int index)
+ void MoveCenterCardOnHand(JButton but, Player p)
+ void UpdateTurn(Player p)
+ void UpdateScore(Player p)
+ void UpdateTrainDeckSize()
+ void UpdateDestDeckSize()
+ ArrayList<JButton> getP1TCards()
+ ArrayList<JButton> getP2TCards()
+ ArrayList<JButton> getP1DCards()
+ ArrayList<JButton> getP2DCards()
+ JButton[] get_actionButtons()
+ JButton get_trainDeckButton()
+ JButton get_destDeckButton()
+ JButton[] get_centerCButtons()
+ int get_YesNoButton()

5. Η Αλληλεπίδραση μεταξύ των κλάσεων – Διαγράμματα UML

Επιπλέον των προηγούμενων UML διαγραμμάτων που δείξαμε στη προηγούμενη ενότητα, παραθέτουμε το UML διάγραμμα όλης της εργασίας



6. Λειτουργικότητα (Β Φάση)

Το πρόγραμμα αρχικά διαλέγει τυχαία τη σειρά των παικτών, και αφού την ανακοινώσει , ο παίκτη που είναι πρώτος επιλέγει αναγκαστικά τουλάχιστον μία κάρτα προορισμού, έπειτα μπορεί να συνεχίσει τις κινήσεις του κανονικά σύμφωνα με τους κανόνες. Το ίδιο ισχύει και για τον επόμενο παίκτη όταν αλλάξει η σειρά πρώτη φορά. Έπειτα κάθε φορα που αλλάζει η σειρά, ο κάθε παίκτη υποχρεωτικά πρέπει να μεταφέρει τις κάρτες τραίνου του από το rail-yard στο on-the-track, ακόμα κι αν αυτό δεν περιέχει καμία κάρτα, χάριν ευκολίας προγραμματισμού. Επειτά συνεχίζει τις κινήσεις του κανονικά σύμφωνα με τους κανόνες. Επιπλέον όταν επιλέγει μία κάρτα τραίνου να παίξει, αυτή ανυψώνεται σε σχέση με τις υπόλοιπες κάρτες του παίκτη, και παίζει τις επιλεγμένες πατώντας το κουμπί Play Cards. Αν έχει δικαίωμα να πάρει μία bonus card τότε αυτή πηγαίνει αυτόματα στις bonus cards του παίκτη τις οποίες μπορεί να δεί πατώντας το κουμπί Big City Bonus Cards.

7. Συμπεράσματα

Στην εργασία δεν προτέθηκε καμία επιπλέον μέθοδος ή κλάση σε σχέση με της σχεδίαση, εκτος δύο accessors στην κλάση Controller , για χάριν ευκολίας των test. Επιπλέον τα tests βρίσκονται στο φάκελο test.