Design

At the beginning, it is the explanation for the reason for the project design. There are three kinds of items in this project, farmer, animal and transporter. First of all, the project needs a grid to put these items and record the coordinate of every item. The farmers' job is producing foods that supply to animals. The transporter will transport the food to animals. Then, animals will consume food and transform into nutrition. Moreover, there will be a score for the relationship between the production and the consumption, which shows in the grid. Thus, in the grid class, it needs param nutrition and consumption param and the array to store the item's coordinate. There are some similar things for farmers, transporters and animals. All of them do the operation on food stock. Therefore, it is possible to create an abstract class to define arguments or methods that are useful for all items. The critical point of the game is the consumption of the production. The food stock will change in each step, so it is necessary for every item in the grid to have methods that can select or update the food stock.

Implementation

The main character of the project are farmers, animals and the transporters. Different kinds of objects have diverse functions, for example, farmers produce foods, and the animals consume foods. However, as for different farmers or transporters, like corn farmer and the radish farmer, they have some differences. In the following words, it will indicate the implementation of different objects.

Farmer

There are two kinds of farmers in this project. Although two kinds of farmers produce different number of food, the same thing is both of them need to produce food. Therefore, the only thing is to judge when the farmer will produce, then add the food into stock. Except for the step will affect farmers produce, the position of the farmer will decide farmers prudence or not as well. Thus, before executing the process method, it is necessary to check the position of the farmer. Based on the requirement of different farmers, traverse the coordinate around the farmer to judge if the farmer meet the condition, then return a Boolean param. If the Boolean param is true, then the farmer start producing, or else the farmer will do nothing.

Transporter

Transporters need to transport foods from farmers to animals. There three kinds of the transporter in my projects, HT, VT and NT. The first thing that they need to do is to judge if they are in the middle of the farmer and the animal. Then, the transporter also needs to judge if the stock of farmers is not empty. The

animals can get the food from the transporter only when the two conditions are satisfying. During the transport, the transporter will firstly take foods from farmers stock then the stock of farmers will decrease. Moreover, if the stock of farmers is not enough for transporter's carry number, the transporter will take all stock of the farmer away. The method for VT and HT to judge if they are in the middle of farmers and animals are very similar. In terms of HT, it needs to get the y coordinate of the HT, and traverse the x coordinate. As for VT, it will get the y coordinate and traverse the x coordinate to find farmers and animals. The implementation of the NT is more complex, it needs to find the closest farmer and the closest animal from it. First of all, NT will get the coordinate of farmers and animals; then it will calculate the length by the sum of absolute value (|x1-x2|+|y1-y2|). By this way, NT can figure out which farmer or animal is closer to it. If farmers or animals have the same length to NT, NT will not transport. Otherwise, NT will find the closest farmer to take foods, then NT will record the closest animal's coordinate.

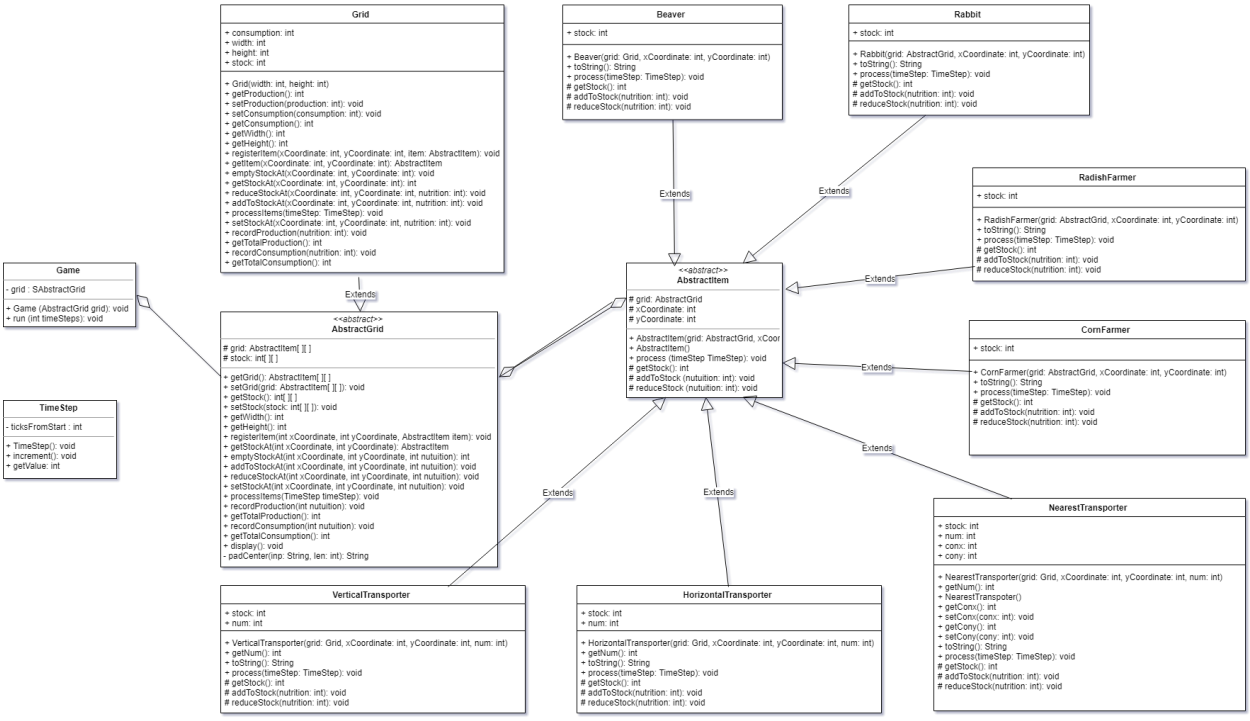
Animals

The animal can get food from the transporter and consume the food. Animals will find the transporter near to them. The search way is similar to the method used in transporters. Based on the animal's x coordinate, traverse the y coordinate to find if there is a VT next it. If the transporter is NT, the animal will check the closest animal's coordinate in NT, if the coordinate equals to its coordinate, the animal will take stock of the transporter if transporter's stock is not zero, then the stock of transporter will be deleted. There is another function that animals need to do in every step which is consumption. Animals need to judge the stock of them first; if it is not empty, then they will consume food and record the consumption. After consuming, animals need to reset the stock. Rabbit needs to delete all the rest stock, and the deleted foods become to waste. As for beaver, it will have one more judgement, because it can store up to 50 foods.

UML

The following words are the description for UML. There are twelve classes in this project, so twelve blocks in the UML picture. Each block involves the params and the method that the class owns. The plus in the block means the method or the param is public. Otherwise, the minus means private; the number sign means the method or the params are protected. If the method or the argument is protected, it can only use in the same class; other classes cannot call it or utilise it. The methods and arguments name will be after the symbol in each line. After that is the type of methods and params. The main classes in this project are AbstractItem, AbstractGrid, Game and TimeStep. Among these four classes, the AbstractItem and the AbstractGrid are abstract classes. Other classes can extend the abstract class. Then, the son class can use

the method and the param that creates in the father class. Moreover, the son class will write the content of the method belongs to the father class. Therefore, as the UML picture shows that the Beaver, Rabbis, CornFarmer, RadishFarmer, HorizontalTransporter, NearestTransporter and VerticalTransporter extends AbstractItem and the line with the arrows between them indicates that their relationship is a generalisation. Except for the arguments in the AbstractItem, I add an int param (stock) in these son classes, because I think that the stock should put in each object, and it will be more convenient to call it by the name of the object rather than an array with the coordinate. As for the grid class, because the abstract class AbstractGrid cannot use directly, the grid class extends it and implements the functions. Besides, the relationship between the AbstractGrid and the AbstractItem is aggregation because they both have the argument belongs to each other. For example, the Game has the param (grid) belongs to AbstractGrid. Thus, their relationship is aggregation as well.



Evaluation

By this practical, I got a clearer understanding of the abstract class and the relationship between the father class and the son class. Although my project can pass the 46 tests, there are many codes that I think it is complex and not efficient. For instance, the implementation of the farmers. The famer has to judge that if there are other farmers on left, then the famer has to judge the right again. I hope to make it clearer and more logical in next practical.