**Πανεπιστήμιο Κρήτης – Τμήμα Επιστήμης Υπολογιστών**

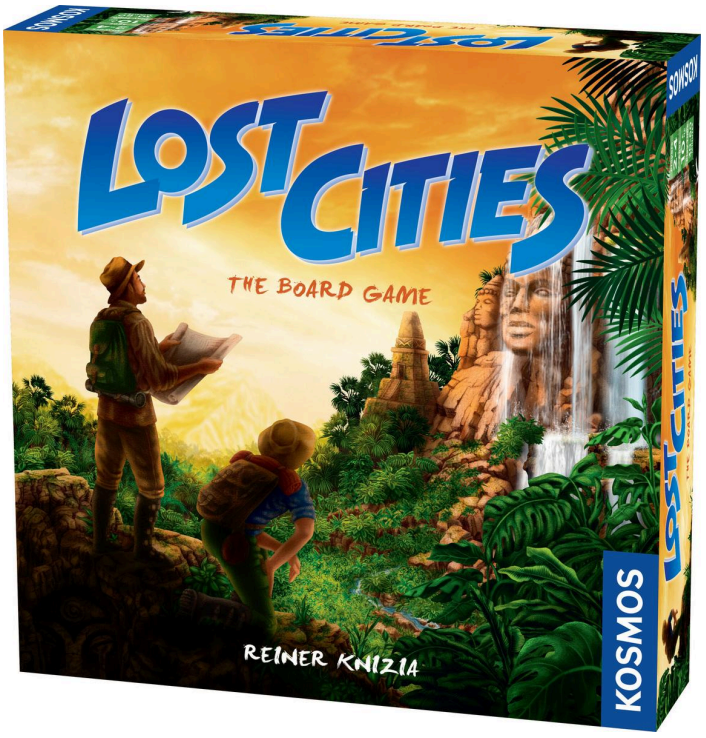
**Αντικειμενοστραφής Προγραμματισμός**

**Διδάσκων: Ι. Τζίτζικας**

**Χειμερινό Εξάμηνο 2024-2025**

# Β’ ΦΑΣΗ PROJECT

**Υλοποίηση επιτραπέζιου παιχνιδιού “Lost Cities”**




***Παπαδάκης Γεώργιος 4975***
***Ιανουάριος 2025***

# *Περιεχόμενα*



## *1. Εισαγωγή*

Σε αυτό το project υλοποιούμε το επιτραπέζιο παιχνίδι "Lost Cities" το οποίο στα πλαίσια του μαθήματος έχει θέμα τον Μινωικό Πολιτισμό.

Για τη δημιουργία του παιχνιδιού χρησιμοποιούμε το προγραμματιστικό μοντέλο MVC (Model View Controller), βάσει του οποίου η υλοποίηση χωρίζεται σε 3 ενότητες:

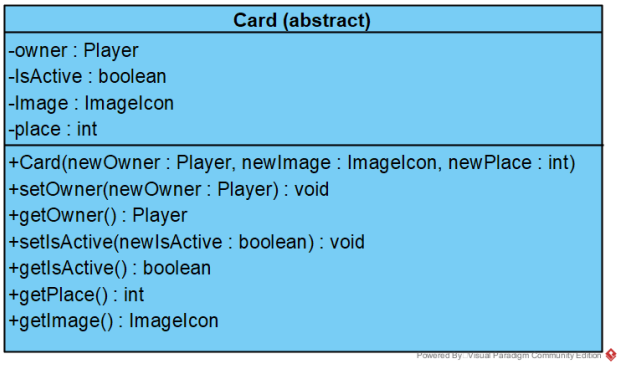
1. Model: Η υλοποίηση της λειτουργικότητας του παιχνιδιού. Δηλαδή το πως λειτουργούν οι κάρτες, τα πιόνια, αλλαγές στο ταμπλό, κ.α.
2. View: Η υλοποίηση της εμφάνισης στον χρήστη. Δηλαδή τα UI και GUI.
3. Controller: Ο κώδικας που διαχειρίζεται αποτελεσματικά τα κομμάτια των Model και VIew, καθώς και την μεταξύ τους επικοινωνία, ώστε να παίζουμε το παιχνίδι κανονικά.

Σε αυτή την αναφορά θα παρουσιαστεί ο τρόπος υλοποίησης κάθε ενός από τα κομμάτια του μοντέλου MVC, μαζί με διαγράμματα UML και κομμάτια κώδικα, συνοδευμένα από βοηθητικά σχόλια τύπου Javadoc.

# *2. Η Σχεδίαση και οι Κλάσεις του Πακέτου Model*

Το model θα πρέπει να περιλαμβάνει όλα τα περιεχόμενα του παιχνιδιού, δηλαδή θα πρέπει να υπάρχουν κλάσεις για τις κάρτες, τις θέσεις του ταμπλό του παιχνιδιού, τον παίκτη, τα πιόνια του παίκτη, το ταμπλό κλπ. Ας δούμε την κάθε κλάση αναλυτικά:

- **Card**




Η αφηρημένη κλάση Card αποτελεί τη βάση για κάθε κάρτα του παιχνιδιού.

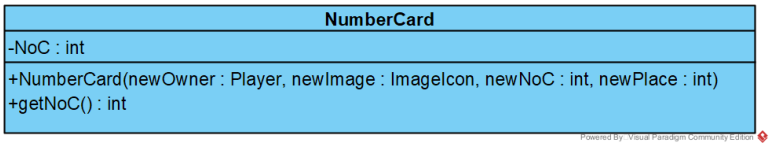
Δεδομένα:

- *"private Player Owner"* Ο ιδιοκτήτης της κάρτας. Μπορεί να είναι τύπου Player ή null αν δεν έχει χρησιμοποιηθεί ακόμα.
- *"private boolean IsActive"* Αν η κάρτα έχει παιχτεί στο παρελθόν τότε η isActive είναι FALSE, αλλιώς είναι TRUE.
- *"private ImageIcon Image"* Η εικόνα που αναπαριστά την εκάστοτε κάρτα.
- *"private int place"* Το μονοπάτι στο οποίο ανήκει αυτή η κάρτα. (Κνωσός=1, Μάλια=2, Φαιστός=3, Ζάκρος=4)

Μέθοδοι:

- *"public Card(Player newOwner, ImageIcon newImage, int newPlace)"* Ο constructor της κλάσης.
- *"public void setOwner(Player newOwner)"* Θέτει καινούργιο ιδιοκτήτη για τη κάρτα.
- *"public Player getOwner()"* Επιστρέφει τον ιδιοκτήτη της κάρτας.
- *"public void setIsActive(boolean newIsActive)"* Αλλάζει την διαθεσιμότητα της κάρτας. Αν έχει παιχτεί τότε η IsActive γίνεται FALSE.
- *"public boolean getIsActive()"* Επιστρέφει boolean σχετικά με το αν η κάρτα είναι ενεργή ή όχι.
- "*public int getPlace()*" Επιστρέφει το μονοπάτι στο οποίο ανήκει η κάρτα.
- "*public ImageIcon getImage()*" Επιστρέφει την εικόνα της κάρτας.

❖ **NumberCard**




Η κλάση NumberCard αφορά τις απλές, αριθμημένες κάρτες του παιχνιδιού.

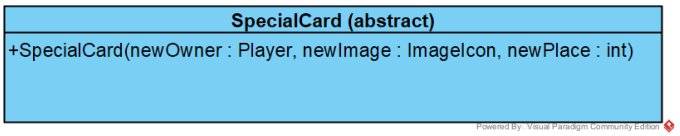
Δεδομένα:

- *"private int NoC"* Ο αριθμός της κάρτας.

Μέθοδοι:

- *"public NumberCard(Player newOwner, ImageIcon newImage, int newNoC, int newPlace)"* Ο constructor της κλάσης.
- *"public int getNoC()"* Επιστρέφει τον αριθμό της κάρτας.

❖ **SpecialCard**

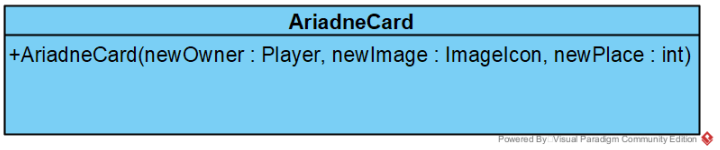



Η αφηρημένη κλάση SpecialCard αφορά τις κάρτες επιπλέον των αριθμημένων, δηλαδή τη κάρτα-Αριάδνη και τη κάρτα-Μινώταυρο.

Μέθοδοι:

- *"public SpecialCard(Player newOwner, ImageIcon newImage, int newPlace)"* Ο constructor της κλάσης.

❖ **AriadneCard**

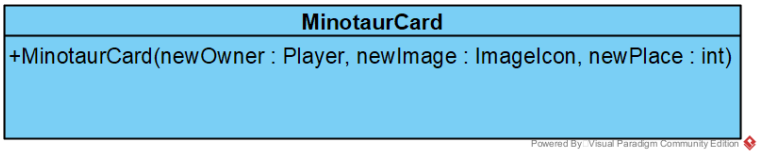



Η κλάση AriadneCard αφορά τη κάρτα-Αριάδνη.

Μέθοδοι:

- *"public AriadneCard(Player newOwner, ImageIcon newImage, int newPlace)"* Ο constructor της κλάσης.

❖ **MinotaurCard**

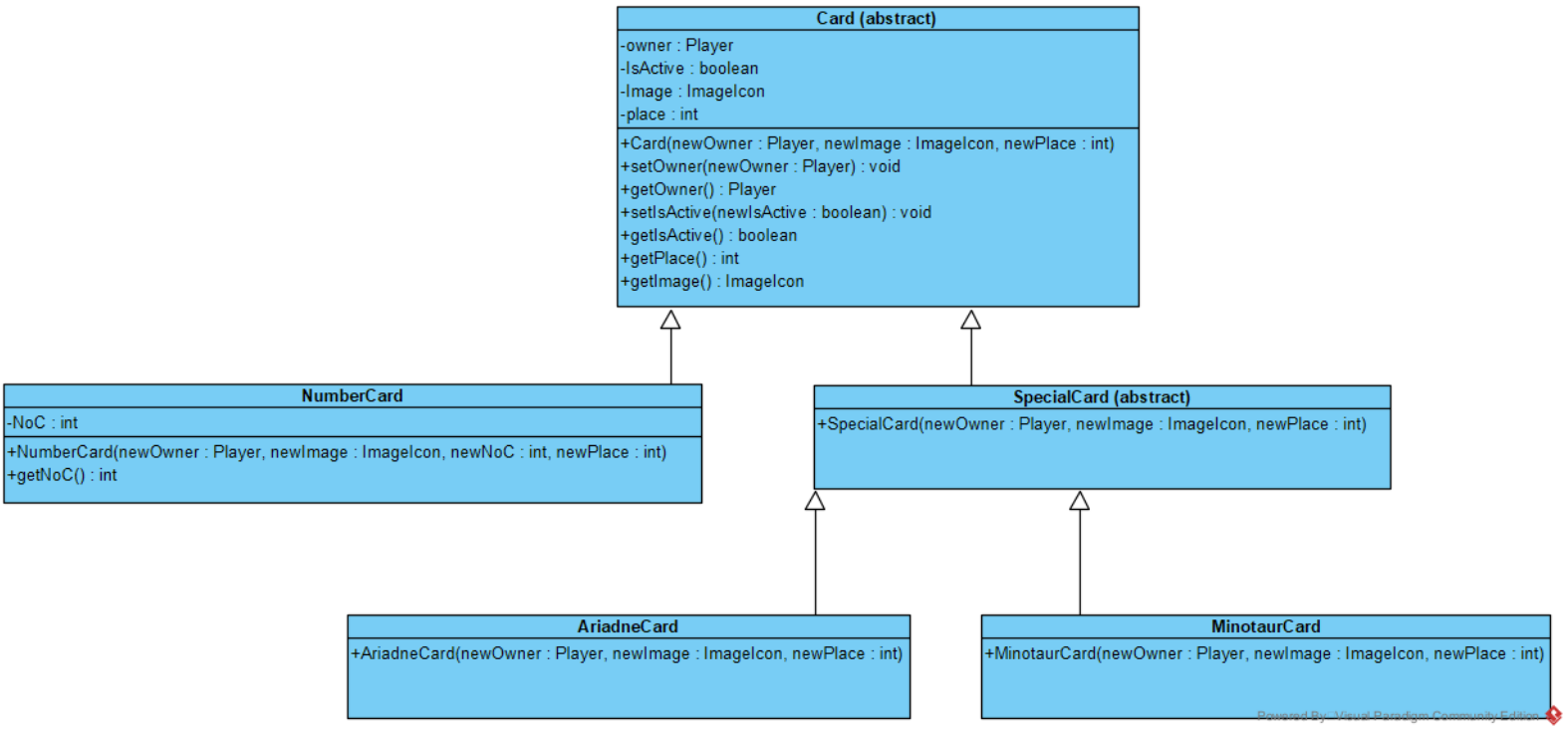


Η κλάση MinotaurCard αφορά τη κάρτα-Μινώταυρο.

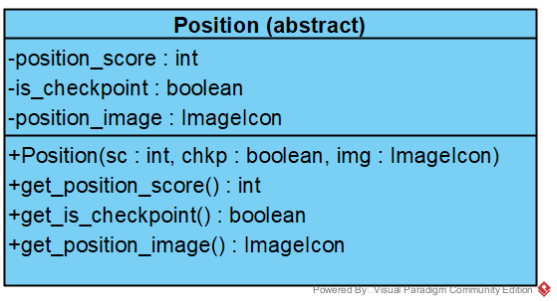
Μέθοδοι:

- *"public MinotaurCard(Player newOwner, ImageIcon newImage, int newPlace)"* Ο constructor της κλάσης.

**ΠΛΗΡΕΣ ΔΙΑΓΡΑΜΜΑ ΤΩΝ ΚΛΑΣΕΩΝ (ΜΕ ΤΗ ΚΛΗΡΟΝΟΜΙΚΟΤΗΤΑ ΤΟΥΣ)**




- **Position**




Η αφηρημένη κλάση Position αποτελεί τη βάση για κάθε θέση του παιχνιδιού.

Δεδομένα:

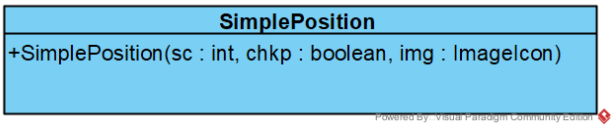
- *"private int position_score"* Το score της κάθε θέσης.

- *"private boolean is_checkpoint"* Boolean η οποία έχει τιμή TRUE εάν η θέση είναι στο checkpoint ή μετά από αυτό, και τη τιμή FALSE αν είναι πριν το checkpoint.
- *"private ImageIcon position_image"* Η εικόνα της θέσης.

Μέθοδοι:

- *"public Position(int sc, boolean chkp, ImageIcon img)"* Ο constructor της κλάσης.
- *"public int get_position_score()"* Επιστρέφει το score της θέσης.
- "*public boolean get_is_checkpoint()*" Επιστρέφει τη boolean τιμή για το εάν η θέση αυτή είναι checkpoint ή όχι.
- "*public ImageIcon get_position_image()*" Επιστρέφει την εικόνα της θέσης.

### ❖ SimplePosition

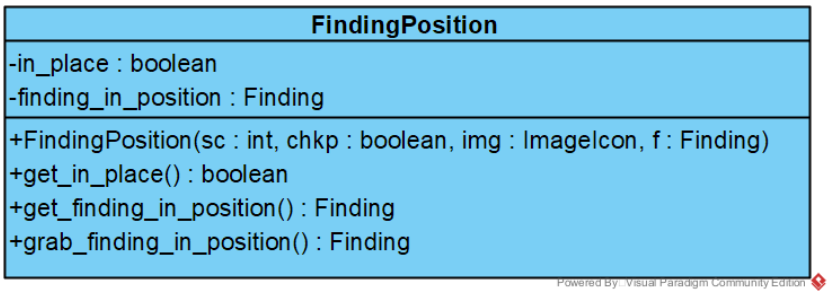



Η κλάση SimplePosition αφορά τις θέσεις οι οποίες **δε** κρύβουν κάποια αρχαιότητα.

Μέθοδοι:

- *"public SimplePosition(int sc, boolean chkp, ImageIcon img)"* Ο constructor της κλάσης.

### ❖ FindingPosition




Η κλάση FindingPosition αφορά τις θέσεις οι οποίες κρύβουν κάποια αρχαιότητα.

Δεδομένα:

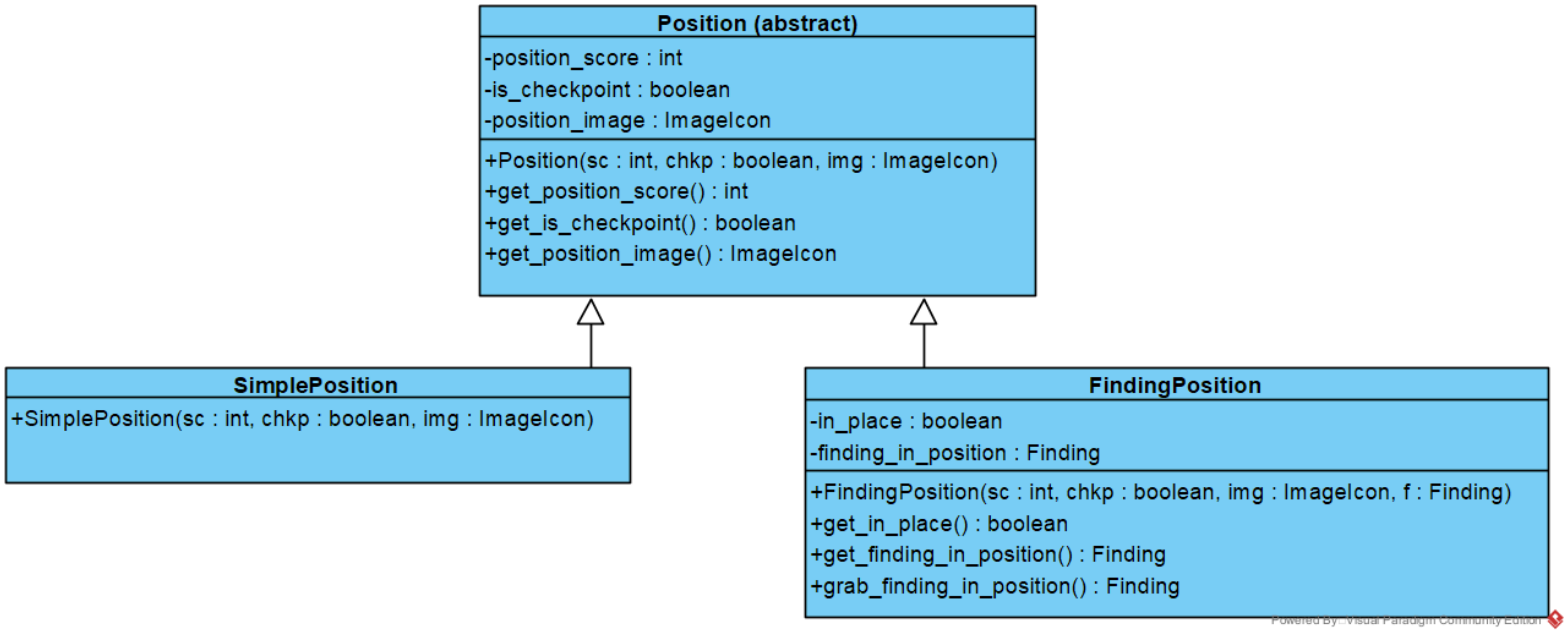
- *"private boolean in_place"* Boolean η οποία έχει τιμή TRUE αν η αρχαιότητα είναι ακόμα κρυμμένη σε αυτή τη θέση, και τιμή FALSE αν κάποιος παίκτης την έχει πάρει ή την έχει καταστρέψει.
- *"private Finding finding_in_position"* Η αρχαιότητα η οποία είναι κρυμμένη σε αυτή τη θέση.

Μέθοδοι:

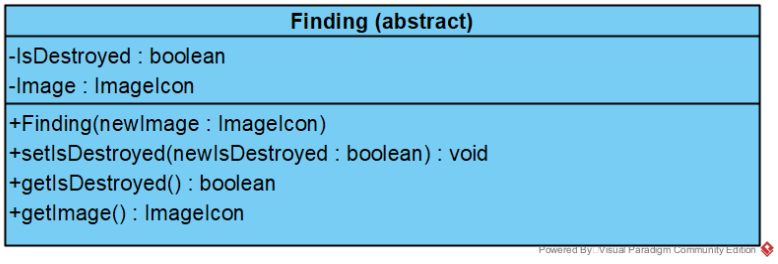
- *"public FindingPosition(int sc, boolean chkp, ImageIcon img, Finding f)"* Ο constructor της κλάσης.
- *"public boolean get_in_place()"* Επιστρέφει τη διαθεσιμότητα της αρχαιότητας αυτής της θέσης.
- *"public Finding get_finding_in_position()"* Επιστρέφει την αρχαιότητα που κρύβεται σε αυτή τη θέση(χωρίς να την αφαιρεί).

- *"public Finding grab_finding_in_position()"* Επιστρέφει την αρχαιότητα που κρύβεται σε αυτή τη θέση, βάζει στη μεταβλητή finding_in_position τιμή null και στη μεταβλητή in_place τιμή FALSE. (χρησιμοποιείται όταν κάποιος παίκτης μπορεί να πάρει την αρχαιότητα και δεν είναι τοιχογραφία)

**ΠΛΗΡΕΣ ΔΙΑΓΡΑΜΜΑ ΤΩΝ ΚΛΑΣΕΩΝ (ΜΕ ΤΗ ΚΛΗΡΟΝΟΜΙΚΟΤΗΤΑ ΤΟΥΣ)**




❖ **Finding**




Η αφηρημένη κλάση Finding αποτελεί τη βάση για κάθε αρχαιότητα του παιχνιδιού.

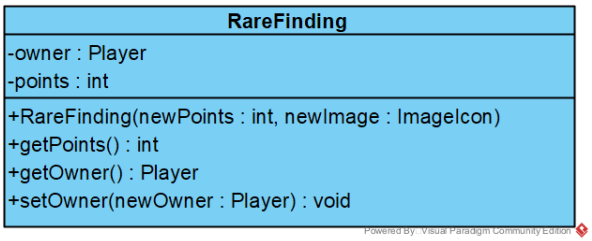
Δεδομένα:

- *"private boolean IsDestroyed"* Boolean η οποία έχει τιμή TRUE εάν η αρχαιότητα έχει καταστραφεί από τον Θησέα, και τη τιμή FALSE αν έκατσε ήρεμα σήμερα ο αθηναίος.
- *"private ImageIcon Image"* Η εικόνα της αρχαιότητας.

Μέθοδοι:

- *"public Finding(ImageIcon newImage)"* Ο constructor της κλάσης.
- *"public void setIsDestroyed(boolean newIsDestroyed)"* Θέτει καινούργια τιμή στη boolean IsDestroyed.
- *"public boolean getIsDestroyed()"* Επιστρέφει τη τιμή της boolean IsDestroyed.
- *"public ImageIcon getImage()"* Επιστρέφει την εικόνα της αρχαιότητας.

❖ **RareFinding**




Η κλάση RareFinding αφορά τις 4 πολύτιμες αρχαιότητες, καθεμία από τις οποίες κρύβεται και σε ένα από τα 4 μονοπάτια του παιχνιδιού.

Δεδομένα:

- *"private Player owner"* Ο ιδιοκτήτης αυτής της αρχαιότητας. Αν δεν έχει ανακαλυφθεί ακόμα έχει τιμή null.
- *"private int points"* Οι βαθμοί που κερδίζει όποιος ανακαλύψει αυτή την αρχαιότητα.

Μέθοδοι:

- *"public RareFinding(int newPoints, ImageIcon newImage)"* Ο constructor της κλάσης.
- *"public int getPoints()"* Επιστρέφει τον αριθμό των πόντων που κερδίζει όποιος παίκτης ανακαλύψει αυτή την αρχαιότητα.
- *"public Player getOwner()"* Επιστρέφει τον ιδιοκτήτη αυτής της αρχαιότητας.
- *"public void setOwner(Player newOwner)"* Θέτει ιδιοκτήτη για αυτή την αρχαιότητα.

❖ **SnakeGoddessFinding**


Η κλάση SnakeGoddessFinding αφορά τα 10 αγαλματάκια της Θεάς των Φιδιών τα οποία είναι κρυμμένα σε 10 θέσεις στα μονοπάτια του παιχνιδιού.

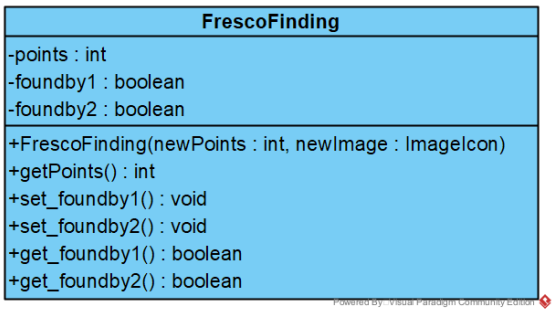
Δεδομένα:

- *"private Player owner"* Ο ιδιοκτήτης αυτής της αρχαιότητας. Αν δεν έχει ανακαλυφθεί ακόμα έχει τιμή null.

Μέθοδοι:

- *"public SnakeGoddessFinding(ImageIcon newImage)"* Ο constructor της κλάσης.
- *"public Player getOwner()"* Επιστρέφει τον ιδιοκτήτη αυτής της αρχαιότητας.
- *"public void setOwner(Player newOwner)"* Θέτει ιδιοκτήτη για αυτή την αρχαιότητα.

❖ **FrescoFinding**




Η κλάση FrescoFinding αφορά τις 6 τοιχογραφίες οι οποίες είναι κρυμμένες σε 6 θέσεις στα μονοπάτια του παιχνιδιού.

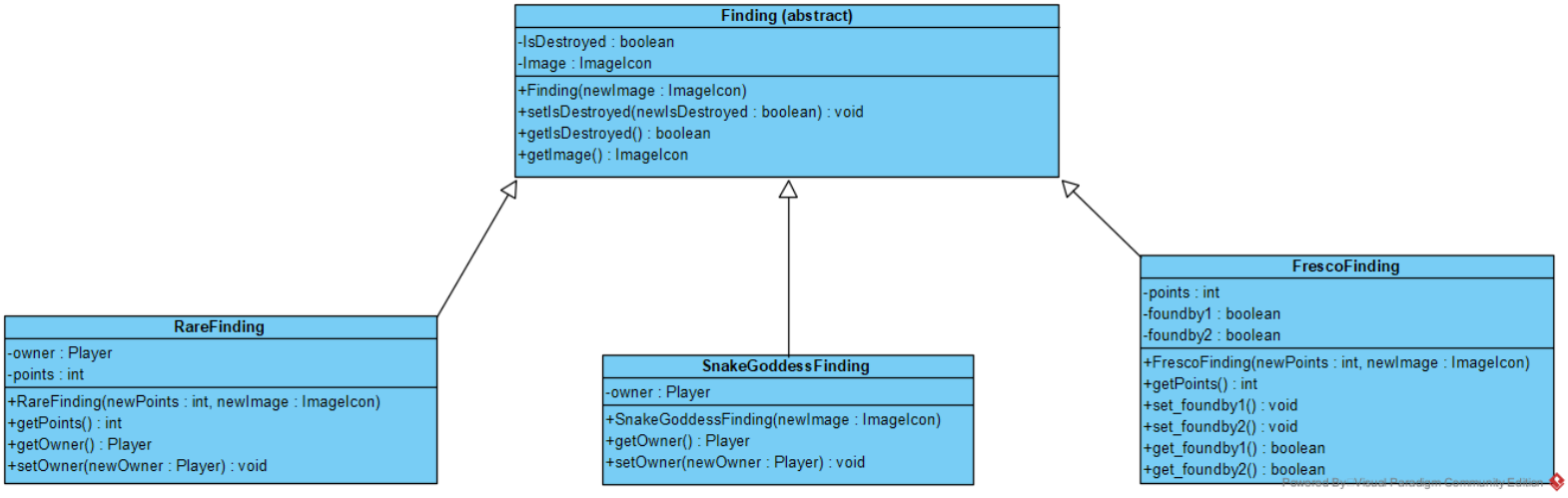
Δεδομένα:

- *"private int points"* Οι βαθμοί που κερδίζει όποιος ανακαλύψει αυτή την αρχαιότητα.
- *"private boolean foundby1"* Boolean μεταβλητή η οποία αποθηκεύει αν η τοιχογραφία έχει φωτογραφηθεί από τον Player1.
- *"private boolean foundby2"* Boolean μεταβλητή η οποία αποθηκεύει αν η τοιχογραφία έχει φωτογραφηθεί από τον Player2.

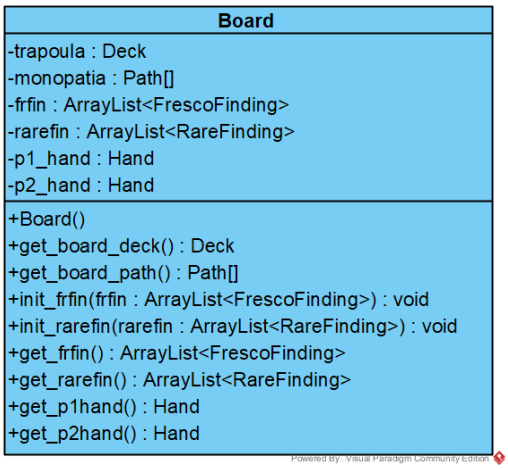
Μέθοδοι:

- *"public FrescoFinding(int newPoints, ImageIcon newImage)"* Ο constructor της κλάσης.
- *"public int getPoints()"* Επιστρέφει τον αριθμό των πόντων που κερδίζει όποιος παίκτης ανακαλύψει αυτή την αρχαιότητα.
- "*public void set_foundby1()"* Δίνει στην boolean μεταβλητή foundby1 τιμή TRUE.
- "*public void set_foundby2()"* Δίνει στην boolean μεταβλητή foundby2 τιμή TRUE.
- "*public boolean get_foundby1()"* Επιστρέφει τη τιμή της boolean μεταβλητής foundby1.
- "*public boolean get_foundby2()"* Επιστρέφει τη τιμή της boolean μεταβλητής foundby2.

**ΠΛΗΡΕΣ ΔΙΑΓΡΑΜΜΑ ΤΩΝ ΚΛΑΣΕΩΝ (ΜΕ ΤΗ ΚΛΗΡΟΝΟΜΙΚΟΤΗΤΑ ΤΟΥΣ)**



❖ **Board**




Η κλάση Board αφορά το ταμπλό του παιχνιδιού.

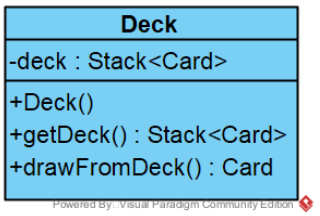
Δεδομένα:

- *"private Deck trapoula"* Η τράπουλα με τις αχρησιμοποίητες κάρτες του παιχνιδιού.
- *"private Path[] monopatia"* Πίνακας με τα 4 μονοπάτια του παιχνιδιού.
- *"private ArrayList<FrescoFinding> frfin = new ArrayList<>()"* Είναι το ArrayList με τις 6 τοιχογραφίες του παιχνιδιού.
- *"private ArrayList<RareFinding> rarefin = new ArrayList<>()"* Είναι το ArrayList με τα 4 σπάνια ευρήματα του παιχνιδιού.
- *"private Hand p1_hand"* Το χέρι με τις 8 κάρτες του Player1.
- *"private Hand p2_hand"* Το χέρι με τις 8 κάρτες του Player2.

Μέθοδοι:

- *"public Board()"* Ο constructor της κλάσης.
- *"public Deck get_board_deck()"* Επιστρέφει τη τράπουλα του παιχνιδιού.
- *"public Path[] get_board_path()"* Επιστρέφει τον πίνακα με τα 4 μονοπάτια του παιχνιδιού.
- "*public void init_frfin(ArrayList<FrescoFinding> frfin)"* Αρχικοποιεί το ArrayList με τις 6 τοιχογραφίες του παιχνιδιού.
- "*public void init_rarefin(ArrayList<RareFinding> rarefin)"* Αρχικοποιεί το ArrayList με τα 4 σπάνια ευρήματα του παιχνιδιού.
- "*public ArrayList<FrescoFinding> get_frfin()"* Επιστρέφει το ArrayList με τις 6 τοιχογραφίες του παιχνιδιού.
- "*public ArrayList<RareFinding> get_rarefin()"* Επιστρέφει το ArrayList με τα 4 σπάνια ευρήματα του παιχνιδιού.
- "*public Hand get_p1hand()"* Επιστρέφει το χέρι με τις 8 κάρτες του Player1.
- "*public Hand get_p2hand()"* Επιστρέφει το χέρι με τις 8 κάρτες του Player2.

❖ **Deck**




Η κλάση Deck αφορά τη τράπουλα με τις κάρτες οι οποίες **δεν** έχουν παιχτεί ακόμα.

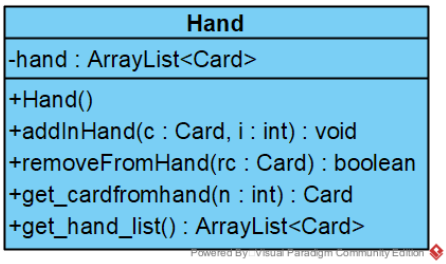
Δεδομένα:

- *"private Stack<Card> deck"* Στοίβα η οποία αποθηκεύει τα χαρτιά που **δεν** έχουν χρησιμοποιηθεί ακόμα.

Μέθοδοι:

- *"public Deck()"* Ο constructor της κλάσης.
- *"public Stack<Card> getDeck()"* Επιστρέφει τη στοίβα με τα χαρτιά.
- *"public Card drawFromDeck()"* Δίνει τη κάρτα στη κορυφή της τράπουλας. Κάνει pop από τη στοίβα και δίνει τη κάρτα στον παίκτη που κάλεσε αυτή τη μέθοδο.

❖ **Hand**




Η κλάση Hand αφορά τις 8 κάρτες που έχει στο χέρι του κάθε παίκτης.

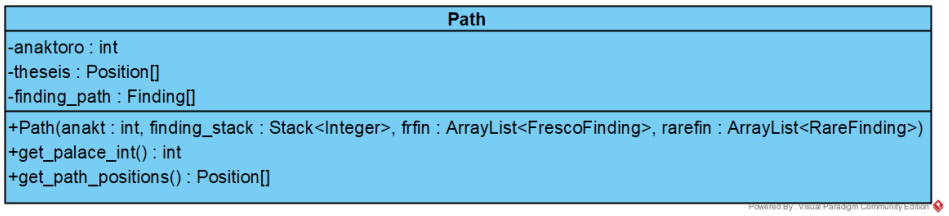
Δεδομένα:

- *"private ArrayList<Card> hand"* Λίστα με τα 8 χαρτιά που κρατάει κάθε παίκτης.

Μέθοδοι:

- *"public Hand()"* Ο constructor της κλάσης.
- *"public void addInHand(Card c, int i)"* Τοποθετεί τη κάρτα c στο χέρι του παίκτη, στην θέση i. Αν υπάρχει ήδη μια κάρτα εκεί την αντικαθιστά με τη c.
- *"public boolean removeFromHand(Card rc)"* Αφαιρεί τη κάρτα rc από το χέρι του παίκτη.
- "*public Card get_cardfromhand(int n)"* Επιστρέφει τη κάρτα στη θέση n του χεριού.
- "*public ArrayList<Card> get_hand_list()"* Επιστρέφει το ArrayList με τις κάρτες του χεριού.

❖ **Path**




Η κλάση Path αφορά κάθε ένα από τα 4 μονοπάτια του παιχνιδιού.

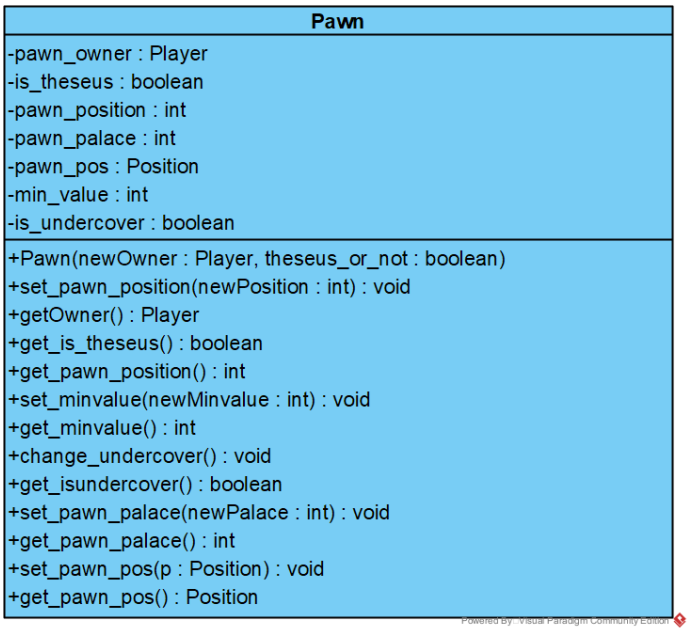
Δεδομένα:

- *"private int anaktoro"* Ακέραια μεταβλητή η οποία υποδεικνύει το ανάκτορο στο οποίο ανήκει αυτό το μονοπάτι. (1=Κνωσός, 2=Μάλια, 3=Φαιστός, 4=Ζάκρος)
- *"private Position[] theseis"* Πίνακας με τις 9 θέσεις κάθε μονοπατιού.
- *"private Finding[] finding_path"* Πίνακας με τα 5 ευρήματα κάθε μονοπατιού.

Μέθοδοι:

- *"public Path(int anakt, Stack<Integer> finding_stack, ArrayList<FrescoFinding> frfin, ArrayList<RareFinding> rarefin)"* Ο constructor της κλάσης.
- *"public int get_palace_int()"* Επιστρέφει την ακέραια μεταβλητή που υποδεικνύει το ανάκτορο στο οποίο ανήκει το μονοπάτι αυτό.
- *"public Position[] get_path_positions()"* Επιστρέφει τον πίνακα με τις θέσεις του μονοπατιού.

❖ **Pawn**




Η κλάση Pawn αφορά κάθε ένα από τα 8 πιόνια του παιχνιδιού.

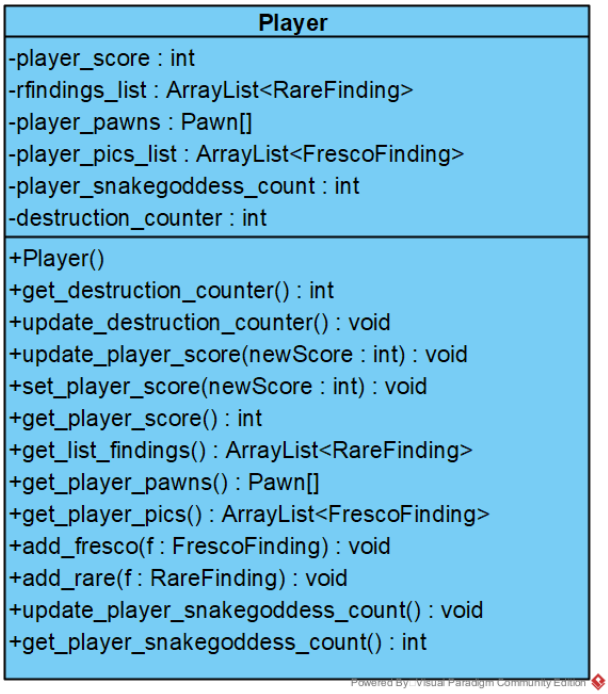
Δεδομένα:

- *"private Player pawn_owner"* Ο ιδιοκτήτης του πιονιού.
- *"private boolean is_theseus"* Boolean η οποία είναι TRUE αν είναι πιόνι-Θησέας και FALSE αν είναι αρχαιολόγος.
- *"private int pawn_position"* Η θέση στην οποία είναι τοποθετημένο το πιόνι ως ακέραιος δείκτης του πίνακα theseis στο Path.
- *"private int pawn_palace"* Ακέραιος που δείχνει σε ποιό ανάκτορο ανήκει κάθε πιόνι.
- *"private Position pawn_pos"* Η θέση στην οποία είναι τοποθετημένο το πιόνι ως Position object.
- *"private int min_value"* Η ελάχιστη τιμή που πρέπει να έχει μια κάρτα ώστε να μετακινήσει το πιόνι.
- *"private boolean is_undercover"* Boolean η οποία είναι TRUE αν το πιόνι έχει αποκαλύψει τη ταυτότητα του και FALSE αν δεν την έχει αποκαλύψει ακόμα.

Μέθοδοι:

- *"public Pawn(Player newOwner, boolean theseus_or_not)"* Ο constructor της κλάσης.
- *"public void set_pawn_position(int newPosition)"* Θέτει καινούργια τιμή στην ακέραια μεταβλητή pawn_position.
- *"public Player getOwner()"* Επιστρέφει τον ιδιοκτήτη του πιονιού.
- *"public boolean get_is_theseus()"* Επιστρέφει τη boolean που υποδεικνύει αν το πιόνι αυτό είναι Θησέας.
- *"public int get_pawn_position()"* Επιστρέφει την ακέραια μεταβλητή pawn_position.
- *"public void set_minvalue(int newMinvalue)"* Θέτει καινούργια ελάχιστη τιμή στο πιόνι.
- *"public int get_minvalue()"* Επιστρέφει την ελάχιστη τιμή του πιονιού.
- *"public void change_undercover()"* Αλλάζει την boolean is_undercover από false σε true αν το πιόνι αποκαλύπτει τη ταυτότητα του.
- *"public boolean get_isundercover()"* Επιστρέφει την boolean is_undercover.
- *"public void set_pawn_palace(int newPalace)"* Θέτει καινούργια τιμή στην ακέραια μεταβλητή pawn_palace.
- *"public int get_pawn_palace()"* Επιστρέφει την ακέραια μεταβλητή pawn_palace.
- *"public void set_pawn_pos(Position p)"* Θέτει καινούργια θέση για το πιόνι.
- *"public Position get_pawn_pos()"* Επιστρέφει τη θέση του πιονιού.

❖ **Player**




Η κλάση Player αφορά κάθε ένα από τους 2 παίκτες του παιχνιδιού.

Δεδομένα:

- *"private int player_score"* Το score του παίκτη.
- *"private ArrayList<RareFinding> rfindings_list"* Λίστα με τα σπάνια ευρήματα που έχει ανακαλύψει ο παίκτης.
- *"private Pawn[] player_pawns"* Πίνακας με τα 4 πιόνια του παίκτη.
- *"private ArrayList<FrescoFinding> player_pics_list"* Λίστα με τις 6 φωτογραφίες από τοιχογραφίες του παίκτη. Στο dialog "My Frescoes" φαίνονται ως μη γκριζαρισμένες όσες έχουν ως ιδιοκτήτη τον συγκεκριμένο Player.
- *"private int player_snakegoddess_count"* Μετρητής των αγαλματιδίων της Θεάς των Φιδιών που έχει ανακαλύψει ο παίκτης.
- *"private int destruction_counter"* Μετρητής των ευρημάτων που μπορεί να καταστρέψει το πιόνι-Θησέας του κάθε παίκτη. Αρχικοποιείται με τιμή 3 και κάθε φορά που καταστρέφει ένα εύρημα ο Θησέας ο μετρητής μειώνεται κατά ένα.

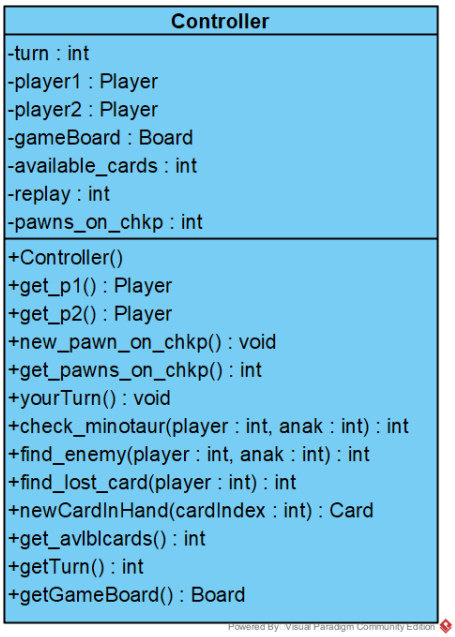
Μέθοδοι:

- *"public Player()"* Ο constructor της κλάσης.
- *"public int get_destruction_counter()"* Επιστρέφει τον μετρητή των ευρημάτων που μπορεί να καταστρέψει ο Θησέας.
- *"public void update_destruction_counter()"* Μειώνει τον μετρητή των ευρημάτων που μπορεί να καταστρέψει ο Θησέας κατά ένα εάν δεν είναι μηδέν.
- *"public void update_player_score(int newScore)"* Προσθέτει στο υπάρχον score του παίκτη τιμή newScore.
- *"public void set_player_score(int newScore)"* Θέτει ως τιμή του score του παίκτη το newScore.
- *"public int get_player_score()"* Επιστρέφει το score του παίκτη.
- *"public ArrayList<RareFinding> get_list_findings()"* Επιστρέφει τη λίστα με τα σπάνια ευρήματα που έχει ανακαλύψει ο παίκτης.

- *"public Pawn[] get_player_pawns()"* Επιστρέφει τον πίνακα με τα 4 πιόνια του παίκτη.
- *"public ArrayList<FrescoFinding> get_player_pics()"* Επιστρέφει τη λίστα με τις φωτογραφίες που έχει τραβήξει από τοιχογραφίες ο παίκτης.
- *"public void add_fresco(FrescoFinding f)"* Προσθέτει τη τοιχογραφία f στη λίστα με τις φωτογραφίες από τοιχογραφίες που έχει τραβήξει ο παίκτης.
- *"public void add_rare(RareFinding f)"* Προσθέτει το σπάνιο εύρημα f στη λίστα με τα σπάνια ευρήματα που έχει ανακαλύψει ο παίκτης.
- *"public void update_player_snakegoddess_count()"* Προσθέτει ένα στον μετρητή των αγαλματιδίων της Θεάς των Φιδιών που έχει βρει ο παίκτης.
- *"public int get_player_snakegoddess_count()"* Επιστρέφει τον μετρητή των αγαλματιδίων της Θεάς των Φιδιών που έχει βρει ο παίκτης.

# *3. Η Σχεδίαση και οι Κλάσεις του Πακέτου Controller*

❖ **Controller**




Δεδομένα:

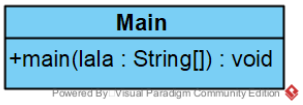
- *"private int turn"* Ακέραια μεταβλητή που κρατάει τη σειρά των παικτών. Για turn=1 παίζει ο player1 και για turn=2 παίζει ο player2.
- *"private Player player1"* Ο πρώτος παίκτης.
- *"private Player player2"* Ο δεύτερος παίκτης.
- *"private Board gameBoard"* Το ταμπλό του παιχνιδιού.
- *"private int available_cards"* Μετρητής για τις διαθέσιμες κάρτες στη στοίβα των καρτών.
- *"private int replay"* Μετρητής ο οποίος αρχικοποιείται με τιμή μηδέν. Κάθε παίκτης ρίχνει ξανά κάρτα τόσες φορές όσες η τιμή αυτού του μετρητή. Όταν ένας παίκτης παίξει κάρτα-Μινώταυρο και το αντίπαλο πιόνι είναι Θησέας, τότε αυτός ο μετρητής αυξάνεται κατά ένα και ο παίκτης που έριξε τη κάρτα ρίχνει ξανά κάρτα.

- *"private int pawns_on_chkp"* Μετρητής ο οποίος αποθηκεύει πόσα πιόνια βρίσκονται σε θέση checkpoint.

Μέθοδοι:

- *"public Controller()"* Ο constructor της κλάσης.
- *"public Player get_p1()"* Επιστρέφει τον Player1.
- *"public Player get_p2()"* Επιστρέφει τον Player2.
- *"public void new_pawn_on_chkp()"* Προσθέτει ένα στον μετρητή pawns_on_chkp.
- *"public int get_pawns_on_chkp()"* Επιστρέφει τη τιμή του μετρητή pawns_on_chkp.
- *"public void yourTurn()"* Δίνει την σειρά στον επόμενο παίκτη.
- *"public int check_minotaur(int player, int anak)"* Ελέγχει αν ο παίκτης player μπορεί να ρίξει τη κάρτα-Μινώταυρο και επιστρέφει τον αριθμό των θέσεων που θα πάει πίσω το αντίπαλο πιόνι.
- *"public int find_enemy(int player, int anak)"* Επιστρέφει τον δείκτη του πιονιού του παίκτη player το οποίο είναι στο μονοπάτι anak.
- *"public int find_lost_card(int player)"* Επιστρέφει τον δείκτη στο χέρι του παίκτη player όπου υπάρχει κάρτα με boolean is_active να έχει τιμή false.
- *"public Card newCardInHand(int cardIndex)"* Τραβάει μια κάρτα από τη στοίβα και τη τοποθετεί στο χέρι του παίκτη που έχει σειρά. Επιστρέφει τη κάρτα που τραβήχτηκε.
- *"public int get_avlblcards()"* Επιστρέφει τη τιμή του μετρητή των διαθέσιμων καρτών.
- *"public int getTurn()"* Επιστρέφει τον αριθμό του παίκτη που παίζει τώρα.
- *"public Board getGameBoard()"* Επιστρέφει το ταμπλό του παιχνιδιού.

❖ **Main**




Μέθοδοι:

- *"public static void main(String[] lala)"* Η main του παιχνιδιού.

***Παπαδάκης Γεώργιος 4975***