Farm Simulation

Josh Casey (21361783) & Jakub Czerniejewski (21646694)

Working Functionality

Description	**Status**	**Note**
More than One Farmer	Working	The number of farmers can be specified in a config file, it has to be at least 5 as we are using a parallel solution and there is at least one farmer for each field.
Prioritisation of Stocking Animals	Working	We have a farmer taking care of each field this means that as soon as the delivery comes the farmer picks its assigned animals and goes to the field.
Limited Field Capacity	Working	Fields have limited capacity which can be set manually causing the Farmer to wait at a field with its stock if the field is full until the field is empty, then the farmer puts all the stock he can and returns to the enclosure for the next delivery.
Breaks By Farmer	Working	A farmer will be randomly assigned a break time between every 200 - 300 ticks. The Brake counter is reset when given a break allowing other farmers to take breaks preventing starvation and fairness issues.
Configurable Parameters	Working	Uses a scanner for custom input. Type “Yes” to set your own parameter or type anything else for default values. Note: Farmers must be 5 or more due to parallel implementation.
It takes a farmer: Ten Ticks to get to a Field. Ten Ticks to get back to the enclosure. One tick for each animal.	Working	Once the farmer picks up the animals from the enclosure he ‘waits’ ten ticks to get to the field, then for each animal he ‘waits’ a tick at the field and then he ‘waits’ ten to simulate going back to the enclosure to wait for a new delivery.
Buyer waits to buy an animal until there is enough stock to buy from.	Working	The buyer tries to buy an animal for example a chicken, if there aren't any chickens left a waiting flag is set, once notified that there has been a delivery the farmer will try to buy the chicken if successful he can go try to buy a different animal if not he will keep on waiting.

Running the code

Compile with Javac - javac Main.java

Run with Java - java Main.java

Custom parameters will be presented if you type “yes” into the system and press enter.
If “no” is inputted and enter pressed the default values will be used.

Tasks and Dependencies

Farmer: Depends on no buyers being at the fields, field limit and deliveries made to the enclosure.
Stocks a specified field with a number of animals in its possession if it does not exceed the field limit, otherwise the farmer waits at the field until it is empty.

Buyer: Depends on the fact if the field is empty or a farmer is currently stocking the field

Enclosure: Generates a random batch of animals and notifies farmers when it arrives.

Fields: Store the animals stocked by farmers and then notify buyers when animals become available

Tick Counter: Atomic variable used to increment time and synchronise the threads and activities.

Patterns and Strategies

Farmers:

- For breaks each farmer will be randomly assigned a break time between 200 and 300 ticks using `**ThreadLocalRandom.current().nextInt(200, 301);**`.

- For the mutual exclusion of fields we implemented monitors using a wait and notifying process, waking threads once a resource becomes available or waiting while another thread is accessing or modifying that resource.

Buyers:

- Acquire resources from the fields at regular intervals when the resources are available else they will wait at the field until it becomes available.

Fields:

- Store the resources and act as a critical section allowing only a buyer or farmer to take or add to a field. When a resource is available in a field that a buyer wants they are notified

Enclosure:

- Generates a random batch of animals at intervals notifying the farmers when the resources arrive.
- Critical section

Threads:

- Runnable is implemented in the classes that are supposed to be run by threads: Farmers Buyers and Enclosure.
- Executors.newCachedThreadPool is used to create threads and to submit the runnable tasks to them.
- Monitors are used to manage concurrency by the use of Notify, NotifyAll and Wait.

Fairness, prevention of starvation

Starvation

Breaks: No starvation or unfairness can occur for farmers taking breaks as when a farmer takes a break at their break time their clock is reset. The limited number of farmers on break also ensures that not all farmers halt and can provide stock to the fields not causing the buyers to starve.

Buyer: A buyer waits if a field is empty, then once restocked it is notified to prevent the buyer from waiting indefinitely.

Fairness

Break intervals are random and their time for a break is reset when they have a break allowing farmers who have not received a break yet to be prioritised.

There is no fairness implemented for buyers and farmers as synchronized notify does not keep a queue and it does not consider how long a thread has been waiting but the actions performed are very quick updates so the chances of farmers not being able to access any stock or buyers waiting forever are quite low.
We explored a solution with a Reentrant Lock as there is a parameter which can be supplied to it to implement fairness but we found out that the execution loop is busy spinning meanwhile trying to acquire the lock instead of waiting in the synchronized solution on top of that it make some prints of information very tricky or inconsistent.

Division of work

Jakub: Main thread idea for working in parallel assigning farmers certain animal types working on certain fields, algorithm for buyers and farmers.
Josh: Main functionality, template for the simulation, Print statement management, Custom Parameters.
Shared: Formatting, debugging, implementation ideas.

Description

Our solution provided is heavily influenced by parallelism when it comes to our farmers, assigning each farmer (Thread) to do a certain task to its dedicated Field (resource). A buyer can buy any animal from any field as long as the field is not being stocked by a farmer or if it is not empty The fields are our main critical sections managed by monitors to prevent race conditions among the farmers and buyers relying on wait(), notify() and notifyAll() inside of Java. If a farmer is not on a break and is trying to stock a field that is currently full the farmer will wait at that field until buyers have bought all the stock. If a buyer tries to buy an animal that is out of stock he waits until it becomes available only then can he try to buy different animals which are randomly selected. Our enclosure is also a critical section which is managed by monitors as it should not be accessed by more than one farmer at a time.

Miscellaneous

Sample statistics gathered from default parameters to show fairness

Fig1. Equal share of breaks and resources can be seen allocated between farmers