Homework Assignment

Vidrighin George Bogdan CEN 3.3B

8 January 2022

# Application design

## Architectural overview

* Coordinates = represents the coordinates
* Hen = simulates the behavior of a hen
* every hen must have the parameters position and id, and must know to which farm it belongs to
* the hen will continue to run and try to move in order to create eggs while it is not retired
* in order to create eggs, the hens needs to move in a parameter newPosition
* HenGenerator = creates multiple hens and gives them a farm to lay eggs on
* it needs the data from Headquarters
* Hens are spawning randomly in each farm at a random place in the farm (random time between 500-1000 milliseconds)
* Farm = represents a farm that has some hens. Every time a hen is

moving it creates an egg that will be taken by a farm employee in order to send them to the Headquarters

* the Farm needs @param farmSize, @param nrHens and @param id in order to start working
* when the farm starts to run, also ObserverHens will start
* a hen will be added if the maximum number of hens was not reached
* when a @param egg is created, it will be added in the list
* a FarmEmployee will try to take an egg from list
* Egg = represents an egg
* an egg has a random name
* MainClass = simulates the activities of Uncle John and his staff
* ObserverHens = used by every farm in order to see where the hens are
* FarmEmployee = simulates the behavior of a farm employee
* Farm employees have access to all farms and send eggs to the address indicated by serverAddress, serverPort
* the farm’s employee goal is to take eggs from farms
* if an egg was taken, the farm employee sends it to the Headquarters
* FarmEmployeeGenerator = creates multiple farm employees and gives them a farm to work on.

\* RetireHen = retires a hen

* at one moment of time a random farm from headquarters is picked and an hen from this farm will be retired
* UncleJohn = UncleJohn is waiting for the eggs
* Headquarter = creates the following entities: Hen generator, employee Generator ,farms and retire hen class
* using nrFarms (number of farms) and nrEmployees (number of employees)

, the Headquarter will create:\* hen generator, farm employee generator

,farms and retire hens class

* CreateFarms creates the farms with a random size and put them in a list
* AddHenToFarm adds a hen to the farm indicated by farmId
* AddEmployee adds an employee to the list
* RetireHen - the hen with the id equal to @param henId will be removed from the farm indicated by @param farmId

## Implementation decisions

For the farm I used 2 locks:

1. ReentrantLock oneHenTrytoMove - this lock is used to ensure the correct- ness of the hen move method
2. ReentrantLock setPriorityOperation - this lock is used to used to deter- mine which operation has priority in this class at a given time.

The operations are:a hen try to move, an employee try to get a egg and the farm try to ask all the hens where they are.

For employees I used a semaphore to limit their number to a maximum of 10 employees that can read from a farm at a time.Every employee has its own id.

# Observations

Employees must be quick to get eggs from the factories because hens are ex- tremely fast (they only sleep 30 milliseconds between laying eggs).

UncleJohn can receive eggs from multiple employees, so he must be quick to accept them (UncleJohn gets no rest between receiving gifts).

# Bibliographic References

1. Baeldung

**https://**[**www.baeldung.com/java-semaphore**](http://www.baeldung.com/java-semaphore) **https://**[**www.baeldung.com/java-concurrent-locks**](http://www.baeldung.com/java-concurrent-locks)