**Πανεπιστήμιο Κρήτης –Τμήμα Επιστήμης Υπολογιστών**

**ΗΥ252– Αντικειμενοστρεφής Προγραμματισμός**

**Διδάσκων: Ι. Τζίτζικας**

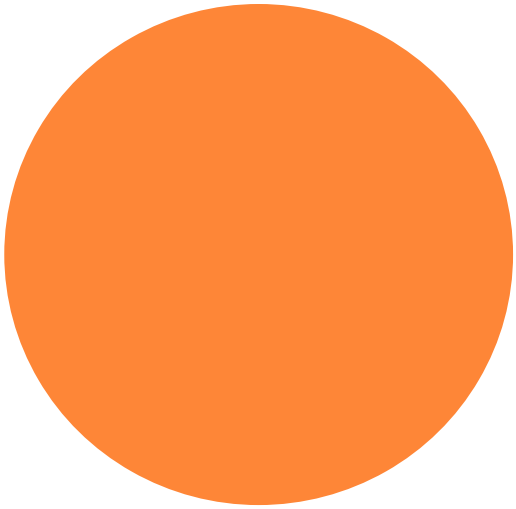
**Χειμερινό Εξάμηνο 2020-2021**

# [HY252 PROJECT 2021-2022]

[Κουμακης Εμμανουηλ]
[csd4281]
[/1/2022]

# Περιεχόμενα





# 1. Εισαγωγή

Σε αυτο το Project μας ζητειται να υλοποιησουμε το επιτραπεζιο παιχνιδι “PayDay”.Για την υλοποιηση του βασιζομαστε στο μοντελο MVC το οποιο ειναι ενα μοντελο αρχιτεκτονικης λογισμικου. Σε αυτο το μοντελο η εφαρμογη που φτιαχνουμε χωριζεται σε τρια μερη (Model – View – Controller).

-Στο Model υλοποιουμε την λειτουργικοτητα του παιχνιδιου. Πιο συγκεκριμενα το model διαχειριζεται τα δεδομενα του παιχνιδιου (δηλαδη το πως δουλευουν οι καρτες, πως μετακινουνται τα πιονια στο ταμπλο κτλ).

-Στο View υλοποιουμε το γραφικο κομματι του προγραμματος το οποιο ειναι αυτο που βλεπει τελικα ο χρηστης.

-Στο Controller υλοποιουμε κομματια κωδικα που ενορηχστρωνουν την επικοινωνια του Model με το View. Μπορουμε να πουμε οτι ο Controller λειτουργει ως ο εγκεφαλος του προγραμματος.

Στην αναφορα αυτη θα αναλυσουμε τα κομματια του Model, Controller και View καθως θα δουμε και καποια διαγραμματα UML.

Στην Β φαση του προτζεκτ η αναφορα ανανεωθηκε. Οτι καινουριο προστεθηκε (μεταβλητες, μεθοδοι) σημειωνεται με ενα : + Οτι αλλαγες πραγματοποιηθηκαν σε μεθοδους σε σχεση με την πρωτη φαση (τι κανουν τελικα ή γιατι αφαιρεθηκαν) σημειωνονται με ενα : -

## 2. Η Σχεδίαση και οι Κλάσεις του Πακέτου Model

Σε αυτο το πακετο περιεχονται οι παρακατω κλασεις: Abstract κλαση card, οι κλασεις DealCard και MailCard οι οποιες επεκτεινουν την κλαση card καθως και αρκετες κλασεις που την επεκτεινουν κλαση MailCard τις οποιες θα περιγραψουμε παρακατω. Υπαρχει επισης η abstract κλαση tile με 10 ακομη υπο-κλασεις να την επεκτεινουν. Τελος υπαρχουν και οι

κλασεις player, Board , jackpot, dice, MailDeck και DealDeck τις οποιες θα δουμε αναλυτικα παρακατω.

## Abstract class card and all other sub-classes about card

Η abstract κλαση card περιεχει τα attributes:

```
1)private String Type;     //Το ειδος μιας καρτας (Deal ή καποιο
                                                    ειδος Mail)

2)private String message; //Το μηνυμα το οποιο θα εχει πανω η καρτα

3)private String icon;   // Το path για το εικονιδιο της καρτας
```

Και τις παρακατω μεθοδους:

### Transformers:

```
1. public abstract void performAction(player p); //Μεθοδος την
οποια ολες οι καρτες κληρωνομουν και πρεπει να υλοποιησουν με
διαφορετικο τροπο προκειμενου να πραγματοποιηθει η ενεργεια της
καθε καρτας.

2. public void SetType(String Type); //Οριζει το ειδος μιας καρτας

3.public void setMessage(String message); //Οριζει το μηνυμα μιας
                                                    κρατας

4.public void setIcon(String icon); // Οριζει το μονοπατι για το
                                        εικονιδιο μιας καρτας
```

### Accessors:

```
5. public String getType();      //Επιστρεφει το ειδος μιας καρατας

6. public String getMessage();  //Επιστρεφει το μηνυμα μιας καρτας

7. public String getIcon()     //Επιστρεφει το μονοπατι για το
                                                    εικονιδιο μιας
καρτας
```

Στην συνεχεια βλεπουμε τις κασεις DealCard και MailCard οι οποιες επεκτηνουν την αφηρημενη κλαση card:

## Class DealCard :

Η κλαση DealCard περιεχει τα attributes:

1) **private int** value; //Η τιμη αγορας της καρτας

2) **private int** cost; //Η τιμη πωλησης της καρτας

3) **private** String choice1, choice2; //Θα χρησιμοποιηθουν για να παρεχεται η δυνατοτητα στον παικτη να διαλεξει αν θελει να κρατησει ή να απορριψει την καρτα

Και τις παρακατω μεθοδους:

### Constructor:

//Δημιουργει ενα στιγμιοτυπο της κλασης αρχικοποιωντας ολες τις απαραιτητες μεταβλητες

**public** DealCard(String message, String icon, **int** cost, **int** value)

### Transformers:

1.**public void** performAction(player p); //Ο παικτης διαλεγει αν επιθυμει να κρατησει αυτη την καρτα, επομενως τοποθετειται στην στην στοιβα με τις καρτες του αλλιως την βαζουμε στην στοιβα απορριψης

### Accessors:

2. **public int** getCost(); //Επιστρεφει την τιμη αγορας της καρτας

3. **public int** getValue(); //Επιστρεφει την τιμη πωλησης της καρτας

4. **public** String getChoice1(); // Επιστρεφει το μηνυμα της πρωτης επιλογης (για το κουμπακι που θα παταει ο παικτης οταν επιθυμει να κρατησει την καρτα)

5. **public** String getChoice2(); //Επιστρεφει το μηνυμα της δευτερης επιλογης

## Class MailCard :

Η κλαση MailCard περιεχει τα attributes:

**1)** **private** String choice; //Για το κουμπακι που θα παταει ο παικτης οταν τραβαει αυτη την καρτα

**2)** **private int** Euro; //Το ποσο που πληρωνει (ή που παιρνει) ο παικτης οταν τραβαει την καρτα

Και τις παρακατω μεθοδους:

### Constructor:

//Δημιουργει ενα στιγμιοτυπο της κλασης αρχικοποιωντας ολες τις απαραιτητες μεταβλητες

**public** MailCard(String Type, String message, String icon, String choice, **int** Euro)

### Transformers:

**1. public void** performAction(player p) //Δεν κανει κατι εδω,

γινεται overide απο την υπερκλαση (card) και την χρειαζομαστε στις υποκλασεις της MailCard ο οποιες θα την υλοποιησουν με διαφορετικο τροπο η καθε μια

### Accessors:

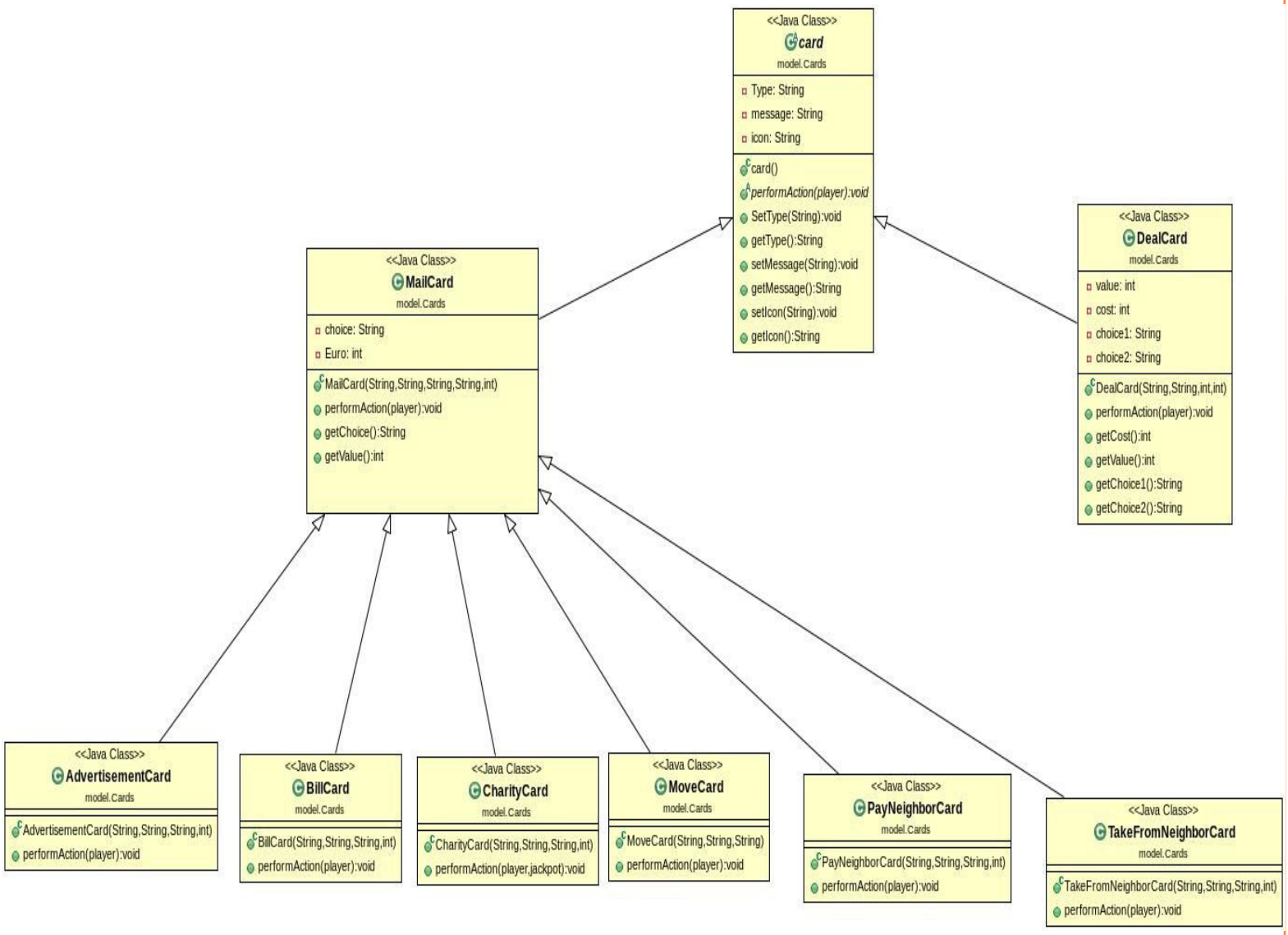
2.**public** String getChoice() // Επιστρεφει το string για το κουμπακι που θα παταει ο παικτης

3. **public int** getValue() //Επιστρεφει την αξια της καρτας (ποσο που λαμβανει ο παικτης ή που πληρωνει)

## Class AdvertisementCard, Class BillCard, Class CharityCard, Class MoveCard, Class PayNeighborCard, Class TakeFromNeighborCard:

Αυτες οι κλασεις επεκτεινουν την κλαση MailCard και μεσω της εντολης super αποκτουν προσβαση στην κλαση MailCard αρχικοποιωντας ετσι τις απαραιτητες μεταβλητες. Καθε μια πο αυτες τις κλασεις κανει overide την μεθοδο "performAction()" προκειμενου να πραγαματοποιηθει η καταληλη ενεργεια που οριζει η καθε καρατα.

Τελος εδω βλεπουμε μια αναπαρασταση των κλασεων που εχουν σχεση με τις καρτες μεσω ενος διαγραμματος UML.



## Abstract class Tile and all the sub-classes:

Η abstract κλαση tile περιεχει τα attributes:

1) **private** String name; // Το ονομα (ειδος) ενος κουτακιου

2) **private int** day; //Η θεση του στο ταμπλο

3) **private** String image; //Το μονοπατι για το εικονιδιο του tile

Και τις παρακατω μεθοδους:

### Transformers:

1. **public abstract void** performAction(player p); // Αυτη την μεθοδο την κληρωνομουν ολα τα κουτακια και την υλοποιουν με διαφορετικο τροπο το καθε ενα προκειμενου να πραγματοποιηθει η ζητουμενη ενεργεια.

2. **public void** setName(String name) //Οριζει το ονομα του tile

3. **public void** setDay(**int** day) //Οριζει την θεση του κουτακιου στο ταμπλο

4. **public void** setImage(String image) // Οριζι το μονοπατι του εικονιδικου

### Accessors:

5. **public** String getName() // Επιστρεφει το ονομα του tile

6. **public int** getDay() // Επιστρεφει την θεση του κουτακια στο ταμπλο

7. **public** String getImage()//Επιστρεφει το μονοπατι του εικονιδικου

### Observers:

-1. **public boolean** isSunday() // Επιστρεφει αν η ημερα του κουτακιου ειναι Κυριακη ή οχι

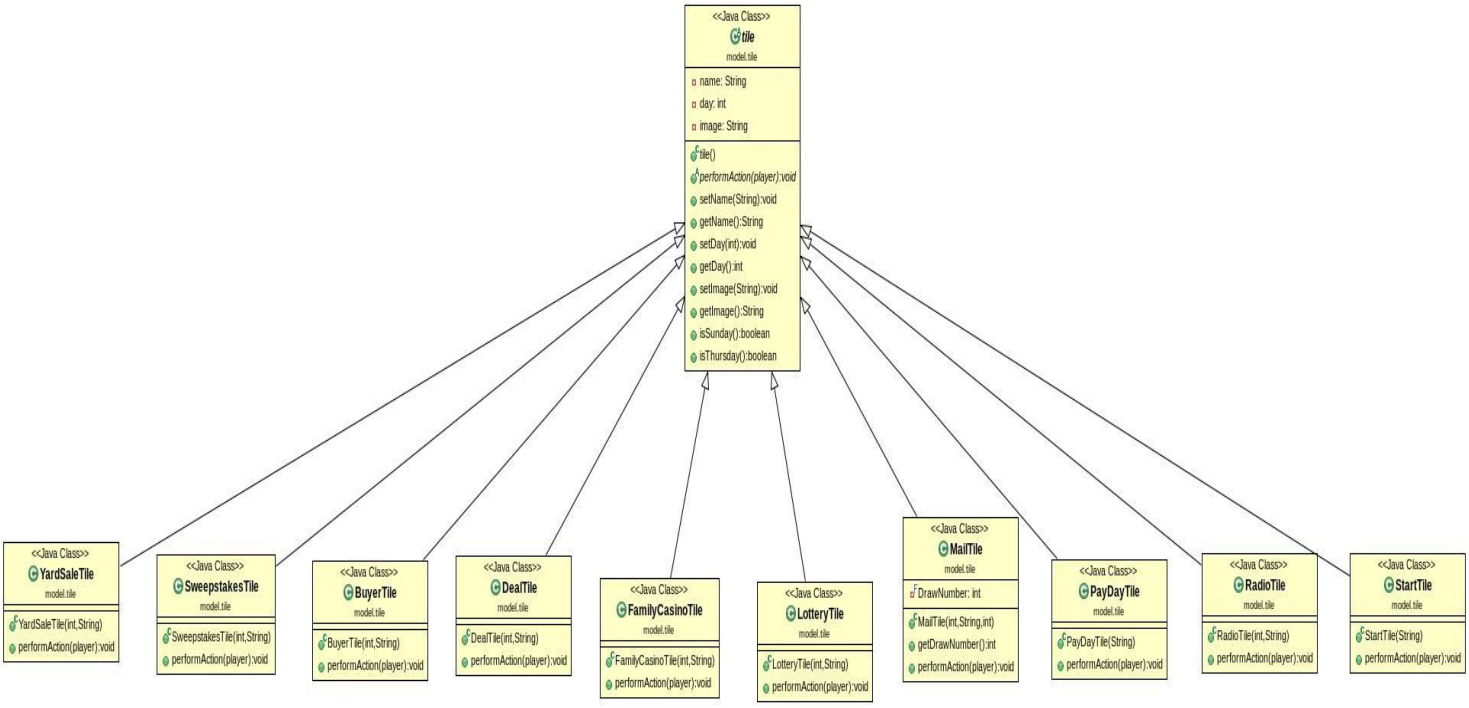
-2. **public boolean** isThursday() // Επιστρεφει αν η ημερα του κουτακιου ειναι Πεμπτη ή οχι

/*Οι δυο παραπανω συναρτησεις αφαιρεθηκαν καθως δεν ειχαν καποια χρησιμοτητα τελικα, καθως ολοκληρη η υλοποιηση του SundayMatch και ThursdayCrypto εγινε στην κλαση player οπου εκει ηταν πιο αμεση. */

## Class BuyerTile, Class DealTile, Class FamilyCasinoTile, Class LotteryTile, Class MailTile , Class PayDayTile, Class RadioTile, Class StartTile, Class SweepstakesTile, Class YardSaleTile:

Αυτες οι κλασεις επεκτεινουν την αφηρημενη κλαση tile. Καθε μια απο αυτε εχει ενα δικο της constructor οπου αρχικοποιει καταλληλα τα πεδια name, day και image. Ολες αυτες οι κλασεις υλοποιουν την μεθοδο "performaction" με διαφορετικο τροπο η καθε μια προκειμενου να πραγματοποιηθουν οι ζητουμενες ενεργειες.

Τελος εδω βλεπουμε μια αναπαρασταση των κλασεων που εχουν σχεση με τα tile μεσω ενος διαγραμματος UML.



## Class Board:

Αυτη η κλαση ειναι υπευθηνη για την αρχικοποιηση του ταμπλο. Αρχικοποιει δηλαδη ολα τα κουτακια του παιχνιδιου.

### Η κλαση Board το attribute:

- `public tile[] tiles = new tile[32]; //Τα κουτακια του ταμπλο (1 για καθε μερα + το αρχικο κουτακι) //Το αλλαξα απο private σε public καθως ηταν αναγαιο`

### Ο Construtor : `public Board() // Δημιουργει ενα νεο ταμπλο`

### Και τις μεθοδους:

1. `public void initBoard() // Transformer: Αρχικοποιει ολα τα tile`

- `public void SetRandom() //Αυτη η μεθοδος δεν μου χρειαστηκε τελικα καθως εκανε τα πραγματα πιο δυσκολα...Τα κουτακια παιρνουν τυχαιες θεσεις στο initialization με την βοηθεια της κλασης RandomPosition που δημιουργησα την οποια περιγραφω ακριβως παρακατω`

## +Class RandomPosition:

Αυτη ειναι μια βοηθητικη κλαση που δημιουργησα για να δινω στα τετραγωνακια μια τυχαια τιμη απο το 1 μεχρι το 30.

```
private ArrayList<Integer> list = new ArrayList<Integer>(); // Η λιστα που κραταει τους αριθμους απο το 1 μεχρι το 30
```

### Ο Construtor :

```
public RandomPosition() //Αρχικοποιει την λιστα
```

Και η μεθοδος :

```
public int getPosition() //Επιστρεφει εναν τυχαιο αριθμο που υπαρχει στην λιστα καθως τον αφαιρει.
```

## Class Dice:

Αυτη η κλαση αναπαριστα το ζαρι του παιχνιδιου.

## Η κλαση Board τα attributes:

1) `private int number; //Ο αριθμος που φερνει καθε φορα το ζαρι`

2) `private boolean roll; // Λογικη τιμη για να μην αφηνουμε τον παικτη να ριχνει το ζαρι πανω απο μια φορα στον γυρο του (εκτος και αν ζηταει καμια καρτα να ριξει ζαρι που εκει θα τον αφηνουμε)`

`+ 3) private static Random r = new Random(); //Το χρησιμοποιουμε για να δινουμε μια τυχαια τιμη στο ζαρι (απο 1 μεχρι 6).`

## Και τις μεθοδους:

### Constructor:

//Δημιουργει ενα στιγμιοτυπο της κλασης αρχικοποιωντας τις μεταβλητες number και roll

```
public dice();
```

### Transformers:

1. `public void rollDice() // Δινει μια τυχαια τιμη στην μεταβλητη number`

2. `public void rollNext() //Θετει την τιμη του roll σε false προκειμενου να μπορει να ριξει ξανα το ζαρι ο παικτης στον επομενο γυρο ( η αν το ζηταει καμια καρτα)`

Accessors:

1. `public int getRoll()` // Επιστρεφει την τιμη του number
2. `public boolean isRolled()` //Επιστρεφει την τιμη του roll

## Class jackpot:

Αυτη η κλαση αναπαριστα το jackpot του παιχνιδιου.

Εχει το attribute : `private int balance;` // Το ποσο που υπαρχει στο jackpot

Constructor : `public jackpot()` // Δημιουργει ενα στιγμιοτπο του jackpot

Και τις εξης μεθοδους :

Transformers:

1. `public void addToJackpot(int amount)` //Προσθετει amount λεφτα στο jackpot
2. `public void winJackpot(player p)` // Ο παικτης p κερδιζει τα λεφτα που υπαρχουν στλ jackpot

Accessor: `public int getBalance()` //Επιστρεφει το ποσο που υπαρχει στο jackpot

## Class player:

Αυτη η κλαση αναπαριστα τους παικτες του παιχνιδιου.

Εχει τα εξης attributes :

1. `private String name;` //Το ονομα του παικτη
2. `private int money;` //Τα λεφτα που διαθετει ο παικτης
3. `private int loan;` //Το δανειο που εχει παρει ο παικτης

4. **private int** bill; **//**Τα λεφτα που χρωσταει ο παικτης στην τραπεζα (λογαριασμοι)

5. **private** dice Dice; //Το ζαρι του παικτη

6. **private int** monthsLeft; //Ποσοι μηνες παιχνιδιου μενουν στον παικτη

7. **private int** day; //Σε ποια μερα του μηνα βρισκεται ο παικτης (χρησιμοποιειται για την θεση του παικτη στο ταμπλο)

8. **private** ArrayList<DealCard> MyDealCards; //Οι καρτε συμφωνιας που εχει στην κατοχη του ο παικτης

9. **private boolean** turn; //Λογικη τιμη για να ελεγχουμε αν ειναι η σειρα του παικτη

10. **private boolean** finished; //Λογικη τιμη για να ελεγχουμε αν ο παικτης εχει φτασει στο τελος του ταμπλο (Στο κουτακι payday)

11. **private** player opponent; //Ο αντιπαλος του παικτης. Η υπαρξη αυτης της μεταβλητης κανει τις συναλλαγες αναμεσα στους δυο παικτης πιο ευκολες (να πληρωσει ο ενας τον αλλο κτλ).

**+ private boolean** actionLeft; //Λογικη τιμη για να μπορουμε να ελεγχουμε αν απομενει κατι αλλο να κανει ο παικτης στον γυρο το ή αν μπορει να πατησει endturn διαφορετικα

## Και τις εξης μεθοδους:

### Transformers:

1. **public void** setMoney(**int** money)//Προσθετει στον λεφτα

2. **public void** pay(**int** money) //Αφαιρει λεφτα απο τον παικτη

3. **public void** setLoan(**int** loan) // Αυξανει το δανειο του παικτη

**-** 4. **public void** payLoan(**int** amount)//Στην Α φαση νομιζα οτι ο παικτης πρεπει να πληρωσει ολο το δανειο στο τελος του μηνα το οποιο δεν ισχυει τελικα. Αρα προστεθηκε η μεταβλητη amount στην συναρτηση και ετσι ο παικτης επιλεγει ποσο θα πληρωσει.

5. **public void** setBill(**int** bill) //Αυξανει τους λογαριασμους που χρωσταει ο παικτης

6.**public void** payBills() //Πληρωνει τους λογαριασμους του παικτη

7. **public void** setStartPoint() //Θετει την αρχικη θεση του παικτη

8. **public void** move()  //Μετακινει τον παικτη τοσες θεσεις οσες εφερε το ζαρι του

**+ public void** movetoDeal(**int** steps) //Βοηθητικη συναρτηση χρησιμοποιειται μονο στην περιπτωση που ο παικτης πρεπει να μετακινηθει στην πλησιεστερη θεση συμφωνιας/αγοραστη

**- public void** endTurn()  //Αυτη η συναρτηση δεν χρειαζοταν και εγινε μια συναρτηση μαζι με την hasPlayed(). Η νεα συναρτηση λεγεται setTurn και ειναι η ακριβως απο κατω.

**+ public void** setTurn(**boolean** turn) //Αρχιζει η τελειωνει τον γυρο του παικτη αντιστοιχα

10. **public void** setFinished() //Θετει την μεταβλητη Finished = true

11. **public void** TakeDealCard(DealCard card) //Προσθετει μια καρτα στην συλλογη καρτων του παικτη

12. **public void** SellDealCard(DealCard card) // Αφαιρει μια καρτα απο την συλλογη καρτων του παικτη

13. **public void** setOpponent(player opponent) //Οριζει τον αντιπαλο του παικτη

-14. **public void** SundayMatch() // Ο παικτης στοιχηματιζει στον Κυριακατικο αγωνα (αν θελει). Αν πιασει το στοιχιμα κερδιζει λεφτα διαφορετικα τα χανει.

-15. **public void** ThursdayCrypto() //Ο παικτης ποντάρει λεφτα αν θελει σε καποιο κρυπτονομισα και κερδιζει, χανει ή παιρνει τα λεφτα του πισω.

**-** Οι δυο παραπανω συναρτησεις παιρνανε ως παραμετρο την επιλογη του παικτη. Αυτο αφαιρεθηκε και η επιλογη γινεται μεσω Dialog Menu.

**+ public void** setMonthsLeft(**int** x) //Οριζει αλλα και ανανεωνει ποσοι μηνες παιχνιδιου απομενουν στον παικτη

**+ public void** setActionLeft(**boolean** b) //Οριζει αν απομενει στον παικτη να ολοκληρωσει καποια ενεργεια

## Accessors:

16. **public** String getName() // Επιστρεφει το ονομα του παικτη

17. **public int** getMoney()  // Επιστρεφει τα λεφτα που διαθετει ο παικτης

18. **public int** getLoan()  //Επιστρεφει το δανειο του παικτη

19. **public int** getBill()  //Επιστρεφει τους λογαριασμους του παικτη

20. **public int** getPosition() //Επιστρεφει την θεση που βρισκεται ο παικτης

21. **public int** getMonthsLeft() //Επιστρεφει ποσοι μηνες απομενουν

**-** 22. **public boolean** hasPlayed() //Αυτη η συναρτηση δεν χρειαζοταν και εγινε μια συναρτηση μαζι με την endTurn()

23. **public boolean** hasFinished() // Επιστρεφει αν ο παικτης εχει φτασει στην θεση payday ή οχι

24. **public** ArrayList<DealCard> viewMyDealCards() // Επιστρεφει τις καρτες που υπαρχουν στην συλλογη του παικτη

25.**public player** getOpponent() //Επιστρεφει τον αντιπαλο του παικτη

**+** 26. **public void** payInterest() //Ο παικτης πληρωνει φορο 10% του δανειου του και στην συνεχεια μπορει να επιλεξει αν θελει να πληρωσει τωρα το υπολειπομενο δανειο.

**+ public** boolean getTurn() //Επιστρεφει αν ειναι σειρα του παικτη

**+ public** dice getDice() **//**Επιστρεφει το ζαρι του παικτη

**+ public int** getMonthsLeft() //Επιστρεφει ποσοι μηνες παιχνιδιου απομενουν

**+ public boolean** getActionLeft() //Επιστρεφει true αν ο παικτης εχει κατι αλλο να κανει στον γυρο του διαφορετικα επιστρεφει false

## Class DealDeck και Class MailDeck:

Αυτες οι κλασεις αναπαριστουν τις δυο στοιβες με τις καρτες του παιχνιδιου (Στοιβα με τις καρτες μηνυματος και στοιβα με τις καρτες συμφωνιας).

## DealDeck attributes:

1. **private** ArrayList<DealCard> Deck; //Η στοιβα με τις διαθεσιμες καρτες συμφωνιας

2. **private** ArrayList<DealCard> RejectedStack; // Η στοιβα απορριψης

## DealDeck μεθοδοι:

Transformers:

1. **public void** init_deck() //Αρχικοποιει την στοιβα με τις διαθεσιμες καρτες

2. **public void** reconstruct() //Μεταφερει τις καρτες απο την στοιβα απορριψης στην στοιβα με τις διαθεσιμες καρτες

3. **public void** shuffle() //Ανακατεβει τις καρτες

4. **public void** AddToDeck(DealCard card) // Προσθετει μια καρτα σε μια στοιβα

5. **public** DealCard DrawCard() //Αφαιρει μια καρτα απο την στοιβα διαθεσιμων καρτων και την προσθετη στην συλλογη του παικτη

+ **public void** RejectCard(DealCard card) // Προσθετει μια καρτα στην στοιβα απορριψης

Observer: **public boolean** isEmpty() // Επισρεφει αν η στοιβα ειναι αδεια ή οχι

## MailDeck attributes:

1. **private** ArrayList<MailCard> Deck; //Η στοιβα με τις διαθεσιμες καρτες μηνυματος

## MailDeck μεθοδοι:

Transformers:

1. **public void** init_deck() //Αρχικοποιει την στοιβα με τις διαθεσιμες καρτες

2. **public void** shuffle() //Ανακατεβει τις καρτες

3. **public void** AddToDeck(MailCard card) // Προσθετει μια καρτα στην στοιβα

5. **public** MailCard DrawCard() //Αφαιρει μια καρτα απο την στοιβα

Observer: `public boolean isEmpty() // Επισρεφει αν η στοιβα ειναι αδεια ή οχι`

+ Class ReadCards

Βοηθητικη κλαση που χρησιμοποιειται για την αρχικοποιηση των καρτων, η οποια περιεχει την συναρτηση που μας δινοταν ετοιμη :

```
public static String[][] readFile(String path, String type)
```

## 3. Η Σχεδίαση και οι Κλάσεις του Πακέτου Controller

Σε αυτο το πακετο υπαρχουν οι κλασεις Menu και Controller.

-Η κλαση Menu καθοριζει αν θα ξεκινησει ενα νεο παιχνιδι ή αν θα συνεχιστει καποιο παιχνιδι που εχει ξεκινησει ηδη. Πιο συγκεκριμενα στην αρχη της εκτελεσης του προγραμματος θα παρεχεται η δυνατοτητα στον χρηστη να επιλεξει αν θελει να ξεκινησει ενα καινουριο παιχνιδι ή αν θελει να φορτωσει, καποιο παιχνιδι που εχει ξεκινησει ηδη, απο καποιο αρχειο.

-Το bonus ερωτημα της ασκησης για την αποθηκευση του παιχνιδιου δεν υλοποιηθηκε τελικα επομενως η κλαση Menu δεν εχει καποια χρησιμοτητα επομενως την αφαιρεσα.

Η κλαση Controller μπορουμε να πουμε οτι ειναι ο εγκαιφαλος του παιχνιδιου. Ειναι υπευθηνη για την αρχικοποιηση ολων των βασικων “συστατικων” του παιχνιδου (Ταμπλο , παικτες, στοιβες καρτων κτλ) αλλα και για τον τερματισμο του παιχνιδιου (αναδειξη του νικητη). Ελεγχει επισης για την ομαλη διεξαγωγη του παιχνιδιου, δηλαδη το ποιος παιχτης

θα παιξει πρωτος, ποιανου παικτη ειναι η σειρα, ποσοι μηνες απομενουν στο παιχνιδι , αν το παιχνιδι εχει τελειωσει κτλ.

Αξιζει να σημειωθει οτι η κλαση View εχει προσβαση στο model μεσω του Controller και ετσι μπορει και παιρνει οτι πληροφορια χρειαζεται προκειμενου να αναπαραστει με γραφικο τροπο το παιχνιδι στην οθονη του χρηση.

## Εχει τα εξης attributes :

```
- public  player P1, P2;     //Απο private το αλλαξα σε public
- public Board board;      //Απο private το αλλαξα σε public
- public  jackpot Jackpot; //Απο private το αλλαξα σε public
- public  DealDeck DealStack; //Απο private σε public
- public  MailDeck MailStack; //Απο private σε public
//Αυτο εγινε για να υπαρχει αμεση προσβαση στο view χωρις να
χρειαστει να φτιαξω 6 συναρτησεις που να μας δινουν προσβαση
- private View view;   //Δεν ειχε καποια χρησιμοτητα τελικα
+ private int months;
```

## Και τις εξης μεθοδους:

### Transformers :

```
1. public void SetMonths()     //Οριζει ποσους μηνες θα διαρκεσει το
                                                        παιχνιδι
2. public void SetStartingPlayer() //Καθοριζει ποιος παικτης θα
                                 παιξει πρωτος (διαλεγει τυχαια)
+ public void initPlayers() //Οι παικτες επιλεγουν τα ονοματα τους
+ public void updateMonths(int months) // Ανανεωνει το ποσοι μηνες
                                          παιχνιδιου απομενουν
```

### Observers :

```
-3. public void playerTurn()  //Δεν χρειαζεται τελικα, το με ποια
σειρα παιζουν οι παικτες υπολογιζεται αλλιως μεσα στο προγραμμα

4. public boolean gameFinished() //Επιστρεφει αν εχει τελειωσει το
                                                        παιχνιδι ή οχι

- 5. public void SaveGame() // Δεν θα υλοποιησω το bonus τελικα..

+ public int getMonths()  // Επιστρεφει ποσοι μηνες παιχνιδιου
                                                        απομενουν
```

Accessor:

```
6. public void gameWinner() //Ανακοινωνει τον νικητη του παιχνιδιου

+ public player getPlayerTurn() //Επιστρεφει ποιος παικτης παιζει
```

## 4. Η Σχεδίαση και οι Κλάσεις του Πακέτου View

Σε αυτο το πακετο υπαρχουν οι κλασεις View και playerField.

### Class PlayerField:

Η κλαση playerField επεκτεινει το Jpanel και δημιουργηθηκε για να υπαρχει επαναχρησιμοποιηση του κωδικα και να κραταμε πληροφοριες σχετικα με τους πακτες αλλα και ολα τα κουμπια που ειναι απαραιτητα για να παιξουν το παιχνιδι στον γυρο τους.

Εχει τα εξης attributes :

```
//Αναπαρασταση των πληροφοριων για τον καθε παικτη

private JTextField PlayerName, Money, Loan, Bills;

private JLabel Dice;  //Η γραφικη απεικονηση του ζαριου

private JButton RollDice, myDealCards, GetLoan, EndTurn; //Τα
            κουμπια που εχει στην διαθεση του ο καθε παικτης
```

**Οι τρεις παρακατω μεταβλητες υπαρχουν για να διαβαζουμε (φορτονουμε ) τις εικονες απο τα αρχεια, να κραταμε το path τους και στην συνεχεια να περναμε την εικονα σε οποιο label, κουμπι κτλ χρειαζεται**

```
private ClassLoader cldr;

private URL imageURL;

private Image image;


private player Player; //βοηθητικη μεταβλητη που μας παρεχει
προσβαση στις πληροφοριες του παικτη
```

## Και τις εξης μεθοδους:

Constructor : `public void initComponents() //Δημιουργει ενα στιγμιοτυπο την κλασης playerField και αρχικοποιει ολα τα απαραιτητα κουμπια,labels κτλ για να ειναι ετοιμα να εμφανιστουν στην οθονη.`

Transformer : `public void updateTextInfo() // Χρησιμοποιειται για να ανανεωνονται οι πληροφοριες στο του καθε παικτη`

Μεσα στην κλαση playerField υπαρχουν nested οι κλασεις DiceListener, getLoanListener, myDealCardsListener, EndTurnListener οι οποιες υλοποιουν την κλαση ActionListener και μεσω της συναρτηση

`public void actionPerformed(ActionEvent e)` που υλοπιουν, εκετελουν τις καταλληλες ενεργειες οταν πατηθει το αντιστοιχο κουμπι.

## Class View:

Η κλαση view δημιουργει ενα frame και μεσα σε αυτο ενα κεντρικο panel το οποιο μπορουμε να το φανταστουμε ως το ταμπλο του επιτραπεζιου παιχνιδιου. Πανω σε αυτο το panel υπαρχουν 2 PlayerFields που οπως περιγραψαμε προηγουμενως κρατανε πληροφοριες και τα κουμπια για τον καθε παικτη. Πανω στο panel υπαρχει επισης ενα InfoBox στο οποιο κραταμε πληροφοριες για το ποιος παικτης παιζει, τι ενεργεια πρεπει να πραγματοποιησει κτλ. Υπαρχουν επισης 2 κουμπια (ενα για καθε στοιβα καρτων) 32 labels (ενα για καθε τετραγωνακι του ταμπλο) καθως και αλλα 32 "αορατα" panels τα οποια χρειαζονται για να μπορουμε να αναπαρησταμε το πιονι πανω σε καθε κουτακι. Τελος υπαρχει ενα εμα ακομη label καθως και ενα textField τα οποια χρησιμοποιουνται για την αναπαρασταση του jackpot του παιχνιδιου.

Ο Constructor view : **public View()** //Οπου δημιουργει ενα στιγμιοτυπο της κλασης view, δημιουργει ολα τα απαραιτητα panels, labels, buttons κτλ. Και μεσν της συναρτησης initComponents() τα αρχικοποιει ολα καταλληλα και τα κανει visible στην οθονη του χρηστη.

Transformer : **public static void** UpdateInfoBox(String action) //Ενημερωνει τις πληροφοριες του πεδιου InfoBox

Transformer : **public static void** UpdateJackpot() //Ενημερωνει τις πληροφοριες του πεδιου Jackpot TextField

Transformer :**public static void** movePawn(player Player ,**int** oldPosition) //Ανανεωνει την θεση του πιονιου στην οθονη

Transformer : **public static void** TileAction(player p) //Οταν ο παικτης πατησει πανω σε ενα τετραγωνακι ανανεωνει το InfoBox και ετσι ενημερωνει τον παικτη τι ενεργεια πρεπει να πραγαματοποιησει. Στην συνεχεια με την βοηθεια του controller εκτελουνται οι καταλληλες ενεργειες.

Transformer : **public static void** resetPawn(player Player) // Πηγαινει το πιονι του παικτη πισω στην αρχικη θεση.

Μεσα στην κλαση View υπαρχουν nested οι κλασεις DealCardListener και MailCardListener οι οποιες υλοποιουν την κλαση ActionListener και μεσω της συναρτησης `public void actionPerformed(ActionEvent e)` που υλοπιουν, εκετελουν τις καταλληλες ενεργειες οταν πατηθει το αντιστοιχο κουμπι και με την βοηθεια των συναρτησεων `public void showDealCard(DealCard card)` και `public void showMailCard(MailCard card)` αντιστοιχα, εμφανιζονται οι καρτες στην οθονη του παικτη.

Ακολουθουν μερικα screenshot απο την γραφικη αναπαρασταση του παιχνιδιου :


Οταν ξεκινησει το παιχνιδι μεσω ενος παραθυρου διαλογου διαλεγουμε ποσους μηνες θελουμε να διαρκεσει το παιχνιδι


Στην συνεχεια παρεχεται η δυνατοτητα στους παικτες να εισαγουν τα ονοματα τους.

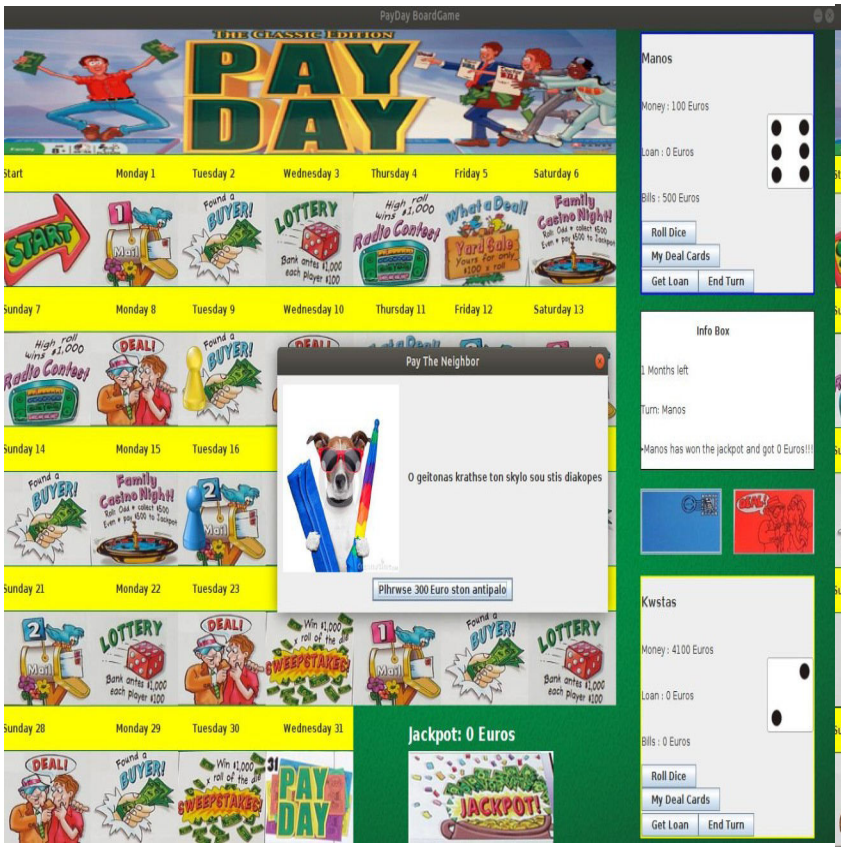
Το παιχνιδι ξεκιναει :

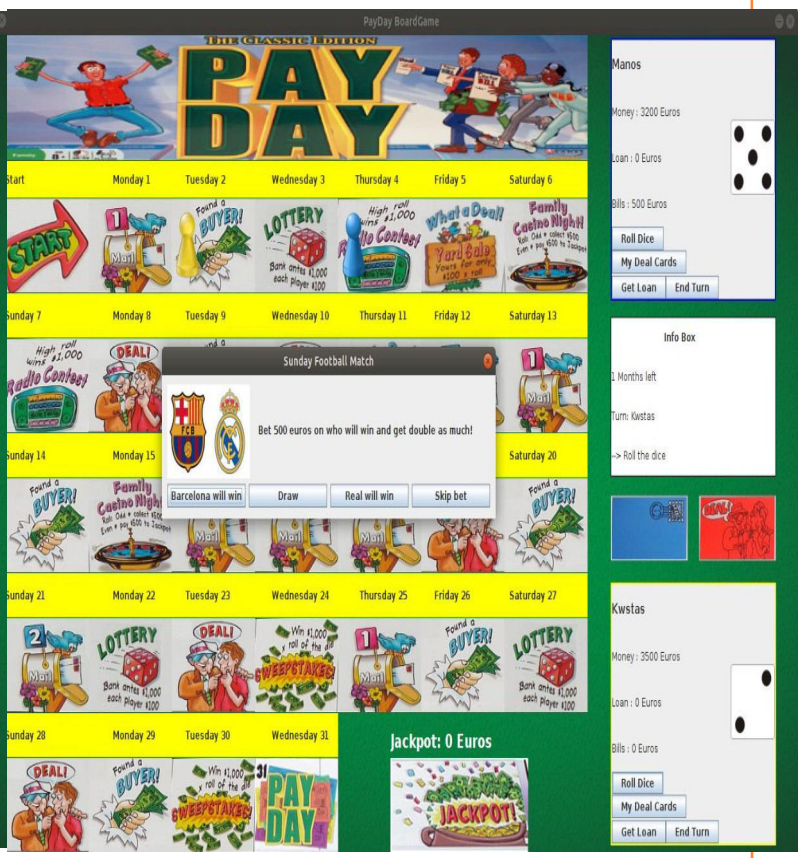

PayDay BoardGame
THE CLASSIC EDITION
PAY DAY
Start
Monday 1
Tuesday 2
Wednesday 3
Thursday 4
Friday 5
Saturday 6
Sunday 7
Monday 8
Tuesday 9
Wednesday 10
Thursday 11
Friday 12
Saturday 13
Sunday 14
Monday 15
Tuesday 16
Wednesday 17
Thursday 18
Friday 19
Saturday 20
Sunday 21
Monday 22
Tuesday 23
Wednesday 24
Thursday 25
Friday 26
Saturday 27
Sunday 28
Monday 29
Tuesday 30
Wednesday 31
Jackpot: 0 Euros
JACKPOT!
Manos
Money : 3500 Euros
Loan : 0 Euros
Bills : 0 Euros
Roll Dice
My Deal Cards
Get Loan
End Turn
Info Box
1 Months left
Turn: Manos
-> Roll the dice
Kwstas
Money : 3500 Euros
Loan : 0 Euros
Bills : 0 Euros
Roll Dice
My Deal Cards
Get Loan
End Turn

Οποια ενεργεια πραγματοποιειται αυτοματα (μεσα στον κωδικα του προγραμματοσ) και εμφανιζεται στην οθονη μεσω ενος παραθυρου :

πχ ενας παικτης πηρε αυτοματα δανειο

ΟΙ καρατες αλλα και οι ειδικες θεσεις (Football και Crypto) εμφανιζονται ως εξης :



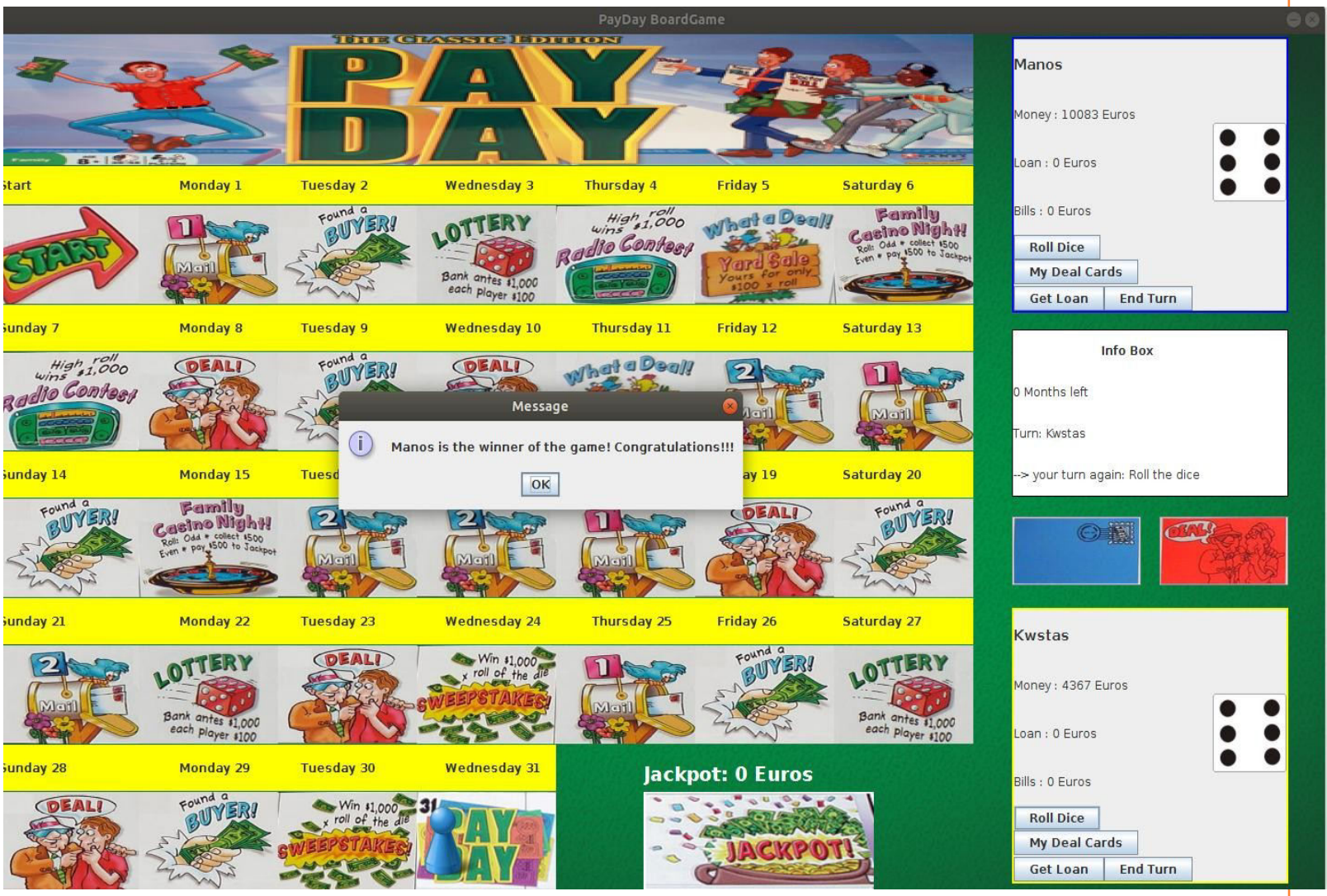





Οταν ο παικτης φταση στην θεση payday μεσω

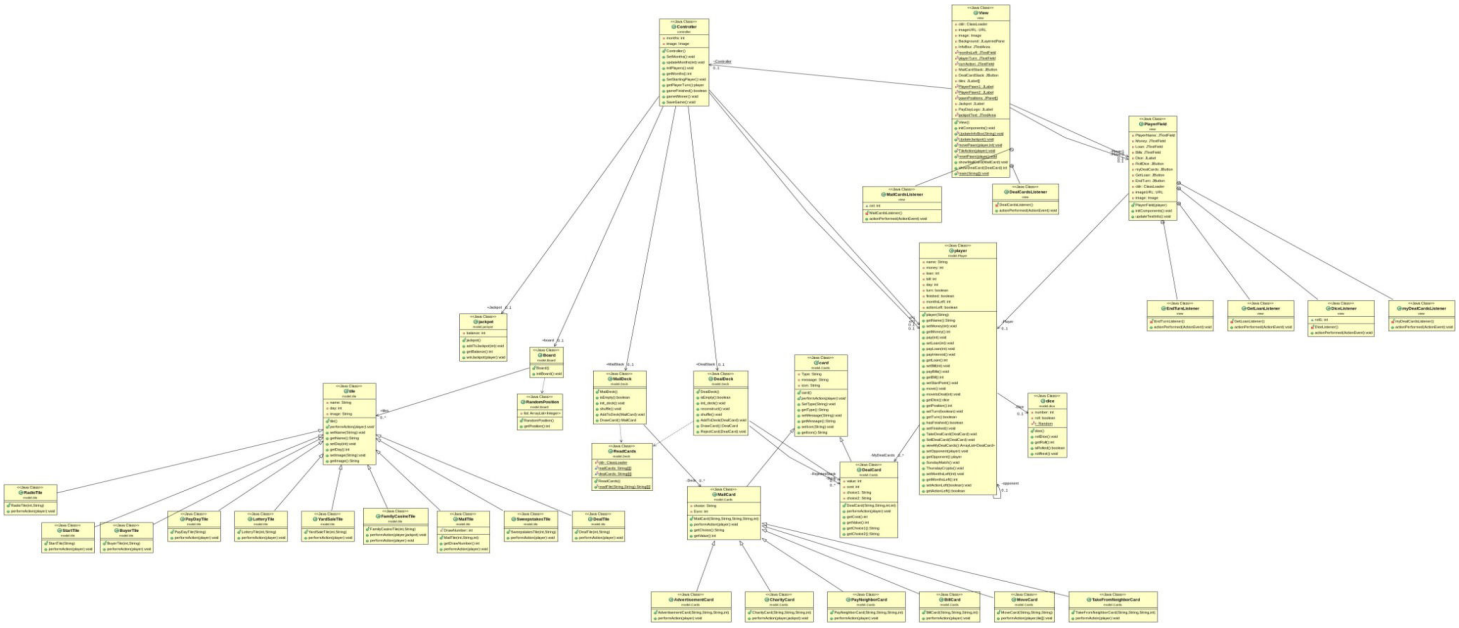
ενος παραθυρου εχει την δυνατοτητα να επιλεξει αν θελει να πληρωσει ενα ποσο απο το δανειο του.

Και τελος η αναδειξη του νικητη :




## 5. Η Αλληλεπίδραση μεταξύ των κλάσεων – Διαγράμματα UML

**Εδω βλεπουμε ενα ανανεωμενο διαγραμμα UML του προγραμματος**



Υπαρχει και αποθηκευμενο ως εικονα στον φακελο με ονοα UMLDiagrams phase2 οπου ειναι πιο ευκολη η αναγνωση του διαγραμματος.

-Οι γραμμες που καταληγουν σε βελακι αναπαριστουν τις επεκτασεις των κλασεων.

-Οι διακεκεκομμενες γραμμες δειχνουν τις συναρτησιακες εξαρτησεις.

-Ενω οι γραμμες που καταληγουν σε ενα κυκλακι αναπαριστουν τα nesting relationships (δειχνουν τις κλασεις δηλαδη που δηλωνονται μεσα σε μια αλλη κλαση).

# 6. Λειτουργικότητα (Β Φάση)

Καταφερα να υλοποιησω τα παντα που ζητουσε η ασκηση και σχεδον ολα με τον τροπο τον οποιο τα ζητουσε.

Δεν ασχοληθηκα τελικα με το bonus της ασκησης (αποθηκευση του παιχνιδιου σε αρχειο) καθως υπηρχε αρκετη πιεση απο εργασιας και σε αλλα μαθηματα και ετσι προτιμησα να παραδωσω νωριτερα ωστε να κερδισω χρονο για τα υπολοιπα μαθηματα.

Τι  εκανα με διαφορετικο τροπο :

-Το Lottery Tile υλοποιηθηκε με λιγο διαφορετικο τροπο απο οτι ζητουσε η ασκηση : οι παικτες διαλεγουν κανονικα εναν αριθμο απο το 1 μεχρι το 6, ξεκινωντας απο τον παικτη που πατησε πανω σε αυτο το κουτακι ομως μετα η κληρωση γινεται τυχαια μεσα στο προγραμμα.

Τι ειναι λαθος και το γνωριζω :

-Οταν ενας παικτης πατησει πανω σε ενα mail Tile και πρεπει να τραβηξει 2 καρτες αλλα η πρωτη καρτα ειναι καρτα μετακινησης, τοτε ο παικτης μετακινηται χωρις καν να τραβηξει την δευτερη καρτα.

Διευκρινισεις για ομαλη λειτουργεια :

-Στο Radio tile και οι δυο παικτες ριχνουν το ζαρι του παικτη πατησε πανω στο τετραγωνακι ξεκινωντας απο αυτον που πατησε.

-Οταν βρισκομαστε στον τελευταιο μηνα και ενας παικτης φτασει στην τελευταια θεση payday, τοτε ο παικτης που εχει μεινει πρεπει να παταει καθε φορα end turn κανονικα, και στην συνεχεια να ξανα ριχνει το ζαρι μεχρι να τερματισει και αυτος και να αναδειχθει ο νικητης του παινχιδιου.

Ολες οι υπολοιπες ενεργεις στο προγραμμα εχουν υλοποιηθει ετσι οπως ζητουσε η ασκηση. Οτι ενεργεια πραγματοποιηθει αυτοματα (πχ οταν καποιος παικτης παιρνει αυτοματα δανειο) εμφανιζεται στην οθονη μεσω ενος παραθυρου οπως φαινεται και παραπανω στην περιγραφη του πακετου view.

Η σειρα τηρειται επισης κανονικα και το προγραμμα δεν αφηνει καποιον παικτη να “κλεψει ή να μην παιζει δικαια”.

# 7. Συμπεράσματα

Σε γενικες γραμμες ηταν μια ενδιαφερουσα εργασια η οποια μου προσφερε καινουριες γνωσεις αλλα και καταλαβα καλυτερα καποια πραγματα για την γλωσσα.

Δεν μπορω να πω οτι την εκανα με ευκολια καθως μου πηρε πολυ περισσοτερο χρονο απο οτι περιμενα.

Θεωρω οτι ο κωδικας μου θα μπορουσε να ειναι αρκετα πιο κομψος οσον αφορα την αναγνωση αλλα και λιγο την σχεδιαση, ομως δεν μπορουσα να αφιερωσω περισσοτερο χρονο και ενεργεια για να διωρθωσω καποια πραγματα καθως υπηρχε και αρκετο φορτο απο εργασιες σε αλλα μαθημα. Ετσι εστιασα στο να κανω το παιχνιδι να λειτουργει σωστα και οπως ζητουσε η ασκηση ακομα και αν αυτο σημαινε οτι ο κωδικας μου δεν θα ηταν τοσο κομψος.

Το τελευταιο που μου ειχε μεινει να υλοποιησω ειναι οι ενεργειες των tiles του ταμπλο (performAction). Οταν εφτασα σε αυτο το σημειο καταλαβα τοτε οτι υπαρχουν καποια σχεδιαστικα προβληματα στην εργασια μου τα οποια ομως δεν ειχα το κουραγιο να αλλαξω.

Παρ' αυτα πιστευω πως το παιχνει λειτουργει σωστα και οπως ζητουσε η ασκηση (περα απο αυτη την μικρη αλλαγη που εκανα στο Lottery Tile).

Αξιζει να σημειωθει βεβαια πως μπορει σε καποιες πολυ ειδικες περιπτωσεις τα πραγματα να μην λειτουργουν ακριβως οπως πρεπει...