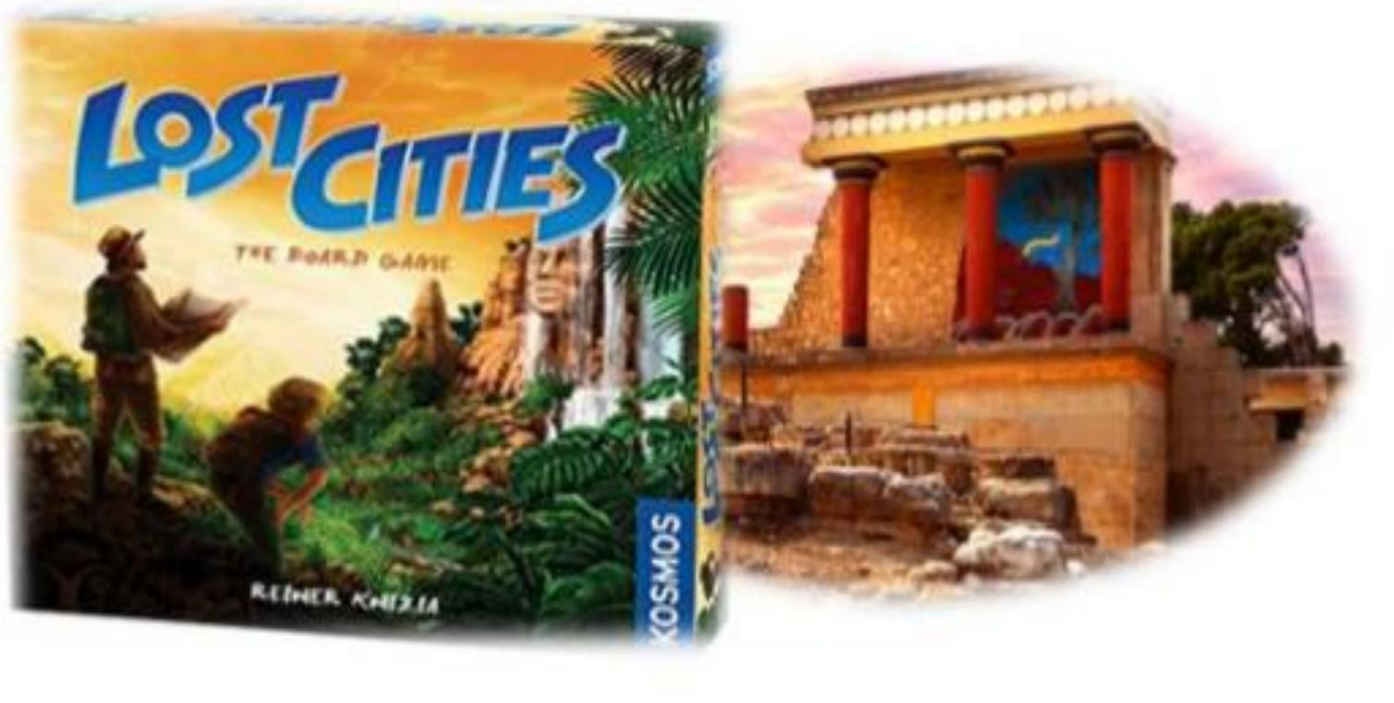
SEARCHING FOR THE LOST MINOAN PALACES

Samaritaki Georgia AM3840

Project| Hy252 -Object Oriented Programming|
December 2017



Planning

The implementation of the work will be based on the MVC (Model View Controller) model.
Thus, our goal is for the Controller to be the connecting link between Model and view. So, in the continuation of our report, we will analyze the parts of the Model and Controller that are important for this phase and finally we will refer a little to the view.

Phase B

The changes made in the second phase of the project were mainly in the Controller and View parts related to graphics and digital design. Some minor cuts have also been made to the model
but these mainly consist of deleting entire redundant functions. The changes have the ! .

Contents

Model Package

Enum Palace

It consists of the 4 Minoan palaces in the game

Knossos, Malia, Phaistos, Zakros enumeration values

and is used by almost all other hells.

Methods:

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

returns the String name of the Palace

! public String getDescription(); // Accessor returns a short description of each palace

Abstract Card Class and subclasses

Attributes:

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

Methods:

public Palace getPalace();	Accessor Returns the palace this card belongs in
! public int getValue() (removed from subclasses added here)	Returns the value number of each card Ariadne has 11 and minotaur has 12
public abstract String to String();	Accessor(overridden) Returns the name of the card
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 boolean matchCard(Card c); Observer	Returns true if the card is equal or more than the last card played
public String toString()	Returns String “Numbered Card of value” with value of card

Abstract Class SpecialCard(extends Card)

SpecialCard consists of its two subclasses Ariadne and Minotaur

ÿ Minotaur(extends SpecialCard)

Methods:

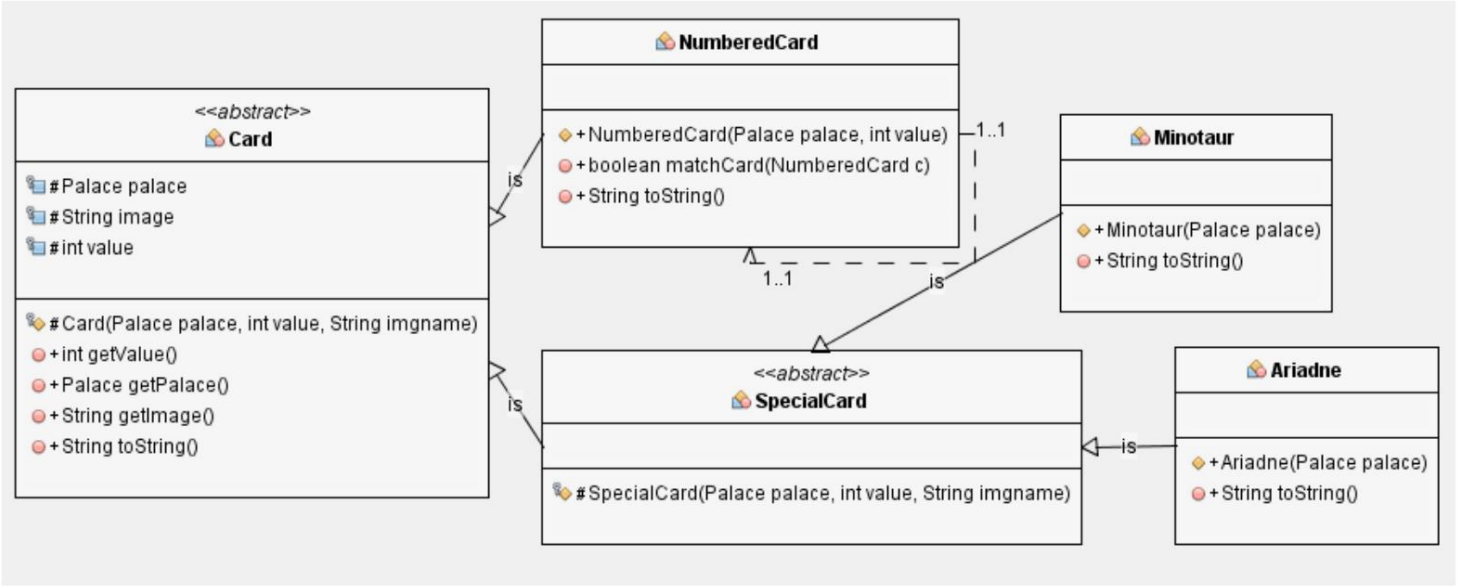
public boolean matchCard(Card c)	Observer Checks if Card c can be played over minotaur
public String toString()	Accessor Returns “Minotaur Card” with card's palace

ÿ Ariadne(extends SpecialCard)

Methods:

public boolean matchCard(Card c)	Observer Returns true because Ariadne can be played over all the cards
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 String getImage(); //Methods for graphic environment
public String getDescription();

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

Methods: All methods inherited plus

public String toString()	Accessor Returns the name of the enum
! public String getValue()	Returns the points value of the enum

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?

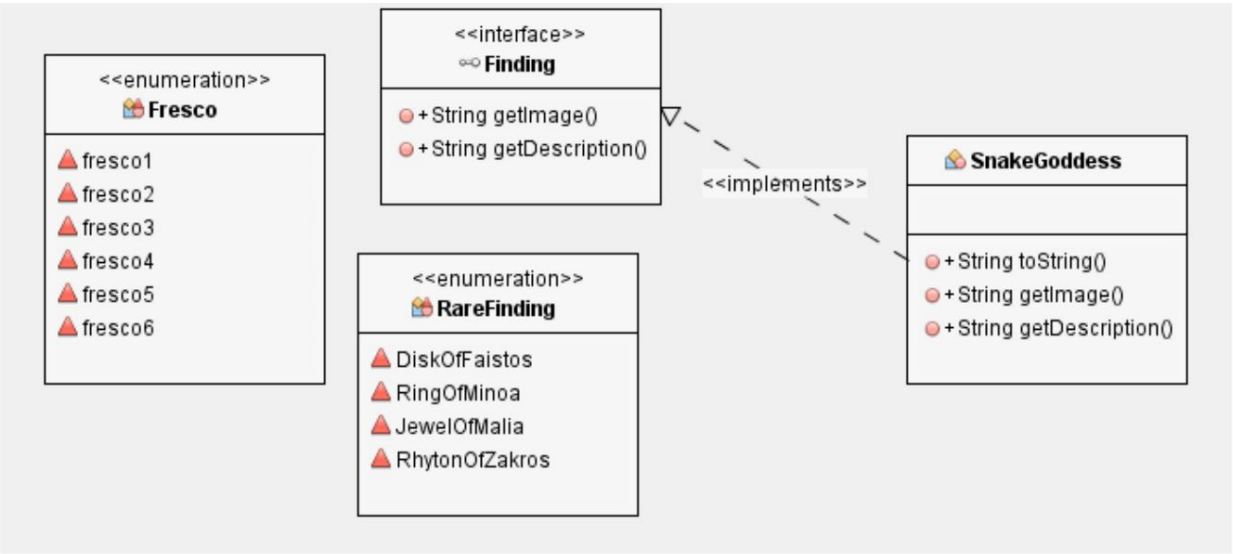
Methods: All methods inherited plus

public String toString()	Accessor Returns the name of the enum
public int getValue()	Accessor Returns the value of the rare finding

Class SnakeGoddess

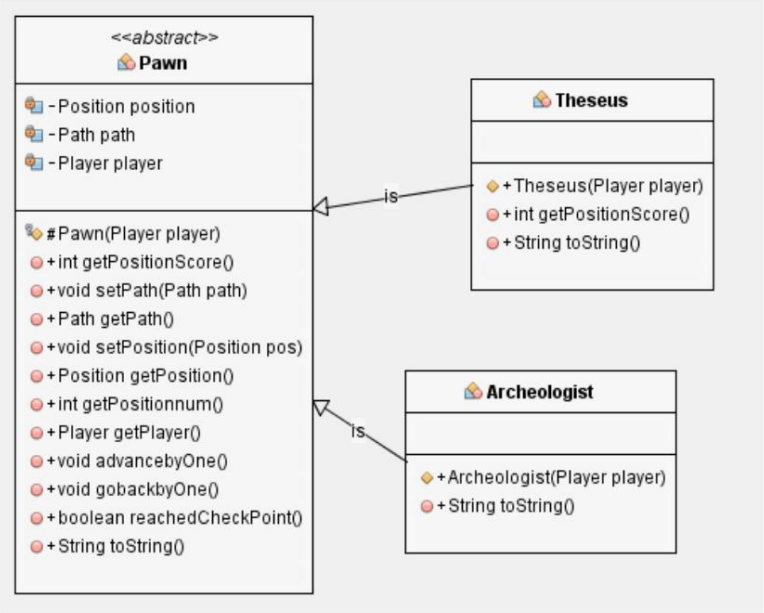
Method: All Methods inherited

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 int getPositionnum();	Accessor Returns the number position the pawn is in the path
public Position getPosition()	Accessor Returns the position of the pawn
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
Public abstract String toString();	Overridden method Returns the name of the pawn

Class Theseus(extends Pawn)

Methods: All inherited plus:

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;
- ! protected boolean hasFinding; // updated by each class

Methods:

public int getPoints()	Accessor Returns the points specified in this position Using posnumber(in path)
! public int getNum()	Accessor Returns the posnumber
! public int getPoints();	Accessor Return the points specified in this position
! public boolean hasFinding();	Observer Returns true if the position has been found else false
public Path getPath()	Accessor Returns Path the path the position belongs to

Class FindingPosition

Attributes:

- Finding finding? // the finding buried in this position

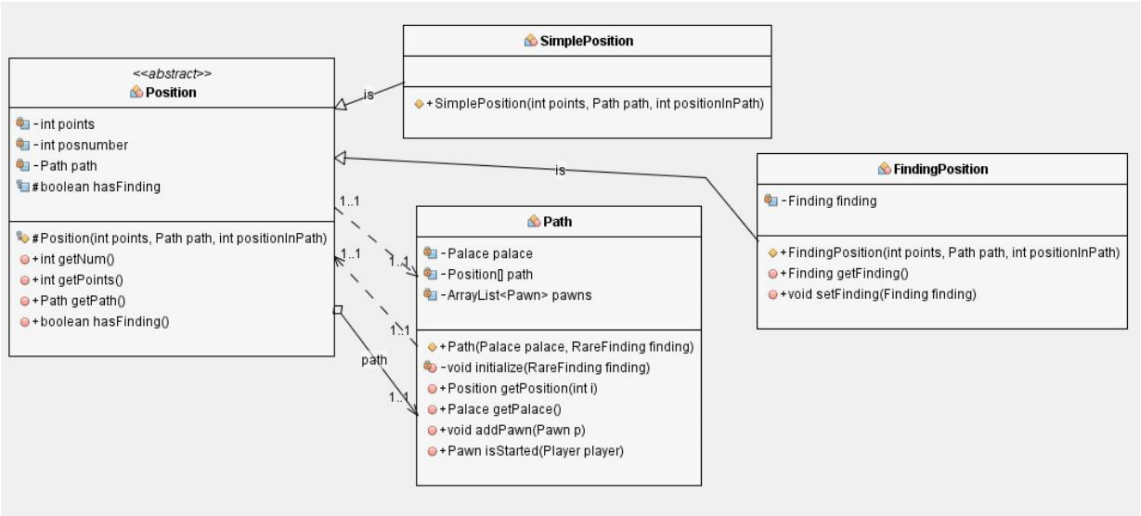
Methods:

public Finding getFinding()	Accessor Returns finding in the current position And sets hasFinding boolean to false
public void setFinding(Finding finding) Transformer	Puts the finding in this position and sets hasFinding boolean to true

Class SimplePosition

! Sets the hasFinding boolean to false

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 Palace getPalace()	Accessor Returns the enum value of the palace of 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 cards 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 an array with the available cards of the player
~~public void discardCard(Card c)~~ ! public void ReplaceCard(int index, Card c)	Transformer Replaces card c in hand on index position
public Card getLastCard(Palace palace)	Accessor Returns the last card played on the specified palace
public void AddCard(Card C)	Transformer Precondition: Player does not have 8 cards on hand Postcondition: Adds card c to palyers hand
! public void updateLastCard(NumberedCard add)	Transformer Changed the last card of same palace in lastCard array
public void takeFinding(FindingPosition pos)	Transformer Postcondition : Checks the type of the finding in position and adds it accordingly to player
! public int getScore()	Accessor Calculates the sum of pawn's positions rare findings values, statues, frescos values and updates Scores Returns score
! public int statuesCollected()	Accessor Returns the int value of the sumOfStatues

Class Board

Attributes:

- private final Path paths[] = new Path[4];

Methods:

private void distributeFindings()	Transformer Creates all frescos and statues and distributes them randomly
!public Path getPath(Palace palace)	Transformer Returns the path of the specified palace

Model Controller

Class Controller

Attributes:

- private final Player player1, player2;
- private final Board board;
- private final Deck deck;
- View view?
- private boolean phaseB,turn;

Methods:

public void init()	Transformer Initializes views and listeners
private void init_player_cards()	Transformer initializes players cards in the beginning
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 String getWinner()	Accessor PreCondition: The game has ended Postcondition Returns the winner comparing the two scores
private void setListeners()	Transformer Connects buttons with listeners
public int checkPointsReached()	Accessor Calculates from the player's pawns how many checkpoints have been reached
public void availableMoves(Player player, position pos)	Transformer Updates player class and view of the item in position
public Pawn choosePawn(Pawn playerPawn[])	Transformer Displays a popup dialog that allows the player to choose a pawn
public void updateView()	Transformer Postcondition updates the information displayed on the screen

Listeners:

private class CardListener implements MouseListener	Card listener
private class DeckListener implements MouseListener	Deck button listener
private class ButtonListener implements MouseListener	Fresco buttons

Model View

The view is the entire graphics part of the game. It consists of a main JFrame which has 3 Jpanels (pane1, pane2, mainpane) one for each player and a main one which has the board with the paths. Each player's cards are buttons for the game and after he chooses (or discards) a card I have conventionally chosen to have an indication of a valid move to make it gray as I move into the second phase of each turn.

The view also uses a JExtension that subclasses JLayeredPane and provides the ability add image for background?

Class View extends JFrame

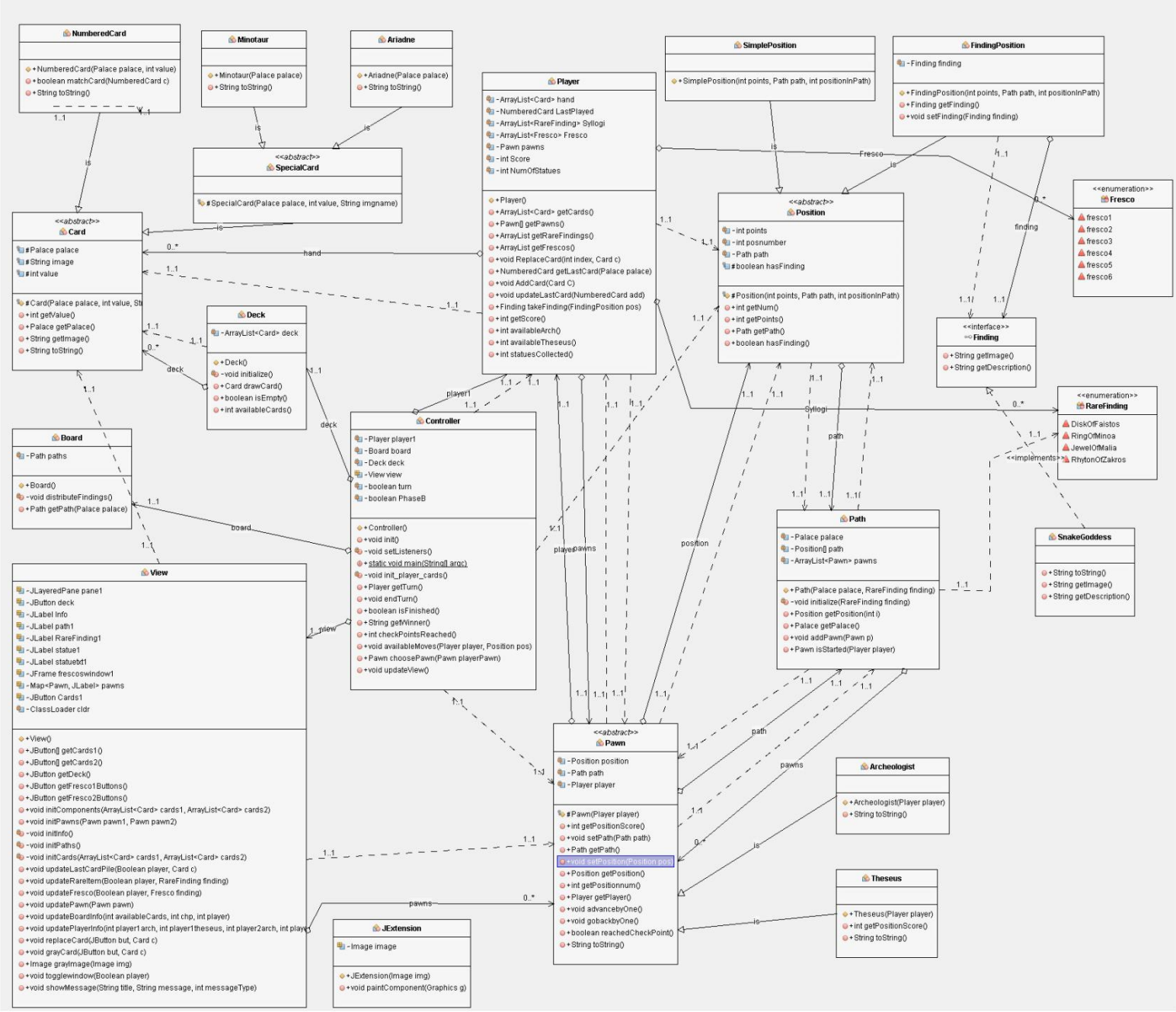
Attributes:

- JLayeredPane pane1, pane2, mainpane;
- JButton deck, F1, F2, Cards1[] = new JButton[8], Cards2[] = new JButton[8];//8 for each
- JLabel Info, availablePawns1, player1LastCard[], player2LastCard[], availablePawns2, Score1, Score2, Statues1, Statues2;
- JLabel path1[], path2[], path3[], path4[], pathPoints[];
- JLabel RareFinding1[], RareFinding2[], Frescos1[], Frescos2[];
- JLabel statue1, statue2, statuetxt1, statuetxt2;
- JFrame frescoswindow1, frescoswindow2;
- Map<Pawn, JLabel> pawns = new HashMap<>(9);
- private final ClassLoader cldr;

Methods: Apart from getter classes

private void initComponents() ~and other init… methods	Transformer Initializes buttons and labels
public void updateLastCardPile(Boolean player, Card c)	Transformer Updates Last card of player
public void updateRareItem(Boolean player, RareFinding finding)	Updates rare items of player ~ “ungrays” the image of rare item
public void updateFresco(Boolean player, fresco finding)	Updates the window with the frescos of Player with the new fresco
public void updatePawn(Pawn pawn)	Transformer Updates the position of pawn ~ moves the JLabel in path
public void updateBoardInfo(..) public void updatePlayerInfo(…)	Transformer Update info on screen
public void replaceCard(JButton but, Card c)	Transformer Replaces the image in JButton but of that of card c
public void grayCard(JButton but, Card c)	Transformer Sets image of card c grayed in JButton but
public Image grayImage(Image img)	Transformer Returns the grayed version of img
public void toggleWindow(Boolean player)	Transformer Toggles visibility of fresco window of player
public void showMessage(String title, String message, int messageType)	Brings up an information-message dialog titled "title".

Project UML



End