Assignment 2 - Documentation

According to assignment 1 specification we needed to consider our design again, and Delete or Add some new Classes to make the code work and more readable. In this documentation we will consider and explain all the changes and show how the classes and methods work in our code.

Dinosaurs:

First, we started by constructing the Dinosaurs again from the Beginning and we eventually decided to Have 4 classes: **Dinosaur**, **Velociraptors**, **Protoceratops** and **Dinosaur Egg**.

Dinosaur Class is a subclass of Actor and Both **Velociraptors** and **Protoceratops** are its subclass as well. We know that both types of dinosaurs are Actors but in the meantime they have some similar attributes and methods which we need to have them in both classes. To avoid repetition in the code we decided to put those similar methods and attributes in the Dinosaur class like **hunger()** and **chanceOfBreed().** But there are some other methods that we saw we can't use them in this class and out them separately in their own classes.

DinosaurEgg class is a subclass of Item, because egg must be an Item in the which let the player to pickup or drop the item during the whole game, Also we can create an object of this item when Dinosaurs have the chance to breed.
In this class we check all the eggs in the map and see their ages, if the eggAge becomes 10, they will hatch into a baby dinosaur and we also differentiate the babies by bad adding different skills to them. So if the **Protoceratops** egg hatches, it becomes 'p' and if **Velociraptors** egg hatches, it becomes 'v'.

Behaviour:

In the Behaviour Interface we decided to add 2 new classes, one is VeloFindFoodBehaviour and ProFindFoodBehaviour, and the reason is that they do not have the same behaviour for having food. The Protoceratops search to find Tree and Grass, on the other hand, Velociraptors search for Dinosaur's egg (ProEgg) and corpse. Because this finding food method is part of the for loop we couldn't combine them as one method or class.

Shop:

In this class we just put the menu and the options that the player can choose to buy or sell in the shop, and all of the numbers in the menu do something. We also set the item prices in this class.

Tree:

In this class we set a grow() method as set a random chance number to grow trees. In each Turn we will find the trees in the game and set the nearest dirt location to check by this method. If we have dirt nearby we will replace them by young trees every turn and in the meantime the other trees grows and get old.

Player:

Most of our important functions are placed in this player class, like selling and buying, dropping and showing the price. For Each Transaction we will record and show the price of any item which has sold or bought and let the player choose to drop the item on the map. Our player also has a budget which will be shown after any kind of Transaction and record it in the system.

Status:

We created a Status class as a Enum<> class, and we put all our skills inside it like as, TREE, GRASS, PROEGG,...