Πανεπιστήμιο Κρήτης –Τμήμα Επιστήμης Υπολογιστών

ΗΥ252– Αντικειμενοστρεφής Προγραμματισμός

Διδάσκων: Ι. Τζίτζικας

Χειμερινό Εξάμηνο 2020-2021

Β ΦΑΣΗ ΠΡΟΤΖΕΚΤ

Εισαγωγή

Νικολαος Μπαρμπαρουσης

4690

22/01/2022

Περιεχόμενα

1. Εισαγωγή

Για την υλοποιηση του παιχνδιου ακολουθηθηκε το μοντελο MVC. Η εργασια χωριστηκε σε 3 βασικα πακετα: Model, View και Controller.

- Το Model περιγραφει τα δεδομενα και την λογικη του παιχνδιου οπως το Tile, Board ή Card καθως και τον τροπο με τον οποιο αλληλεπιδρουν αυτα τα αντικειμενα μεταξυ τους.
- Το View εχει ως μοναδικο σκοπο να δημιουργει το γραφικο περιβαλλον του προγραμματος που θα βλεπει ο χρηστης.
- Το Controller χρησιμοποιεται σαν μεσαλοβητης αναμεσα στο View και στο Model, διοικώντας την συμπεριφορα του Model και τις ενεργειες που θα λαβει, αναλογα με τις ενεργειες του χρηστη.

Για παραδειγμα, ο χρηστης εχοντας μπροστα του το GUI του παιχνιδιου, πατωντας το κουμπι για να τραβηξει μια καρτα, στελνεται απο το View, σημα στον Controller. Ο Controller πλεον χρησιμοποιώντας τα αντικειμενα του Model, εκτελει αυτην την ενεργεια και ανανεωνει επισης με τα νεα δεδομενα και το View.

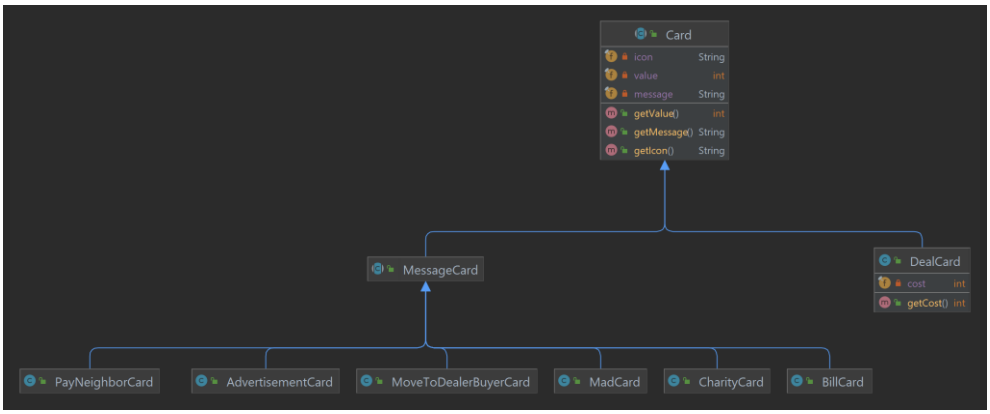
2. Η Σχεδίαση και οι Κλάσεις του Πακέτου Model

Το πακετο Model ειναι χωρισμενο σε 8 συνολικα υποπακετα, που το καθενα τους εκφραζει και ενα δεδομενο και τον μηχανισμο του, του παιχνδιου.

1) Cards: Το πακετο αυτο υλοποιει καθε μια καρτα του παιχνιδιου χρησιμοποιώντας μια abstract κλαση Card.
   a) Η κλαση αυτη εχει 3 μεταβλητες :
      i) message: Η προταση της καρτας
      ii) value: Η τιμη της καρτας
      iii) icon: Η εικονα της καρτας

   b) Η κλαση αυτη εχει και μια abstract μεθοδο cardAction, που κληρομονειται απο καθε διαφορετικη καρτα και υλοποιει την ενεργεια της.

   c) Στη συνεχεια χωριζεται σε δυο πακετα:
      i) DealCard: Κλαση που περιγραφει την συμπεριφορα μιας καρτας Deal.
      ii) MessageCard: Abstract κλαση που στην συνεχεια χρησιμοποιεται για την δημιουργια των:
         (1) AdvertisementCard
         (2) BillCard
         (3) CharityCard
         (4) MadCard
         (5) MoveToDealerBuyerCard
         (6) PayNeighbourCard



UML διαγραμμα της κλασης Cards

2) Deck: Το πακετο αυτο υλοποιηθηκε για να μοντελοποιησουμε τις τραπουλες των Deal και Message καρτων που θα χρειαστουμε.

   a) Στο πακετο οριζονται 2 κλασεις, η DealDeck και η MessageDeck, που υλοποιουν τις αντιστοιχες τραπουλες του παιχνιδιου.
      - Ισως, με την χρηση Generics, να μπορουσε να αποφευχθει αυτος ο διαχωρισμος των κλασεων, και να υπηρχε μια μονο κλαση που να περιγραφει την καθε τραπουλα, αλλα δυστυχως ο περιερισμενος χρονος που υπηρχε για την δημιουργια του προτζεκτ, δεν το επετρεψε.

   b) Σε περιπτωση που τελειωσουν τα φυλα καποιας τραπουλας, ο μηχανισμος του reshuffle των discarded καρτων γινεται εσωτερικα στην κλαση, μεσω της μεθοδου reshuffleDeck, που μπορει επισης να χρησιμοποιηθει οποιαδηποτε στιγμη καθως ειναι public .

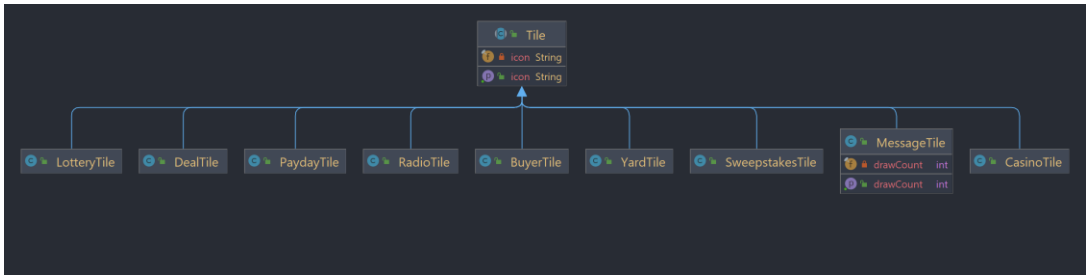
   c) Το μονο που θα πρεπει να προσεξει κανεις ειναι οτι η μεθοδος drawCard παρολο που επιστρεφει την καρτα στην κορυφη της τραπουλας, αφου την αφαιρεσει απο το stack, δεν εκτελει καποια αλλη ενεργεια, και θα πρεπει δηλαδη να:

      i) Προσθεσουμε την καρτα στο playedDeck χρησιμοποιώντας την moveToPlayed, σε περιπτωση που ο παιχτης δεν θελει να κρατησει την καρτα που τραβηξε.
      Η
      ii) Προσθεσουμε την καρτα στο χερι του παιχτη καλωντας την addPlayerCard της κλασης Player και αφαιρωντας την απο το stack της activeDeck, καλωντας την removeCard, σε περιπτωση που ο παιχτης επιλεξει να κρατησει την καρτα που τραβηξε.

3) Tile: Το πακετο αυτο υλοποιει καθε tile του παιχνιδιου χρησιμοποιωντας μια abstract κλαση Tile.
   a) Η κλαση αυτη εχει 1 μεταβλητη:

i) Icon: Η εικονα του tile

b) Στη συνεχεια χρησιμοποιεται αυτη η abstract κλαση για την δημιουργια των:
   i) BuyerTile
   ii) CasinoTile
   iii) DealTile
   iv) LotteryTile
   v) MessageTile
   vi) PaydayTile
   vii) RadioTile
   viii) SweepstakesTile
   ix) YardTile



UML διαγραμμα της κλασης Tile

4) Board: Το πακετο αυτο υλοποιει το ταμπλο του παιχνιδιου.
   a) Η κλαση εχει 1 μεταβλητη:
      i) Board: Ενας πινακας απο στιγμιοτυπα Tiles που περιγραφει καθε κουτακι του ταμπλο

5) Player: Το πακετο αυτο υλοποιει τα δεδομενα και την συμπεριφορα του καθε παιχτη του παιχνιδιου.
   a) Η κλαση εχει 11 μεταβλητες:
      i) balance: Το χρηματικο ποσο που διαθετει ο παιχτης
      ii) loanAmount: Το χρηματικο ποσο των δανειων που εχει παρει ο παιχτης
      iii) billsAmount: Το χρηματικο ποσο των λογαριασμων του παιχτη
      iv) position: Η θεση του παιχτη στο ταμπλο
      v) currentMonth: Ο αριθμος του μηνα του παιχτη
      vi) score: Το score του παιχτη
      vii) playerCards: ArrayList που περιεχει ολες τις Deal καρτες που εχει στην κατοχη του ο παιχτης
      viii) pawn: Το πιονι του παιχτη
      ix) isFinished: Boolean μεταβλητη που εκφραζει αν τελειωσε το παιχνιδι ο παιχτης
      x) isTurn: Boolean μεταβλητη που εκφραζει αν ειναι η σειρα του παιχτη(Ισως στην τελικη υλοποιηση αλλαξει και χρησιμοποιηθει ενα στιγμιοτυπου τυπου Player που εκφραζει σε καθε γυρο τον παιχτη που εχει σειρα στο Controller)
      xi) completedAction: Boolean μεταβλητη που εκφραζει αν ο παιχτης εχει εκτελεσει την αντιστοιχη ενεργεια στο tile που βρισκεται

b) Η κλαση, εκτως απο του κλασικους Accessors που προσφερει, περιεχει και τις μεθοδους:
   i) updateBalance(int num)
   ii) updateLoanAmount(int num)
   iii) updateBillsAmount(int num)
   iv) updatePosition(int num)
   v) updateCurrentMonth(int num)
   Οι μεθοδοι αυξανουν το αντιστοιχο πεδιο που επιλεγεται, με την τιμη που περασουμε.

c) Επισης, εχουμε και τις μεθοδους:
   i) addPlayerCard: Προσθετει την καρτα που δωσουμε στο DealDeck του παιχτη
   ii) removePlayerCard(Card c), removePlayerCard(int index): Που διαγραφει την καρτα που θελουμε ή την καρτα στην αντιστοιχη θεση.
   iii) emptyHand: Διαγραφει καθε καρτα που εχει στην κατοχη ο παιχτης, χρησιμοποιεται οταν ο παιχτης εχει τελειωσει το παιχνιδι.

6) Pawn: Το πακετο αυτο υλοποιει το πιονι του καθε παιχτη
   a) Η κλαση εχει 1 μεταβλητη:
      i) icon: Η εικονα του πιονιου

7) Dice: Το πακετο αυτο υλοποιει το ζαρι του καθε παιχτη
   a) Η κλαση εχει 3 μεταβλητες:
      i) icon: Η εικονα του ζαριου
      ii) num: Το νουμερο που εφερε το ζαρι
      iii) wasRolled: Boolean μεταβλητη που εκφραζει αν το ζαρει εχει παιχτει
   b) Για την προσομοιωση του ζαριου χρησιμοποιεται η μεθοδος roll

8) Jackpot: Το πακετο αυτο υλοποιει την λειτουργικοτητα του Jackpot του παιχνιδιου
   a) Η κλαση εχει 1 μεταβλητη:
      i) balance: Το χρηματικο ποσο του Jackpot

   b) Η κλαση, εκτως απο τους κλασικους Accessors που προσφερει, περιεχει και την μεθοδο win(Player p), που πληρωνει τον παιχτη που εχουμε επιλεξει και επαναφερει το balance στο 0.

Στην Α φαση του project εχει οριστει και η κλαση Model, που χρησιμοποιει τα παραπανω πακετα για να δημιουργησει το μοντελο του παιχνιδιου. Στην τελικη υλοποιηση, ισως αφαιρεθει και αντικατασταθει, οριζοντας κατευθειαν τα στιγμιοτυπα των κλασεων στο Controller, μιας και η λογικη του παιχνιδιου δεν ειναι τοσο συνθετη, που να απαιτει, εναν τοσο μεγαλο κατακερματισμο των λειτουργιων.

Για την τελικη υλοποιηση του προτζεκτ, η κλαση Model, δεν διαγραφτηκε καθως κριθηκε αναγκαια για την ομαλη και ευκολη επικοινωνια μεταξυ του View και του Controller.

Πρεπει επισης να σημειωθει οτι στην τελικη υλοποιηση, μπορει οι ενεργειες των καρτων, αντι να υλοποιουνται χρησιμοποιωντας την abstract μεθοδο cardAction, να οριζονται σε μια μεθοδο στο Controller ή σε μια κλαση σαν αυτη που περιγραφθηκε παραπανω(Model), οπου τα στιγμιοτυπα των αντικειμενων που χρειαζομαστε θα υπαρχουν ηδη, και δεν θα χρειαστει να περναμε για καθε διαφορετικη καρτα, και διαφορετικα ορισματα καθως και διαφορετικο πληθος ορισματων.

Η ενεργεια της καθε καρτας, στην τελικη υλοποιηση, γινεται μεσω της μεθοδου cardAction(Player p, Model m) που βρισκεται στην abstract κλαση MessageCard.

i) Player p: Ο παιχτης που θα εκτελεσει την αντιστοιχη ενεργεια
ii) Model m: Το μοντελο του παιχνιδιου που θα χρησιμοποιηθει απο την μεθοδο για την αλλαγη των δεδομενων.

Για παραδειγμα, πρεπει να περασουμε και το μοντελο στην μεθοδο, για να εκτελεσουμε την ενεργεια της PayNeighborCard, αφου πρεπει εκτως απο τον παιχτη που πληρωνει, να ανανεωσουμε και τα δεδομενα του αντιπαλου του.

3. Η Σχεδίαση και οι Κλάσεις του Πακέτου Controller

Το πακετο Controller ειναι υπευθυνο για την επικοινωνια μεταξυ του Model και του View μας. Θα οριζει τις ενεργειες των κουμπιων του View, και θα χρησιμοποιει τις μεθοδους των κλασεων της Model, λειτουργωντας σαν ενα 'μυαλο' του παιχνιδιου.

Ειναι προφανες λοιπον, πως το controller θα πρεπει να ορισει αρκετες μεθοδους για την αρχικοποιηση/ελεγχο/ενημερωση του παιχνιδιου.

Αυτες ειναι οι :

1) setDealDeck, setMessageDeck: Για την αρχικοποιηση των 2 τραπουλων του παιχνιδιου
2) setTiles: Για την αρχικοποιηση των tiles του ταμπλο
3) setListeners: Για την υλοποιηση των action των μεταβλητων της View(π.χ rollDiceButton)
4) setMonthsPlayed: Για την αρχικοποιηση των αριθμων των μηνων που θα διαρκεσει το παιχνιδι
5) isGameFinished: Για τον ελεγχο αν το παιχνδι εχει τελειωσει
6) getWinner: Για την ευρεση του νικητη του παιχνιδιου
7) decideFirstTurn: Για την επιλογη του παιχτη που θα ξεκινησει το παιχνιδι
8) decideNextTurn: Για την ευρεση ποιου παιχτη θα παιξει στην επομενη σειρα

9) sundayMatch, thursdayCrypto: Για την υλοποιηση των ενεργειων του παιχνδιου στα tiles που αντιστοιχουν στην Κυριακη και στην Πεμπτη.

10) takeLoanDialog: Για την υλοποιηση των ενεργειων που θα εκτελεστουν, οταν ενας παιχτης ζηταει να παρει δανειο.

11) dealCardsDialog: Για την υλοποιηση των ενεργειων που θα εκτελεστουν, οταν ενας παιχτης ζηταει να δει, τις καρτες που εχει στην κατοχη του.

12) loanPaymentDialog: Για την υλοποιηση των ενεργειων που θα εκτελεστουν, οταν ενας παιχτης, βρισκεται στο PayDayTile, και πρεπει να ξεπληρωσει το δανειο ή μερος του δανειου του.

13) Στη συνεχεια εχουμε την υλοποιηση των ενεργειων του καθε Tile του παιχνιδιου:

    a) sweepstakesTileAction
    b) lotteryTileAction
    c) buyerTileAction
    d) casinoTileAction
    e) yardTileAction
    f) paydayTileAction

    Καθε μεθοδος, εκτελει τις αντιστοιχες ενεργειες που πρεπει στο μοντελο, παιρνοντας, οπου χρειαζεται, και πληροφοριες μεσω διαλογων απο τον παιχτη.

    Καποιος θα μπορουσε να πει οτι αυτες οι μεθοδοι θα επρεπε κανονικα να βρισκονται στο μοντελο του παιχνιδιου και οχι στον Controller. Η αποφαση να μεινουν σε αυτην την κλαση, ηταν επειδη η εκτελεση των ενεργειων αυτων απαιτουσαν ταυτοχρονα, και την χρηση του μοντελου αλλα και του View, καθως και επειδη η λογικη του καθε tile θεωρηθηκε αρκετα “απλη ”, σε βαθμο δηλαδη που δεν χρειαζεται να διαχωρισουμε τις μεθοδους.

    Π.χ. Οταν χρειαζεται ο παιχτης να επιλεξει μια DealCard για να πουλησει, αν βρισκεται σε buyerTile ή Οταν πρεπει οι παιχτες να διαλεξουν ενα νουμερα, αν καποιος βρισκεται σε lotteryTile.

14) playerTurnAction: Η μεθοδος δεχεται ως ορισματα 2 παιχτες, τον παιχτη που εχει τωρα σειρα p1, και τον αντοιστιχο αντιπαλο p2.

    Στη συνεχεια, ριχνει το ζαρι του παιχτη και εκτελει ολες τις ενεργειες που πρεπει, οπως τον Αγωνα Κυριακης αν βρεθει σε tile που ειναι Κυριακη, καθως και καλει την μεθοδο που αντιστοιχει στο καινουργιο tile που βρισκεται.

    Επισης, ενημερωνει, οπου χρειαζεται, το view καθως και το infoBox, και στο τελος καλει την decideNextTurn(), οπου πλεον σειρα εχει ο επομενος παιχτης.

15) actionPerformed: Η μεθοδος αυτη ειδοποιεται για τα πατηματα των κουμπιων της View, και αναλογως την καταστση του παιχνιδιου, καλει τις αντιστοιχες μεθοδους.

Η μεθοδος calculateScore που βρισκοταν στο Controller στην Α φαση, εχει μεταφερθει στην κλαση Player.

Ενω η μεθοδος jackpot, που θα χρησιμοποιωνταν για τις ενεργειες του Jackpot, εχει διαγραφθει, καθως κριθηκε οτι δεν χρειαζεται να βρισκεται σε ξεχωριστη μεθοδο. Οι ενεργειες του Jackpot βρισκονται πλεον στην μεθοδο playerTurnAction.

4. Η Σχεδίαση και οι Κλάσεις του Πακέτου View

Το πακετο View ειναι χωρισμενο σε συνολικα 3 κλασεις, εκφραζοντας το καθενα, ενα μερος του GUI, καθως και στην κλαση View που, χρησιμοποιωντας της παραπανω, υλοποιει το GUI μας.

Αναλυτικα για καθε κλαση:

1) BoardPanel: Η κλαση αυτη δημιουργει το panel που περιεχει, το ταμπλο του παιχνιδιου καθως και το Jackpot.

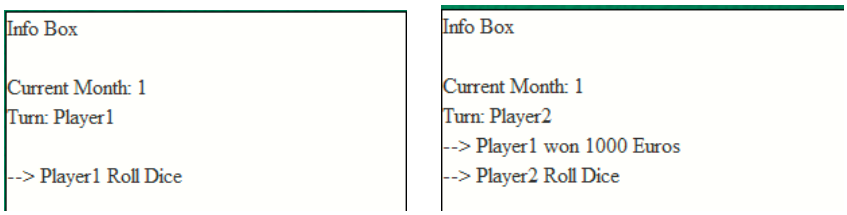
   (a) Επεκτεινει την κλαση JLayeredPane, και οριζει τα απαραιτητα JLabel οπως το pawnA,pawnB ή JackpotTile.
   (b) Δεχεται σαν ορισμα ενα Model, που περιγραφει την τωρινη κατασταση του παιχνιδιου που θελουμε να δειξουμε στον χρηστη.
   (c) Προσφερει 1 public μεθοδο:

i) void update(): Η μεθοδος αυτη, οταν καλειται, ανανεωνει την θεση των πιωνων καθως και την τιμη του Jackpot, στα καινουργια πλεον δεδομενα του Model m.

Ενα instance της BoardPanel

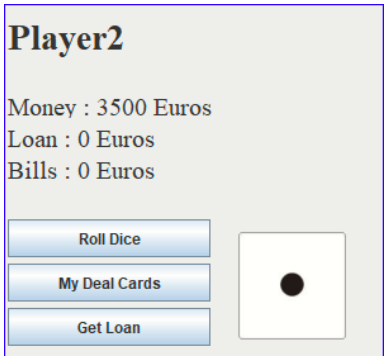
2) InfoBoxPanel: Η κλαση αυτη δημιουργει το panel που θα χρησιμοποιηθει για να ενημερωνει τους παιχτες για την κατασταση του παιχνιδιου, καθως και για τις ενεργειες που πρεπει να εκτελεσουν. Οι 2 πρωτες σειρες περιγραφουν τα δεδομενα του παιχτη που εχει τωρα σειρα, ενω οι 2 τελευταιες, τις 2 τελευταιες ενεργειες που πρεπει να εκτελεστουν.

   (a) Επεκτεινει την κλαση JTextArea και οριζει εναν πινακα infoMessages, στον οποιον περιεχονται ολα τα μυνηματα που θα προβαλουμε.
   (b) Προσφερει 2 public μεθοδους:
      (i) void updatePlayer: Η μεθοδος αυτη ενημερωνει τις 2 πρωτες σειρες του πεδιου, τον παιχτη που εχει πλεον σειρα καθως και τον μηνα στον οποιον βρισκεται, αναλογα με ποιον παιχτη περασουμε.
      (ii) void updateMessage: Η μεθοδος αυτη ενημερωνει την τελευταια σειρα του πεδιου με αυτη που περασουμε, και θετει το προηγουμενο μυνημα στην προτελευταια σειρα.



2 Καταστασεις του InfoBoxPanel

3) PlayerBoxPanel: Η κλαση αυτη δημιουργει το Panel που περιεχει ολες τις πληροφοριες του παιχτη.
   (a) Επεκτεινει την κλαση JPanel και οριζει τα:
      1. nameLabel
      2. balanceField
      3. loanField
      4. billField
      5. rollDiceButton
      6. dealCardsButton
      7. loanButton
      8. dice
   (b) Δεχεται σαν ορισμα ενα Player p, που ειναι ο παιχτης που θελουμε να εμφανισουμε.
   (c) Προσφερει μια public μεθοδο:
      (i) void update: Η μεθοδος αυτη ενημερωνει τα διαφορα Jlabel, στις καινουργιες τιμες που εχει πλεον ο παιχτης p.



Ενα instance της PlayerBoxPanel

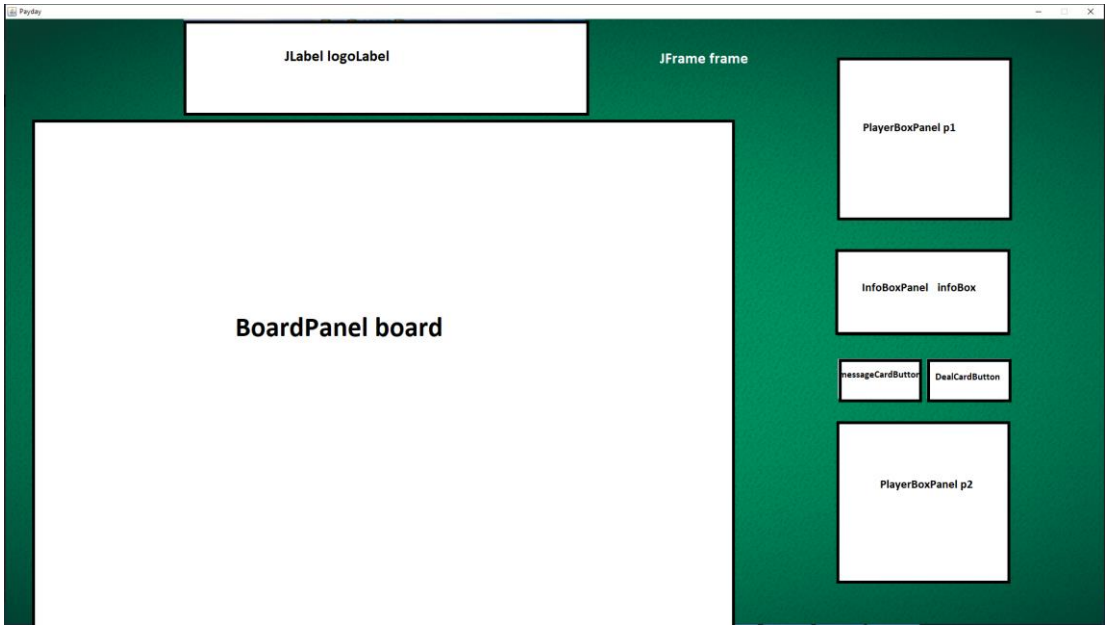
4) View: Η κλαση αυτη χρησιμοποιει τις κλασεις που εχουμε δημιουργησει, προσθετοντας καποια αντικειμενα και μεθοδους, για να δημιουργησει το GUI που θα προβαλεται στους παιχτες.

    (a) Οριζει:
        (i) 1 JFrame: Το βασικο frame στο οποιο θα βρισκεται καθε αντικειμενο
        (ii) 2 PlayerBoxPanel: Ενα για τον καθε παιχτη
        (iii) 1 BoardPanel: Για το ταμπλο/jackpot
        (iv) 1 InfoBoxPanel: Για το πανελ ενημερωσεων
        (v) 2 JButtons: Ενα για την καθε τραπουλα
        (vi) 1 JLabel: Για το logo του παιχνιδιου

    (b) Δεχεται σαν ορισμα ενα Model m, που ειναι το μοντελο του παιχνιδιου που θελουμε να προβαλουμε.

    (c) Προσφερει συνολικα 7 public μεθοδους:
        (i) void update: Η μεθοδος αυτη καλει τις αντιστοιχες update() στα PlayerBoxPanel και BoardPanel, και ξαναζωγραφιζει το main frame μας.
        (ii) void updateInfoPanelPlayer: Η μεθοδος αυτη καλει αντιστοιχα την updatePlayer της InfoBoxPanel, με τον παιχτη που εχουμε περασει.
        (iii) void updateInfoPanelMesage: Η μεθοδος αυτη καλει ανιστοιχα την updateMesage της InfoBoxPanel, με το μυνημα που εχουμε περασει.
        (iv) Getters για τα 2 PlayerBoxPanel και τα 2 κουμπια των τραπουλων.

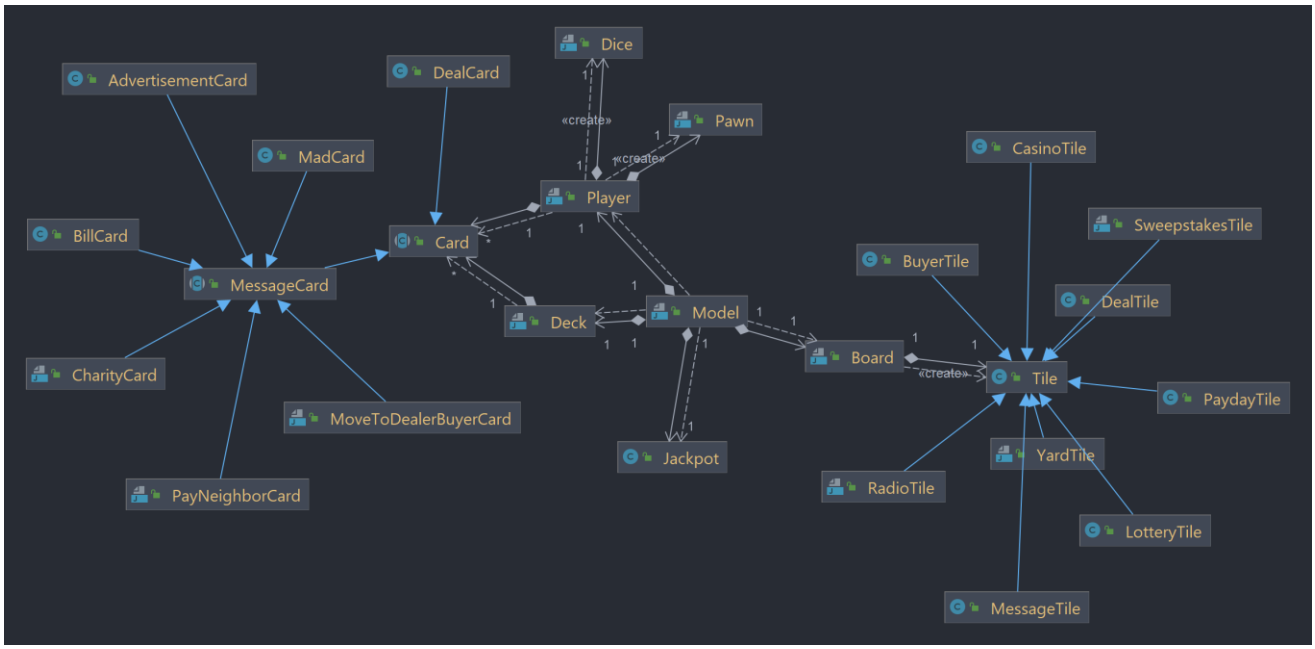
    Εδω πρεπει να σημειωθει οτι τα Getters χρησιμοποιουνται απο το Controller, για τα actionListeners που εχουμε ορισει. Ομως γνωριζουμε οτι δεν θα επρεπε κανονικα, εξωτερικες κλασεις να γνωριζουν για την υπαρξη των αντικειμενων αυτων. Μια καλυτερη υλοποιηση θα εκρυβε αυτα τα αντικειμενα, αλλα θα προσφερε καποιες μεθοδους οπου θα μπουρουσαμε να περναμε απο το Controller στο View, αυτους τους actionListeners.



Ο τροπος με τον οποιο ειναι τοποθετημενα τα αντικειμενα της View

5. Η Αλληλεπίδραση μεταξύ των κλάσεων – Διαγράμματα UML

Για να παρουμε μια καλυτερη εικονα του πως ειναι ορισμενο το Model, παρακατω ειναι το UML διαγραμμα του πακετου.

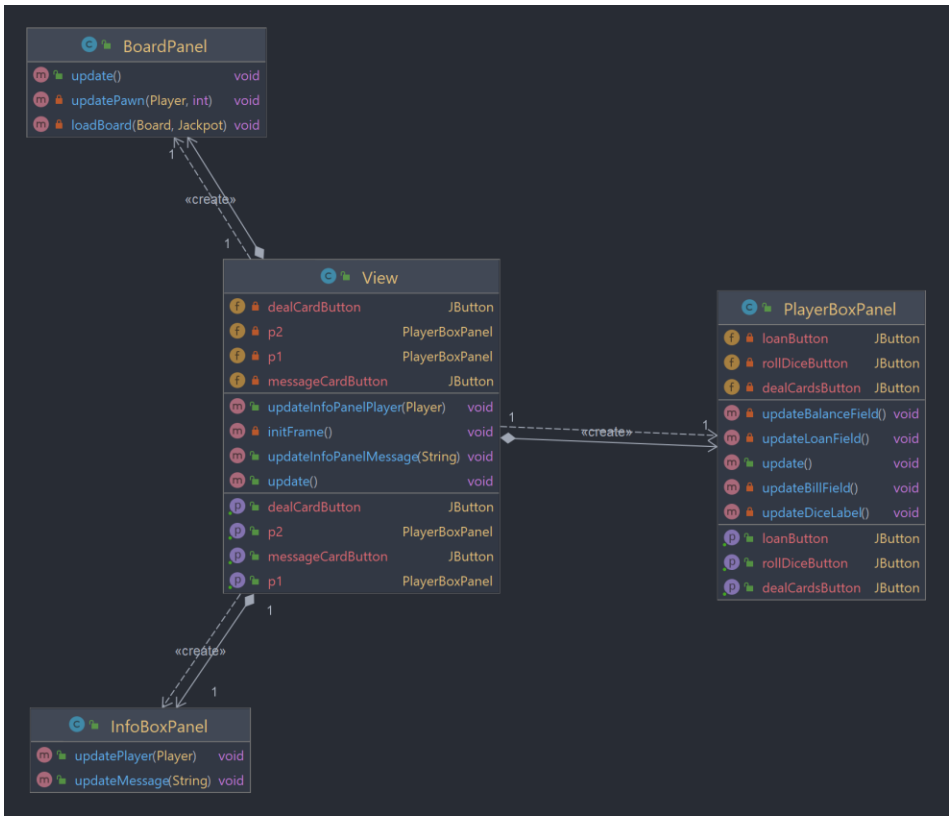


Απο οτι παρατηρουμε, εχουμε 3 συνολικα βασικες ‘οικογενειες’,

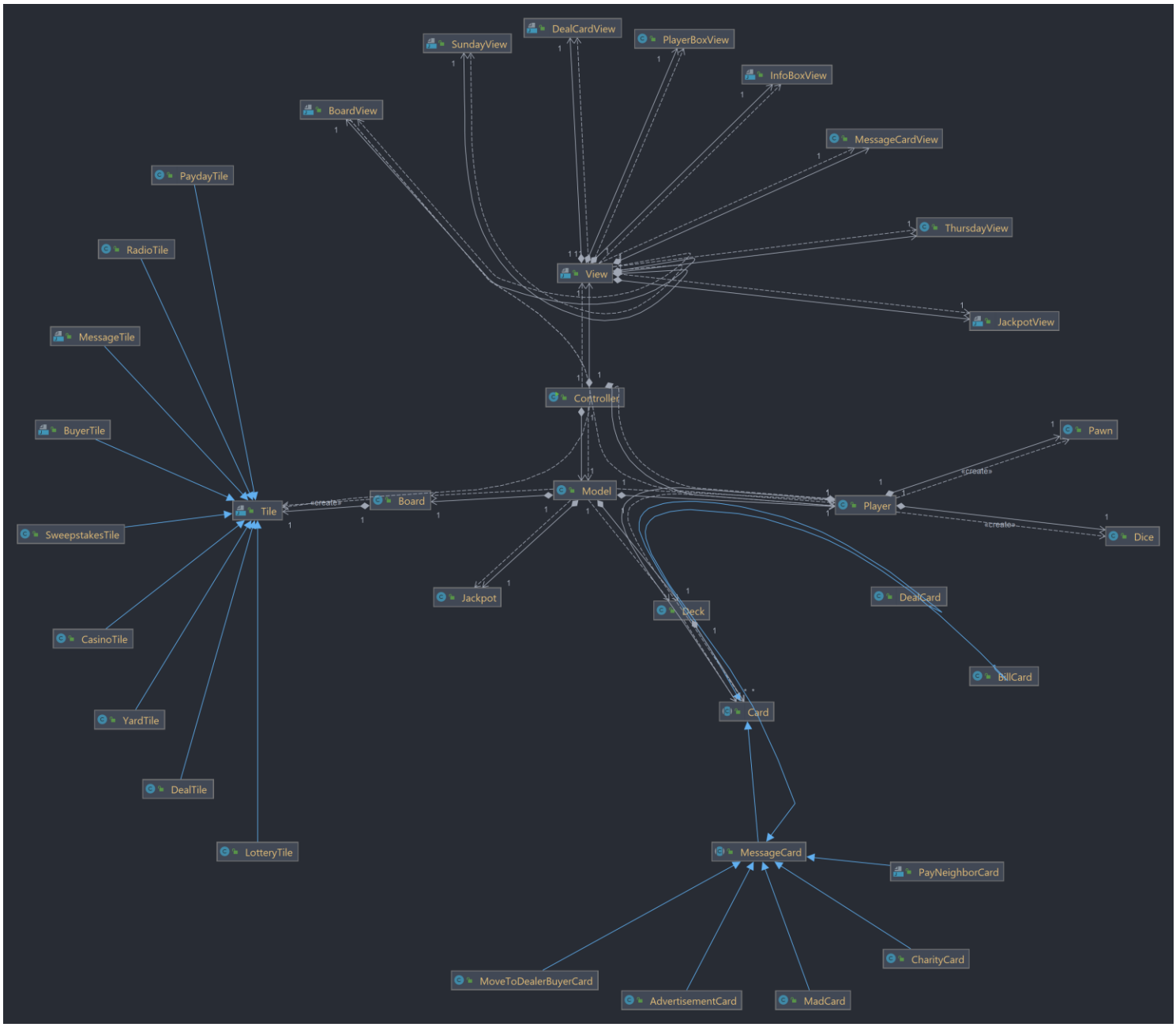
Η 1η αφορα τις καρτες που χρησιμοποιουνται στο παιχνιδι, η 2η περιγραφει το καθε tile του ταμπλου μας, ενω η 3η υλοποιει τις ιδιοτητες του καθε παιχτη.

Στην μεση του διαγραμματος, βλεπουμε την κλαση Model, που χρησιμοποιει αυτες τις ‘κατηγοριες’ που εχουμε ορισει, για να δημιουργησει το μοντελο του παιχνιδιου.

Παρακατω βλεπουμε και το UML διαγραμμα του Πακετου View.



Και για να καταλαβουμε πιο ευκολα την συνολικη εικονα του Project, στην συνεχεια παρατηρουμε το UML διαγραμμα του προγραμματος.



Στο κατω μισο του διαγραμματος παρατηρουμε την υλοποιηση του Model μας, ενω στο πανω μισο βλεπουμε πως εχει οριστει το View μας.

Στο κεντρο του διαγραμματος υπαρχει το Controller που εποικοινωνει αμεσα μονο με το View και το Model, αλλα εμμεσα συνδιαζει καθε εννοια του προγραμματος.

Το μονο που εχει αλλαξει απο την Α φαση του προτζεκτ σε αυτο το διαγραμμα, ειναι οτι ορισμενες κλασει της View δεν υπαρχουν πλεον, και για αυτο χρησιμοποιηθηκε το ιδιο.

6. Λειτουργικότητα (Β Φάση)

Το προτζεκτ αυτο υλοποιει καθε ενεργεια του παιχνδιου PayDay.

Η μονη διαφοροποιηση ειναι οτι το κουμπι End Turn δεν υπαρχει καθως, ο Controller εσωτερικα ελεγχει την σειρα τον παιχτων και τις ενεργειες που επιτρεπεται να εκτελεσουν.

7. Συμπεράσματα

Οπως αναφερθηκε και προηγουμενως, αν υπηρχε παραπανω χρονος για το προτζεκτ αυτο, ισως η υλοποιηση των τραπουλων καθως και των ActionListeners των διαφορων κουμπιων, γινοταν με ενα πιο μεθοδικο τροπο.