Πανεπιστήμιο Κρήτης –Τμήμα Επιστήμης Υπολογιστών

ΗΥ252– Αντικειμενοστρεφής Προγραμματισμός

Διδάσκων: Ι. Τζίτζικας

Χειμερινό Εξάμηνο 2020-2021

Εισαγωγή

Think and describe what you plan to do and why it will be useful.

Περιεχόμενα

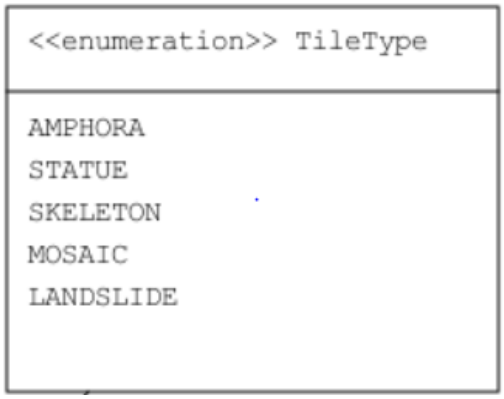
- Εισαγωγή

I used the MVC model which stands for Model-View-Controller. This model keeps the code more organized and helps coordination when working with a large team. In the next few paragraphs i'm going to explain my approach on creating the Amphipolis Board Game while using the MVC model.

- Η Σχεδίαση και οι Κλάσεις του Πακέτου Model

TILES

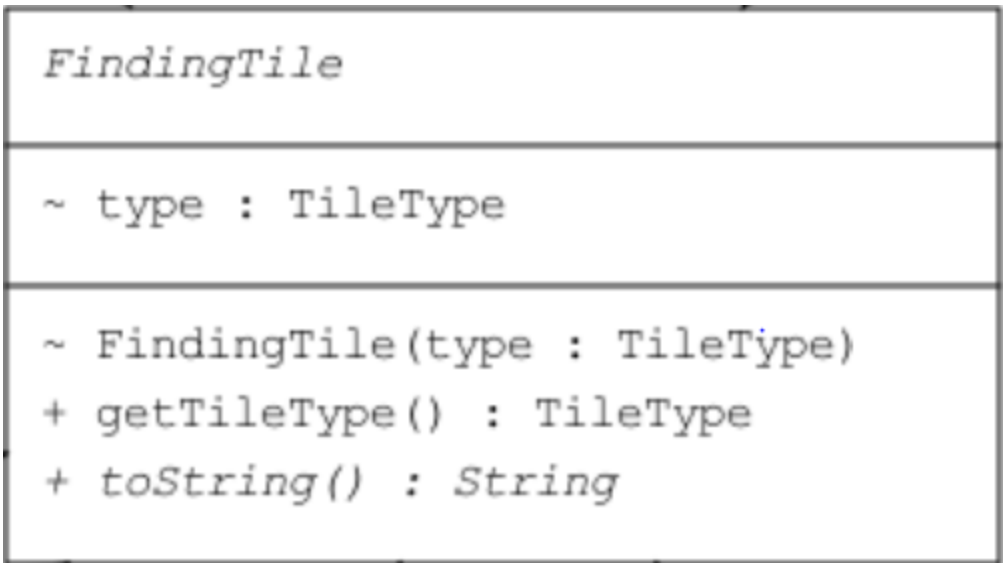
I started by creating the model package as it is the database of the project and has all the objects that will be used. I firstly created the tiles by creating a Tile interface that has two functioons, getTileType() and toString() (The toString() function is mostly for debug and visual purposes so I will not go in depth as to what it does). The getTileType() helps distinguish the different tile from one another. Every class that implements this interface has a different type. I created an enum to create the different tiles.



The getTileType() method returns one of these instances of the enum.

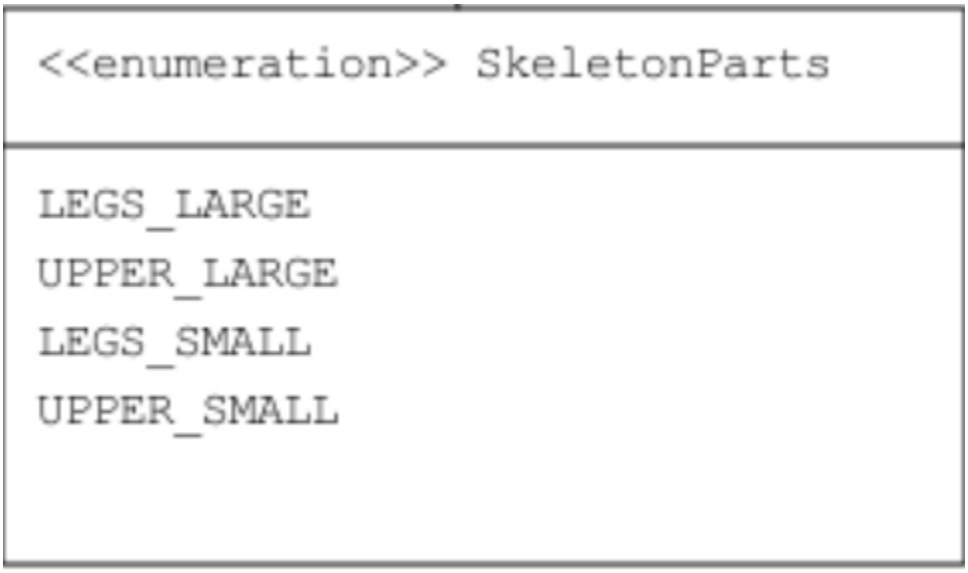
Creating the Landslide Tile was straight forward it implements the Tile interface and when the getTileType() is called it returns LANDSLIDE.

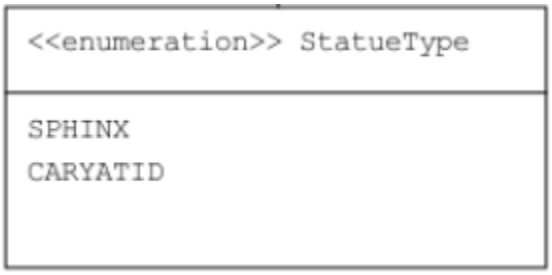
For the other Tiles I had to create a class that implements the Tile interface that is going to be the parent class of all the other classes. This class is called FindingTile and stores the type of the tile.



It implements the getTileType and returns the type that is stored to it's variable type. This constructor is called from the children of this class and the children set the type variable to whatever type they are.

To create the other Tiles that will be children of the FindingTile I had to create some enums to help me with the implementation.



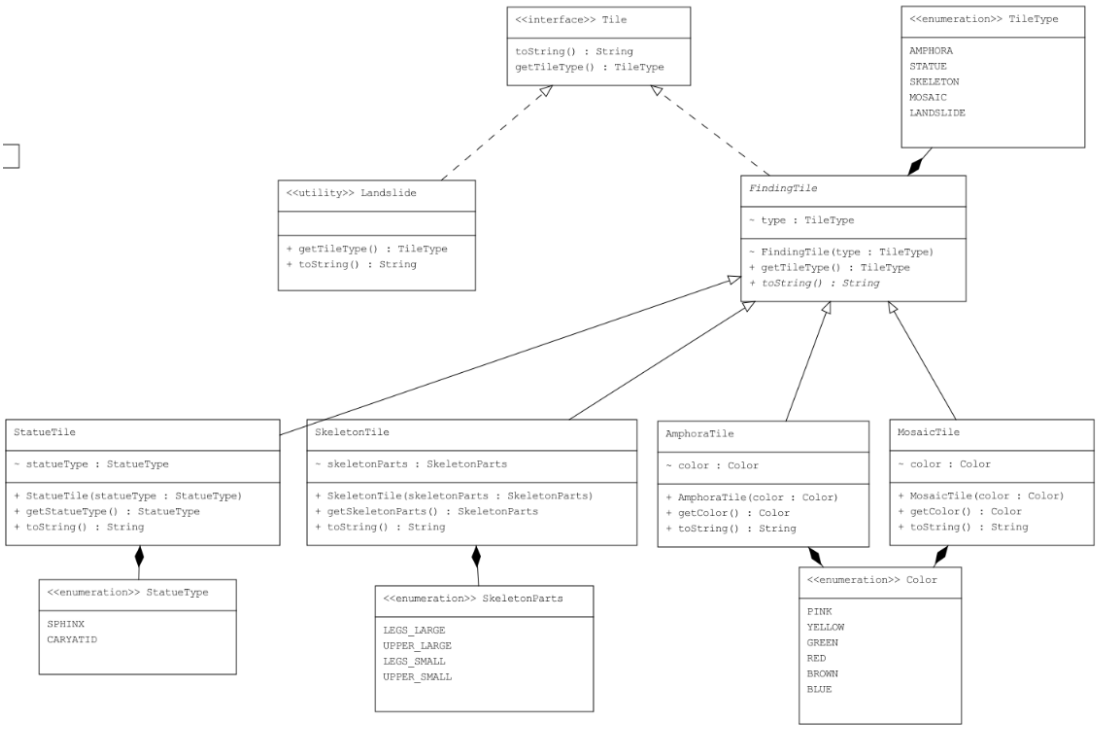


The enum color is used to help me differentiate the different colored Mosaics and Amphoras. For those children classes I have created a variable called color that stores the color of the instance and sets the type to FindingTile according from what children the super constructor was called. For Amphoras and Mosaics I have also created a method getColor() and it returns the color of the Tile.

The enum SkeletonParts is used to help me differentiate the parts of the skeletons as well as their size. Simillar to the color enum I have created the method getSkeletonParts() that returns the part of the skeleton that this Tile has. Also when the contsructor is called it sets the super variable type to the Enum. SkeletonTile.

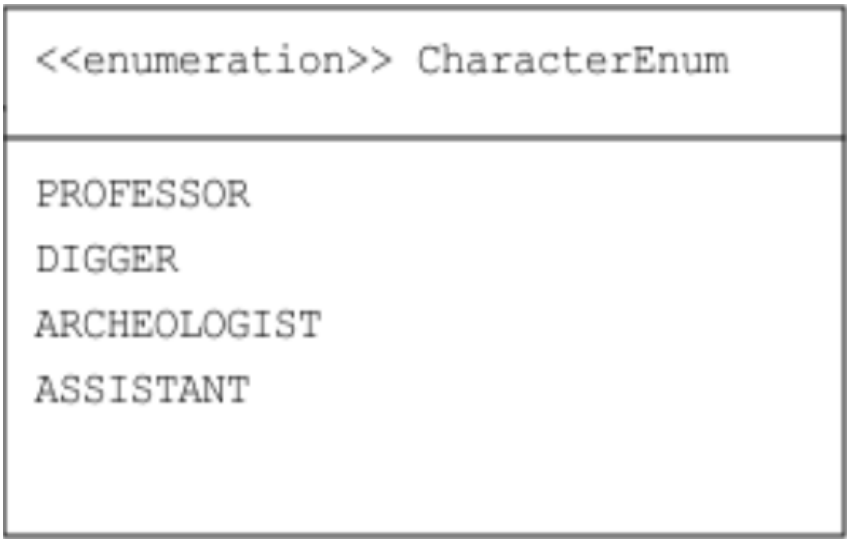
Lastly the enum StatueType is used to help me differentiate the type of the statue that the tile holds. Their are two types of statues SPHINX and CARYATID. Again similar to the previous classes I have created the method getStatueType() which returns an enum containg the type of the statue and when the constructor is called from this children it sets the super variable type to Enum.StatueTile.

This detailed and useful system creates a Tile system that is easy to work with and the enums make it staight forward so that anyone who is working on the project can understand what everything is used for.

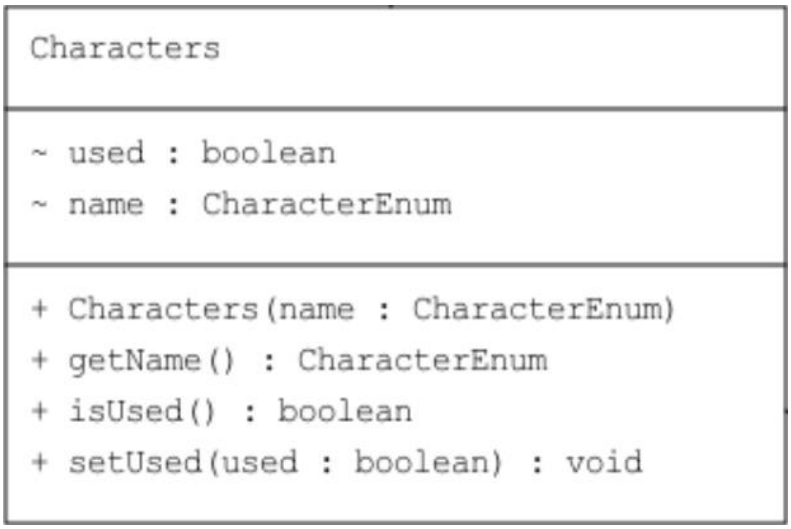


CARDS

Then after creating the tiles I the special character cards. This cards represent a character. This character has a unique ability and can be used only one time for each player during the game. To diffirentiate the characters from one another I created yet another enum.



This enum holds the names of the four different characters. So I need a class that will represent a card and will store one instance of this enum and wether the card has been used or not. So I created the Characters class :



The constructor sets the name of the card so the instance of the character and sets the boolean variable used to false.

The getName() and isUsed() functions are sraight forward they are getters and return the value of the variables and the setUsed() is a setter which sets the value of a variable.

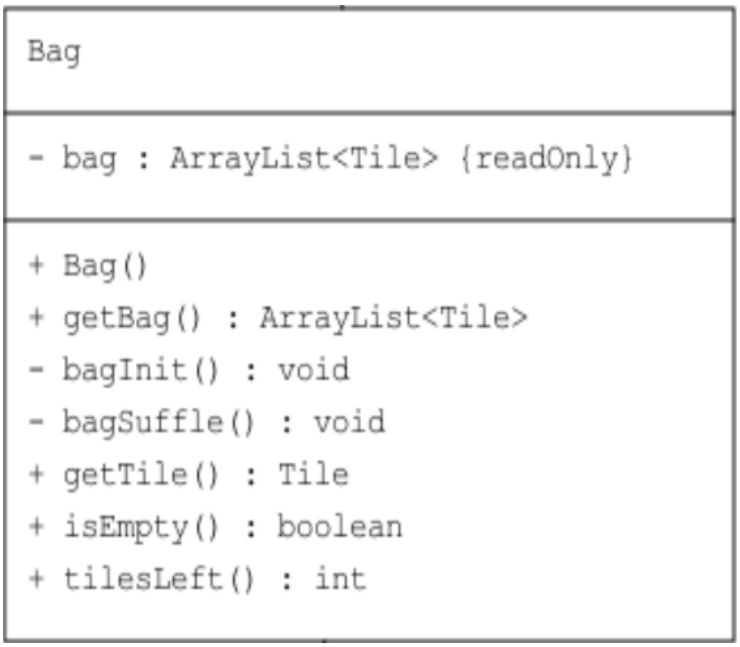
P.S. I know I din't have to create enums to be able to diffirentiate different classes from one another because I can use instanceof. But after some though I came to the conclusion that Enums are more diverse and I can do many things with them that I couldn't if I just checked for the instance Example: parts of skeleton, color. And also the enums guide other people when reading my code and make it easy to understand what i'm doing. In this project sometimes I didn't go for the optimal solution because I wanted to make my code accessible and easy to pick by anyone. Also when I will continue this project I will understand and remember my code better. Sometimes you have to sacrifice speed and memory for more approachable code. Now that that is out of the way let's continue.

BAG

After I was done creating all the neccessary pieces I made the Bag. The Bag class is exactly like the real life bag that the board game is coming with. The Bag has a variable that is an ArrayList of Tiles meaning I can store all the Tiles that I want inside the Bag and access them easily whenever I want.

The contstructor method allocates the space for the ArrayList and then I inittiliaze it with the method bagInit(). What bagInit() does is inserting the right tiles that are needed for the game to play 12 caryatids,12 sphinx, 27 mosaics with different colors, 30 skeleton parts with differnt parts and sizes and 30 Amphoras also with different colors. After inserting them in the bag they are in a way sorted so if a player would draw the first index of the array he would get the same type of Tile until they run out and start getting another type of Tile. That's why I implemented a method called bagSuffle() and as the name suggests it suffles the tiles inside the bag so every time the player draws from the bag I am sure that it's a radom Tile.

Also for the Bag to be completely read we have to implement some more methods such as getTile() that draws the first tile inside the Bag. The isEmpty() which returns true if the Bag is empty(it has no Tiles left). And the method tilesLeft() which returns how many tiles are left in the bag.



PLAYER

After all this preparation I was ready to create the Player class which holds all the information for a player.

First it holds the collection of Tile of the player in an ArrayList so that the player can easily expand his collection. Next I created the drawnTiles an array of 4 spaces, which is a temp storing point where the player draws from the bag and places the tiles he drew to the board. We also need the player to hold his characters cards that are 4 some I created an array of 4 Characters. Last but not least the player needs to store his score and to diffirentiate him from other players I have the variable ID which is different for every player.

The constructor simply initilliazes every variable and creates the 4 different character cards with the help of initCharacters().

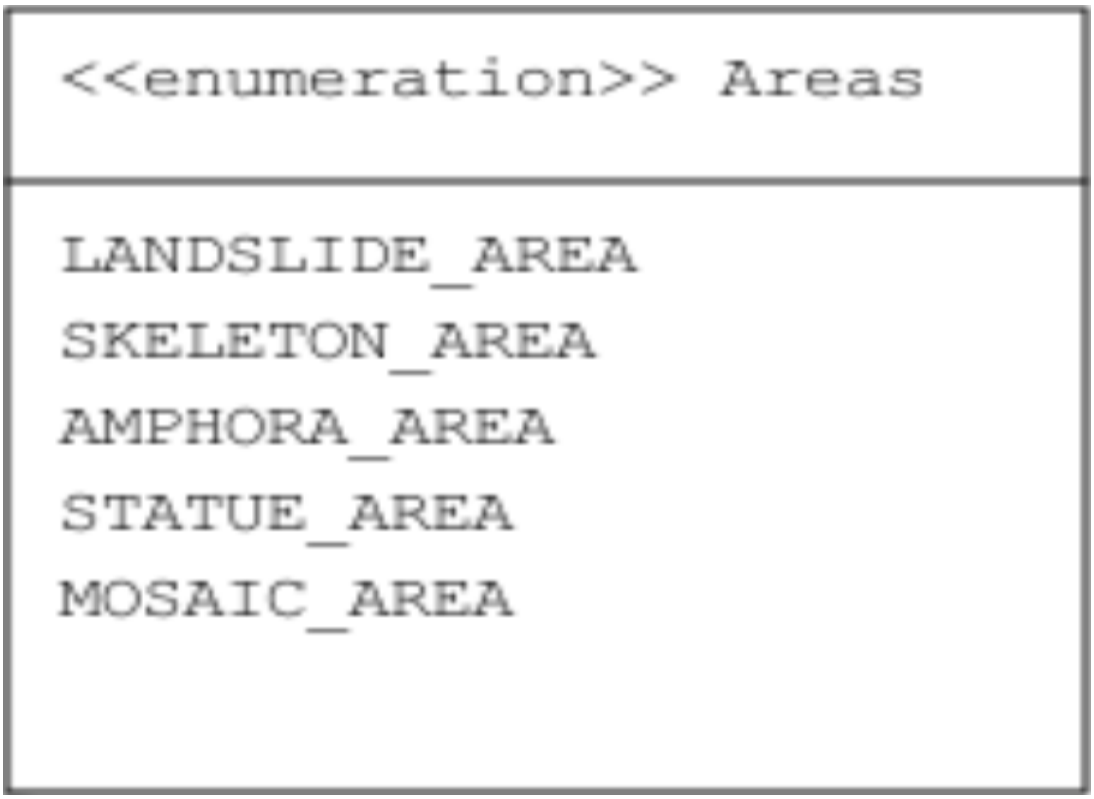
I have getters for all the variables getID(), getScore(), getCharacters(), getDrawnTiles(), getCollection() which return the value that the variables hold.

I also created some other useful methods such as findScore() which traverses through the collection and calculates the score of the player and sets it and returns it. To do that efficiently and correctly I also created a sortCollection() method which sorts the collection of the Tiles by type to make the calculation of the score much more easier. Another useful method is findCharacter(name) which returns the character with name = name. This method can be used to check if the character of the player has been used again by accessing its isUsed() method. A very crucial role plays the setDrawnTilesToBoard() which takes the tiles inside the drawTiles and puts them in the corresponding areas in the board. Lastly I created addToCollection(Tile) which takes the Tile it is given and inserts it in the collection.

With all those methods we can emulate a real player and we can do everything a player must do in the game.

BOARDS

The last class we must create in model is the Board class which holds information for all the Tiles that are in the board. I have created an enum to help me manage this in a very interesting way. The enum is called Areas and holds the 5 areas that tiles can be placed in the board. But the clever thing about this enum is that its of his value represents a number *MOSAIC_AREA = 0, STATUE_AREA = 1…LANDSLIDE_AREA =4.* That is easily done by creating a private variable in the enum and giving it an int value based on what enum constructor is called and also the enum is put on a map based on its value as a key. So for example in the map the value with the key 1 is STATUE_AREA. That doesn't only allow enums to be represented as numbers but to represent enums as numbers also.



Now for the implementation of the board. The board has 2 variables an Array of 16 Tiles that are the landslide area and store all the landslide Tiles and TileAreas an ArrayList of 4 ArrayLists that store Tiles. Basically its an array container that for every index an ArrayList of Tiles are stored that is why I wanted to be able to translate enums to numbers so I can access the area of the statue_tiles without needing to know in what index I have stored it. Example:

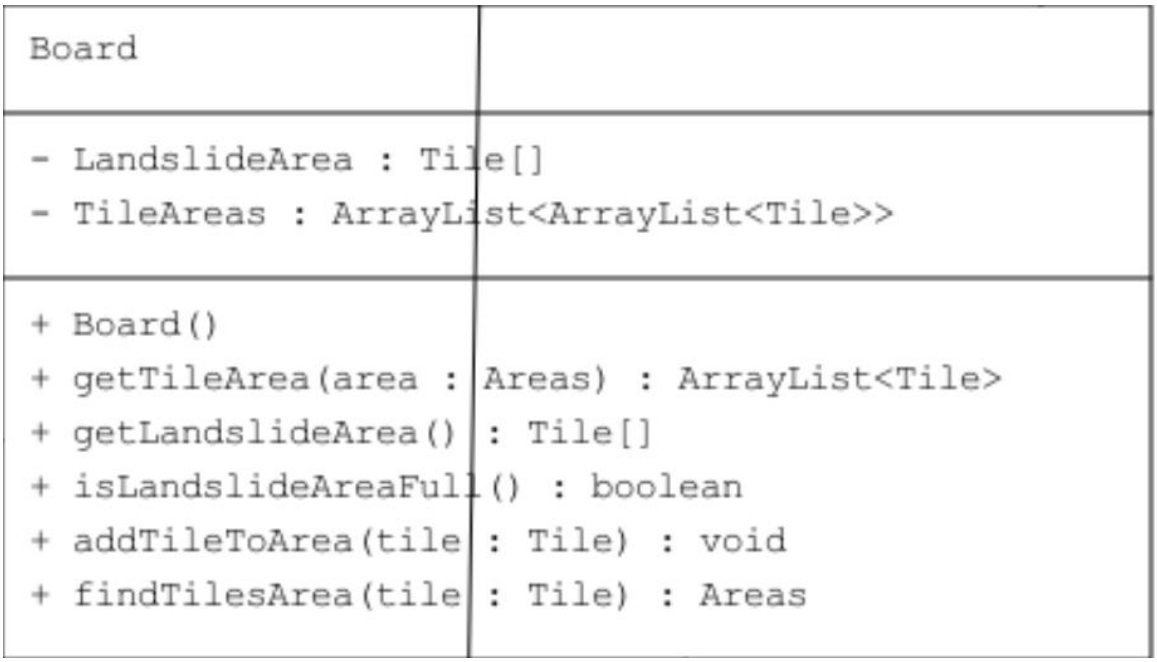
To access the SKELETON_AREA I just use:

```
TileAreas.get(Areas.SKELETON_AREA.getValue())
```

As I said in this project I want to make my code accessible and easy to pick up and not create the optimal solution. But I think this solution is pretty fast too.

The Board has some methods to help with the implementation. The constructor initializes everything and sets the 16 array of Tiles to null to make finding if the Landslie Area is full easier. We also have the

getTileArea(Area) that takes the area that I want to get and returns the arraylist of tiles of this area. The same goes with getLandslideArea() it returns the arraylist of tiles from the Landslide Area. The method is LandslideAreaFull() is used to determine wether the game should end or continue. If the area is full the game ends, we check if the area is full by checking if there are any null Tiles left. That is why we initillized it in the constructor. Lastly we have the addTileToArea(Tile) which takes a Tile and by using the findTilesArea(Tiles) method we place the Tile in the right area(ArrayList).



- Η Σχεδίαση και οι Κλάσεις του Πακέτου Controller

The Controller Class is where the logic of the game is written. We have for every turn who are the players p1,p2,p3,p4 the state of the board that changes every turn, the bag from which tiles are drawn and for specifically this turn the turn that tells which players turn is, the action which tells how many Tiles has the player gotten from the board (at two tiles he can only activate a characters skill if he hasn't in this round or End his turn) and weather if he has used a character in this round (because a player can only use one character each round).

The controller's contstructor sets everything up and allocates space in the memory and with initBoard() we initiallize everything so the game can start.

PlayersTurn() is used at the start of every round puts for random tiles at the corresponding areas.

useCharacter(Character) uses a character of a player if the character has not been used.

gameIsFinished() does nothing if the Landslide Tiles in the area are not 16 but if they are the game ends and the game calculates the PlayersScores to find the Winner.

setScores() calculates the scores of the players and sets them.

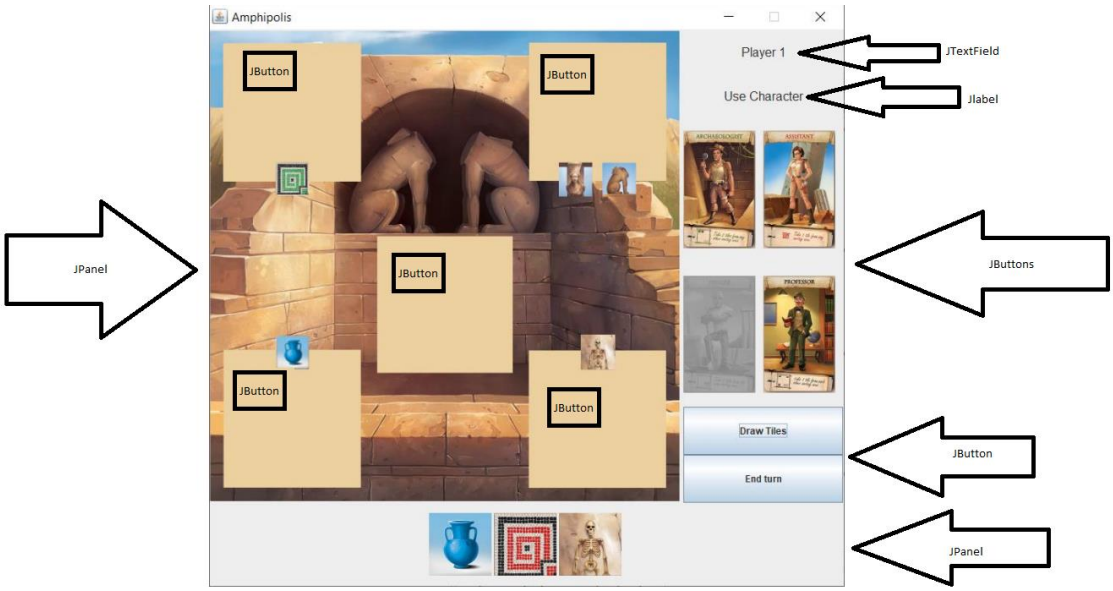
getTiles(Tile) gets the Tile from the board.

finishedActions() Checks if the player has finished his actions.

hasUsedCharacters() Checks if the player has used a character in this round.

- Η Σχεδίαση και οι Κλάσεις του Πακέτου View

The View Class with the help of the controller(the game logic) represents visually all the board game and makes the interaction of the player with the game much more easier.



The constructor allocates space for all the buttons and panels and images and the initView() initiallizes them gives them values and make them functionable.

newTurn() is called every new turn and updates the displayed items.

And to make everything interact with the player we have Listeners that are waiting for an action such as the press of a button and performs the operation to complete the request.

- Η Αλληλεπίδραση μεταξύ των κλάσεων – Διαγράμματα UML

Σε αυτήν την ενότητα μπορείτε να συμπεριλάβετε διαγράμματα UML, και να εξηγήσετε μέσω αυτών την αλληλεπίδραση των κλάσεων (πχ μεταξύ των κλάσεων διαφορετικών πακέτων.

- Λειτουργικότητα (Β Φάση)

Σε αυτήν την ενότητα θα γράψετε στη Β φάση ποια ερωτήματα καταφέρατε να υλοποιήσετε είτε επιτυχώς είτε εν μέρει (και ενδεχομένως ποια όχι).

- Συμπεράσματα

Σε αυτήν την ενότητα θα γράψετε τα συμπεράσματα σας για την εργασία, τυχόν προβλήματα που συναντήσατε και γενικά ότι άλλο κρίνετε απαραίτητο να αναφερθεί.