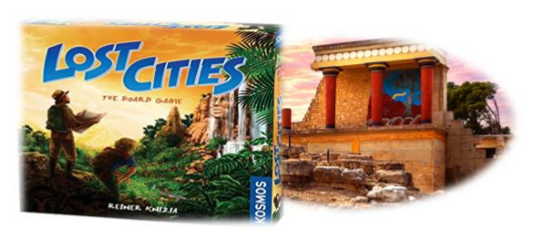
Αναζητώντας τα Χαμένα Μινωικά Ανάκτορα

Σαμαριτάκη Γεωργία AM3840 Project| Hy252 -Αντικειμενοστραφής Προγραμματισμός|

Δεκέμβριος 2017

**Σχεδιασμός**

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

Έτσι , σκοπός μας είναι ο Controller να είναι ο συνδετικός κρίκος των Model και view. Οπότε στη συνέχεια

της αναφοράς μας θα αναλύσουμε τα κομμάτια του Model και Controller που είναι σημαντικά

για αυτή τη φάση και τέλος θα αναφερθούμε και λίγο στο view.

Contents

[**Model Package** 2](#_Toc500686317)

[Enum Palace 2](#_Toc500686318)

[Abstract Card Class and subclasses 2](#_Toc500686319)

* [NumberedCard*(extends Card*) 2](#_Toc500686320)
* [Abstract Class SpecialCard*(extends Card)* 3](#_Toc500686321)
* [CardUML 3](#_Toc500686322)

[Interface Finding 4](#_Toc500686323)

* [Enum RareFinding*(implements Finding)* 4](#_Toc500686324)
* [Enum Fresco*(implements Finding)* 4](#_Toc500686325)
* [Class SnakeGoddess 4](#_Toc500686326)
* [FindingUML 4](#_Toc500686327)

[Abstract Class Pawn 5](#_Toc500686328)

* [PawnUML 5](#_Toc500686329)
* [Class Theseus*(extends Pawn)* 5](#_Toc500686330)
* [Class Archeologist*(extends Pawn)* 5](#_Toc500686331)

[Abstract class Position 6](#_Toc500686332)

* [Class FindingPosition 6](#_Toc500686333)
* [Class SimplePosition 6](#_Toc500686334)
* [UMLPosition+Path 6](#_Toc500686335)

[Class Path 7](#_Toc500686337)

[Class Deck 7](#_Toc500686338)

[Class Player 8](#_Toc500686339)

[Class Board 9](#_Toc500686340)

[**Model Controller** 10](#_Toc500686341)

[Class Controller 10](#_Toc500686342)

[**Model View** 11](#_Toc500686343)

[Class View *extends JFrame* 11](#_Toc500686344)

[**UML** 12](#_Toc500686345)

## Model Package

**Enum Palace**

Αποτελείται από τα 4 μινωικά ανάκτορα του παιχνιδιού

*Κnossos,Malia,Phaistos,Zakros* enumeration values

και χρησιμοποιείται σχεδόν από όλες τις υπόλοιπες κολάσεις.

*Methods*:

public String to String(); //Overridden method to String

returns the String name of the Palace

**Abstract Card Class and subclasses**

*Attributes*:

* Palace palace; //common characteristic of all cards - The palace they belong to
* Private String image;

*Methods*:

|  |  |
| --- | --- |
| public abstract boolean matchCard(Card c); | Observer  Check if the card c can be played over this card |
| public Palace getPalace(); | Accessor  Returns the palace this card belongs in |
| public boolean isSpecial(); | Observer  Returns true for special cards |
| public abstract String to String(); | Accessor(overridden)  Returns the name of the card |
| Setimage(String image) – Getimage() | Methods for graphics |

* **NumberedCard(extends Card)**

Cards with numeric value 1-10 20 for each palace

*Attributes:*

* private final int value; //The value of the numbered card

*Methods:*

|  |  |
| --- | --- |
| public int getValue(); | Accessor  Returns the value of the card |
| public boolean matchCard(Card c); | Observer  Returns true if the card c equal or more of the last card played or |
| public boolean isSpecial() | Overridden method  Returns false always |
| public String toString() | Returns String  “NumberedCard of value” with value of card |

* **Abstract Class SpecialCard(extends Card)**

SpecialCard consists of its two subclasses Ariadne and Minotaur

*Methods:*

|  |  |
| --- | --- |
| public boolean isSpecial() | Observer  Returns true overriding super |
| public boolean isMinotaur() | Observer  Returns false unless overridden |
| public boolean isAriadne() | Observer  Returns false unless overridden |

* **Minotaur(extends SpecialCard)**

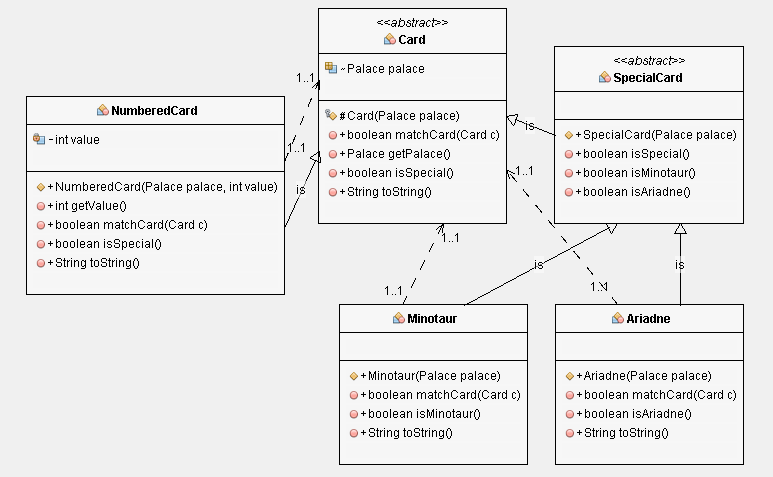
*Methods:*

|  |  |
| --- | --- |
| public boolean matchCard(Card c) | Observer  Checks if Card c can be played over minotaur |
| public boolean isMinotaur() | Observer  Returns true |
| public String toString() | Accessor  Returns “Minotaur Card” with card’s palace |

* **Ariadne(extends SpecialCard)**

*Methods:*

|  |  |
| --- | --- |
| public boolean matchCard(Card c) | Observer  Returns true because Ariadne can be played over all the cards |
| public boolean isAriadne () | Observer  Returns true |
| public String toString () | Accessor  Returns “Ariadne Card” with card’s palace |

**CardUML**

**Interface Finding**

Acts as a connection between subclasses Fresco,RareFinding,SnakeGoddess

*Methods:* public boolean isStatue();

Methods for encapsulation of image and description of each finding

* **Enum RareFinding(implements Finding)**

Consists of the 4 rare findings as Finding(value)

DiskOfFaistos(35), RingOfMinoa(25), JewelOfMalia(25), RhytonOfZakros(25);

*Attributes:*

* final private int value; //value of the finding
* String image,description;

*Methods:* All methods inherited plus

|  |  |
| --- | --- |
| public String toString() | Accessor  Returns the name of the enum |
| public int getValue() | Accessor  Returns the value fo the rare finding |
| public boolean isStatue() | Observer  Returns false |

* **Enum Fresco(implements Finding***)*

Consists of the 6 frescos according to the strg image given

Fresco1(20), fresco2(20), fresco3(15), fresco4(20), fresco5(15), fresco6(15);

*Attributes:*

* final private int value; //value of the finding
* String image,description;

*Methods:* All methods inherited plus

|  |  |
| --- | --- |
| public String toString() | Accessor  Returns the name of the enum |
| public int getValue() | Accessor  Returns the value of the rare finding |
| public boolean isStatue() | Observer Returns false |

* **Class SnakeGoddess**

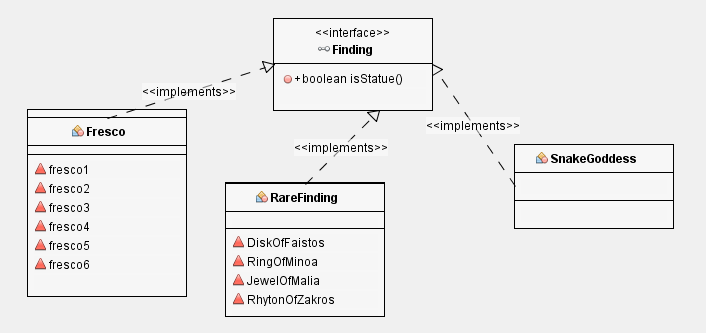
*Attributes:*

* String image,description;

*Method:* All Methods inherited plus

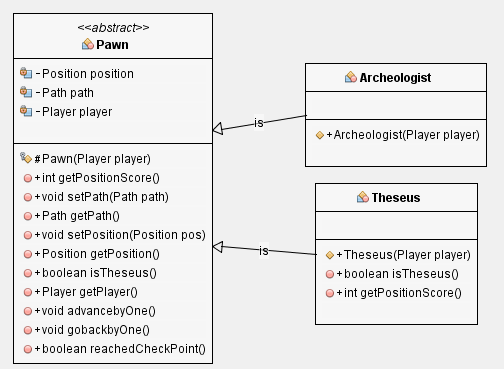
Public boolean isStatue() // Observer Returns true

**FindingUML**



**Abstract Class Pawn**

***PawnUML***

*Attributes:*

* private Position position;

//the position the pawn is on

* private Path path;

//the path the pawn is on

* private final Player player;

//the owner of the piece

*Methods:*

|  |  |
| --- | --- |
| public int getPositionScore() | Accessor  Returns the points of the position the pawn is on |
| public void setPath(Path path) | Transformer  Sets the path the pawn is on to path |
| public Path getPath() | Accessor  Returns the path the pawn is on |
| public void setPosition(Position pos) | Transformer  Sets the pawns position to pos |
| public Position getPosition() | Accessor  Returns the position of the pawn |
| public boolean isTheseus() | Observer  True if the pawn is an instance of theseus |
| public Player getPlayer() | Accessor  Returns the owner of the pawn |
| public void advancebyOne() | Transformer  Advances pawn by one in the path providing that its not in the last place |
| public void gobackbyOne() | Transformer  Returns pawn one place back providing its not in the last place |
| public boolean reachedCheckPoint() | Observer  Returns true if the pawn has passed position 7 of the path providing it has begun a path |

* **Class Theseus(extends Pawn)**

*Methods:*

|  |  |
| --- | --- |
| public boolean isTheseus() | Observer  Returns always true |
| public int getPositionScore() | Accessor  Returns the position score doubled  (theseus earns double the value of the position) |

* **Class Archeologist(extends Pawn)**

**Abstract class Position**

*Attributes:*

* private final int points;
* private final int posnumber; //position number in path
* private final Path path;

*Methods:*

|  |  |
| --- | --- |
| public int getPoints() | Accessor  Returns the points specified in this position  Using posnumber(in path) |
| public Path getPath() | Accessor  Returns Path the path the position belongs to |
| public abstract void availableMoves() | Observer  Informs about the available moves |

* **Class FindingPosition**

*Attributes:*

* Finding finding; // the findingburied in this position

*Methods:*

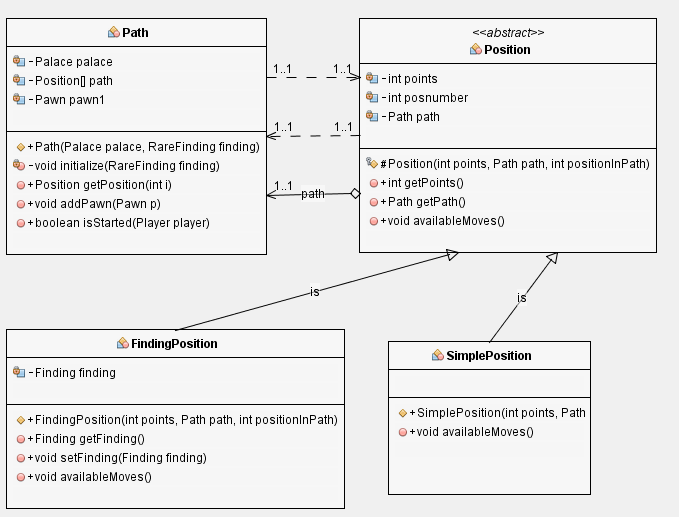
|  |  |
| --- | --- |
| public Finding getFinding() | Accessor  Returns finding in the current position |
| public void setFinding(Finding finding) | Transformer  Puts the finding in this position |
| public void availableMoves() | Accessor  Informs about the available moves of the pawn in this position |

* **Class SimplePosition**

*Methods:* public void availableMoves()

Observer there are no available moves in this position

**UMLPosition+Path**



**Class Path**

*Attributes:*

* private final Palace palace;
* private final Position []path = new Position[9];
* private ArrayList<Pawn> pawns;

*Methods:*

|  |  |
| --- | --- |
| private void initialize(RareFinding finding) | Transformer  Sets position 2,4,6,8,9 to special and rest to simple Adds rare finding randomly to one of FindingPositions |
| public Position getPosition(int i) | Accessor  Returns the i th position of the path |
| public void addPawn(Pawn p) | Transformer  Adds pawn to path |
| public boolean isStarted(Player player) | Observer  Returns true if the player has placed a pawn on this path false |

**Class Deck**

*Attributes:*

* private final ArrayList<Card> deck;

*Methods:*

|  |  |
| --- | --- |
| private void initialize() | Transformer  Initializes all cards of deck |
| public ArrayList getDeck() | Accessor  Returns the array list with the deck |
| public void shuffleDeck() | Transformer  Shuffles card of deck |
| public Card drawCard() | Accessor && Transformer  Draws card from deck removes it and returns it |
| public boolean isEmpty() | Observer  Returns true if there are no more available cards |
| public int availableCards() | Accessor  Returns the number of available cards in deck |

**Class Player**

*Attributes:*

* private final Card hand[];
* private final NumberedCardT LastPlayed[];
* private ArrayList<Finding> Syllogi;
* private ArrayList<Finding> Fresco;
* private final Pawn pawns[];
* private int Score;
* private int NumOfStatues;

*Methods:*

|  |  |
| --- | --- |
| public Card[] getCards() | Accessor  Returns a new array with the available cards in deck |
| public void playCard(Card c) | Transformer  Precondition: Card can be played over the previous card in path and card has to be contained in players hand  Postcondition: Sets the card the play |
| public void playPawn(Pawn p, Path path) | Transformer  Precondition: Player has available pawns  Postcondition: Plays pawn on specified path && adds pawn to path |
| public void discardCard(Card c) | Transformer  Removes card c from hand and returns it to be added to discard pile |
| public Card getLastCard(Palace palace) | Accessor  Returns the last card played on each palace |
| public void AddCard(Card C) | Transformer  Precondition: Player does not have 8 card on hand  Postcondition: Adds card c to palyers hand |
| private void AddStatue() | Transformer  Increases number of statues by one |
| private boolean hasPhotographed(Fresco fresco) | Observer  Checks if the fresco is in the frescos of the player, if it is returns true |
| private void photographFresco(Fresco fresco) | Transformer  Precondition: The fresco has not been photographed  Postcondition: Adds fresco to fresco ArrayList |
| private void addRareFinding(RareFinding finding) | Transformer  Takes the rare finding from its posiition and adds it to Syllogi |
| public void takeFinding(FindingPosition pos) | Transformer  Precondition: It’s the players turn  Postcondition : Checks the type of the finding in position and calls the relevant  private method above |
| private void updateScore() | Transformer  Calculates the sum of pawn's positions rare findings values and frescos values and updates Scores |
| public int getScore() | Accessor  Returns score |
| public Pawn[] availablePawns() | Accessor  Returns a pawn array of the pawns that have yet to be played |
| public Path[] unplayedPaths() | Accessor  Returns A path array of the path that the player has not put a pawn on |

**Class Board**

*Attributes:*

* private final Path paths[] = new Path[4];
* private ArrayList<Card> DiscardPile;

*Methods:*

|  |  |
| --- | --- |
| private void distributeFindings() | Transformer  Creates all frescos and statues and distributes them randomly |
| public int checkPointsReached() | Accessor  Calculates from the player's pawns how many checkpoints have been reached |
| public void discardCard(Card c) | Transformer  Moves card c to discardpile |

## Model Controller

**Class Controller**

*Attributes:*

* private final Player player1, player2;
* private final Board board;
* private Deck deck;
* View view;
* private boolean turn;

*Methods:*

|  |  |
| --- | --- |
| private void init\_player\_cards() | Transformer  Draws 8 cards for each player and adds them to his hands |
| public Player getTurn() | Observer  Returns the player who plays |
| public void endTurn() | Transformer  Switches turn |
| public boolean isFinished() | Observer  Returns True if 4 checkpoints have been reached or the deck has been emptied |
| public Player getWinner() | Accessor  PreCondition: The game has ended  Postcondition Returns the winner comparing the two scores |

*Listeners:*

|  |  |
| --- | --- |
| private class Player1Listener implements ActionListener | Actions for player 1  Left Click on card plays card  Right Click Discards card |
| private class Player2Listener implements ActionListener | Actions for player2  As above |
| private class DeckListener implements ActionListener | Left Click draws card |
| private class FrescoListener implements ActionListener | Fresco button shoes frescos of each player |
| private class FindingInfoListener implements ActionListener | Hovering over rare findings shows description |

## Model View

**Class View extends JFrame**

*Attributes:*

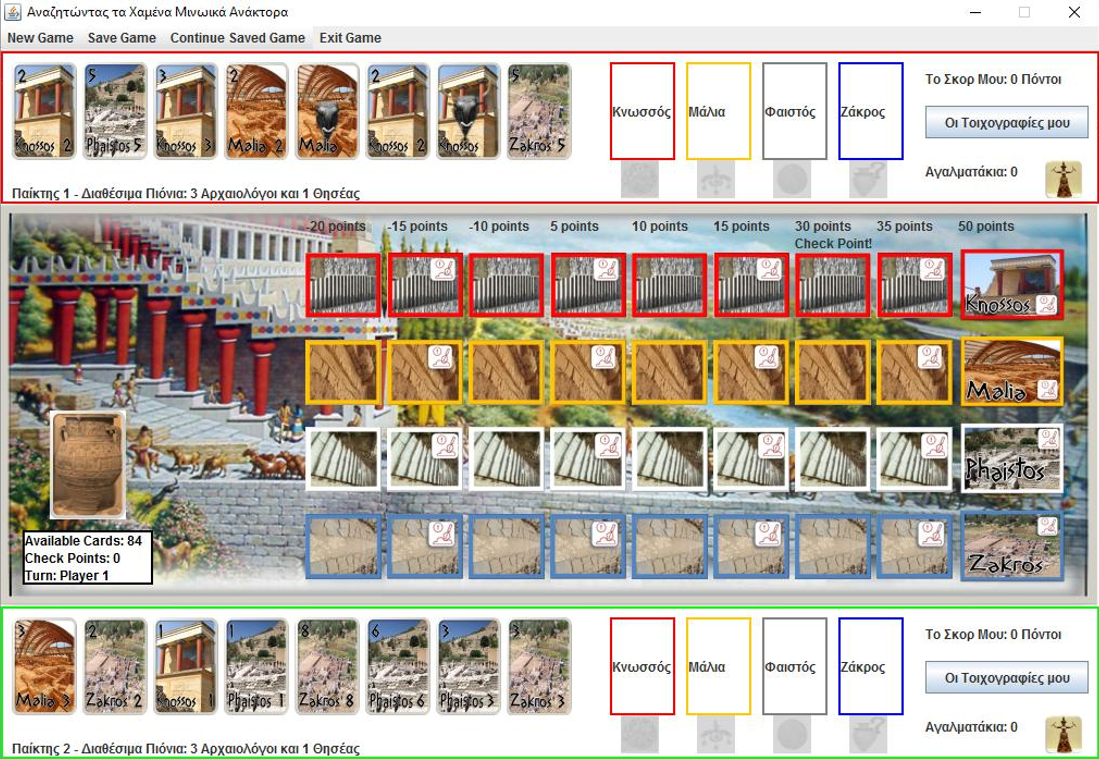
* JMenu menu;
* JLayeredPane pane1, pane2, mainpane;
* JButton Deck, Frescos1, Frescos2;
* JButton Cards1[], Cards2[];
* JLabel Info, pawns1[], pawns2[], availablePawns1[],

availablePawns2[], Score1, Score2, Statues1, Statues2;

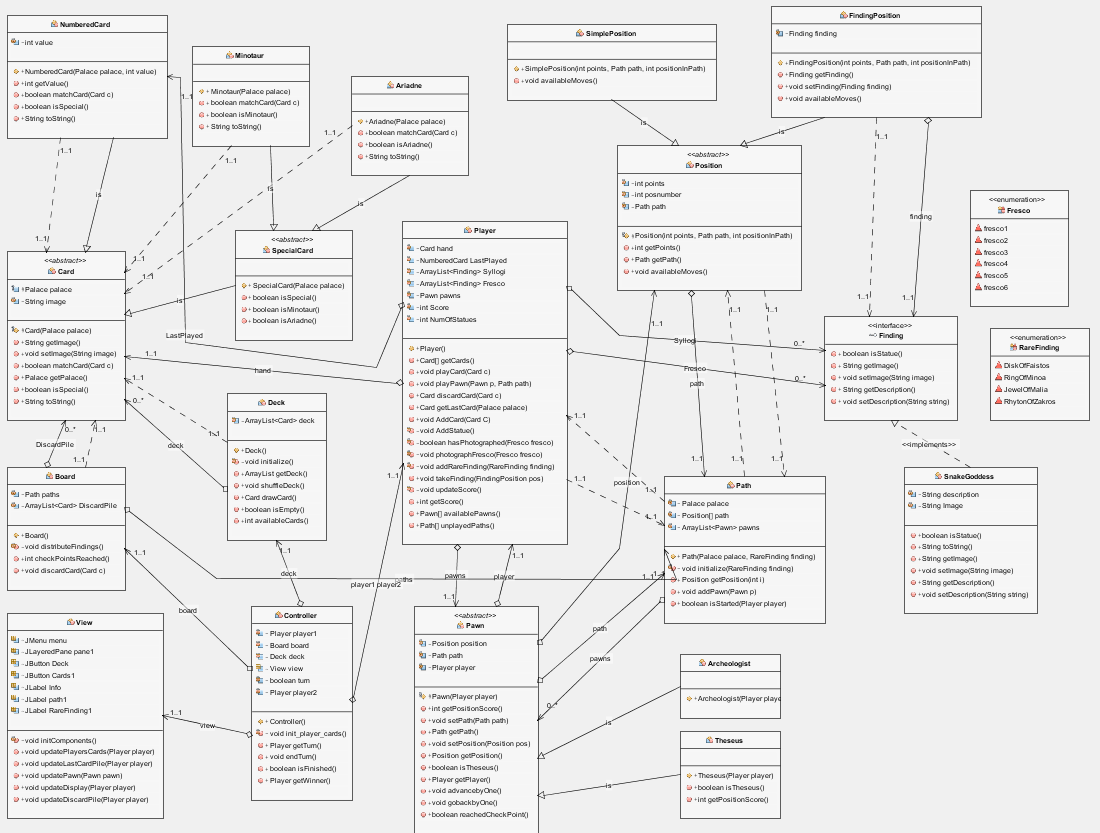
* JLabel path1[], path2[], path3[], path4[];
* JLabel RareFinding1[], RareFinding2[];

*Methods:*

|  |  |
| --- | --- |
| private void initComponents() | Transformer  Initializes buttons and labels |
| public void updatePlayersCard(Player player) | Transformer  Updates the cards of player in display |
| public void updateLastCardPile(Player player) | Transformer  Updates Last card of player |
| public void updatePawn(Pawn pawn) | Transformer  Updates the position of pawn  (removes previous occurrence and repaints) |
| public void updateDisplay(Player player) | Transformer  Updates labels of player |
| public void updateDiscardPile(Player player) | Transformer  Updates discard pile in display |



## Project UML



Τέλος