Peter Maitland, Christopher Pauly, Matus Mikus, Petr Sramek

JOOSE lab group ??????

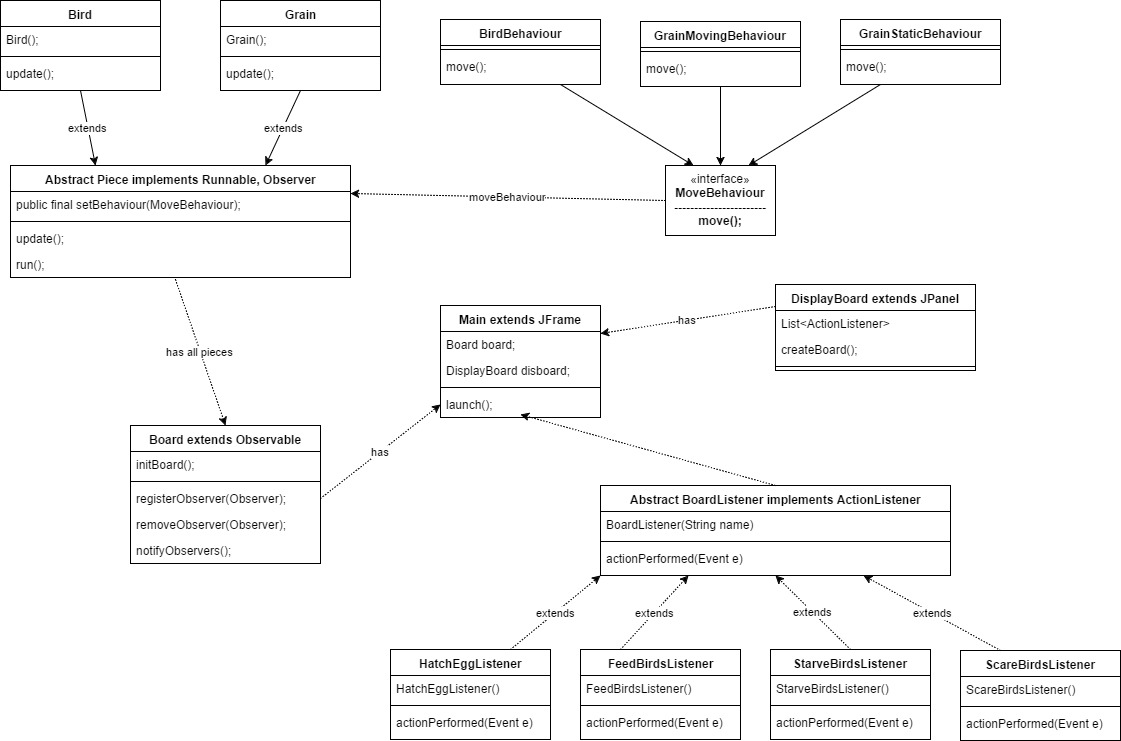
Task 1:

The current design is not appropriate. Apart from using a lot of inheritance which is bad for scalability as changes are not as easily propagated, Board also extends Observable and implements Observer which is not a good practice. The behaviour of Pieces is implemented entirely inside Board which increases complexity of Board and classes inheriting from Board. This means decreased cohesion.

Current design also does all the graphics inside the Board which is also a model which holds the board and all the pieces. This is not a good practice either as the concerns are not separated.

Task 2:

We have decided to use Strategy design pattern to decrease the usage of inheritance therefore the behaviour is implemented inside MoveBehaviour interface and each Piece accepts a behaviour and moves accordingly. This means that adding new behaviour to a Bird or Grain requires only to add a new Behaviour class implementing MoveBehaviour. We have also used Observer/Observable pattern where each Piece observes the Board to know where the other Pieces are. In addition to that, we have decided to change the design in a way that would separate concerns thus we used MVC pattern.



Task 5:

We have decided to extend the behaviour of the simulation to multiply Birds when they feed on Grain. The Strategy design pattern allowed us easy extension since it only involved implementing new MoveBehaviour for Bird. Our design allowed us to add new feature easily but we would like to point out that when running GrainMovingBehaviour there occurs a race condition demonstrated by NullPointerException. It happens because two Birds are accessing the same Grain and one of them removes it.

Task 5:

Our design allowed us to add a new behaviour easily since we have an interface for that. So only by creating new class implementing

that interface and adding an appropriate button we extended the behaviour. This was mainly thanks to usage of Strategy design pattern.