

**Πανεπιστήμιο Κρήτης –Τμήμα Επιστήμης Υπολογιστών**

**ΗΥ252– Αντικειμενοστρεφής Προγραμματισμός**

**Διδάσκων: Ι. Τζίτζικας**

**Χειμερινό Εξάμηνο 2020-2021**

LOST CITIES

Θεόδωρος Παπαλιάγκας

csd4980

15/01/2025

Αναζητώντας τα Χαμένα Μινωικά Ανάκτορα

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)8

[4. Η Σχεδίαση και οι Κλάσεις του Πακέτου View 2](#_Toc530045461)0

[5. Η Αλληλεπίδραση μεταξύ των κλάσεων – Διαγράμματα UML 2](#_Toc530045462)1

6[. Συμπεράσματα 2](#_Toc530045464)1

## Εισαγωγή

Στον τρόπο που υλοποιούμε το έργο μας, χρησιμοποιούμε το προγραμματιστικό μοντέλο MVC (Model-View-Controller). Σύμφωνα με αυτό το μοντέλο, η δημιουργία ολόκληρου του προγράμματος διαιρείται σε τρεις κατηγορίες:

1. "Model": Σε αυτό το στάδιο υλοποιούμε τη λειτουργικότητα του παιχνιδιού. Συγκεκριμένα, ασχολούμαστε με το πώς λειτουργούν οι κάρτες, οι διάφορες περιοχές του παιχνιδιού και τη διαχείριση των καρτών.

2. "View": Σε αυτό το στάδιο υλοποιείται το τμήμα που βλέπει ο χρήστης, δηλαδή το διεπαφικό περιβάλλον χρήστη (UI) ή γραφικό περιβάλλον χρήστη (GUI).

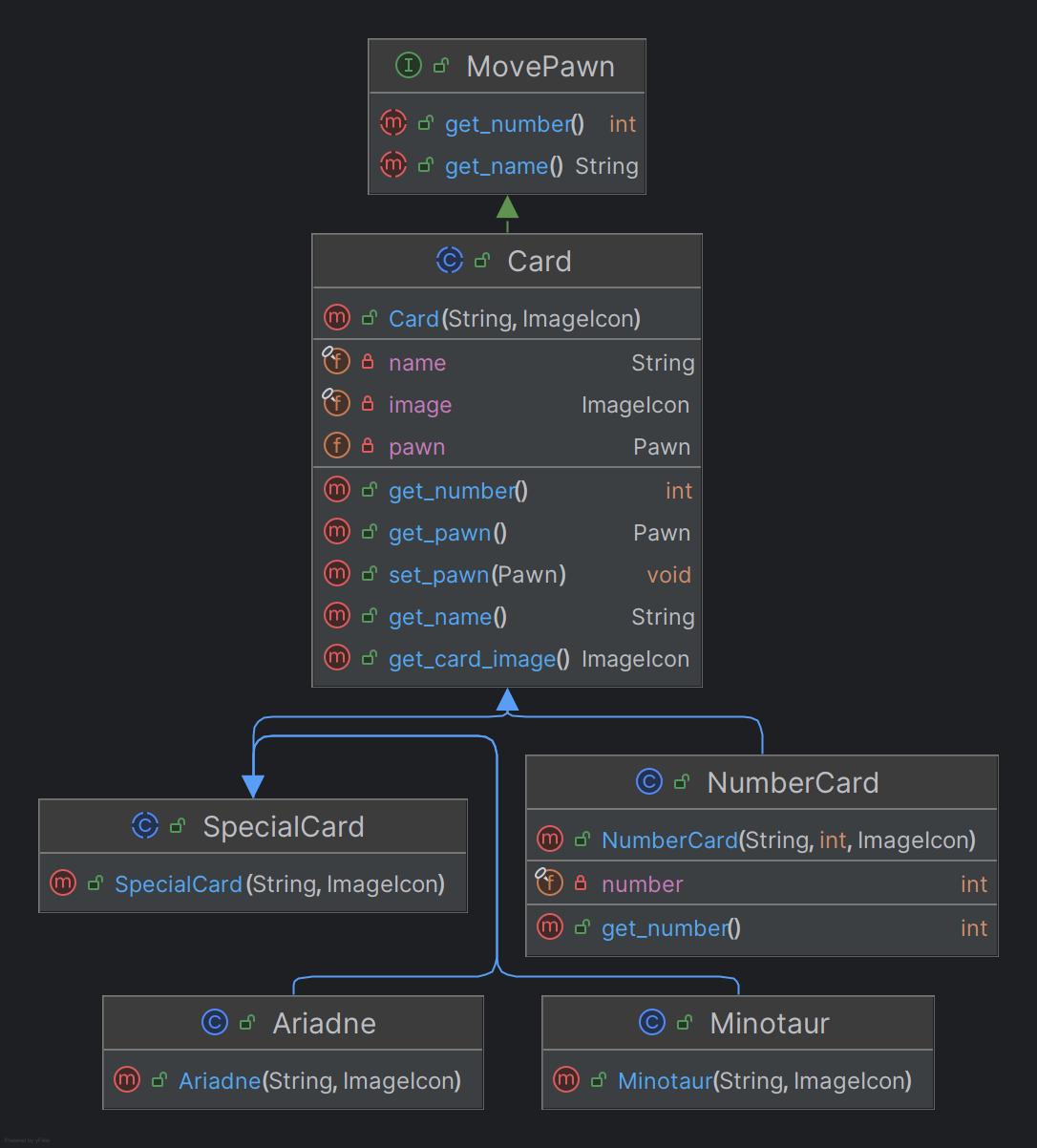
3. "Controller": Σε αυτό το στάδιο περιλαμβάνονται τμήματα κώδικα που οργανώνουν την επικοινωνία μεταξύ του "View" και του "Model".

Με αυτόν τον τρόπο, διευκολύνουμε τη συνολική διαδικασία ανάπτυξης και επιταχύνουμε την εργασία μας, επειδή κάθε κατηγορία είναι υπεύθυνη για ένα συγκεκριμένο κομμάτι του προγράμματος, καθιστώντας τη συντήρηση και την επέκταση του πιο ευέλικτη.

## Η Σχεδίαση και οι Κλάσεις του Πακέτου Model

Συνοπτικά, το πακέτο Model περιλαμβάνει πολλές κλάσεις που εκτελούν μικρές λειτουργίες. Αυτές οι λειτουργίες συνδέονται μεταξύ τους και, με την κατάλληλη οργάνωση από το πακέτο Controller, αποτελούν τον πυρήνα – δομή της λειτουργικότητας του παιχνιδιού. Περιλαμβάνουν όλες τις απαραίτητες κλάσεις όπως Player, Pawn, Card, Deck κλπ.

**Package Cards:**

****

Interface MovePawn

String get\_name();

Επιστρέφει το όνομα της κάρτας

int get\_number();  
 Επιστρέφει το νούμερο της κάρτας

class Card (abstract) implements MovePawn

Δεδομένα:

private final String name;

Το όνομα της κάρτας

private final ImageIcon image;

Η εικόνα της κάρτας

private Pawn pawn;

Ποιό πιόνι έχει η κάθε κάρτα

Methods:

public void set\_pawn(Pawn pawn);

Θέτει το πιόνι της κάρτας  
public String get\_name();  
public ImageIcon get\_card\_image();

Επιστρέφει την εικόνα της κάρτας  
  
public int get\_number();  
public Pawn get\_pawn();

Επιστρέφει το πιόνι της κάρτας  
public Card(String name, ImageIcon image);

Θέτει τις τιμές από τις παραμέτρους που δέχεται

class NumberCard extends Card

Δεδομένα:

private int number;

Ο αριθμός της κάρτας

Methods:

public int get\_number();

Επιστρέφει τον αριθμό της κάρτας  
  
public NumberCard(String name, int number, ImageIcon image);

Θέτει τις τιμές από τις παραμέτρους που δέχεται

class SpecialCard extends Card

Methods:

public SpecialCard(String name, ImageIcon image);

Θέτει τις τιμές από τις παραμέτρους που δέχεται

class Minotaur extends SpecialCard

Methods:

public Minotaur(String name, ImageIcon image);

Θέτει τις τιμές από τις παραμέτρους που δέχεται

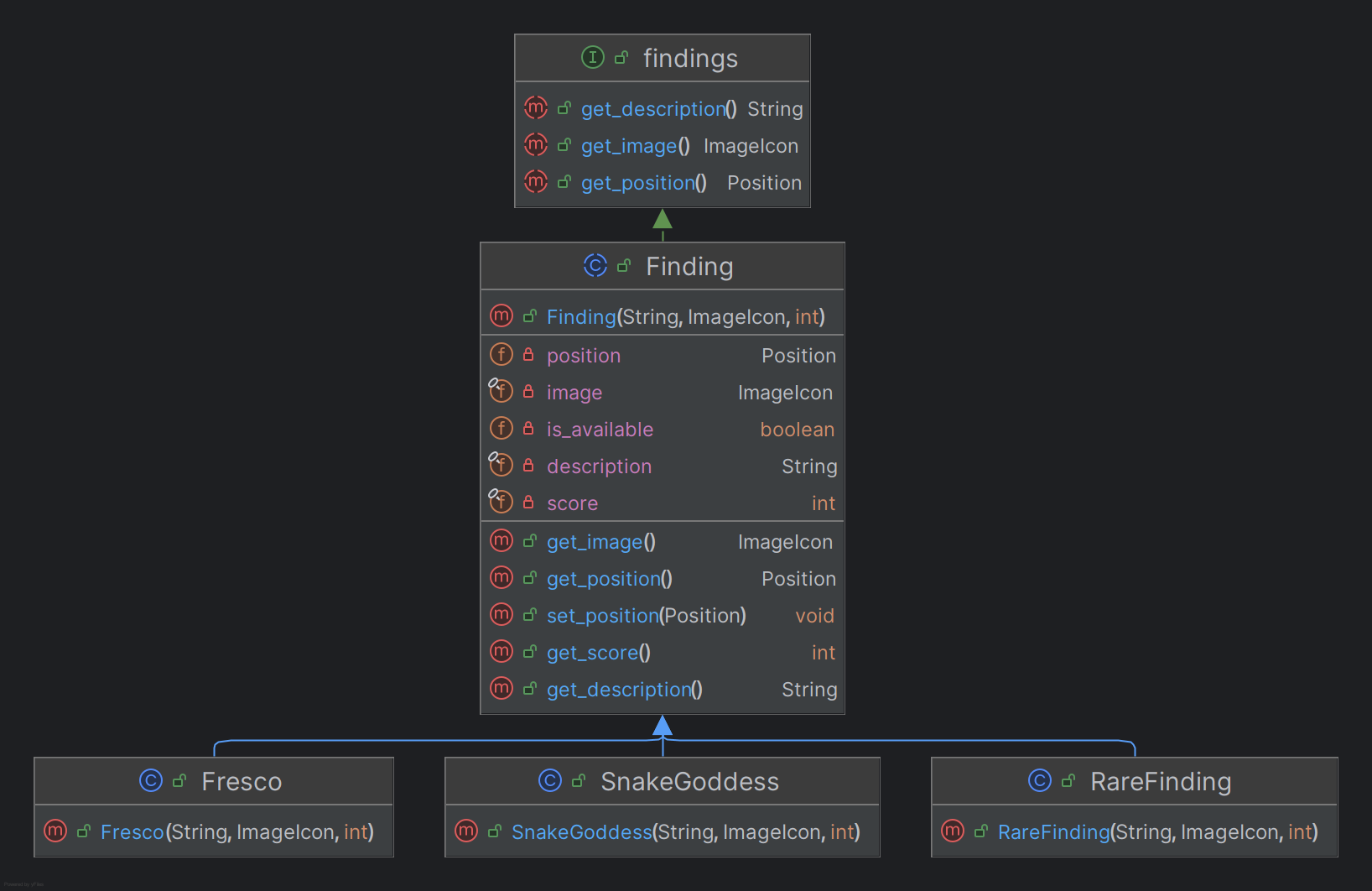
class Ariadne extends Ariadne

Methods:

public Ariadne(String name, ImageIcon image);

Θέτει τις τιμές από τις παραμέτρους που δέχεται

**Package Findings**

****

Interface findings

String get\_description();

Επιστρέφει την περιγραφή του αρχαιολογικού ευρήματος  
  
ImageIcon get\_image();

Επιστρέφει την εικόνα του αρχαιολογικού ευρήματος  
  
Position get\_position();

Επιστρέφει την θέση του αρχαιολογικού ευρήματος

class Finding (abstract) implements findings

Δεδομένα:

private final String description;

Η περιγραφή του αρχαιολογικού ευρήματος

private final ImageIcon image;

Η εικόνα του αρχαιολογικού ευρήματος

private Position position;

Η θέση του αρχαιολογικού ευρήματος

private final int score;

To score του ευρήματος

Methods:

public void set\_position(Position position)

Θέτει την θέση του ευρήματος  
public String get\_description();

public ImageIcon get\_image();

public int get\_score();

Επιστρέφει το score του ευρήματος

public Finding(String description, ImageIcon image, int score);

Θέτει τις τιμές από τις παραμέτρους που δέχεται

class RareFinding extends Findings

Methods:

public RareFinding(String description, ImageIcon image, int score);

Θέτει τις τιμές από τις παραμέτρους που δέχεται

class Fresco extends Findings

Methods:

public Fresco(String description, ImageIcon image, int score);

Θέτει τις τιμές από τις παραμέτρους που δέχεται

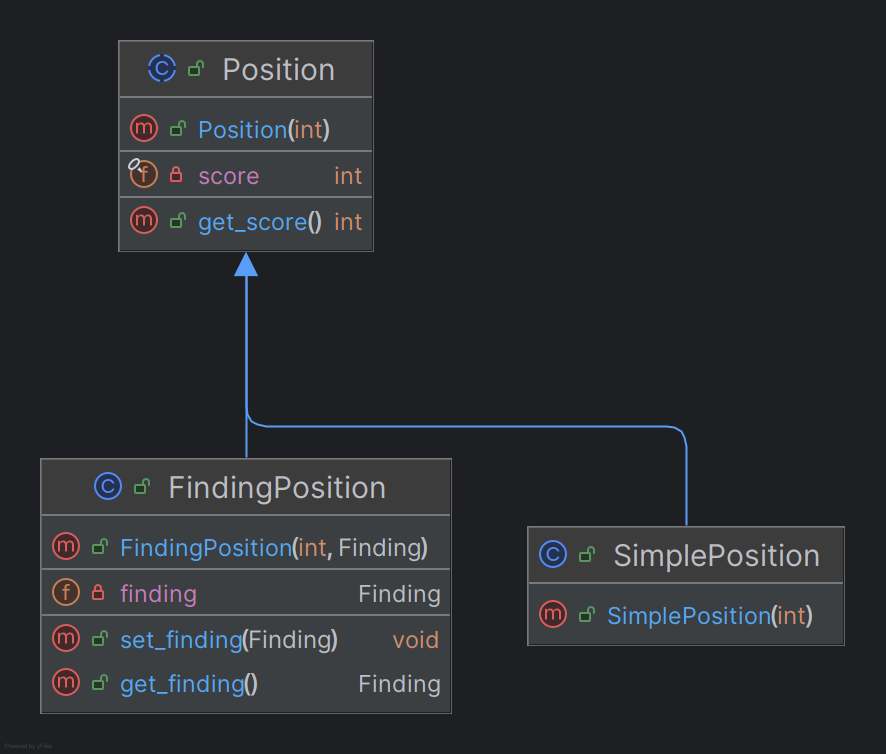
class SnakeGoddess extends Finding

Methods:

public SnakeGoddess(String description, ImageIcon image, int score);

Θέτει τις τιμές από τις παραμέτρους που δέχεται

**Package Position**

****

class Position

Δεδομένα:

private final int score;

Το score της θέσης

## Methods:

public int get\_score();

Επιστρέφει το score της θέσης

public Position(int score);

Θέτει τις τιμές από τις παραμέτρους που δέχεται

class FindingPosition extends Position

Δεδομένα:

private Finding finding;

Το εύρημα της θέσης

## Methods:

public void set\_finding(Finding finding);

Θέτει το εύρημα της θέσης

public int get\_finding();

Επιστρέφει το εύρημα της θέσης

public FindingPosition(int score, Finding finding);

Θέτει τις τιμές από τις παραμέτρους που δέχεται

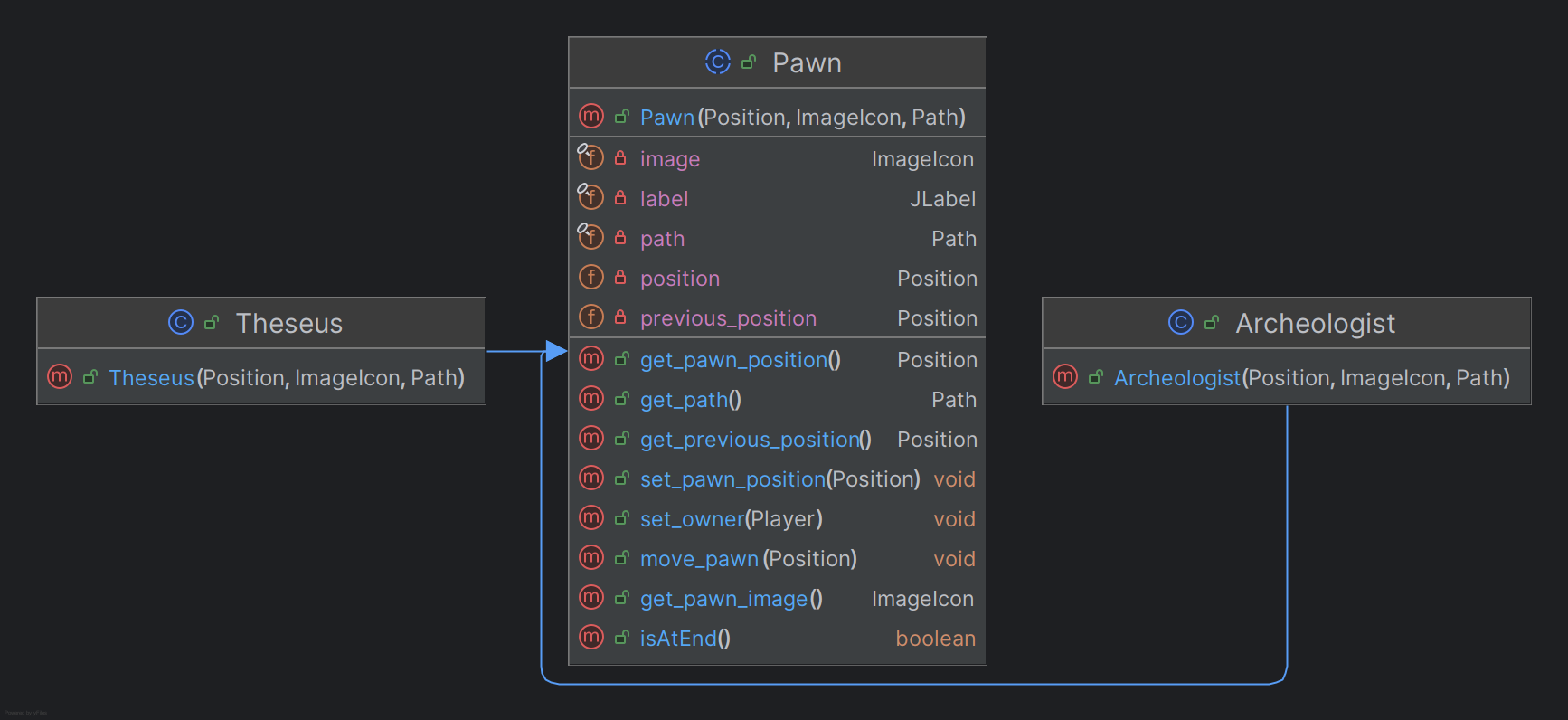
class SimplePosition extends Position

## Methods:

public SimplePosition(int score);

Θέτει τις τιμές από τις παραμέτρους που δέχεται

**Package Pawns**



**class Pawn (abstract)**

Δεδομένα**:**

private Position position;

Η θέση του πιονιού

private final ImageIcon image;

Η εικόνα του πιονιού

private final JLabel label;

Το label του πιονιού

private final Path path;

Το μονοπάτι του πιονιού

private Position previous\_position;

Η προηγούμενη τοποθεσία του πιονιού

Methods:

public void move\_pawn(Position position);

Θέτει την θέση του πιονιού

public void set\_owner(Player owner);

Θέτει τον κάτοχο του πιονιού

public void set\_pawn\_position(Position position);

Θέτει την θέση του πιονιού

public Position get\_pawn\_position();

Επιστρέφει την θέση του πιονιού

public ImageIcon get\_pawn\_image();

Επιστρέφει την εικόνα του πιονιού

public Path get\_path();

Επιστρέφει το μονοπάτι του πιονιού

public Position get\_previous\_position();

Επιστρέφει την προηγούμενη θέση του πιονιού

public boolean isAtEnd();

Επιστρέφει true ή false, ανάλογα με το άμα ένα πιόνι έφτασε στο τέλος του μονοπατιού

public Pawn(Position position, ImageIcon image, Path path);

Θέτει τις τιμές από τις παραμέτρους που δέχεται

class Archeologist extends Pawn

Methods:

public Archeologist(Position position, ImageIcon image, Path path);

Θέτει τις τιμές από τις παραμέτρους που δέχεται

class Theseus extends Pawn

Methods:

public Theseus(Position position, ImageIcon image, Path path);

Θέτει τις τιμές από τις παραμέτρους που δέχεται

**Package Utilities**

class Image\_reader

Methods

public static ImageIcon get\_Image(String name, String folder)

Επιστρέφει την εικόνα από το επιλεγμένο path

public static ImageIcon get\_Image(String name)

Επιστρέφει την εικόνα

class Deck

Δεδομένα:

private final ArrayList<Card> deck = new ArrayList<>(100);

Ένας πίνακας από 100 κάρτες

private ArrayList<Card> cards;

Ένας πίνακας από κάρτες

Methods:

public void initialize\_deck();

Αρχικοποιεί την τράπουλα

public void set\_cards(ArrayList<Card> cards);

Θέτει την κάρτα

public Card draw\_card();

Αφαιρεί μία κάρτα

public ArrayList<Card> get\_deck();

Επιστρέφει την τράπουλαpublic int size();

Επιστρέφει το μέγεθος την τράπουλας

public Deck();

Αρχικοποιεί την τράπουλα και την ανακατεύει

class Path

Δεδομένα:

private static Position[] *positions*;

Ένας πίνακας θέσεων

private final String name;  
 To όνομα του μονοπατιού

private final Map<Integer, JLabel> labels = new HashMap<>();

Ένα hashmap με integer, labels στοιχεία

Methods:

public Position get\_position(int index);

Έπιστρέφει έναν πίνακα θέσεων

public void set\_position(Position position, int index);

Θέτει την θέση του μονοπατιού

public void set\_label(int index, Jlabel label);

Θέτει το label του μονοπατιού

public int get\_index\_of\_position(Position position);

Επιστρέφει την θέση του μονοπατιού

public JLabel get\_label(int index);

Επιστρέφει το label του πιονιού

public int get\_length();

Επιστρέφει το μέγεθος του μονοπατιού

public String get\_name();

Επιστρέφει το όνομα του μονοπατιού

public ImageIcon get\_original\_icon(int index);

Επιστρέφει την εικόνα του μονοπατιού

public Path((String name, int position\_length);

Θέτει τις τιμές από τις παραμέτρους που δέχεται

class Player

Δεδομένα:

private final String name;

Το όνομα του παίκτη

private ArrayList<Pawn> pawns = new ArrayList<Pawn>(4);

Τα 4 πιόνια του παίκτη

private final ArrayList<Finding> findings\_list = new ArrayList<>();

Τα αρχαιολογικά ευρήματα που έχει στην κατοχή του

private ArrayList<Card> hand = new ArrayList<Card>(); Οι κάρτες που έχει στην κατοχή του

private int agalmatakia = 0;

Τα αγαλματάκια του

private final int score;  
 To score του;

Methods:

public void set\_hand(ArrayList<Card> hand);

Θέτει τις κάρτες στον παίχτη

public void add\_findings(Finding findings);

Θέτει τα αρχαιολογικά ευρήματα στον παίχτη

public void add\_agalmatakia();

Προσθέτει στην συλλογή του αγαλματάκια

*\*

public void clear\_hand();

“Καθαρίζει” το χέρι του παίκτη

public ArrayList<Card> get\_hand();

Επιστρέφει το χέρι του παίκτηpublic Finding get\_findings\_list();

Επιστρέφει τα αρχαιολογικά ευρήματα

public ArrayList<Card> get\_cards();

Επιστρέφει της κάρτες του παίχτη

public int get\_score();

Επιστρέφει το score του παίχτη

public int get\_name();

Επιστρέφει το όνομα του παίχτη

public ArrayList<Pawn> get\_pawns();

Επιστρέφει τα πιόνια του παίκτη

public int get\_agalmatakia();

Επιστρέφει τα αγαλματάκια του παίκτη

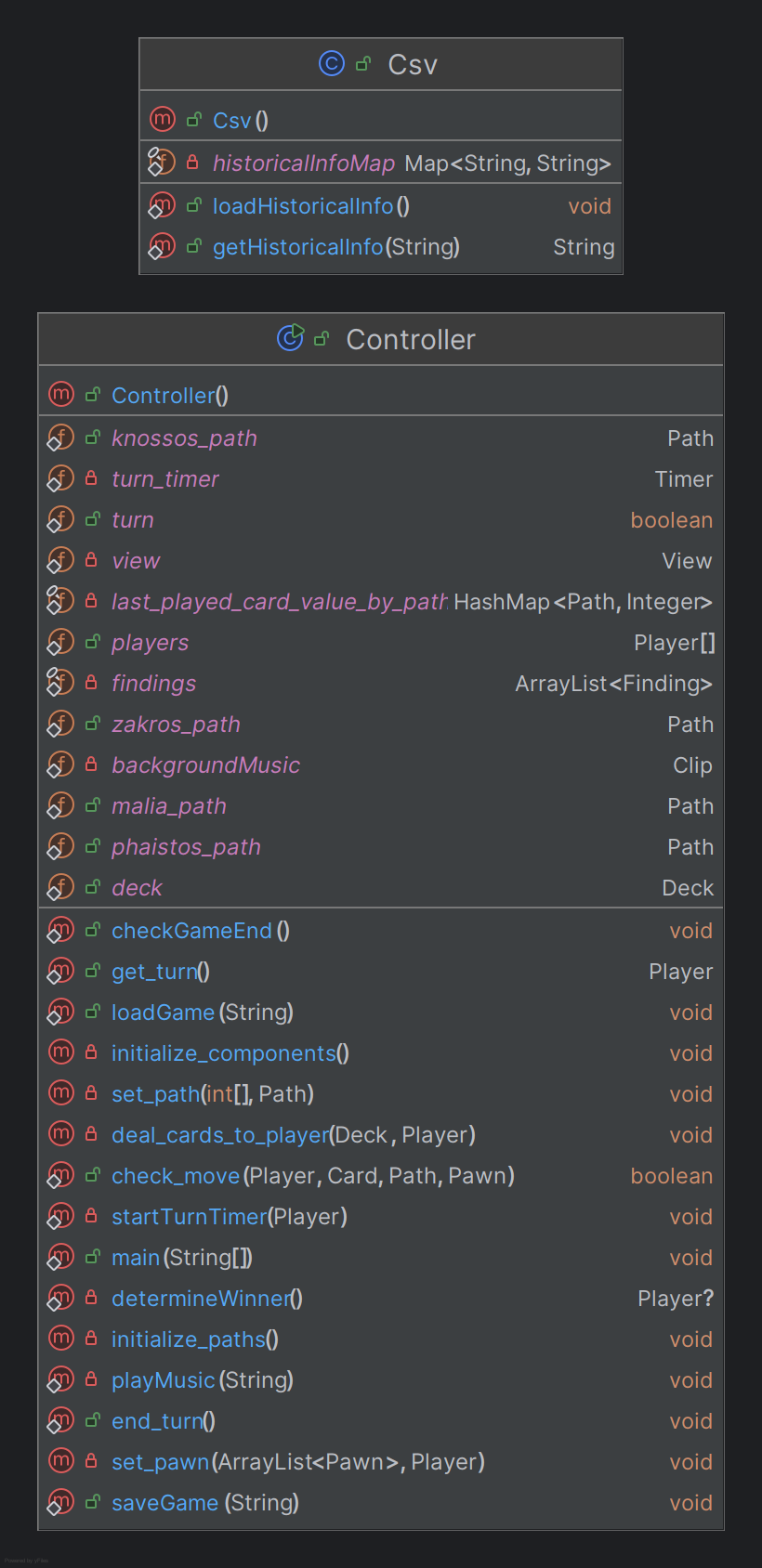
public Player(String name);

Θέτει τις τιμές από τις παραμέτρους που δέχεται

## Η Σχεδίαση και οι Κλάσεις του Πακέτου Controller

O controller είναι ο “εγκέφαλος” του παιχνιδιού και συνδέει to model με το view. Επίσης επικοινωνεί με το view για να παρουσιάσει τις ενημερώσεις από το model στον χρήστη.

**Package Controller**



class Controller

Δεδομένα:

private final Player[] players = new Player[2];

Οι 2 παίχτες

private static boolean turn;

Καθορίζει ποιανού είναι η σειρά

private static final ArrayList<Finding> *findings* = new ArrayList<>(20);

Tα αρχαιολογικά ευρήματα  
  
public static Deck *deck*;

Η τράπουλα

private static View *view*;

Το view

public static Path *knossos\_path*;

Το μονοπάτι knossos

public static Path *malia\_path*;

Το μονοπάτι malia

public static Path *phaistos\_path*;

Το μονοπάτι phaistos

public static Path *zakros\_path*;

Το μονοπάτι zakros

private static final HashMap<Path, Integer> *last\_played\_card\_value\_by\_path* = new HashMap<>();

Ένα hashmap με Path, Integer στοιχεία που αποθηκεύει την τιμή της τελευταίας κάρτας του μονοπατιού

private static Timer *turn\_timer*;

Το timer

private static Clip *backgroundMusic*;

Η μουσική

Methods:

private void initialize\_components();

Αρχικοποιεί τα components (pawns, players, findings, etc)

private void set\_pawn(ArrayList<Pawn> pawns, Player owner);

Αρχικοποιεί τα πιόνια στον παίχτη

private void initialize\_paths();

Αρχικοποιεί τα μονοπάτια

private void set\_path(int[] points\_path, Path path);

Τα θέτει πόντους

private void deal\_cards\_to\_player(Deck deck, Player player);

Μοιράζει κάρτες στους παίχτες

public static boolean check\_move(Player player, Card card, Path path, Pawn pawn);

Ελέγχει αν η κίνηση είναι legal

private static void startTurnTimer(Player currentPlayer);

Αρχίζει το timer

public static void saveGame(String fileName);

Αποθηκεύει το παιχνίδι

public static void loadGame(String fileName);

Φορτώνει το παιχνίδι

private void end\_turn();

Τελειώνει την σειρά του ενός παίχτη και την δίνει στον άλλον

private Player get\_turn();

Επιστρέφει ποιανού παίχτη σειρά είναι

private static void playMusic(String musicFilePath);

Παίζει μουσική ανάλογα ποιανού σειρά είναι

public static void checkGameEnd();

Ελέγχει αν υπάρχει νικητής

private static Player determineWinner();

Καθορίζει τον νικητή

public Controller();

Ο εγκέφαλος του παιχνιδιού, που είναι υπεύθυνος για την σωστή λειτουργία του

public static void main(String[] args);

Δημιουργεί έναν καινούργιο controller

class Csv

Δεδομένα:

private static final Map<String, String> *historicalInfoMap* = new HashMap<>();

Ένα hashmap με στοιχεία String, String για ιστορικές πληροφορίες

Methods:

public static void loadHistoricalInfo();

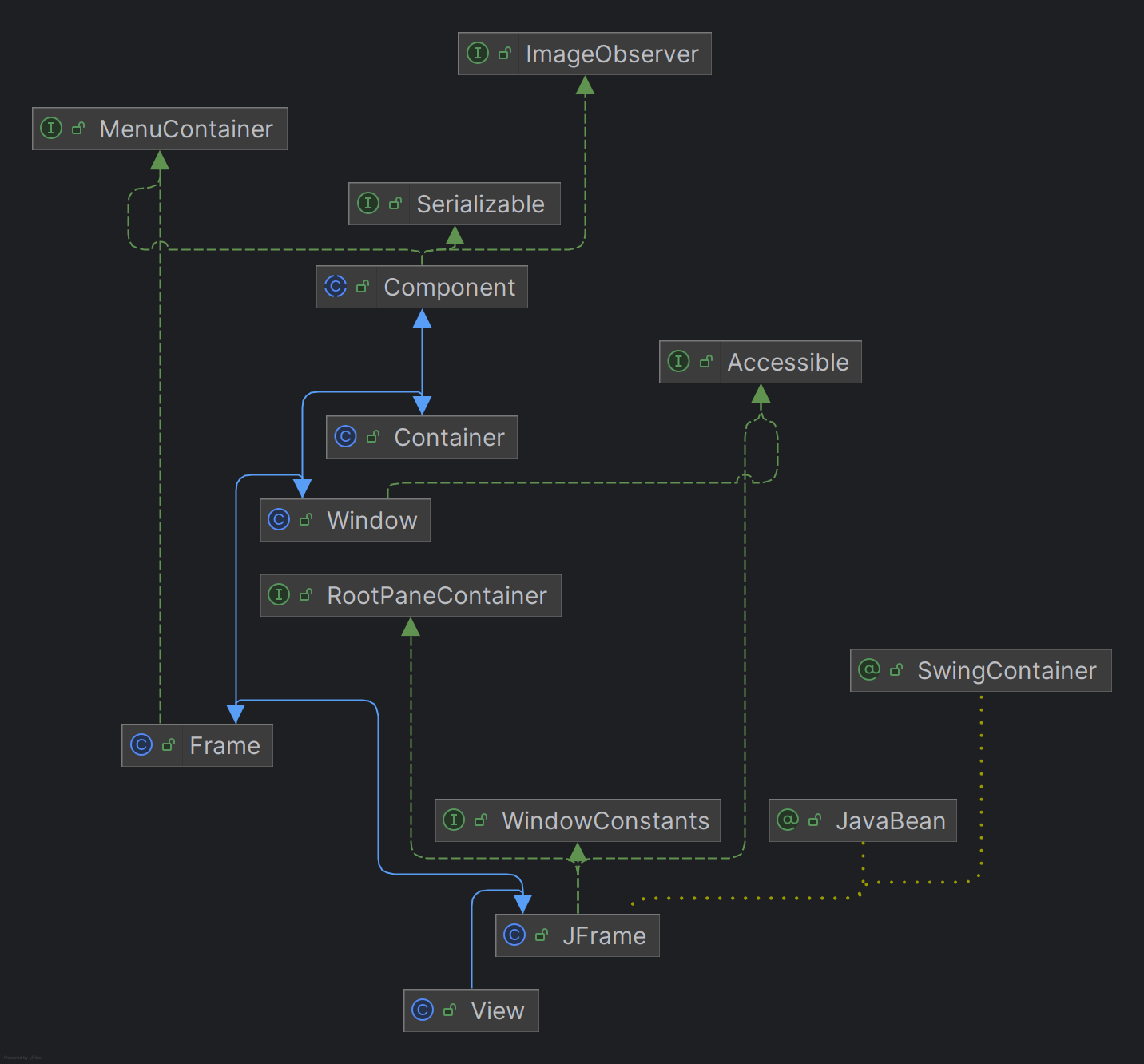
Κάνει fetch τις ιστορικές πληροφορίες

public static String getHistoricalInfo(String findingName

Επιστρέφειτις ιστορικές πληροφορίες

## Η Σχεδίαση και οι Κλάσεις του Πακέτου View

Η view χειρίζεται την παρουσίαση δεδομένων και την αλληλεπίδραση του χρήστη. Η View λαμβάνει δεδομένα από το **Model** (μέσω του **Controller**) και τα εμφανίζει με γραφικά.

**Pac****kage View**

class View extends Jframe

Δεδομένα:

private final Player[] players = Controller.*players*;

Οι 2 παίκτες

private JFrame frame;

Το βασικό frame

public static JLayeredPane *player\_pane1* = new JlayeredPane();

Η “μεριά” του 1ου παίκτη  
public static JLayeredPane *player\_pane2* = new JlayeredPane();

Η “μεριά” του 1ου παίκτη

private JTextArea card\_info;

Το κουτί με τις πληροφορίες

static JLayeredPane *board* = new JlayeredPane();

Το κεντρικό board

private JLabel player1\_findings;

Tα ευρήματα του 1ου παίκτη

private JLabel player2\_findings;

Tα ευρήματα του 1ου παίκτη

private JLabel player1\_score;

Το score του 1ου παίκτη

private JLabel player2\_score;

Το score του 2ου παίκτη

Methods:

public void initialize\_components();

Αρχικοποιεί τα components (JButtons, JLabels, etc)

private void add\_background();

Προσθέτει το υπόβαθρο του παιχνιδιού

private void create\_menu\_bar();

Προσθέτει την μπάρα μενού

private void add\_player\_area();

Προσθέτει την περιοχή των παικτών

private void add\_labels\_to\_player\_pane(JLayeredPane player\_pane);

Προσθέτει τα labels των παικτών

private void add\_board\_area();

Προσθέτει το board

private void add\_deck\_area();

Προσθέτει την τράπουλα

private static void add\_paths\_to\_board(JLayeredPane board, JLabel[] path\_points);

Προσθέτει τα μονοπάτια

private void add\_show\_findings\_button(JLayeredPane board);

Προσθέτει τα κουμπιά των ευρημάτων

private void add\_score\_info();

Προσθέτει το score

private void add\_image\_button();

Προσθέτει εικόνες-κουμπιά

private void add\_end\_turn\_button(JLayeredPane board);

Προσθέτει κουμπί για να αλλάξει η σειρά

private void add\_excavate\_button(JLayeredPane board);

Προσθέτει κουμπί για την εξκαφή

private void add\_cards\_to\_player\_pane(JLayeredPane player\_pane, int player\_index);

Προσθέτει τις κάρτες στους παίκτες

private void display\_findings();

Εμφανίζει τα ευρήματα

private void display\_player1\_frescos();

Εμφανίζει τις τοιχογραφίες του 1ου παίκτη

private void display\_player2\_frescos();

Εμφανίζει τις τοιχογραφίες του 2ου παίκτη

private void display\_frescos\_for\_player(Player player);

Εμφανίζει τις τοιχογραφίες του κάθε παίκτη

private void handle\_card\_discard(Player player, Card card);

Χειρίζεται την διαγραφή των καρτών

private void handle\_draw\_card(Player player);

Χειρίζεται την παραλαβή των καρτών

private void handle\_excavation();

Χειρίζεται την εξκαφή

private void refresh\_player\_findings(Player player);

Ενημερώνει τα ευρήματα των παικτών

private void refresh\_player\_score(Player player);

Ενημερώνει το score των παικτών

private void refresh\_agalmatakia\_count(Player player)

Ενημερώνει τα αγαλματάκια των παικτών

private void handle\_card\_click(Player player, Card card)

Χειρίζεται τις κάρτες

private void refresh\_player\_hand(Player player);

Ενημερώνει το χέρι

public static JLayeredPane get\_player\_pane(Player player)

Επιστρέφει την περιοχή του παίκτη που έχει σειρά

public static void update\_player\_position\_in\_UI(Pawn pawn)

Ενημερώνει τα πιόνια

public void move\_pawn(Pawn pawn, Position newPosition);

Μετακινεί τα πιόνια

public static void clear\_opponent\_pawns(Player currentPlayer);

Αφαιρεί τα πιόνια

public static void clear\_previous\_position(Pawn pawn)

Αφαιρεί την προηγούμενη θέση

public static void force\_board\_refresh()

Ενημερώνει το board

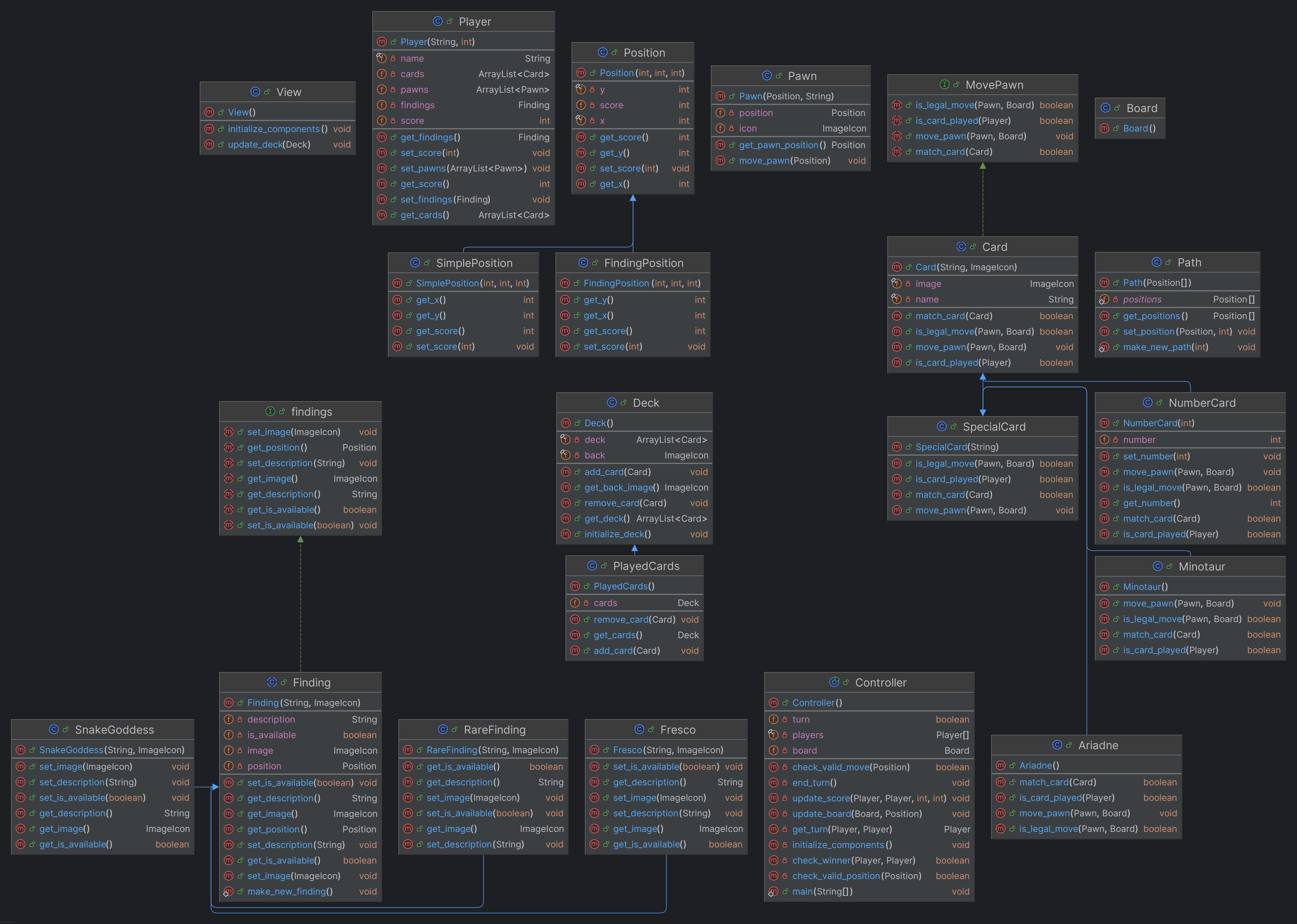
public void refreshPlayerAreas()

Ενημερώνει τις περιοχές των παικτών

public View();

Δημιουργεί ένα παράθυρο με το απαραίτητο UI

## Η Αλληλεπίδραση μεταξύ των κλάσεων – Διαγράμματα UML



## Συμπεράσματα

Δυσκολεύτηκα λιγο στο γραφικό κομμάτι και γιαυτο μερικές φορες δεν κανει clear την παλιά θέση του πιονιού, αλλά μετακινείται.