

Object oriented programming

Homework 7

Question 1a

Decide for each method, whether it is executed by the lifeform class (basic operation in the slides), whether it is executed by a subclass (required operation) or whether it is an optional (a hook). Argue for each function what makes most sense in written form.

- act is implemented in LifeForm and calls all 4 methods in order:
 - move
 - eat
 - reproduce
 - die
- the method above are not defined in Lifeform since the conditions or them to apply vary a lot across species.
- Grass simply inherits act, but since all other methods are overwritten, they have no effect.
- Move get implemented separately in Sheep and Wolf since Wolf would need it to be modified anyway as it would occupy the cell previously occupied by his prey sheep. This could be modified if, for example, another prey would enter the picture. Let's suppose we created a Deer class, that also can be eaten by the wolves, and, like the sheep, can only eat grass. In this scenario, sheep and Deer could inherit the same move method and in the Wolf class, we would just overwrite it with the specification of occupying its prey cell after eating it, as is right now.

Because in the current implementation we only have 2 lifeforms that can move, defining the base version of the method move in LifeForm or in Sheep does not change much.
- Eat implementation also varies depending if the Lifeform eating is a Sheep or a Wolf. Here the difference is much more obvious than for the case of Move. When the sheep eats, it simply gains energy and changes the state of the cell. However, when the wolf eats, it kills the sheep it just ate.
- Reproduce varies across Lifeforms Because the new offspring the method should create would need to call a Wolf or a Sheep class depending on which species the parent belongs to. What can be implemented in LifeForm is the common necessary conditions of energy required to call the reproduce method. This is however was not changed from the last version as the required amount of energy to reproduce vary from sheep to wolf. A longer but simpler approach was chosen as the methods needed to be redefined in any case because of the difference in classes called.
- Die does not change a lot across species however I still chose to implement in separately in Sheep and Wolf since the wolf could also die of chance, as instructed in the last homework. This slight difference made it necessary to rewrite the whole function for

Wolf and again, since we only have two lifeforms that can “die”, it would not change much whether to implement the base no-chance-involved version of die in Sheep or in LifeForm.

Question 1b

In class, we discussed that die may sound weird as generally one doesn't die at each timestep. If a method title is misleading, rename it. Change the name die to survive. What are the advantages and disadvantages of renaming functions in (any of) your project(s)?

I managed to change the name of the die method to survive very quickly using the key F2 on my keyboard. This name simply better captures what the method actually does in all cases it is used. It does not make a LifeForm die, but it simply checks whether it is still alive at each time step or if it needs to die. Survive is a much better suitable option. However, there are some disadvantages in this approach, especially if working with other people. When sharing code, even if not specified, a convention might be established and therefore changes like this might carry evitable confusion. It is important to change the names when the previous one causes even more confusion,

Question 1c

Implement act() as a template method in your lifeform. Also consider its impact for grass. Write down how and why grass can use and profit (or not) from the template method approach.

From the current implementation Grass does not really benefit from the template method act() defined in the lifeform, even though it inherits from it, all the other methods are overwritten and not used. The methods Grass actually uses are simply the ones specific for it, Grow and Consume. This whole implementation is not ideal for grass since it behaves completely different from Sheep and Wolf. This can even lead to some misinterpretation of the inheritance from LifeForm, which, at this stage, does not influence much Grass' behaviour.

Question 3

Imagine, that in the future, you don't want your lifeforms to act on their own but also to interact. In week 8, you will implement some interactions. To prepare yourself for that, the question is how you would structure your program to enable interactions. Explain your choices and provide examples.

I would create a specific interaction phase in the simulation. This next phase will only take place after all lifeforms in the list have acted. This new phase would need to implement new objects as well, as, for example an encounter detector based on cell proximity. At this point I would need more instruction on the type of interactions to implement, whether they will happen always and, if so, how they will change depending on the type of animals. (e.g. will two wolves kill each other or reproduce?). I would also need to understand which animal to give priority to in case two available animals for encounters are equally distant to a third one. For example it would make sense that if a sheep was equally distant to a wolf and another sheep, it will be more likely to be killed and eaten rather than reproduce. It would make sense to execute all interactions at once after the animals have moved so that there would not be any confusion in deciding priorities or where to stop.