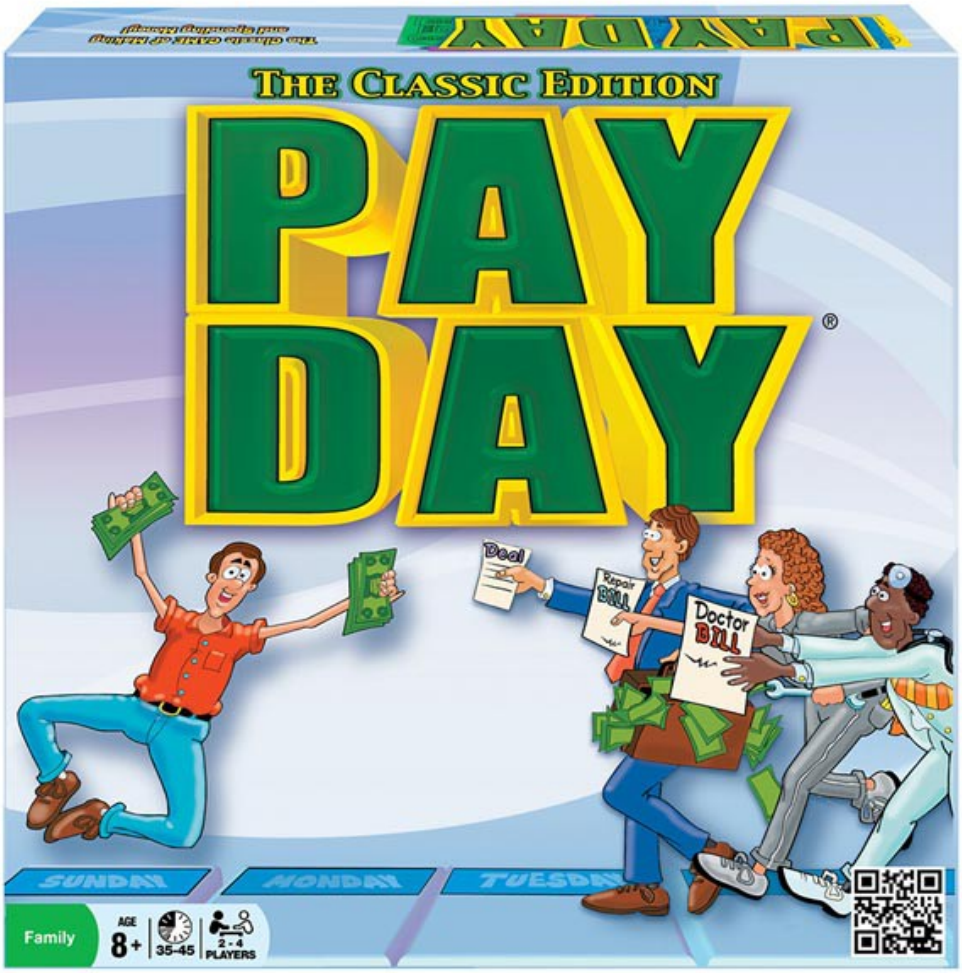
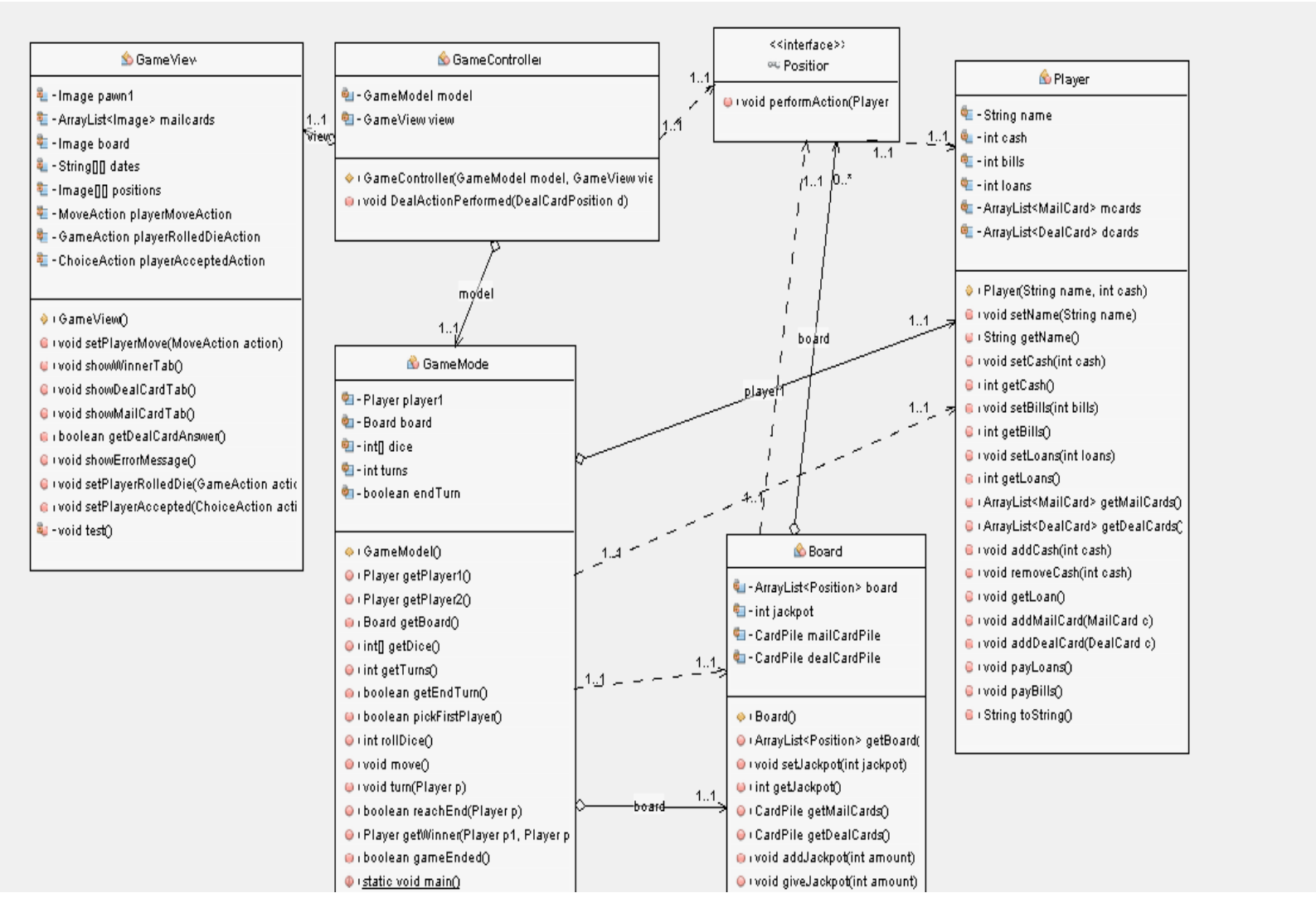
Java HY-252
Project 2016-2017



George Fotopoulos 3317

Σχεδιασμός Εργασίας

Η εργασία αυτη βασιζεται στο MVC μοντέλο.
Εχει λοιπόν έναν Controller που ενώνει το Model
και το view. Σε αυτην την αναφορα θα αναφερθω
κυρίως στο Model.
(Το παρακατω ειναι μια απλουστευμενη εκδοση του UML του project , παρ' ολα
αυτα πανω σε αυτο εχω δουλεψει.)



Model Package

Cards Sub-Package

Περιέχει την διεπαφή Card τις κλάσεις
CardPile , MailCard και DealCard , όπως
και την enum Type.

Αρχικα το interface Card περιέχει τις εξής μεθόδους:

1. public void setType(Type type); Transformer (Mutative)
Sets the type of the card.
2. public Type getType(); Accessor (Selector)
Returns type of the card.
3. public void setMessage(String message);Transformer(Mutative)
Sets the message of the card.
4. public String getMessage(); Accessor (Selector)
Returns the message of the card.
5. public void setImage(Image image); Transformer (Mutative)
Sets the image of the card.
6. public Image getImage(); Accessor (Selector)
Returns the image of the card.

Έπειτα έχουμε τα MailCard και DealCard που υλοποιουν την Card και χρησιμοποιουν το enum Type:

Πρώτον το MailCard εχει κάποια attributes και μερικες αλλες μεθόδους (πέρα απο αυτές της Card).

MailCard attributes:

1. private Type type; // the type of a MailCard
2. private String message; // the message of a MailCard
3. private String choice; // the choice of a MailCard
4. private int euro; // the amount of money in a MailCard
5. private Image image; // the image of a MailCard

Υπόλοιποι μέθοδοι της MailCard:

1. public void setChoice(String choice); Transformer (Mutative)
Sets the choice of the card
2. public String getChoice(); Accessor (Selector)
Returns the choice of the card
3. public void setEuro(int euro); Transformer (Mutative)
Sets the euros of the card
4. public int getEuro(); Accessor (Selector)
Returns the euros of the card

Class Type

Είναι μια enum κλάση που περιέχει τους τύπους των καρτών (PayTheNeighbor, MadMoney, Charity, Bill, MoveToBuyer, Advertisement, Deal).

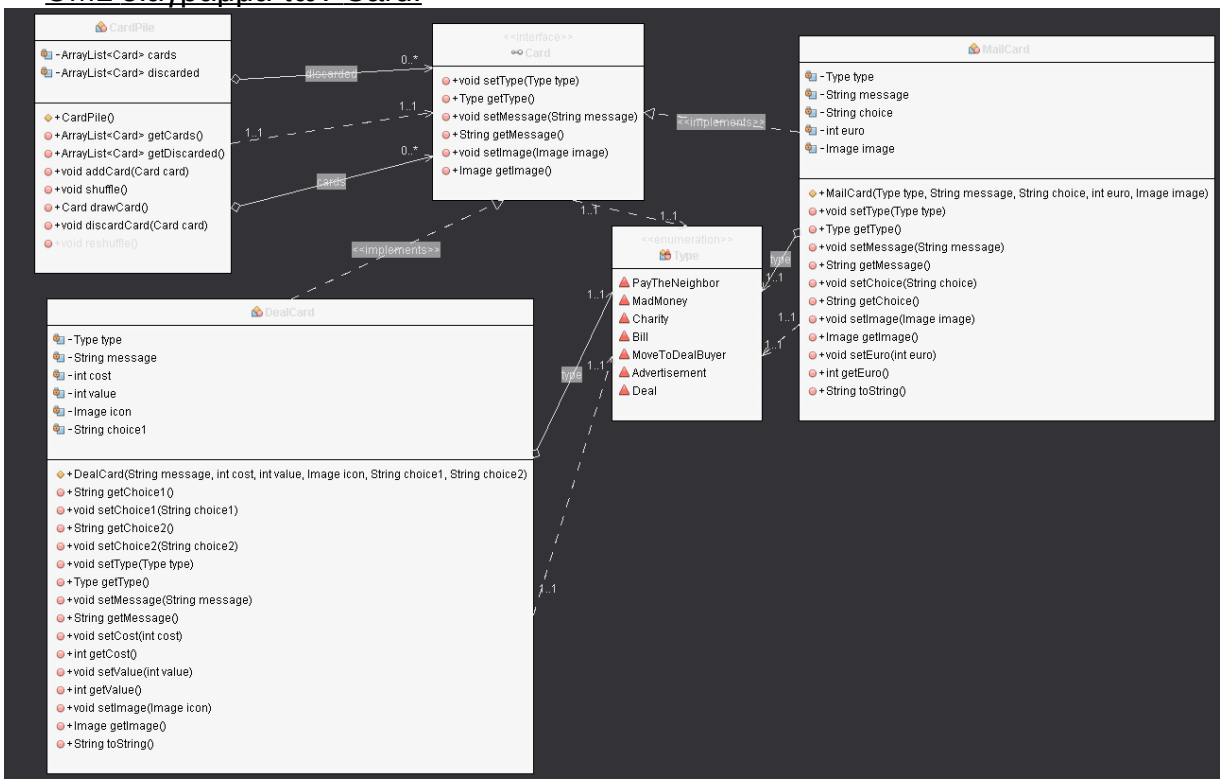
DealCard attributes:

1. private Type type; //the type of a DealCard (μόνο το Deal εδώ)
2. private String message; //the message of a DealCard
3. private int cost; //the cost of a DealCard (to buy it)
4. private int value; //the value of a DealCard (to sell it)
5. private Image image; //the image of a DealCard
6. private String choice1, choice2; //the 2 choices of a DealCard (whether to buy it or not)

Υπόλοιποι μέθοδοι της DealCard:

1. public void setCost(int cost); Transformer (Mutative)
Sets cost of the card
2. public int getCost(); Accessor (Selector)
Returns cost of the card
3. public void setValue(int value); Transformer (Mutative)
Sets value of the card
4. public int getValue(); Accessor (Selector)
Return value of the card
5. public void setChoice1(String choice1); Transformer (Mutative) (choice2 too)
Sets choice 1 of the card
6. public String getChoice1(); Accessor (Selector) (for choice2 too)

UML διάγραμμα των Card:



Position Sub-Package

Περιέχει την διεπαφή Position και τις κλάσεις CardPosition , DealPosition και PayDayPosition (οι CardPosition και DealPosition βρισκονται σε ξεχωριστα ύπο-ύπο-packages του model και εχουν και ειναι και οι δυο abstract με αρκετες υποκλάσεις η καθεμία).

Το interfrace Position περιέχει τις εξής μεθόδους(μια μονο :p):

1. performAction(Player p); Transformer (Mutative)
Πραγματοποει συγκεκριμένες ενέργιες που για κάθε Position είναι διαφορετικές (Αναφέρω συγκεκριμένα τι κανει η κάθε μια στα javadoc σχόλια της ασκησης)

Θα πω αρχικά για το PayDayPosition και το SundayFootballMatch που δεν εχουν δικο τους ύπο-package:

Δεν περιεχουν τιποτα παραπάνω απο το performAction (και constructor).

Μετά εχουμε την DicePosition abstact class που εχει υποκλάσεις τις OnePlayerDicePosition και TwoPlayerDicePosition οι οποίες χρησιμοποιουν το enum TypeDicePosition

Η DicePosition περιέχει μονο τις PerforcmAction , toString και τον constructor της.

Και η OnePlayerDicePosition και η TwoPlayerDicePosition περιεχουν σαν attribute to private TypeDicePosition type; και εχουν ακριβως τις ίδιες μεθόδους με την DicePosition συν getters και setters για το type.

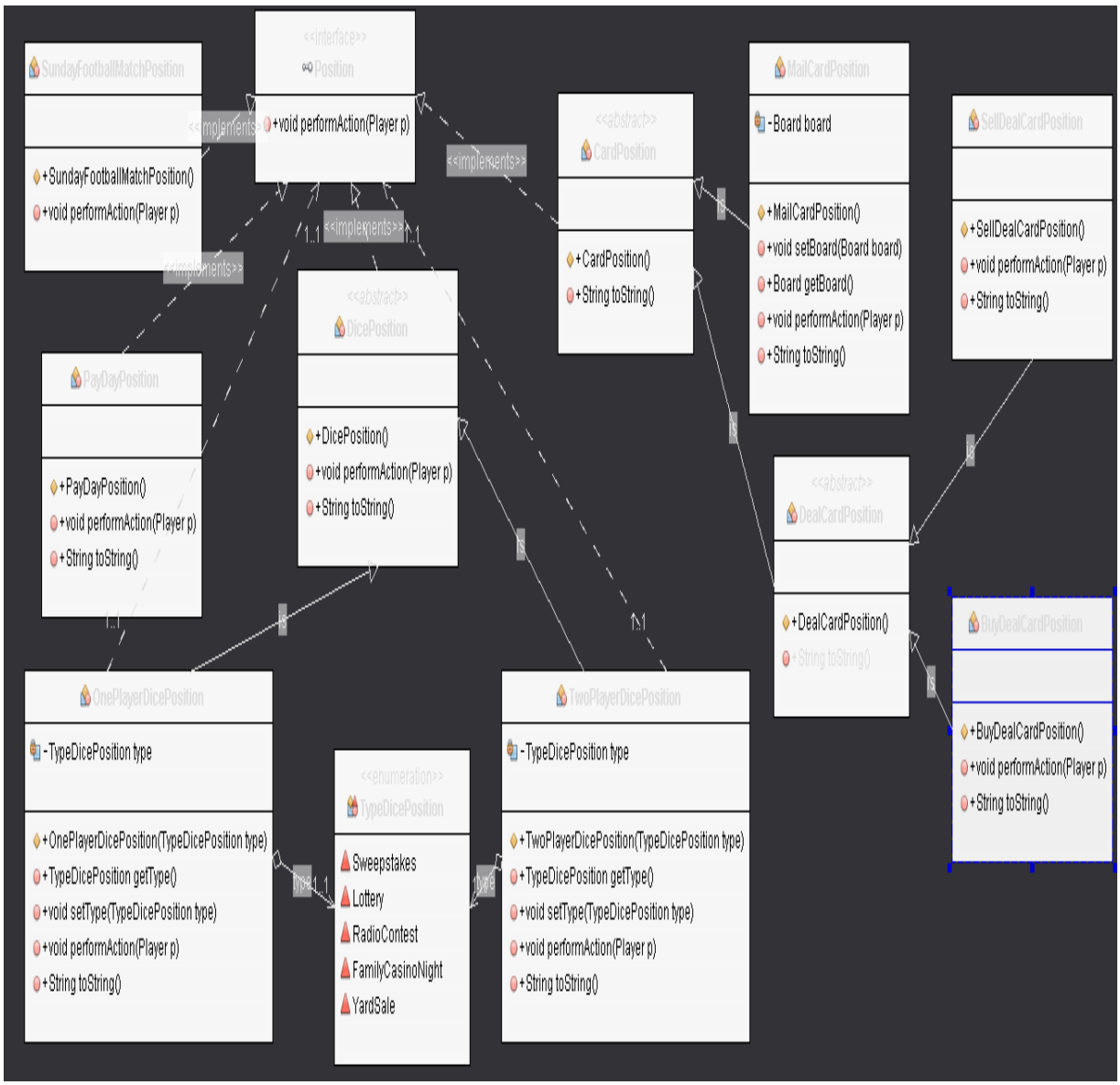
Class TypeDicePositon

Είναι μια enum κλάση που περιέχει τους τύπους των DicePosition (Sweepstakes, Lottery, RadioContest, FamilyCasionNight και YardSale).

Ακολουθεί η CardPosition abstract class που εχει την υπο-κλάση MailCardPosition και την abstract DealCardPosition , η οποία έχει τις υποκλάσεις BuyDealCardPosition και SellDealCardPosition

Η CardPosition δεν περιέχει τίποτα πέρα απ τον constructor και την toString
Το το ίδιο και η abstract DealCardPosition.
Μετά το μόνο παραπανω που εχει η MailCardPosition, η SellDealCardPosition και η BuyDealCardPosition ειναι η performAction(Person p);.
Τέλος η MailCardPosition περιέχει τα ιδια με τις δύο παραπανω τους setters και getters για το board (ειναι reference στην Board κλάση).

Παρακάτω βλέπουμε το UML διάγραμα των Position.



Rest Packages

Μέτα έχουμε τον Player

Player attributes:

1. private String name; //το όνομα του Player
2. private int cash; //τα λεφτά του Player
3. private int bills; //οι λογαριασμοί που έπεται να πληρώσει ο Player
4. private int loans; //τα δάνεια που εχει πάρει ο Player
5. private ArrayList<MailCard> mcards; //η λίστα με τις MailCards του Player
6. private ArrayList<DealCard> dcards; //η λίστα με τις DealCards του Player
7. private int positionIndex; // η θεση του Player στο ταμπλό
8. private int months; //οι μήνες που εμειναν στον Player
9. private int dieRoll; //η ζαριά του Player
10. private int selectedNumber; //ο αριθμός που διαλεξε ο Player
11. private boolean accepted; //μια enum για το αν αποδεχτηκε ο Player κατι
12. private int bet; //το στοίχημα που εβαλε ο παιχτης για τον αγώνα Κυριακής
13. private Player opponent; //ο αντίπαλος του Player
14. private int jackpot; //η μεταβλητη για το reference στο jackpot

Player methods:

1. public void setName(String name); Transformer (Mutative)
Sets the name of the Player
2. public String getName(); Accessor (Selector)
Returns the name of the Player
3. public void setCash(int cash); Transformer (Mutative)
Sets the cash of the Player
4. public int getCash(); Accessor (Selector)
Returns the cash of the Player
5. public void setBills(int bills); Transformer (Mutative)
Sets the bills of the Player
6. public int getBills(); Accessor (Selector)
Returns the bills of the Player
7. public void setLoans(int loans); Transformer (Mutative)
Sets the loans of the Player
8. public int getLoans(); Accessor (Selector)
Returns the loans of the Player
9. public ArrayList<MailCard> getMailCards(); Accessor (Selector)
Returns the ArrayList of the MailCards of the Player
10. public ArrayList<DealCard> getDealCards(); Accessor (Selector)
Returns the ArrayList of the MailCards of the Player
// συν ολοι οι setters και getters που προστεθηκαν μετά
11. public void addCash(int cash); Transformer (Mutative)
Adds cash to Player's cash
12. public void removeCash(int cash); Transformer (Mutative)
Removes cash from Player's cash (getLoan(); if needed)
13. public void getLoan(); Transformer (Mutative)
Adds a x1000 loan to Player's loan and the same amount to his cash
14. public void addMailCard(MailCard c); Transformer (Mutative)
Adds a MailCard to Player's MailCard ArrayList

15. public void addDealCard(DealCard c); Transformer (Mutative)
Adds a DealCard to Player's DealCard ArrayList
16. public void payLoans(); Transformer (Mutative)
Pays (removes from Player's loan) a percentage of the loan at the end of the month, or the whole of it if possible , if Player can't pay the percentage needed then takes another loan
17. public void payBills(); Transformer (Mutative)
Pays (makes Player's bills zero) the Player's bills and takes a loan if needed
18. public void move(int die);
Moves Player according to the die roll
19. public void moveTo(int index);
Moves Player to a certain position
20. public void moveToStart();
Moves Player backto the beginning
21.public int throwDie();
Returns a randomly chosen number between 1 and 6

Afterwards we have the class Board

Board attributes:

1. private ArrayList<Position> board;//ArrayList with the Positions on the board

Board methods:

1. public ArrayList<Position> getBoard(); Accessor (Selector)
Returns Board's ArrayList of Positions
2. public void shuffle(); Transformer (Mutative)
Shuffles all the positions on the board except the first and the last

The Jackpot class

Its only attribure is an int which is the amount of money there are in the jackpot. Then we have the setters and getters. Plus 2 more methods giveJackpot which gives everything contained in jackpotto a Player and addJackpot that it adds money to the amount of the jackpot.

Then the GameModel

Here is where everything needed for the game is created , both players, the board and jackpot . Also in its constructor it calls restart where the months will happed and the other actions needed for each Player's turn.

Model attributes:

1. private Player[] players; //the Array for the 2 players (added later)
2. private Board board; //the board of the game
3. private int activePlayerIndex; // the int for which player plays (its 0 or 1)
4. private int months; //how long will the game last (1-3 turns)
5. private boolean endTurn; //a boolean for when a Player's turn has ended
6. private Jackpot jackpot; //a jackpot

7. private CardPile mailCardPile; //a cardpile of mailcards
8. private CardPile dealCardPile; //a cardpile of dealcards

Model methods:

There are all the setters and getters for each attribute.
public boolean pickFirstPlayer(); picks which is the first player that will play randomly.
public int rollDice(); returns a randomly chosen number between 1 and 6.
public void move(int dice); moves player on the board and perfoms the action on the position landed.
public void turn(); rolls the die , moves the player then switches active player.
public void switchPlayers(); changes the activePlayerIndex from 0 to 1 or from 1 to 0.
public boolean reachEnd(Player p); checks if a player has reached payday position as well as if its the last month that shall be played.
public Player getWinner(Player p1, Player p2); returns the player of the 2 that have won , only if game has ended though.
public boolean gameEnded(Player p1, Player p2); checks if both players have reached end.
private void loadMailCards(); loads all mail cards from a file to the mail card pile
private void loadDealCards(); loads all deal cards from a file to the deal card pile

GameView

Το GameView ειναι ουσιαστικα ολο το visual κομμάτι του παιχνιδιού. Περιέχει τις συναρτήσεις __Action(); και τα __Action interfaces που περιέχει το package του view. Για την επικοινωνία μεταξύ view-controller έχουν οριστεί αυτά τα interfaces με μία μέθοδο performAction στην οποία παρεχεται η καταλληλη παράμετρος για να παρέχει στα controller τα αντίστοιχα δεδομένα. Πχ το interface PlayerRolledDieAction χρησιμοποιείται για να σηματοδοτήσει στον controller οτί ο παίχτης έχει ρίξει το ζάρι. Η μέθοδος performAction αυτού του interface δέχεται μια int παράμετρο η οποία παρέχει στον controller τον αριθμό που εφερε το ζάρι. Ο αριθμός αυτός πρεπει να είναι μεταξυ του 1 και 6.

Μεσα στον View εχω δημιουργησει μια μικρη κλάση αποτελούμενη απο panes και labels , τα οποια αργοτερα στην doFreshStart(); θα βαλω ολα σε ενα πινακα ετσι ωστε να προστιθενται στον πίνακα και να μπερδεύονται σε τυχαιες θέσεις.

Στην doFreshStart(); που αναφερθηκε ειναι επίσης και το μέρος οπου θα αρχικοποιηθουν τα πιόνια στην θέση εκκινησης και αργότερα θα μετακινούνται μεσω της updatePlayerPosition.

Υπαρχει η μέθοδος setDieNumber η οποία παιρνοντας τον Player και την ζαριά απ τον Controller θα αλλαξει το εικονίδιο του ζαριού του κάθε παιχτη.

Για το παραθυράκι που εμφανιζεται στις Κυριακης για τον αγώνα ποδοσφαιρου εχω κανει την requestBet η οποία θα εμφανισει το παραθυρακι και θα επιστρέψει την απαντηση του παιχτη.

Για την εμφανιση των καρτών κάθε φορα που θα πατηθει το label τους εχω κανει μια showMailCard και μια showDealCard , οι οποιες θα εμφανίσουν την καρτα που εχει μολις τραβηχθει απ τον παιχτη.

Έπειτα για τα labels των παιχτών και του info panel εχω κανει διαφορους setters οι οποιοι χρησιμοποιουνται απ την controller ετσι ωστε να αλλαξει η τιμη τους καθε φορα που πρεπει.

Εχω ρυθμιση την view (και τον controller) καθε φορα που χρειάζεται να γινει η οποιαδηποτε κινηση (τραβηγμα καρτας ή ρηψη ζαριας) να αποενεργοποιουνται ολα τα κουμια εκτος απο αυτο που πρεπει να χρησιμοποιηθει.

Τέλος ολη την σχεδιαση των panels labels κλπ εγινε μέσω του design panel που προσφερει το Netbeans.

Controller

Controller methods:

1. private GameModel model; //a GameModel object
2. private GameView view; //and a GameView object

Εδω υπάρχουν Player___Action(); που κανουν implements το καθενα την δικιά του interface απ το GameView package. Ουσιαστικα αυτές ενώνουν τις αντίστοιχες συναρτήσεις που υπαρχουν στο view με το model μέσω της performAction της καθεμίας.

Αρχικα έχουμε την συναρτηση startGame που αρχικοποιει το board , τα labels , και βάζει τον παίχτη να παιξει.

Μεσα σε καποια απο τα Player Action χρησιμοποιώ δυο μεθόδους την updateBoard και την gameStatus. Η πρώτη κάθε φορα που θα τελειωσει ο κάθε γυρος θα ανανεωσει τις τιμες σε ολα τα labels και στο jackpot. Η δευτερη σε καθε τελος γυρου θα τσεκαρει σε ποια φαση του παιχνιδιου βρισκομαστε και θα πει στον επομενο παιχτη (οποτε χρειαζεται) να παιξει , η θα ανακοινωσει νικητη.


Τέλος!