

**Πανεπιστήμιο Κρήτης –Τμήμα Επιστήμης Υπολογιστών**

**ΗΥ252– Αντικειμενοστραφής Προγραμματισμός**

**Διδάσκων: Ι. Τζίτζικας**

**Χειμερινό Εξάμηνο 2020-2021**

Payday

boardgame

Εισαγωγή

Αλέξανδρος Εμμανουήλ Αντωνακάκης

csd4802

11 / 12 / 2021

Think and describe what you plan to do and why it will be useful.

Περιεχόμενα

[1. Εισαγωγή 1](#_Toc530045458)

[2. Η Σχεδίαση και οι Κλάσεις του Πακέτου Model 1](#_Toc530045459)

[3. Η Σχεδίαση και οι Κλάσεις του Πακέτου Controller 1](#_Toc530045460)

[4. Η Σχεδίαση και οι Κλάσεις του Πακέτου View 2](#_Toc530045461)

[5. Η Αλληλεπίδραση μεταξύ των κλάσεων – Διαγράμματα UML 2](#_Toc530045462)

[6. Λειτουργικότητα (Β Φάση) 2](#_Toc530045463)

[7. Συμπεράσματα 2](#_Toc530045464)

## Εισαγωγή

Το μοντέλο σχεδίασης που χρησιμοποιήθηκε είναι το MVC. Αυτό επεκτείνεται σε,

* ***Model***

Στο model υπάρχουν τα δεδομένα που θα χρειαστούμε (player, position, cards) και κάποια βασική λογική πίσω από το καθένα.

* ***View***

Στο view βρίσκεται η γραφική απεικόνιση της εφαρμογής. Είναι επίσης, το κόμματί που θα αλληλοεπιδρά ο χρήστης.

* ***Controller***

Ο controller, είναι υπεύθυνος για την λειτουργία της εφαρμογής, καθώς και για την σύνδεση View και Model.

## Η Σχεδίαση και οι Κλάσεις του Πακέτου Model

***Package Cards:***

**Interface Cards:**

Για το Cards, χρησιμοποιήθηκε ένα interface, το όποιο γίνεται implement σε κάθε υποκατηγορία κάρτας.

int getValue(); (accessor)

void action(Player *p*); (transformer)

**Abstract class MailCards:**

Για την υποκατηγορία MailCards, χρησιμοποιήθηκε ένα abstract class που γίνεται extend μέσα σε κάθε class κάρτας. Αυτό διότι, υπάρχει η abstract method action, που διαφέρει ανά κάρτα, ενώ όλα τα υπόλοιπα δεν αλλάζουν.

**Attributes:**

private int amount; (ποσό κάρτας)  
private String type; (όνομα κάρτας)

**Methods:**

public MailCard(int *amount*, String *type*); (constructor)

public int getValue(); (accessor)

public abstract void action(Player *p*); (accessor) (ιδιότητα καρτών)

**Έπειτα για κάθε κάρτα, υλοποιήθηκε η abstract method action, η όποια άλλαζε ανάλογα με την ιδιότητα της κάθε κάρτας. Δεν υπήρχαν επιπλέον πεδία, και γινόταν απλά κλήση του super constructor της MailCards. Η μόνη εξαίρεση είναι η κάρτα moveToDB που δίνει τιμή 0 στο πεδίο amount.**

**Class DealCard:**

Για την υποκατηγορία DealCards, υλοποιήθηκε **μια** κλάση, αυτό διότι οι deal cards, δεν διαφοροποιούνται μεταξύ τους, μόνο ως προς το κείμενο, και την τιμή τους.

**Attributes:**

private final int BuyPrice; (τιμή προσφοράς)  
private final int sellPrice; (αρχική τιμή πώλησης)

**Methods:**

public DealCard(int *amount*, int *original\_amount*); (constructor)

public int getValue(); (accessor) (τιμή προσφοράς)

public int getSellPrice(); (accessor) (αρχική τιμή πώλησης)

public void action(Player *p*); (transformer) (αποδοχή ή όχι προσφοράς)

***Package Dice:***

**Class Dice:**

Για το ζάρι, οι λειτουργίες υλοποιήθηκαν στο class Dice.

**Attributes:**

private int value; (τιμή ζαριού 1 – 6)

**Methods:**

public Dice(); (constructor)

public int getValue(); (accessor) (επιστρέφει την τιμή του ζαριού

public void rollDice(); (transformer) (ρίψη ζαριού)

***Package player:***

**Class Player:**

Εδώ υλοποιήθηκαν βασικές μέθοδοι και πεδία για την λειτουργία του παίκτη.

**Attributes:**

private final String name; (όνομα παίκτη)  
private int bank\_balance; (υπόλοιπο στην τράπεζα)

private int loans; (ποσό δανείου)

private int due; (οφειλές)  
private int position; (θέση στο board)

private Player neighbor; (γείτονας – αντίπαλος)  
private final Dice dice; (ζάρι)

**Methods:**

public Player(String *name*); (constructor)

public String getName(); (accessor)

public int getPosition(); (accessor)

public void setPosition(int *position*) (transformer)

public int getBank\_balance(); (accessor)

public void setBank\_balance(int *bank\_balance*); (transformer)

public int getLoans(); (accessor)

public void setLoans(); (transformer)

public int getDue(); (accessor)

public void setDue(int *due*); (transformer)

public Player getNeighbor(); (accessor)

public void setNeighbor(Player *neighbor*); (transformer)

public Dice getDice(); (accessor)

***Package position:***

Εδώ υλοποιήθηκε ένα abstract class αφού οι θέσεις έχουν κοινά γνωρίσματα – πεδία, το μόνο που διαφέρει είναι η μέθοδος action, που υλοποιείται ξεχωριστά από κάθε class θέσης (λ.χ. Lottery)

**Abstract class Position:**

**Attributes:**

private final Days day; (ποια μέρα της εβδομάδας είναι)  
private final String imageURL; (εικόνα του τετραγώνου)  
private final int day\_index; (αριθμός ημέρας)

**Methods:**

public Position(Days *day*, String *imageURL*, int *day\_index*); (constructor)

public Days getDay(); (accessor)

public String getImageURL(); (accessor)

public int getDay\_index(); (accessor)

public abstract void posAction(Player *p*);(transformer)

Τα υπόλοιπα classes, κάνουν extend το abstract class Position, δεν έχουν επιπλέον πεδία, και απλά υλοποιούν την abstract method, posAction, που είναι διαφορετική ανάλογα με την θέση.

***Package special:***

Σε αυτό το πακέτο, υλοποιούνται κάποιες ειδικές περιπτώσεις του παιχνιδιού, αυτές είναι το Jackpot, η άνοδος των κρυπτονομισμάτων και ο ποδοσφαιρικός αγώνας της Κυριακής. Επέλεξα να τις διαφοροποιήσω από τις άλλες θέσεις, καθώς κάθε περίπτωση απαιτεί ειδικές μεθόδους και επιπλέον πεδία.

**Class Jackpot:**

**Attributes:**

private int amount; (χρήματα στο jackpot)

private String imageURL; (εικόνα στο ταμπλό)

**Methods:**

public Jackpot(String *imageURL*); (constructor)

public void jackpotAction(Player *p*); (transformer)

public int getAmount(); (accessor)

public void setAmount(int amount); (transformer)

public String getImageURL(); (accessor)

public void setImageURL(String *imageURL*); (transformer)

**Class SundayFootballDay:**

**Attributes:**

private int bet; (στοίχημα)  
private final Bet\_Sports choice; (στοίχημα ναι ή όχι)

**Methods:**

public SundayFootballDay(Bet\_Sports *choice*); (constructor)

public void sundayFootballaction(); (transformer)

**Class ThursdayRiseInCrypto:**

**Attributes:**

private final Bet\_Crypto choice; (ποντάρισμα ναι ή όχι)  
private Dice newRoll; (νέα ρίψη ζαριών)

**Methods:**

public ThursdayRiseInCrypto(Bet\_Crypto *choice*); (constructor)

public void thurdayRiseAction(); (transformer)

## Η Σχεδίαση και οι Κλάσεις του Πακέτου Controller

**Class Controller:**

Εδώ υλοποιήθηκαν όλες οι λειτουργίες του παιχνιδιού και οι απαραίτητες δομές και πεδία.

**Attributes:**

public Player p1;   
public Player p2;  
public Player turn; (τίνος σειρά είναι)  
public int months\_left; (πόσοι μήνες ακόμα)  
public final LinkedList<Card> cardStack; (στοίβα με κάρτες)  
public final LinkedList<Card> rejectedCardStack; (στοίβα με πεταμένες κάρτες)  
public final ArrayList<Position> positions; (θέσεις στο ταμπλό)

**Methods:**

public Controller(); (constructor)

private void init\_cardStacks(); (transformer)

private void init\_board(); (transformer)

## Η Σχεδίαση και οι Κλάσεις του Πακέτου View

Σε αυτήν την φάση, η ιδέα είναι να υπάρχει ένα class με τα κύρια γραφικά, και 2 ακόμη class που θα υλοποιούν τα γραφικά των ειδικών περιπτώσεων, όπως jackpot.

**Attributes:**

JFrame frame; (κύριο frame)  
JPanel board; (panel για το board)  
JPanel p1; (panel για τον πρώτο παίκτη)  
JPanel p2; (panel για τον δεύτερο παίκτη)  
JMenu menu; (menu παιχνιδιού)  
JMenuBar menu\_bar; (μπάρα menu)  
Controller controller; (controller παιχνιδιού)

private SundayFootballDayUI sundayFootballDayUI; (διεπάφη special)  
private ThursdayRiseInCryptoUI thursdayRiseInCryptoUI;(διεπάφη special)

**Methods:**

public GraphicsUI(); (constructor)

private void initialize\_fields(); (transformer)

private void create\_player\_panels(); (transformer)

private void init\_session(); (transformer)

## Η Αλληλεπίδραση μεταξύ των κλάσεων – Διαγράμματα UML

(Λόγω του μεγέθους, είναι σχεδόν αδύνατο να διαβαστεί από εδώ άρα πρόσθεσα την εικόνα μέσα στο project).

