My design patterns:

1. Singleton - the apiary is a singleton. Only one instance of the object is every allowed to be created and any other variables of apiary will all have the same reference.

2. Flyweight - the bees use a flyweight DP for their baseStats. That way a hive can have many many many bees, but all bees of a certain type will share the same base stats. This allows for one flyweight bee of each type to be created--and all bees of that type reference the same object instead of creating hundreds, possibly thousands of base stat objects we keep bees simple and reduce impact on memory.

3. Factory - the hives use a factory to create them. A single AbstractHive class is called in the factory with different enum Types to create unique Hives with different types of Bees. The factory uses a method instead of constructor calls to create whichever hive type is desired.

4. Mediator - the mediator communicates between hives and the Apiary by relaying tick counts to each different hive. It also communicates from the game space aka the user to the hives/apiary when playing.

5. Bonus DP - Adapter. I used an adapter dp both in the mediator and in the game interface. The adapters allow for multiple different messages to all be passed into the same method where separate methods are then called. For example: in the mediator from a single method, based on the parameters given, you can call makeRooms, makeBees, makeHarvesters, and toString for each different hive.

To run the main demo you can use gradle run.

Alternatively you can run gradle **FatJar** and then from:

    build > libs
    Run:
    java -jar assignDP.mcole18.jar

*side note: my codeCoverage is just barely over 60% without running main because the extra credit and demos are all in the jacocoTest.

From here you will see a printout that demonstrates each of the 4 main design patterns functionality.

To run the extraCredit you can run the jar in the additional zip [the code is included in the main assignment zip]

Alternatively you can run gradle **ExtraFatJar** and then from:

build > libs
Run:
java -jar extraDP.mcole18.jar

Once it is running type “QUIT” to end the simulation or create one hive to rule them all to end the game naturally.
Follow the horribly formatted prompts to create new hives and fight. Enjoy attempting to input the exact right values to make things happen and have fun!

For a quick route to victory enter the following commands in order when prompted:

```
Spawn 1 1 k
Spawn 1 1 b
Attack 1 2 10
```

^^clearly the game is not yet balanced! xD