**Πανεπιστήμιο Κρήτης –Τμήμα Επιστήμης Υπολογιστών**

**ΗΥ252– Αντικειμενοστρεφής Προγραμματισμός**

**Διδάσκων: Ι. Τζίτζικας**

**Χειμερινό Εξάμηνο 2020-2021**

[Τίτλοσ αναφορασ]

[Δαμασκηνάκης Κώστας]

[csd3755]

[11/20/2020]

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

Για την εργασία χρησιμοποιώ το μοντέλο MVC, όπως υποδεικνύεται στα παραδείγματα. Αρχικά δημιουργώ τις κλάσεις μου με ότι κρίνω απαραίτητο ή εύχρηστο .Αναλυτικότερη περιγραφή βρίσκεται στις παρακάτω ενότητες.

## Η Σχεδίαση και οι Κλάσεις του Πακέτου Model

Στο πακέτο Model σκοπός είναι η ομαδοποίηση και η αρχικοποίηση όλων των απαραίτητων δεδομένων που θα χρειαστεί το πρόγραμμα μου. Όπως η αρχικοποίηση των Tile, Character, Player, Board και η αρχική ομαδοποίησή τους στις καταλληλες αντιστοιχες δομες.

public class Player: Περιεχόμενα:

Int lastAreaVisited: Αναγνωριστικό της τελευταίας ενέργειας

String id : Αναγνωριστικό του κάθε παίκτη

ArrayList<Character> playerCards : Ομαδοποιει τις κάρτες χαρακτήρα

ArrayList<Tile> playerArea: Ομαδοποιεί τα tiles του παίκτη

Int victoryPoints:

υπολογίζει στο τέλος του παιχνιδιόυ το σκορ του παικτη

public class Tile: Περιεχόμενα :

String image, Int position; (Εικόνα και αναγνωριστικό θέσης)

Υποκλάσεις: (extend)

Mosaic: (extra) int color για αναγνώριση χρώματος

Position = 0

Statue:

Caryatid:

Position = 1

Sphinx:

Position = 1

Amphora: (extra) int color για αναγνώριση χρώματος

Position = 2

Skeleton : (extra) int isWhichPart για αναγνώριση

Position = 3

LandSlide

Position = 4

public class Board: Περιεχόμενα :

ArrayList<mosaic> mosaicArea,

ArrayList<statue> statueArea,

ArrayList<amphora> amphoraArea,

ArrayList<skeleton> skeletonsArea,

Player[] playerList;

ArrayList<Tile> bag;

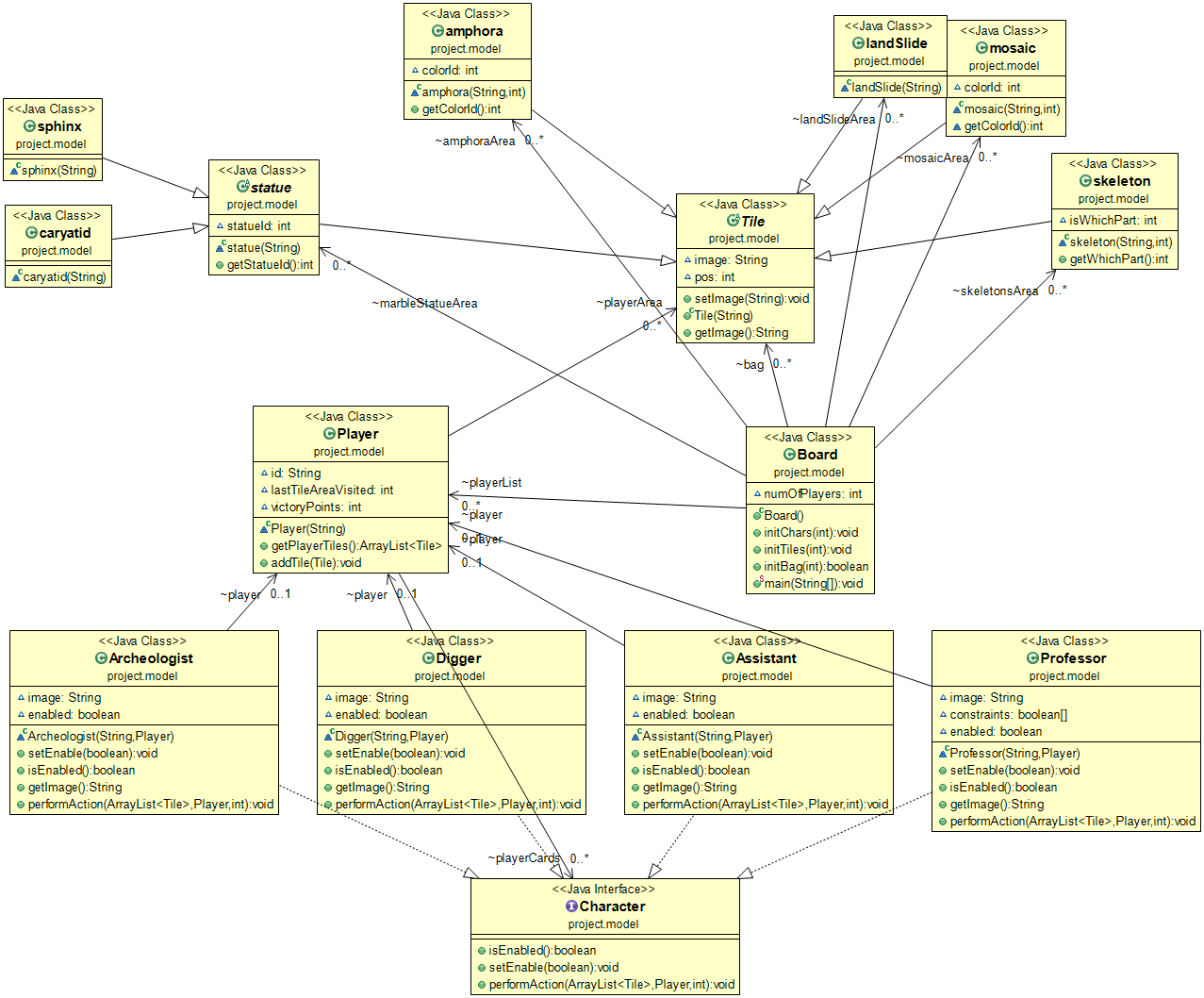
Όλα τα παραπάνω είναι τμήματα του Board και έτσι είναι απαραιτητα.

Χρήσιμες μέθοδοι : InitialiseTiles -> αποθηκεύει όλα τα Tile στο Board

: InitialiseCharacters-> αρχικοποιεί τους παίκτες και τους μοιράζει κάρτες

Interface Character: Περιεχόμενα :

String image;



## Η Σχεδίαση και οι Κλάσεις του Πακέτου Controller

Η κλάση controller πρέπει να έχει ως πεδία της στιγμιότυπα της Board και της View. Πρέπει να κάνει αρχικοποιήσεις στα Board, View. Να κανει implement τα actionEvent των αντίστοιχων ActionListeners (draw button, end\_turn button, Character’s buttons...) καθώς και να παρέχει άλλες βοηθητικές μεθόδους όπως InitializeGame(), UpdateView() , nextPlayerTurn() οι οποίες θα προκαλλούν ενημερώσεις στις δομές μου και στο γραφικό παράθυρο.

Δηλαδή το τι ενέργειες θα κάνουμε όταν ο χρήστης επιλέξει να παίξει το παιχνίδι.

## Η Σχεδίαση και οι Κλάσεις του Πακέτου View

Εδώ θα περιγράψετε τo σχέδιο υλοποίησης της προγραμματιστικής εργασίας του πακέτου view. Συγκεκριμένα, στην Α φάση του Project θα περιγράψετε το πώς σκέφτεστε να σχεδιάσετε το γραφικό περιβάλλον του πακέτου αυτού (πχ κάποιες βασικές κλάσεις, μεθόδους κλπ). Στη Β φάση, θα περιγράψετε με περισσότερες λεπτομέρειες τις κλάσεις και τις μεθόδους που χρησιμοποιήσατε.

Στο πακέτο view αρχικοποιω τη γραφικη διεπαφή και υλοποιω μεθόδους αλλαγής γραφικού περιβάλλοντος, όπως (γραφική) ενημέρωση των ευρημάτων στη συλλογή του παίκτη ή στο board .

Μερικές μεθόδους που σκέφτομαι να υλοποιήσω είναι:

Constructor: δημιουργεί τη γραφική διεπαφή και τοποθετεί κάθε τι εκεί που πρέπει.

/\*\*

\* @brief will insert the selected Tile in player's area

\* @param P is the player i want to customize

\* @precondition must be player's turn

\* @postcondition if move is valid the board loses this icon image

</br>and it goes to the player's area.

\* @invariant the button's position on board

\*/

public void insertInPlayerArea(Player P,Tile t):Τα tiles του board θα είναι buttons και εάν επιτρέπεται η κίνηση αυτή θα γίνεται η ενημέρωση του καμβά.

/\*\*

\* @brief if for example player has 4 mosaics it becomes one image

\* @param tile

\* @precondition must be called after player's move

\* @if move is valid then graphics refresh

\*/

public void combineTiles(Tile tile)

/\*\*

\* @precondition the number of Tile objects must be == 4

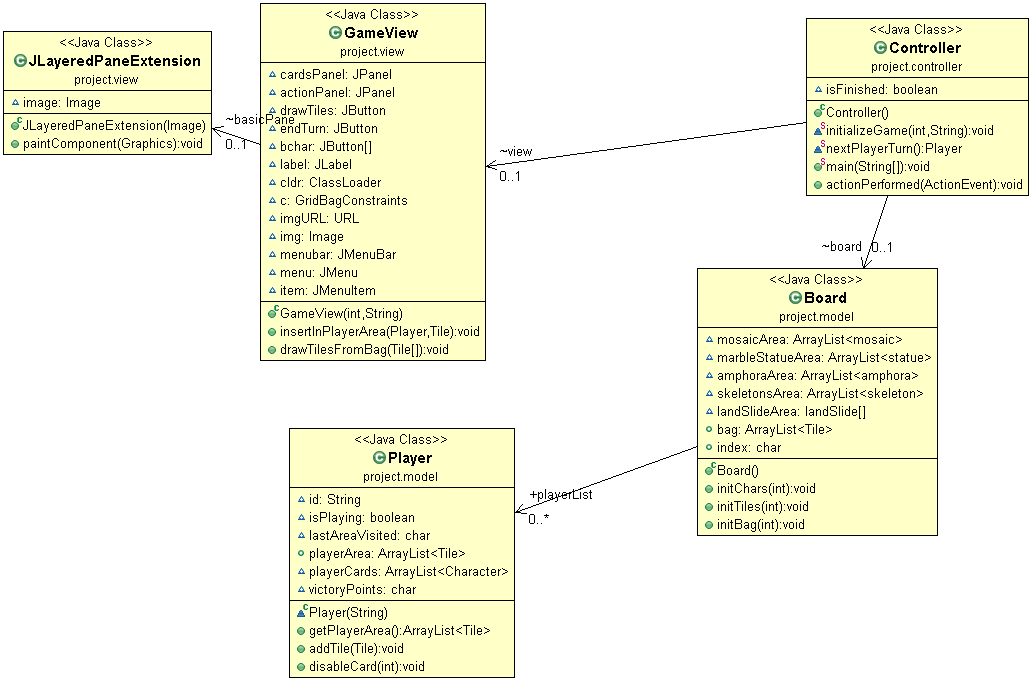
\* @postcondition repaints the board with 4 new Buttons on it

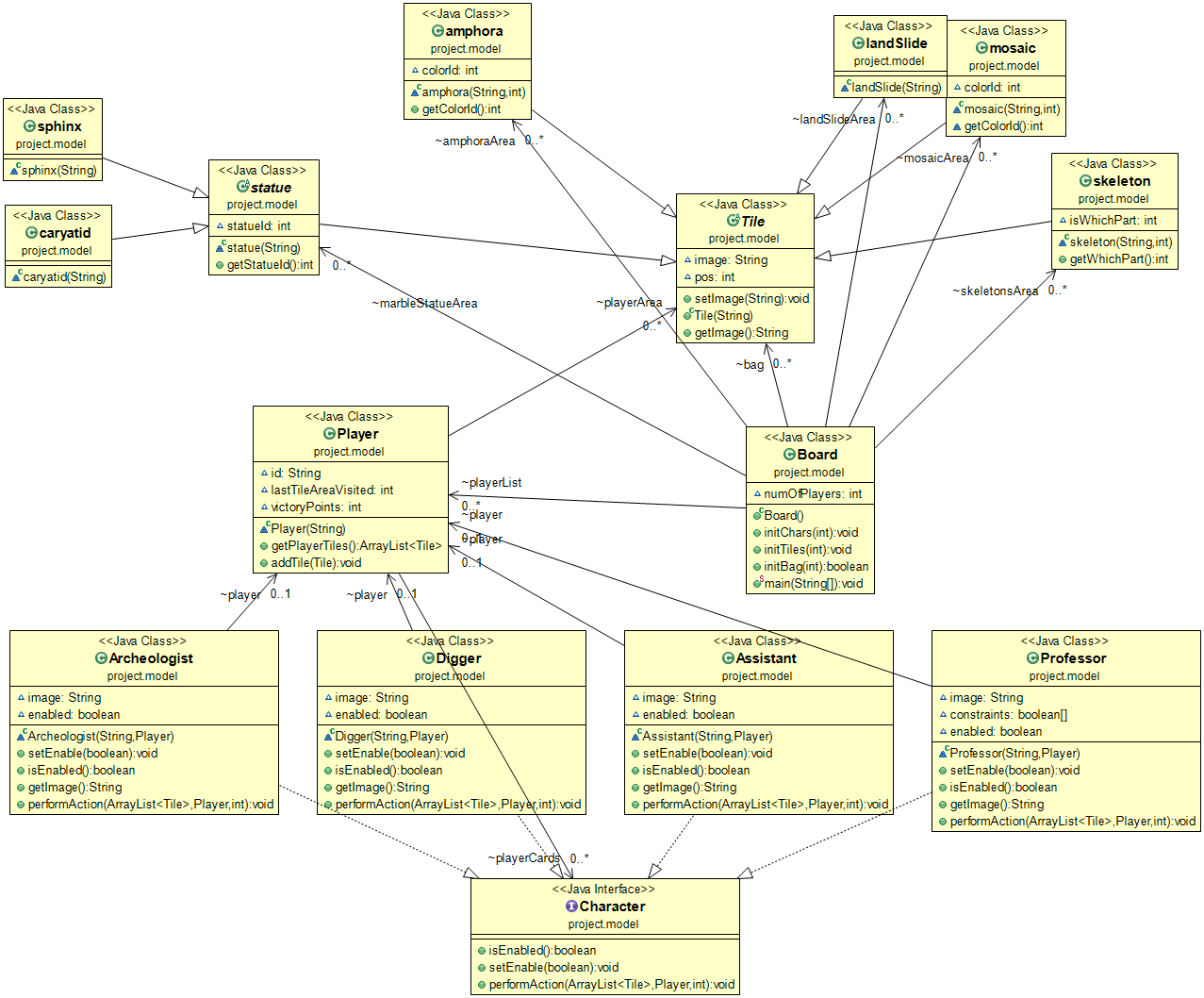
\* @param tilesDrawn

\*/

public void drawTilesFromBag(Tile[] tilesDrawn)

## Η Αλληλεπίδραση μεταξύ των κλάσεων – Διαγράμματα UML





## Λειτουργικότητα (Β Φάση)

Κατάφερα να το υλοποιήσω λειτουργικά και έκανα το μπόνους 2 με το solo gameplay.  
Σχετικά με την υλοποίηση του μπόνους θεωρώ ότι πρέπει στο τέλος κάθε γύρου να μαζεύει ο κλέφτης όλα τα ευρήματα (tiles) του board και όχι όταν βρίσκουμε landslide γιατί έτσι μπορεί ο παίκτης να πατάει endTurn μέχρι να μαζέψει όλα τα ευρήματα και να εξασφαλίσει σίγουρη νίκη.  
 Επομένως στη δική μου υλοποίηση ακολούθησα τις οδηγίες του “κουτιού” όπως δηλαδή αναγράφεται το solo mode στο [HY252-Project\_2020\_AmphipolisΑρχείο](https://elearn.uoc.gr/mod/resource/view.php?id=64111)

.  
Επίσης υλοποίησα και το save.

## Συμπεράσματα

Στην Β φάση απλοποίησα κατά μεγάλο βαθμό τα identifications που νόμιζα στην Α φάση ότι θα χρειαστώ , με τη βοήθεια των Map, εφόσον το Map το διδαχθήκαμε μετά την παράδοση της Α φάσης. Δεν ήμουν σίγουρος ποιες θα ήταν οι ιδανικές δομές για την ολοκλήρωση του προτζεκτ οπότε το είχα γεμίσει με identifications.  
Επίσης αρχικά νόμιζα ότι θα υλοποιήσω actionListener στον Controller , πράγμα μη εφικτό απ’ότι φάνηκε.

Το τελικό UML κατέληξε όπως φαίνεται παρακάτω.