CSE 443
Object Oriented Analysis and Design

Homework - 2

Muharrem Ozan Yeşiller
171044033

Project Definition

The project is based on a two-dimensional puzzle. When we bring the Green, Red and Blue buttons on top of each other or side by side with at least 3 of the same color, we attack the enemy in the relevant row. The attack algorithm here is “Az Laf, Çok İş A.Ş.” it has been implemented as the company requested from us. The artificial intelligence part, on the other hand, has been developed without being far from randomness, but in a somewhat clever way. Just like round robin scheduling, it will attack the 1st character first, then the 2nd character, then the 3rd character with a 50% probability (randomness here) of vertical or horizontal attacks. While the game is being developed, the button listener produces the relevant boolean variables, while the java thread that plays the game I implemented runs the game by consuming these changes. The buttons that enable the game to be played are an inner class called Tile. This class is extended from the JButton class.

** Extra note in the project, the map is shuffled after the user or AI moves. Tiles do not come from the bottom up.

How to Play?

The start button allows us to start the game. Pause button is used to pause the game. Exit button helps us exit the game.
Pressing two buttons after pressing the start button means you are requesting the replacement of these two buttons. If this request is not an illegal move, the move will be executed.

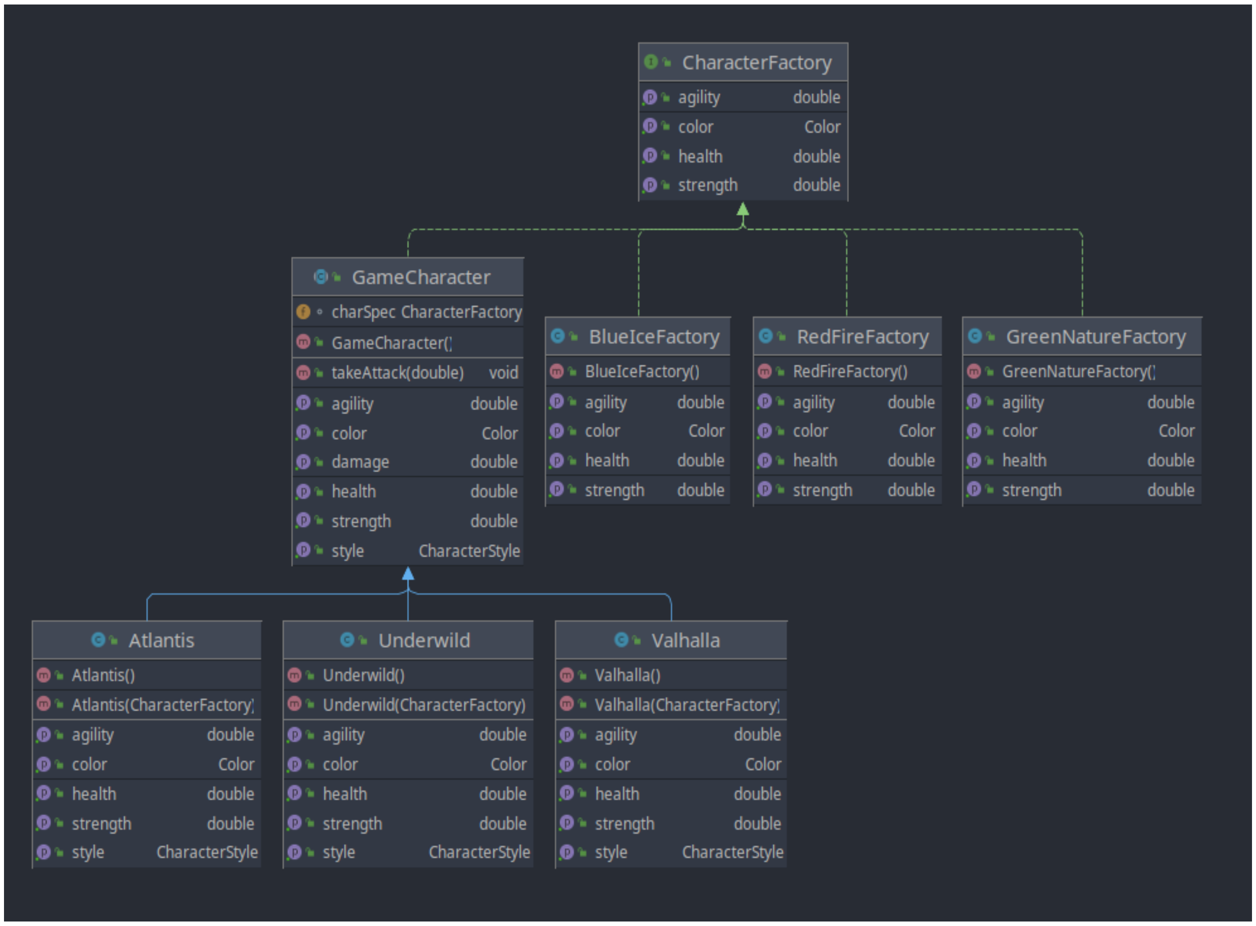
Non Functional Requirements

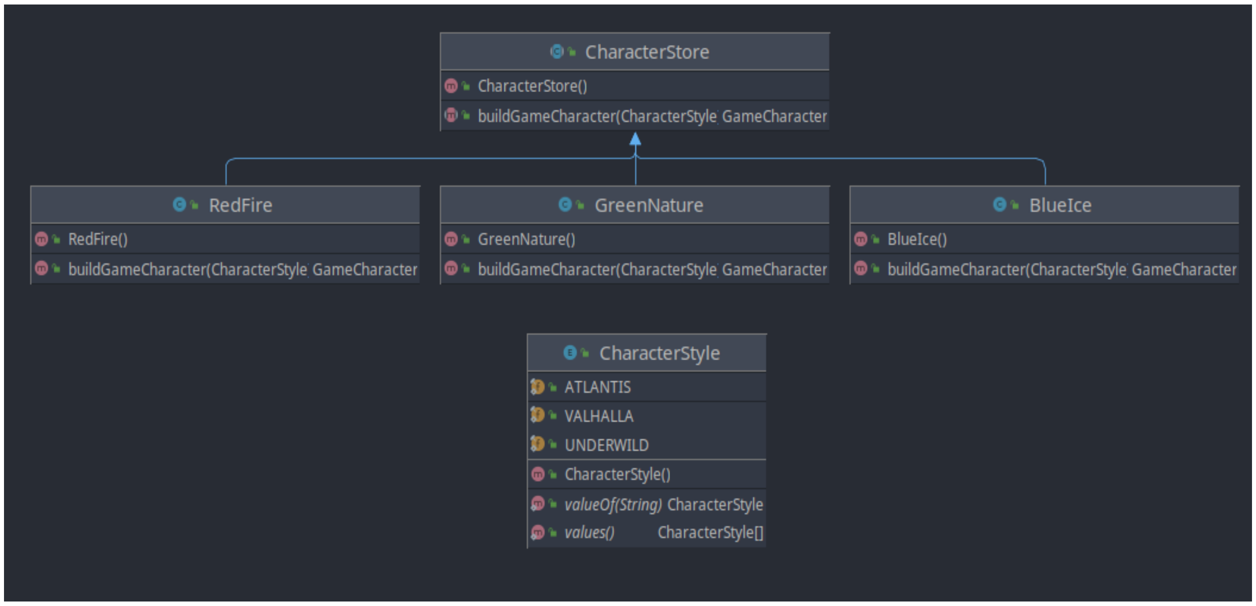
Java Development Kit 14 (JDK 14)
An average computer and operating systems (Windows/Unix/macOS)
Mouse click button should work. (if default)

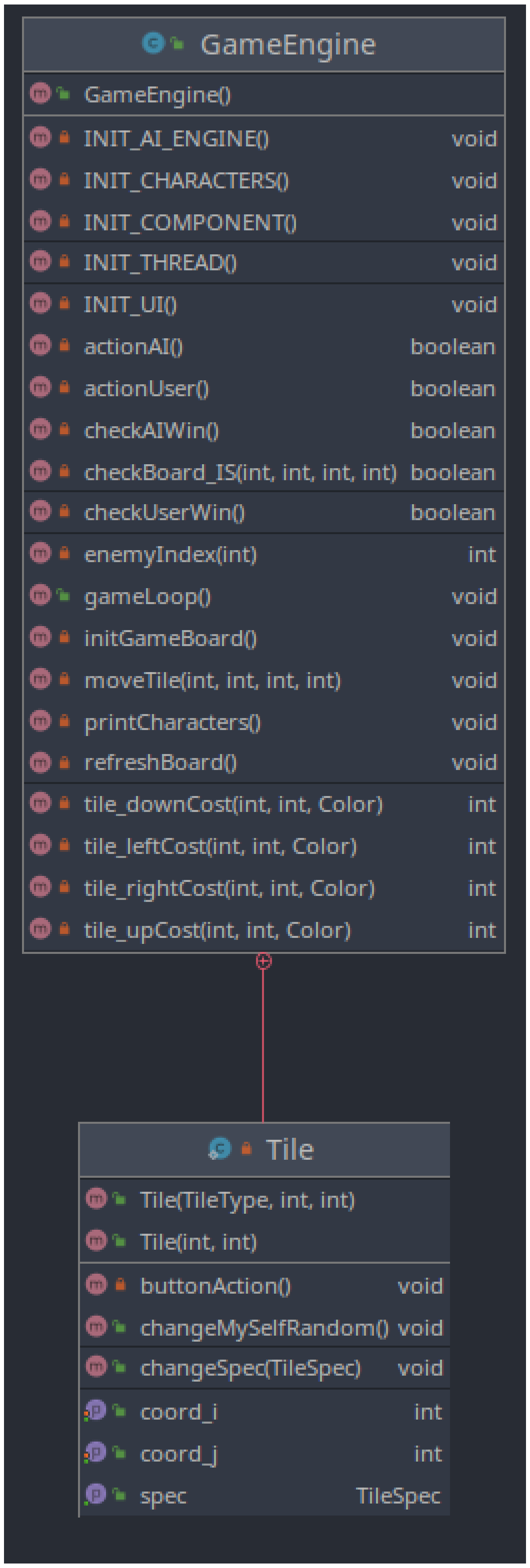
Design Patterns and Class Diagrams

a) <u>Abstract Factory Design Pattern</u>

The Factory design pattern implements a common interface for subclasses with similar properties and assigns the responsibility of creating objects to a factory class that performs through this interface. When the client wants to create a child object, it makes a request to the factory class and the object is created by the factory class. If we exemplify the Factory design pattern, whether we have Mercedes-Audi-BMW vehicles, these vehicles are connected to the Car interface and are created via a factory class over the Car interface when desired.


CharacterFactory
agility double
color Color
health double
strength double
GameCharacter
charSpec CharacterFactory
GameCharacter(
takeAttack(double) void
agility double
color Color
damage double
health double
strength double
style CharacterStyle
BlueIceFactory
BlueIceFactory()
agility double
color Color
health double
strength double
RedFireFactory
RedFireFactory()
agility double
color Color
health double
strength double
GreenNatureFactory
GreenNatureFactory(
agility double
color Color
health double
strength double
Atlantis
Atlantis()
Atlantis(CharacterFactory
agility double
color Color
health double
strength double
style CharacterStyle
Underwild
Underwild()
Underwild(CharacterFactory)
agility double
color Color
health double
strength double
style CharacterStyle
Valhalla
Valhalla()
Valhalla(CharacterFactory
agility double
color Color
health double
strength double
style CharacterStyle


CharacterStore
CharacterStore()
buildGameCharacter(CharacterStyle GameCharacter
RedFire
RedFire()
buildGameCharacter(CharacterStyle GameCharacter
GreenNature
GreenNature()
buildGameCharacter(CharacterStyle GameCharacter
BlueIce
BlueIce()
buildGameCharacter(CharacterStyle GameCharacter
CharacterStyle
ATLANTIS
VALHALLA
UNDERWILD
CharacterStyle()
valueOf(String) CharacterStyle
values() CharacterStyle[]


GameEngine
GameEngine()
INIT_AI_ENGINE() void
INIT_CHARACTERS() void
INIT_COMPONENT() void
INIT_THREAD() void
INIT_UI() void
actionAI() boolean
actionUser() boolean
checkAIWin() boolean
checkBoard_IS(int, int, int, int) boolean
checkUserWin() boolean
enemyIndex(int) int
gameLoop() void
initGameBoard() void
moveTile(int, int, int, int) void
printCharacters() void
refreshBoard() void
tile_downCost(int, int, Color) int
tile_leftCost(int, int, Color) int
tile_rightCost(int, int, Color) int
tile_upCost(int, int, Color) int
Tile
Tile(TileType, int, int)
Tile(int, int)
buttonAction() void
changeMySelfRandom() void
changeSpec(TileSpec) void
coord_i int
coord_j int
spec TileSpec