Πανεπιστήμιο Κρήτης –Τμήμα Επιστήμης Υπολογιστών

ΗΥ252– Προγραμματισμός Αντικειμενοστρεφής

Διδάσκων: Ι. Τζίτζικας

Χειμερινό Εξάμηνο 2020-2021

ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΗ ΕΡΓΑΣΙΑ ΣΤΟΝ ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΕΦΗ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ (ΗΥ-252)

ΓΕΩΡΓΙΟΣ ΠΑΝΑΓΙΩΤΗΣ ΜΕΛΛΙΟΣ

csd4416

09/12/2021

Εισαγωγή

Περιεχόμενα

1 Εισαγωγή

Για την συγκεκριμενη εργασια χρησιμοποιησα το μοντελο MVC (Model, View, Controller). Αυτό σημαινει ότι το προγραμμα χωριζεται σε τρια "πακετα" Model View και Controller αντιστοιχα. Το "πακετο Model περιεχει τις κλασεις και τις μεθοδους οι οποιες θα χρησιμοποιηθουν για την υλοποιηση του προγραμματος, κανοντας τις απαραιτητες λειτουργιες οι οποιες μας δινονται στην εκφωνηση της ασκησης. Στο πακετο "View" θα συναντησουμε τον κωδικα για την υλοποιηση των γραφικων GUI(Graphic User Interface) από το οποιο, μεσω καταλληλων κουμπιων θα δινονται εντολες (ή αλλιως εισοδοι) από τον/τους χρηστη/χρηστες ουτως ώστε να "παιξουν" το παιχνιδι. Τελος, στο πακετο Controller συμεπριλαμβανονται οι εντολες οι οποιες ενωνουν τα δυο παραπανω πακετα τα οποια και συνδυαζονται με καταλληλο τροπο, ουτως ώστε το προγραμμα να λειτουργει ορθα και με τις προδιαγραφες οι οποιες περιγραφει η εκφωνηση.

2 Η Σχεδίαση και οι Κλάσεις του Πακέτου Model

(Θα ξεκινησω την αναφορα αρχιζοντας από τις κλασεις-γονεις και "κατεβαινοντας" προς τις κλασεις-παιδια)

Αρχικα, εχουμε την κλαση Board στην οποια οριζουμε το ταμπλο του παιχνιδιου. Η κλαση αυτή, περιεχει μεθοδους οι οποιες υλοποιουν την αρχικοποιηση του ταμπλο αλλα και των παικτων, περιεχει μια μεθοδο η οποια "ριχνει το ζαρι", αλλα και έναν πινακα ο οποιος είναι τυπου Position (θα εξηγηθει παρακατω τι ειδους τυπος είναι αυτος) που περιεχει ολες τις θεσεις πανω στο ταμπλο (ο πινακας αυτος θα αρχικοποιειται στην initialize μεθοδο οπου και θα γινεται randomize, ουτως ώστε να επιτυχουμε την τυχαια σειρα των θεσεων στο ταμπλο κάθε φορα που αρχιζουμε ένα νέο παιχνιδι). Επειτα εχουμε την κλαση Card η οποια απεικονιζει τις καρτες του παιχνιδιου. Σε αυτην, υπαρχουν δυο δισδιαστατοι πινακες οι οποιοι περιεχουν τα δεδομενα κάθε καρτας (ειτε καρτων μηνυματος ειτε συμφωνιας). Οι πινακες αυτοι αρχικοποιουνται στην κατασκευαστρια μεθοδο της κλασης μεσω του ετοιμου κωδικα που μας δινεται στο moodle. Επειτα, εχουμε μια μεθοδο action() η οποια θα περιγραφει την λειτουργια κάθε καρτας, μιας και κάθε καρτα εχει μια διαφορετικη λειτουργια, και εχουμε μια μεθοδο money_of_the_card() η οποια θα επιστρεφει τα λεφτα (αν αυτά υπαρχουν) της συγκεκριμενης καρτας ουτως ώστε αυτά να χρησιμοποιηθουν για αυξηση-μειωση των χρηματων του παικτη. Άλλη μια βασικη κλαση είναι η κλαση Player. Όπως μαρτυραει και το ονομα, η κλαση αυτή απεικονιζει έναν παικτη του παιχνιδιου. Ο κάθε παικτης εχει στη διαθεση του: χρηματα, δανειο, καρτες συμφωνιας (τα πεδια αυτά αρχικοποιουνται μεσω της

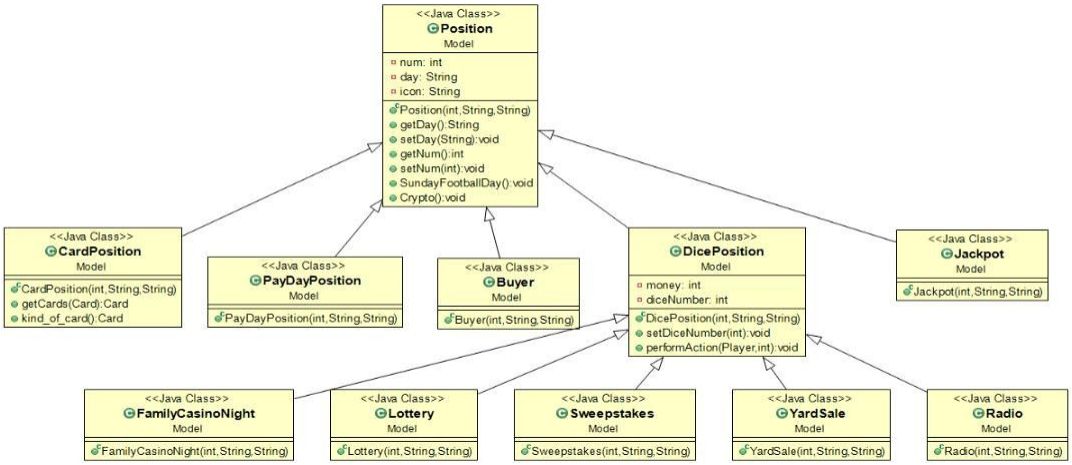
κατασκευαστριας μεθοδου). Εκτος αυτων, μεσω καποιων accessors και observers (για να τηρειται η αρχη της ενθυλακωσης) υπαρχει μια μεταβλητη turn η οποια θα καθοριζει αναλογα με την τιμη της (false ή true) εάν είναι η σειρα του συγκεκριμενου παικτη να παιξει. Επισης σε αυτην την κλαση εχουμε και ένα πεδιο win το οποιο αναλογα με την τιμη του μας φανερωνει αν ο συγκεκριμενος παικτης εχει κερδισει ή όχι. Παρακατω, εχουμε μια κλαση Position η οποια απεικονιζει την κάθε θεση του ταμπλο του παιχνιδιου. Κάθε θεση (Position) εχει έναν αριθμο στο ταμπλο, μια ημερα και ένα εικονιδιο. Αυτά αρχικοποιουνται μεσω της κατασκευαστριας μεθοδου της κλασης. Επειδη όμως κάθε θεση εχει και μια διαφορετικη λειτουργια, εχω φτιαξει επιπλεον τις κλασεις DicePosition και CardPosition. Η πρωτη κλαση απεικονιζει τις θεσεις οι οποιες χρειαζονται αλληλεπιδραση με το ζαρι του παιχνιδιου ουτως ώστε να γινει σωστα η εκαστοτε λειτουργια, ενώ η δευτερη απεικονιζει τις θεσεις στις οποιες όταν ο παικτης βρεθει σε μια από αυτές, πρεπει να τραβηξει μια καρτα ειτε συμφωνιας ειτε μηνυματος. Για να ξεχωριζουμε όμως τι ειδους καρτα θα πρεπει να τραβηχτει (στην περιπτωση που πεσουμε σε CardPosition προφανως) εχω φτιαξει μια μεθοδο kind_of_card η οποια μεσω του εικονιδιου που εχει η θεση του ταμπλο, θα ελεγχει αν ειμαστε σε θεση μηνυματος ή συμφωνιας. Επισης, εχω φτιαξει μια μεθοδο getCards η οποια αναλογα με το τι ειδους καρτα είναι, επιστρεφεται μια αντιστοιχη καρτα από την εκαστοτε στοιβα καρτων. Σε περιπτωση που ειμαστε σε DicePosition, υπαρχει μια μεθοδος performAction η οποια καλειται όταν πρεπει να γινει καποια λειτουργια αφου εχουμε πεσει σε DicePosition. Αυτή η μεθοδος όμως, λογω του ότι υπαρχουν θεσεις στις οποιες χρειαζεται η αλληλεπιδραση και των δυο παικτων ή του ενός, θα πρεπει να υπαρχει μια διαφορετικη της εκδοχη για κάθε τετοια περιπτωση. Αυτό ακριβως επιτυγχανεται με το overriding. Υπαρχουν κλασεις-παιδια της DicePosition (Lottery, Radio, Sweepstakes, YardSale, FamilyCasinoNight) οι οποιες θα κανουν override τη μεθοδο performAction οπου θα υλοποιειται η (ξεχωριστη) λειτουργια της κάθε καρτας. Εκτος αυτων των θεσεων όμως, υπαρχει και η θεση jackpot η οποια θα είναι επισης μια κλαση-παιδι της κλασης Position και θα υλοποιει τη θεση Jackpot του ταμπλο. Τελος για την κλαση Position, υπαρχουν καποιες επιπλεον ενεργειες για τις θεσεις που θα είναι ημερα Πεμπτη και Κυριακη οι οποιες θα υλοποιουνται στην κλαση Position μεσω των αντιστοιχων μεθοδων(SundayFootballDay, Crypto). Επισης, εχουμε καποιες υποκλασεις της κλασης Card, τις MailCard και DealCard οι οποιες απεικονιζουν τις καρτες μηνυματος και συμφωνιας αντιστοιχα. Αρα οι αναφερομενες από την εκφωνηση ως απαραιτητες μεθοδοι για την Α φαση της προγραμματιστικης εργασιας είναι:

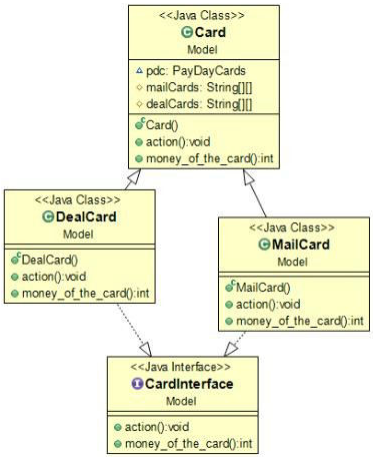
- Μεθοδος για την αρχικοποιηση του παιχνιδιου(ολες οι αρχικοποιησεις θα γινονται σε αυτην την κλαση, για παικτες, ταμπλο, πλακιδια-θεσεις) --> initialize της κλασης Board
- Μεθοδος που καθοριζει τη σειρα --> whose_turn της κλασης Board

- Μεθοδος που ελεγχει αν τελειωσε το παιχνιδι/νικητης --> win της κλασης Board
- Μεθοδος που υπολογιζει το σκορ κάθε παικτη ουσιαστικα θα είναι ο Observer set_euros της κλασης Player και το σκορ του κάθε παικτη θα υπαρχει στο πεδιο euros του κάθε παικτη το οποιο θα λαμβανουμε από τον Accessor get_euros

ΥΠΟΣΗΜΕΙΩΣΗ: Υπαρχουν ορισμενες και καποιες διεπαφες(interfaces) με javadoc σχολια τα οποια εξηγουν επισης τις λειτουργιες της κλασης που τις κανουν implement: BoardInterface, PlayerInterface, CardInterface. Επισης, στο Model υπαρχει και ο ετοιμος κωδικας που μας δοθηκε.

Παρουσιαζω τα UML διαγραμματα για την ιεραρχια κλασεων Card και Position των οποιων η επεξηγηση εγινε παραπανω.





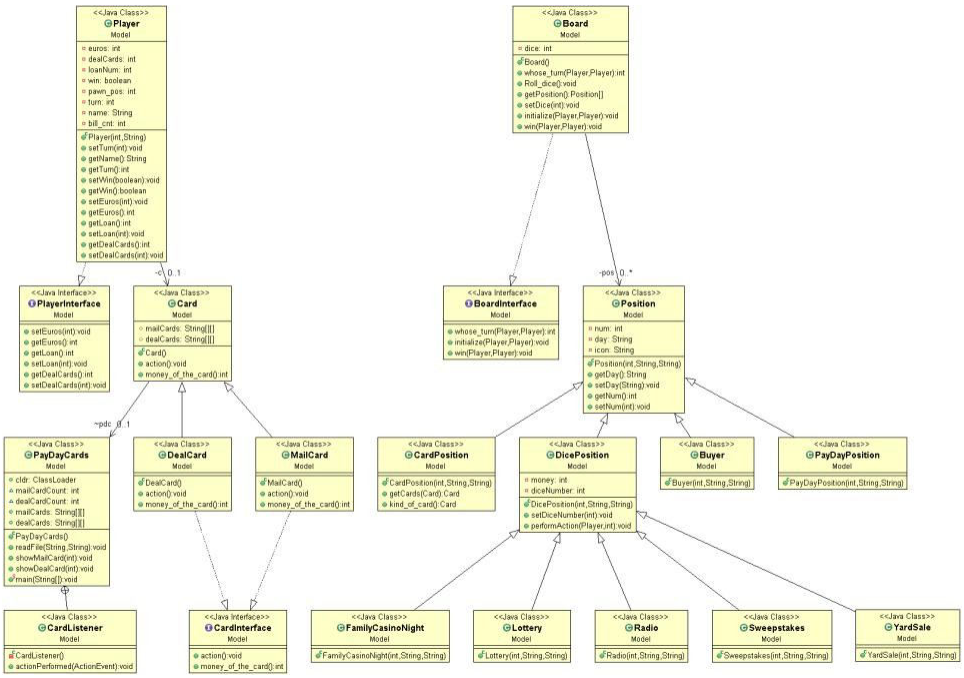
3 Η Σχεδίαση και οι Κλάσεις του Πακέτου Controller

Στο πακετο Controller θα υπαρχει η (μοναδικη) μεθοδος public static void main ολοκληρου του προγραμματος. Οι μεθοδοι που θα χρησιμοποιησω σιγουρα είναι αρχικα η initialize της κλασης Board από το πακετο Model, η κλαση Player από το ιδιο πακετο και καποιες μεθοδους της κλασης αυτης (οι μεθοδοι αυτοι θα χρησιμοποιουνται “αυτοματα” μεσω των αλλων ηδη υλοποιημενων κλασεων του πακετου). Επισης, θα χρησιμοποιησω την (κατασκευαστρια) μεθοδο View του πακετου View, η οποια υλοποιει την γραφικη απεικονιση του παιχνιδιου. Τελος, με καταλληλο συνδυασμο των δυο παραπανω, σε αυτό το πακετο, θα υπαρχει η τελικη υλοποιηση του προγραμματος.

4 Η Σχεδίαση και οι Κλάσεις του Πακέτου View

Το πακετο View θα περιεχει μια κλαση View στην οποια θα υλοποιειται οσο το δυνατον καλυτερα η δοσμενη απο την εκφωνηση εικονα του παιχνιδιου(δηλαδη θα υλοποιουνται τα γραφικα του προγραμματος). Επισης θα περιεχει μια κλαση Pawn η οποια θα οπτικοποιει τα πιονια των παικτων.

5 Η Αλληλεπίδραση μεταξύ των κλάσεων – Διαγράμματα UML

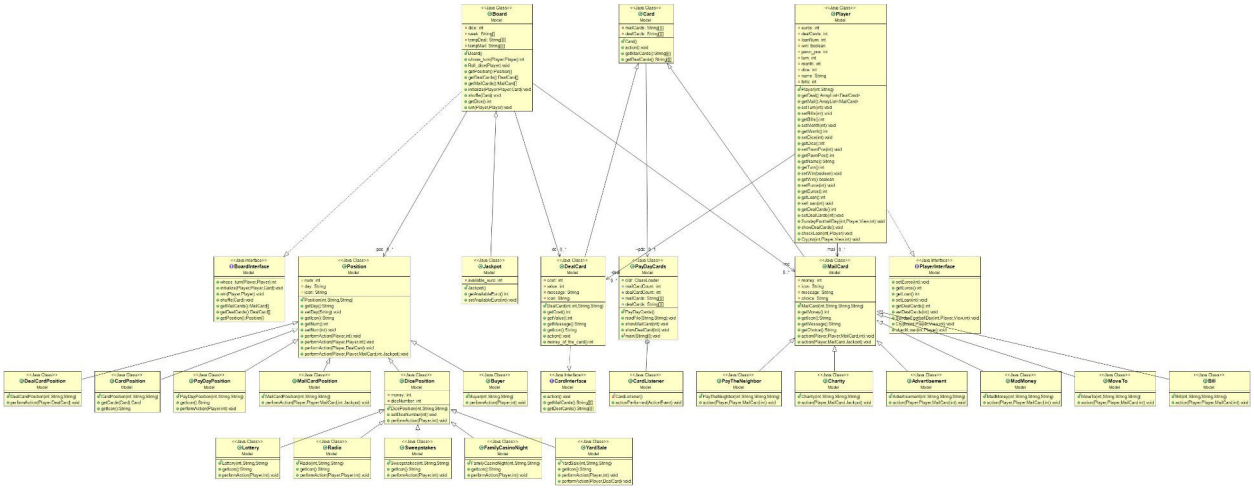


Εδω βλεπουμε το διαγραμμα UML το οποιο δειχνει πως συνδεονται οι κλασεις στο Model. (Στην επομενη φαση θα παρουσιαστει μια πιο ολοκληρωμενη μορφη του διαγραμματος αυτου, μιας και θα εχουν υλοποιηθει και οι κλασεις-πακετα των Controller και View). Πιο συγκεκριμενα παρατηρουμε οτι η κλαση Player συνδεεται με την κλαση Card μιας και καθε παικτης θα εχει στη διαθεση του καρτες προσφορας (αφου εχει παει στο αναλογο Position του ταμπλο). Η ιεραρχια της κλασης Card εξηγηθηκε παραπανω, στην ενοτητα 2. Επισης, βλεπουμε οτι η κλαση Board συνδεεται με την κλαση Position μιας και το ταμπλο(Board) εχει - αν μη τι αλλο - δυνατες θεσεις στις οποιες ειναι δυνατον να παει ο παικτης. Η ιεραρχια της κλασης Position και των υποκλασεων της εξηγηθηκαν επισης στο κεφαλαιο 2 της αναφορας.

6 Λειτουργικότητα (Β Φάση)

Στην εργασια μου καταφερα να υλοποιησω σχεδον ολα τα ζητουμενα (νομιζω) επιτυχως, εκτος απο την καρτα “πηγαινε στην κοντινοτερη θεση αγοραστη”. Επισης, η υλοποιηση μου εχει πολυ μικρο ελεγχο σε λαθη και ειναι καπως ευαλωτη σε τυχον λαθος εισοδος (οχι σε ολες τις περιπτωσεις αλλα σε αρκετες). Στις περιπτωσεις που υπαρχει ελεγχος, ο χρηστης δεν ειδοποιειται για το σφαλμα, αλλα του ζητειται να εισαγει ξανα τις τιμες που πρεπει (πχ. για την θεση λοτταριας, εαν ενας απο τους δυο αριθμους ειναι εκτος οριων 1-6 που μπορει να φερει το ζαρι, τοτε πρεπει και οι δυο να εισαγουν και παλι νεες τιμες). Παρομοια ειναι η αντιμετωπιση και στην επιλογη μηνα (εαν ειναι εκτος οριων 1-3 ο χρηστης πρεπει να εισαγει νεα τιμη). Επισης, στις θεσεις ζαριου(λαχειο, λοτταρια, ραδιο κλπ.) εμφανιζω ενα παραθυρο τυπου JoptionPane και ειδοποιω τον χρηστη για το τι εφερε το ζαρι. Επομενως σε αυτες τις θεσεις, το ζαρι ριχνεται αυτοματα και οχι “χειροκινητα”απο τον χρηστη και ο χρηστης ειδοποιεται για το συμβαν αυτο. Οσον αφορα το τελος του παιχνιδιου, για να ανακηρυχθει ο νικητης, πρεπει και οι δυο να εχουν φθασει στον αναλογο μηνα που πρεπει, να εχουν φθασει στην τελευταια θεση(PayDay position) και μετα οταν ενας εκ των δυο πατησει το Roll Dice κουμπι, ανακηρυσσεται ο νικητης και το παιχνιδι τελειωνει. Οταν ενας εκ των δυο φθασει στην τελευταια θεση στον τελευταιο μηνα, ειδοποιειται οτι τελειωσε και δεν μπορει να συνεχισει το παιχνιδι, παρολα αυτα λαμβανει μερος σε τυχον διαγωνισμους. Το γραφικο περιβαλλον το εχω υλοποιησει κανονικα. Το μονο που θα μπορουσε κανεις να πει οτι λειπει ειναι το Info Box που εχει δωθει στην προτυπη υλοποιηση, και αυτο, επειδη το θεωρησα περιττο, μιας και εχω παντου ενα JoptionPane για να ειδοποιω τον χρηστη για καθε τυχουσα ενεργεια. Στην υλοποιηση μου σχετικα με την αναφορα της Α φασης, αλλαξα μερικα πραγματα. Αυτα ειναι οτι προσθεσα τις μεθοδους Crypto και SundayFootballMatch οι οποιες υλοποιουν αυτα που μαρτυραει το ονομα τους, εφτιαξα επιπλεον κλασεις για καθε καρτα μηνυματος (advertisement, charity, bill κλπ.) και εφτιαξα επιπλεον δυο κλασεις-θεσεις, τις MailCardPosition και DealCardPosition οι οποιες υλοποιουν θεσεις καρτας

μηνυματος/συμφωνιας. Στην κλαση DealCardPosition, εχω κανει την μεθοδο performAction η οποια γινεται override απο την υπερκλαση Position και υλοποιει τις καρτες συμφωνιας. Στην κλαση MailCardPosition γινεται το ιδιο, μονο που μεσω της performAction καλειται μια αλλη μεθοδος, η action, η οποια κανει καποια δραση, αναλογα με το τι ειδους καρτα μηνυματος ετυχε στον παικτη. Επισης, σε σχεση με την Α' φαση αλλαξα την μεθοδο performAction και της εδωσα πολλες υπογραφες με διαφορετικα ορισματα (πολυμορφισμος) οι οποιες χρησιμοποιουνται ολες καταλληλα, η καθεμια για διαφορετικο σκοπο.



Εδω βλεπουμε την τελευταια εκδοση του UML διαγραμματος για το πακετο Model εχοντας προσθεσει τις τελευταιες κλασεις-μεθοδους. Τα πακετα Controller και View εχουν μια κλαση και πολυ λιγες μεθοδους(σε σχεση με το model) και για αυτο δεν επισυναψα τα αντιστοιχα UML διαγραμματα.

7 Συμπεράσματα

Στην υλοποιηση του Controller στην μεθοδο ActionListener το προβλημα που αντιμετωπισα(το οποιο θα μπορουσα να το αντιμετωπισω γραφοντας μια μεθοδο με καταλληλα ορισματα) ηταν οτι επρεπε για καθε κουμπι του καθε παικτη να γραψω τον ιδιο κωδικα απλα ξεχωριστα για καθε παικτη. Οπως προειπα αυτο θα μπορουσε να υλοποιηθει διαφορετικα, αλλα λογω ελλειψης χρονου (και λογω του οτι το αντιληφθηκα καπως αργα αυτο) αφησα τον κωδικα ετσι.