Assignment 7 - Design Pattern

Version: March 21, 2018

Due: Monday April 2th, 11:59pm

Objectives

- Use design patterns to build the application

General

1. Each Design Pattern below is worth 25
2. You need to create 4 design patterns for the application specified below (3 for task 1, 1 for task 2)
3. The last one (see task 2) must be created using the PatternBox plugin
4. You do NOT:
   a) need to meet all the functional requirements specified in Task 1 but the general idea should be visible
   b) have to have a running/working application (but it should not have compiler errors).
5. We will grade you based on the code that you've written.
6. 25% bonus (12.5 points) if your application is working and at least 2 of the criteria mentioned in the Interface requirements are met (so you can get up to 62.5 points in this assignment)
7. Please indicate (as comments in submission box on BB) which design patterns you implemented and the class file or piece of code where it is implemented.

As stated before you do not have to implement all the requirements and you can choose any 3 design patterns for task 1 that you see fit. The HINTS might help you. You are relatively free of how to do things. Please mark where you see your Design Patterns in your code. This is relatively coding extensive again, start early it will take a while to figure things out.

Task 1: Introduce Design Patterns (37.5 points, 12.5 points for each design pattern)

Create a small application that utilizes some simple Design Patterns into the code. Below are some of the functional requirements for the application.

- An apiary can have many beehives in them. For this assignment, only ever allow one apiary to exist.

- An apiary should be capable of spawning a logically unlimited number of beehives.
- A beehive should be comprised of a network of hives or rooms and a beehive can't contain another beehive.
- A beehive should have hives for spawning more bees.
- Bees should be able to battle other bees (unconventional). When they do, the loser of the battle should die (it's beehives can no longer command it), and the winner of the battle should get the attributes of the bee.
- A beehive should only have one species of bees. Each species should have some kind of bonus to them. Perhaps they harvest nectar from flowers faster. Perhaps they have a higher chance to kill other bees (or anything else).
- If a member of a beehive kills the queen of another beehive, the killer's queen should assume control of the dead queen's population of bees. all bees should now also have the attributes of both species of bees. (So, if one species was strong, and the other species was efficient, now both populations are merged into a single population that is both strong and efficient.)
- Bees should have to rest every so often. When they do, they should consume food. A beehive should be limited in capacity of how many bees can rest at a time based on the number of hives (X amount per resting hive).
- The simulation should end when, at the end of a tick, there is 1 or less active beehives/queens. This means you should spawn at least 2 beehives before your first tick.
- Rooms in the beehives should require a certain (probably large) number of worker-ticks to build. So, if it would take 1 worker 100 ticks to dig a room, it would take 50 workers 2 ticks, etc. But there should be a significant cost to building a room, since rooms determine how much you can rest.

You can be creative on how you define the different bee types and their "special" ability etc. It is more about using the Design Patterns and setting them in contect.

Some HINTS:

- The apiary class MUST be a singleton.
- A beehive should be built by building rooms into a beehive. (Builder pattern)
- Use the decorator pattern to keep track of a bee's attributes.
- Beehives should be templated (using Java generics) to hold any type of bee, and the beehives should specify at runtime which type of bee it holds.
- The simulation should be tick-based. (Mediator pattern) Each tick, every bee (in a random order) should perform some action based on its surroundings. Drones should look for food, warriors should hunt enemies (or go back for food if hungry) and queens should spawn an egg.

Task 2: Create a new Feature using PatternBox (12.5 points)

The PatternBox plugin (patternbox.com) provides a design wizard for 16 of the Go4 pattern. Download this plugin into Eclipse (it is on the Marketplace, and a PDF of instructions are on their website) and use it to implement a new feature of your choosing (to the application from task 1) that uses one or more patterns (only 1 is required).

The PB plugin will generate an .xdp file that should be in your submission. It will generate code for you but you may have to enhance it. Put comment(s) in your code "SER316 PATTERNBOX" where you have written the code according to the PB plugin.

Extra Credit (12.5 points)

Make your application work, so that we can run it and fulfill at least 2 of the interface requirements from below. You can basically again decide what exactly you want to do and how to exactly implement it. Interface requirements The interface to the apiary should be a command line interface with the following commands:

- spawn X Y T - should create a new beehive at position X,Y of species T. The output should give an identifier for the beehive so it can be controlled later.
- Example spawn 14 -32 Killer creates a beehive of "Killer" bees at position x-14 y-32.
- give I R A - should give the beehive identified by I (at creation) resource R of amount A.
- Example give 1 food 60 gives beehive 1 60 food.
- Example give 3 warrior 20 gives beehive 3 20 warriors that spawn at beehive 3's base.
- tick [T] - should perform T tick operations. For convenience, allow T to not be specified, and simply tick once.

  Example tick 10

  Example tick
- summary I - should give a summary of beehive I. A summary gives information about that beehive.
- Example summary 1 might give the following output:

  Output:

  Species: Killer

  Workers: 14

  Warriors: 10

  Bee Kills: 18

  beehive kills: 2 (2:Pansy 4:Gatherer)

Ticks alive: 143

Status: Alive

- Example summary 2 might give the following output:

  Output:

  Species: Pansy

  Workers: 4

  Warriors: 0

  Bee kills: 2

  beehive kills: 0

  Ticks Alive: 25

  Status: Killed by 1:Killer

If you do this part then please include it into your code and only submit one zip and give us the information of what you think you did achieve from this extra credit part in the Blackboard submission comment.

Submission

You need to submit

1. a zip file assignDP.<asurite>.zip. Make sure you commented your code well and marked where you see your Design Patterns.
2. the .xdp file from task 2

Going Forward:

I do not expect you to include Design Patterns in your code, since I think it will mean major refactoring. I do however require you to discuss in your group where you think (theoreticly) using Design Pattern would make sense. Document on your Wiki as "Design Pattern page" where you think you could use Design Patterns, which one and explain why.

Also check if you already see some Design Patterns used in Memoranda, either because it was already there or because you introduced. it.