

Link to Repo: <https://github.com/JakeHresh/ser316-spring2021-C-jhreshch-DP/tree/Submission2>

Be sure to look in the Submission2 branch for the implementation.

Link to Travis CI: <https://travis-ci.com/github/JakeHresh/ser316-spring2021-C-jhreshch-DP/builds/223238884>

Link to Screencast:

https://drive.google.com/file/d/1OclWyyVsBLtWxKQqvtdtn7q_2x3vHsW9/view?usp=sharing

This submission completes Version B by using intertwining Decorator, Mediator, and Factory patterns to complete the requirements as outlined in the README.

Specifically, the Decorator Pattern is used to fulfil the three requirements of assigning crop affinities to crops, assigning animal affinities to animals, and assigning special affinities to farmers. The classes responsible for the fulfilling of these requirements include AnimalDecorator, CropDecorator, FarmDecorator, FarmerDecorator, AnimalFarm, AnimalMilkYieldAff, AnimalWoolYieldAff, ConcreteAnimal, ConcreteCrop, ConcreteFarm, ConcreteFarmer, CropFarm, CropYieldAff, FarmerAnimalAff, FarmerCropAff, and FarmerMoneyAff. These classes work together to allow for the dynamic assignment of behavioral modifiers that give different boosts to the revenue generation.

The Mediator Pattern is used to fulfil the three requirements of generating crop revenue once every three cycles, sending predators out to kill livestock on night cycles, and simulating a natural animal life cycle. The classes responsible for the fulfilling of these requirements include ConcreteMediator, Mediator, and the aforementioned decorator classes like AnimalDecorator, CropDecorator,

FarmDecorator, and FarmerDecorator. Also communicating with the mediator is the Predator, which waits for ticks sent by the ConcreteMediator to kill crops and animals on night cycles.

The Factory Pattern is used to fulfil the three requirements of creating different types of farms, creating farmers, and creating animals and crops. The classes directly responsible for the fulfilling of these requirements include Factory and InteractorFactory. The InteractorFactory abstracts the creation of Interactors, which in turn abstracts the different entities that exist within the system, such as animals, crops, farmers, etc. The InteractorFactory creates variants of Interactors by employing the decorator classes to add affinities to entities during creation.

Checkstyle Reports:

CheckStyle Audit

Designed for use with [CheckStyle](#) and [Ant](#)

Summary	
Files	Errors
27	0

Files	
Name	Errors
D:\SER316_DP\ser316-spring2021-C-jhreshch-DP\src\main\java\AbstractDecoratorClasses\AnimalDecorator.java	0
D:\SER316_DP\ser316-spring2021-C-jhreshch-DP\src\main\java\AbstractDecoratorClasses\CropDecorator.java	0
D:\SER316_DP\ser316-spring2021-C-jhreshch-DP\src\main\java\AbstractDecoratorClasses\FarmDecorator.java	0
D:\SER316_DP\ser316-spring2021-C-jhreshch-DP\src\main\java\AbstractDecoratorClasses\FarmerDecorator.java	0
D:\SER316_DP\ser316-spring2021-C-jhreshch-DP\src\main\java\AbstractDecoratorClasses\AnimalFarm.java	0
D:\SER316_DP\ser316-spring2021-C-jhreshch-DP\src\main\java\AbstractDecoratorClasses\AnimalMilkYieldAff.java	0
D:\SER316_DP\ser316-spring2021-C-jhreshch-DP\src\main\java\AbstractDecoratorClasses\AnimalWoolYieldAff.java	0
D:\SER316_DP\ser316-spring2021-C-jhreshch-DP\src\main\java\ConcreteDecoratorClasses\ConcreteAnimal.java	0
D:\SER316_DP\ser316-spring2021-C-jhreshch-DP\src\main\java\ConcreteDecoratorClasses\ConcreteCrop.java	0
D:\SER316_DP\ser316-spring2021-C-jhreshch-DP\src\main\java\ConcreteDecoratorClasses\ConcreteFarm.java	0
D:\SER316_DP\ser316-spring2021-C-jhreshch-DP\src\main\java\ConcreteDecoratorClasses\ConcreteFarmer.java	0
D:\SER316_DP\ser316-spring2021-C-jhreshch-DP\src\main\java\ConcreteDecoratorClasses\CropFarm.java	0
D:\SER316_DP\ser316-spring2021-C-jhreshch-DP\src\main\java\ConcreteDecoratorClasses\CropYieldAff.java	0

Spotbugs Reports:

Summary

Warning Type	Number
Performance Warnings	4
Dodgy code Warnings	1
Total	5

Warnings

Click on a warning row to see full context information.

Performance Warnings

Code	Warning
UrF	Unread field: main.java.AbstractDecoratorClasses.AnimalDecorator.animalToDecorate
UrF	Unread field: main.java.AbstractDecoratorClasses.CropDecorator.cropToDecorate
UrF	Unread field: main.java.AbstractDecoratorClasses.FarmDecorator.farmToDecorate
UrF	Unread field: main.java.AbstractDecoratorClasses.FarmerDecorator.farmerToDecorate

Dodgy code Warnings

Code	Warning
IM	Check for oddness that won't work for negative numbers in main.java.AbstractDecoratorClasses.FarmDecorator.updateTick()

I believe there were a few false positives in this report. First, while the fields ending in “ToDecorate” were not directly read, they were important in the implementation of the Decorator design pattern to allow a reference to the decorated objects. Additionally, while there is the use of a percent symbol to check for odd numbers when handling ticks, there is an additional check that the values are not less than 0, rendering the warning unnecessary. Further, the tick values never become negative, so this would never be an issue.

Junit/Jacoco Reports:

Test Summary

27 tests	0 failures	0 ignored	0.109s duration	100% successful
-------------	---------------	--------------	--------------------	--------------------









Packages

Classes

Package	Tests	Failures	Ignored	Duration	Success rate
default-package	27	0	0	0.109s	100%

Generated by [Gradle 6.6.1](#) at Apr 17, 2021, 9:40:14 PM

ser316-spring2021-C-jhreshch-DP

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed Cxty	Missed Lines	Missed Methods	Missed Classes
main.java.FactoryClasses		73%		87%	25 59	35 119	22 43	0 7
main.java.ConcreteDecoratorClasses		74%		37%	16 48	23 103	10 40	0 12
main.java.MediatorClasses		91%		75%	12 47	7 88	1 25	0 2
main.java.AbstractDecoratorClasses		100%		81%	3 24	0 54	0 16	0 4
Total	202 of 1,188	82%	28 of 108	74%	56 178	65 364	33 124	0 25