Problem Set 09: Design Patterns

Given that we had to adopt a design pattern in this exercise, we decided to use the 'command pattern' and implemented the 'Life Expectancy' gene, the 'Speed' gene and the 'Movement 1' gene with it.

We are fully aware that using the 'Strategy' or 'Decorator' patterns would have been the more elegant solutions, but the development stage of 'Ursuppe' has already been far too much advanced to make it possible to easily implement these more sophisticated design pattern. To keep time and effort within a reasonable limit we thus implemented the command pattern which – in our opinion – also fits quite well to the problem. As can be seen in *Figure 1*, our genes are organized as follows: There is a superclass called 'GeneCard' which has three subclasses 'GeneCardLifeExpectancy', 'GeneCardSpeed' and 'GeneCardMovement1'.

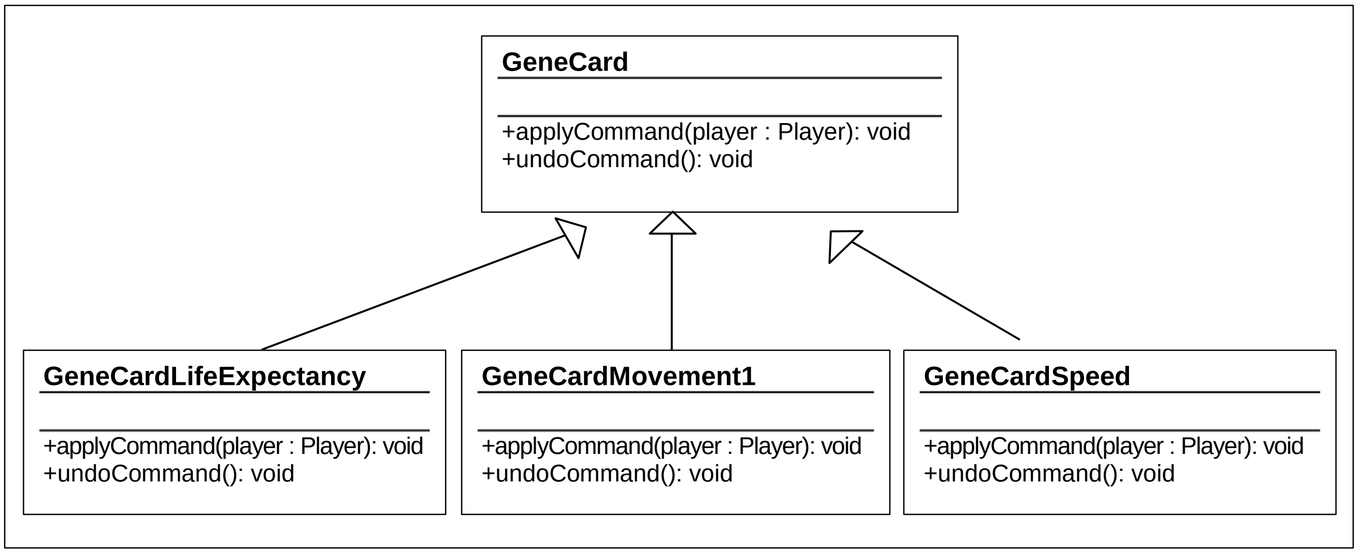
GeneCard
+applyCommand(player : Player): void
+undoCommand(): void
GeneCardLifeExpectancy
+applyCommand(player : Player): void
+undoCommand(): void
GeneCardMovement1
+applyCommand(player : Player): void
+undoCommand(): void
GeneCardSpeed
+applyCommand(player : Player): void
+undoCommand(): void

Figure 1

When a player buys a GeneCard, he calls the #applyCommand method and passes *itself* to the GeneCard as an argument. The #applyCommand method then modifies certain parameters of this Player object, depending on what function its gene has: The #applyCommand method in the 'GeneCardSpeed' class changes the movement points of the player from one to two. In the 'GeneCardMovement1' class the #applyCommand method changes the dies available for this player from one to two and in the 'GeneCardLifeExpectancy' this method adds an additional point to the maximal damage points of every amoeba of this player.

When a player sells a GeneCard, he calls the #undoCommand method of this GeneCard and passes *itself* to this GeneCard object as an argument. The #undoCommand method makes sure that all the changes to the Player object done by the #applyCommand method are cancelled. The #undoCommand method of the 'GeneCardSpeed' class for example, sets the movement points of the player back to one.

Especially with more genes (up to 20 as in the game rules) implementing the command pattern can help to hold the Player class more or less simple when not all the functions of the genes have to be implemented inside this class but can be outsourced to separate classes. However, there is also a downside of this: When we build a separate class for every gene, there are 20 new classes in our package which is not especially helpful for keeping the overview of all the classes...