ΗΥ252 – Αντικειμενοστρεφής Προγραμματισμός

ΣΤΑΥΡΟΣ ΑΡΓΥΡΟΥ

ΑΜ: **2962**

ΑΝΑΦΟΡΑ ΕΡΓΑΣΙΑΣ: **STRATEGO ICE VS FIRE**

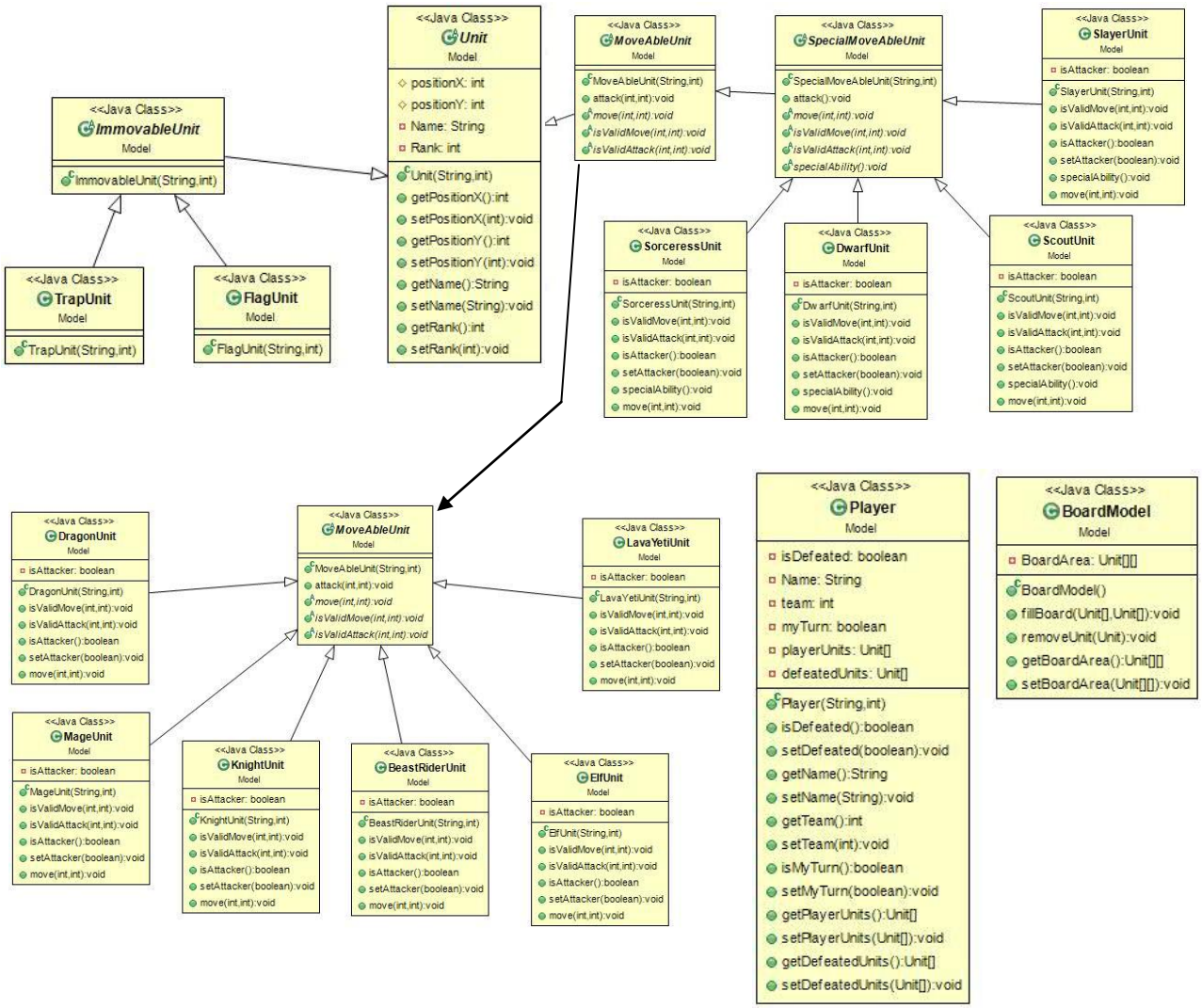
ΦΑΣΗ: Α’

The Project is structured using the MVC principle (Model View Controller), where the model and the View classes are completely unaware of each other and are simply bridged together using a Controller class.

As this phase is mostly focused in breaking the game down to MVC not much work has been done with the View classes, that will be completed for the second phase of this project – game. For detailed method analysis check out the javadocs.

The Model:

The Model classes are responsible for the game's logic, and represent the various objects required from the game. The main objects of the game are of course the Units.



The **Units** are divided into **two categories**. The **Immovable Units** that consists of the flag and the trap, and the **Movable units**. Movable units break down to the normal units that are able to move, but there are also units that have special abilities. Those units are children of the class ***SpecialMoveableUnits***, an abstract class with the abstract method special ability, that it's children must implement accordingly. The Units also have Rank that indicates their power with an integer from 1 through 10, a **name** and finally their **positions X** and **Y** on the board, which are **protected**, so **only subclasses can see**.

Besides the Immovable units, the rest of the Units have methods responsible for moving, attacking, and **checking if a move or an attack is valid**.

There is also the class **Player** , to represent the two players who will play the game, for the Red and Blue teams accordingly. The Player class has the **name** and **team** of a player, but also holds their units that are in game and their defeated units using an array of type Unit. There is also a Boolean variable **isTurn**, that will be used to cycle through the players back and forth until a winner is found. Finaly the variable **isDefeated** will indicate if a player is still in-game(false) or if the player is defeated(true).

The **Board Model** Class represents the board that the game will be played on, and its responsible to fill the board with the units given from the players, and rearrange the board accordingly when the players move their units. **The board is a two dimensional array of type Unit.**

The Controller:

The class **Controller** consists the Controller part of the **MVC** and it includes among others the **main** method of the game.

The Controller counts the turns and holds the Players in an array.

Using the method populate Board the controller gets the board from the model and then using the View classes it creates the appropriate graphics and puts the Units on the board visually.

The controller in general handles all the changes and interactions.

When a user does something the controller uses the model and view to calculate the results and display them

The View:

The **view** is responsible only for creating the graphics using the JFrame and JPanel in this case. It is not responsible however to decide when to move graphics around on its own, this is done by the controller as well.

In this case there is an abstract class View with a method called Render to render the visuals, and there are also two subclasses to that. The Main view , will create everything but the Unit's visual representations. The Unit View will create the Buttons and graphics required to represent the Units. And all of these classes together will be used by the controller for the graphical aspect of this game.