

**Πανεπιστήμιο Κρήτης –Τμήμα Επιστήμης Υπολογιστών**

**ΗΥ252– Αντικειμενοστρεφής Προγραμματισμός**

**Διδάσκων: Ι. Τζίτζικας**

**Χειμερινό Εξάμηνο 2020-2021**

Αναφορά Project

Εισαγωγή

Εμμανουήλ Πλατάκης

4730

29-11-2021

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

Η υλοποίηση της εργασίας θα βασιστεί πάνω στο μοντέλο MVC (Model View

Controller). Έτσι, σκοπός μου είναι ο Controller να συνδέει ομαλά το Model και το View. Οπότε στη συνέχεια της αναφοράς μου θα αναλύσω

λίγο ιδιαίτερα τα κομμάτια του Model και Controller που είναι σημαντικά για αυτή τη φάση και τέλος θα αναφερθούμε και λίγο στο view.

## 

## Η Σχεδίαση και οι Κλάσεις του Πακέτου Model

Σε αυτό το πακέτο θα περιέχεται μια abstract class Position με τις υποκλάσεις DicePosition, CardPosition, PayDayPosition, OnePlayerDicePosition και BothPlayersDicePosition, εκ των οποίων οι 3 πρώτες θα κάνουν extend την Position, ενώ οι 2 τελευταίες θα κάνουν extend την DicePosition. Ακόμη, θα υπάρχουν enumerations που θα αφορούν στις κατηγορίες θέσεων του παιχνιδιού. Θα περιέχεται επίσης μια abstract class Card, με υποκλάσεις MailCard, DealCard και ένα enumeration MailCardsType που θα περιέχει τον τύπο της κάρτας μηνύματος. Τέλος, θα έχουμε μια abstract Class Player, μια Class Board και μια Class Turn.

**Abstract Class Position**

**Attributes:**

1. protected int num; //number of the position(month)
2. protected String day; //day of the position
3. protected Player p = new Player(); //a player
4. protected Board b = new Board(); //the board
5. protected Sullogi sulD = new Sullogi("DealCard"); //DealCards Deck
6. protected Sullogi sulM = new Sullogi("MailCard"); //MailCards Deck
7. protected String image; //image of a position
8. protected Controller c=new Controller(); //the controller

**methods:**

1. public Position(int num, String day, String image); Constructor

Creates a new Position with the new number of the day and the new value for the day itself.

1. public String getImage(); Accessor(selector)

Returns the image of the position

1. public void setImage(String image); Transformer(mutative)

Changes the image of the position to image

1. public void setNum(int num); Transformer(mutative)

Changes the number of the position to num

1. public void setDay(String day); Transformer(mutative)

Changes the day of the position to day

1. public abstract void performAction(Player P, int dnum);

Method that defines the action that has got to be performed for the specific position(its overridden in every subclass of positions)

**Class CardPosition**

Κληρονομεί την Position και μέσω της εντολής super αποκτούν πρόσβαση σε αυτή και αρχικοποιούν τις num, image και day.

**Methods:**

1. public CardPosition(int num, String day, String image); Constructor
2. public Card getCards(Card C);

Returns the card on the top of the wanted deck(we will find the wanted deck from the parameter C)

1. public void performAction(Player P, int dnum)

Performs a specific action only for Card Positions.

**Class DicePosition**

Κληρονομεί την Position και μέσω της εντολής super αποκτούν πρόσβαση σε αυτή και αρχικοποιούν τις num, image και day.

**Attributes:**

1. private int money;//player’s money
2. private int diceNumber;//the number on the dice that got the player in this position

**Methods:**

1. public DicePosition(int num, String day, String image, int diceNumber, int money)

Creates a new DicePosition and sets new num, image, day, money and diceNumber

1. public void performAction(Player P, int dnum)

Performs a specific action only for Dice Positions.

**Classes OnePlayerDicePosition and BothPlayersDicePosition**

Κληρονομούν την DicePosition και μέσω της εντολής super αποκτούν πρόσβαση σε αυτή και αρχικοποιούν τις num, image, money, diceNumber και day. Επίσης, κάνουν override την public void performAction(Player P, int dnum).

**Class PayDayPosition**

Κληρονομεί την Position και μέσω της εντολής super αποκτούν πρόσβαση σε αυτή και αρχικοποιούν τις num=31, image(payday image(there is only one)) και day = “Wednesday”. Επίσης κάνει override τις setNum(int num) /\* num=31\*/, setDay(String day) /\* day= “Wednesday” \*/ και public void performAction(Player P, int dnum).

**Enums CardPositionsNames, OnePlayerDicePositionNames, BothPlayersDicePositionNames**

They contain the names of the cards of each type of position.

**Abstract Class Card**

**Attributes:**

1. private String image; //The image of the card

**Methods:**

1. public Card(String image); Constructor

Creates a new Card with a new image.

1. public String getImage(); Accessor(selector)

Returns the image of the card.

1. public void setImage(String image); Transformer(mutative)

Sets a new image for the card.

**Class DealCard**

Κληρονομεί την Card και μέσω της εντολής super αποκτοά πρόσβαση σε αυτή και αρχικοποιεί την image και έπειτα με τον δικό της constructor αρχικοποιεί και τις buyprice, sellprice. Επίσης, κάνει override την public void action().

**Attributes:**

1. private int buyprice; //the buy price of the card
2. private int sellprice; //the sell price of the card

**Class MailCard**

Κληρονομεί την Card και μέσω της εντολής super αποκτά πρόσβαση σε αυτή και αρχικοποιεί την image και έπειτα με τον δικό της constructor αρχικοποιεί και την type. Επίσης, κάνει override την public void action().

**Attributes:**

1. private MailCardsType type; //the type of the mail card.

**Enum MailCardsType**

Θα περιέχει τους τύπους των καρτών μηνύματος για να μπορώ να κάνω action() ανάλογο του type.

**Class Board**

**Attributes:**

1. private Position[] tiles= new Position[31];
2. private Sullogi deal=new Sullogi("DealCard");
3. private Sullogi mail=new Sullogi("MailCard");

**Methods:**

1. public Board(Position[] tiles , Sullogi deal, Sullogi mail); Constructor

Creates a new Board with new tiles (a Position array), Sullogi deal and mail.

1. public void initPositions();

initializes the positions.

**Class Player**

**Attributes:**

1. private String name;
2. private int money;
3. private int bills;
4. private int loan;
5. private Position pos;
6. private static int plnum=1;
7. private Sullogi sullogi= new Sullogi("DealCard");

**Methods:**

1. public Player(String name, int money, int bills, int loan, Position pos); Constructor

Creates a new Player with new name, money, bills, loan, pos

1. public Player(); Default constructor

sets name= "Player"+String.valueOf(plnum)

1. public void setName(String name); Transformer(mutative)

Sets the name of a player

1. public String getName(); Accessor(selector)

Returns the name of a player

1. public void setMoney(int money); Transformer(mutative)

Sets the money of a player

1. public int getMoney(); Accessor(selector)

Returns the money of a player

1. public void setLoan(int loan); Transformer(mutative)

Sets the loan of a player

1. public int getLoan(); Accessor(selector)

Returns the loan of a player

1. public void setBills(int bills); Transformer(mutative)

Sets the bills of a player

1. public int getBills(); Accessor(selector)

Returns the bills of a player

1. public void setPos(Position pos); Transformer(mutative)

Sets the pos of a player

1. public Position getPos(); Accessor(selector)

Returns the pos of a player

**Class Turn**

**Attributes:**

1. private final int pnum = 2; //number of players
2. protected int lastPlayer; //last player played
3. public int hasturn=0; //index of whom has the turn to play

**Methods:**

1. public Turn(); Default Constructor

Creates a new Turn with a new hasturn and a new lastPlayer value.

1. public Turn(int hasturn); Constructor

Creates a new turn with the given hasturn and the corresponded value of lastPlayer

1. public void setLastPlayer(int lastPlayer); Transformer(mutative)

Sets the last player that played

1. public void setHasturn(int hasturn); Transformer(mutative)

Sets the player that has turn to play

1. public int getLastPlayer(); Accessor(selector)

Returns the last player that played

1. public int getHasturn(); Accessor(selector)

Returns the player that has turn to play

**Class Sullogi**

**Attributes:**

1. public Stack<DealCard> DealCardStack; //Deck of deal cards
2. public Stack<MailCard> MailCardStack; //Deck of mail cards
3. private String type; //type of card (“DealCard” or “MailCard”)

**Methods:**

1. public Sullogi(String type); Constructor

Creates a Sullogi with a new type (of card) and a new deck that depends on the type.

1. public Stack<DealCard> getDealCardStack(); Accessor(selector)

Returns all the deal cards (the deal card deck)

1. public Stack<MailCard> getMailCardStack(); Accessor(selector)

Returns all the mail cards (the mail card deck)

1. public void setDealCardStack(Stack<DealCard> arr); Transformer(mutative)

Sets the new stack(deck) of deal cards to arr

1. public void setMailCardStack(Stack<MailCard> arr) Transformer(mutative)

Sets the new stack(deck) of mail cards to arr

1. public void initCards();

A method that initializes all the cards and shuffles them.

1. public boolean isEmptyDeal();

Returns true if the DealCardStack is empty

1. public boolean isEmptyMail();

Returns true if the MailCardStack is empty

1. public void addCard(DealCard dealCard);

Adds a deal card to the DealCardStack/PlayerCollection.

1. public void addCard(MailCard mailCard);

Adds a mail card to the MailCardStack.

1. public void removeCard(int i);

Removes a card from the Collection or Stack

1. public int sizeMail();

Returns the size of the MailCardStack;

1. public int sizeDeal();

Returns the size of the DealCardStack;

1. public DealCard getCardD(int index); Accessor(selector)

Returns the deal card in position 'index'

1. public MailCard getCardM(int index); Accessor(selector)

Returns the mail card in position 'index'

1. public void clearAll();

Clears DealCardStack and MailCardStack

## 

## Η Σχεδίαση και οι Κλάσεις του Πακέτου Controller

Η Controller αποτελεί το μυαλό του παιχνιδιού. Δημιουργεί ένα νέο παιχνίδι μαζί με όλα τα παρελκόμενα(όπως π.χ. παίκτες) και συνδέει το package Model με το package View, τα γραφικά. Περνάει τις επιλογές του κάθε παίκτη μέσω των γραφικών και εκτελεί ό,τι ενέργεια χρειάζεται κάθε φορά, ώστε το παιχνίδι να παίζεται σωστά. Μέσω αυτής, βγαίνει προφανώς ο νικητής αφού υπολογίζει τα κερδισμένα η χαμένα λεφτά του κάθε παίκτη και, τέλος, φροντίζει να ενημερώσει για το τέλος του παιχνιδιού.

**Class Controller**

**Attributes:**

1. private int months;
2. private Turn turn = new Turn();
3. private boolean notstarted;
4. private Board board;
5. private Player player1, player2;
6. private ArrayList<Player> players = new ArrayList<Player>();
7. private int dice;
8. private Sullogi dealcards = new Sullogi("DealCard");
9. private Sullogi mailcards = new Sullogi("MailCard");

**Methods:**

1. public Controller(); Constructor

Creates a new Controller with two new players and initializes the two decks.

1. public int rollTheDice();

Method that returns a random integer between 1-6 and this number is assigned to the dice

1. public int seeTurn();

Returns the index of the player that has turn to play.

1. public void set\_started();

Sets notstarted to false.

1. public boolean not\_started();

Returns true if the game has not started

1. public void init\_board();

Initializes some things(dealcards, mailcards, turn) for a new game

1. public boolean game\_has\_finished();

Return true if the game is finished.

1. public void set\_Turn(int index);

Gives the turn to player<index>

## Η Σχεδίαση και οι Κλάσεις του Πακέτου View

Στο πακέτο View εμπεριέχεται ο φάκελος που μας δόθηκε για τις κάρτες του παιχνιδιού, και μερικές ακόμα κλάσεις όπως η GraphicUI και η Jackpot, που θα αναλυθούν παρακάτω.

**Class GraphicUI:**

Attributes:

1. private Image PayDayImage;
2. private Image background;
3. private Image dice1;
4. private Image dice2;
5. private Image paw1;
6. private Image paw2;
7. private JButton RollDice1, MyDealCards1, GetLoan1, EndTurn1;
8. private JButton RollDice2, MyDealCards2, GetLoan2, EndTurn2;
9. private myDesktopPane basic\_panel;
10. private JLayeredPane player2\_field;
11. private JLayeredPane player1\_field;
12. private GridLayout board;
13. private PayDayCards cards;
14. private JTextField info\_box;
15. private Jackpot jackpot;

Methods:

1. public GraphicUI(); Constructor

Creates a new Window and initializes some buttons and panels

1. public void init\_infobox();

Initializes the info box

1. public void refresh\_infobox();

Refreshes the info box

1. public void init\_board();

Initializes the board

1. public void refresh\_board();

Refreshes the board (moving the paws etc)

1. public void init\_playerfields();

Initializes the players’ field

1. public void refresh\_player1field();

Refreshes the field of player1

1. public void refresh\_player2field();

Refreshes the field of player2

1. public void initComponents();

Initializes some buttons and labels

**Class myDesktopPane:**

Attributes:

1. private Image backImage;

Methods:

1. public myDesktopPane(); Constructor

Creates a new myDesktopPane and is used for putting a background image to a jdesktoppane

1. public void paintComponent(Graphics g); Overridden

**Class Jackpot:**

Attributes:

1. private int money;
2. private Image image;

Methods:

1. public Jackpot(); Constructor

Creates a new Jackpot

1. public void refresh\_money();

Refreshes the money of jackpot

1. public void init\_money();

Initializes the money of jackpot

There are also other 9 private classes that implement class ActionListener and are used for doing some action when a specific button is pushed (one class for every button).

## Η Αλληλεπίδραση μεταξύ των κλάσεων – Διαγράμματα UML

Σε αυτήν την ενότητα μπορείτε να συμπεριλάβετε διαγράμματα UML, και να εξηγήσετε μέσω αυτών την αλληλεπίδραση των κλάσεων (πχ μεταξύ των κλάσεων διαφορετικών πακέτων.

## Λειτουργικότητα (Β Φάση)

Σε αυτήν την ενότητα θα γράψετε στη Β φάση ποια ερωτήματα καταφέρατε να υλοποιήσετε είτε επιτυχώς είτε εν μέρει (και ενδεχομένως ποια όχι).

## Συμπεράσματα

Σε αυτήν την ενότητα θα γράψετε τα συμπεράσματα σας για την εργασία, τυχόν προβλήματα που συναντήσατε και γενικά ότι άλλο κρίνετε απαραίτητο να αναφερθεί.