**AntFarm**

**1. Factory Pattern:**

Based on whether meadow spawn antfarms or antfarms spawns ant the interface spawn will give appropriate spawn method definition based on which subclasses are implemented.

**2.Decorator:-** Now battle can be between two ants or between ant and queen. So at runtime objects are assigned different properties. And battleBetween is the decorator which does run time task of allocating properties depending on who is winner.

**3. Mediator:-** Now for each tick ant should do some task based on surrounding. So here tick is a method used in intermediator class. Now antFarm has provides functionality to rest() or doWork().

Each ant which is the object of antFarm class have to interact with eachother and do task at each tick and also there is limitation on no. Of ants that can rest at a time so here for this communication problem a intemediator class is introduced. Hence mediator design pattern is implemented.

Public class intermediator{

public static void tick(){

-------

}

}

public class antFarm{

public void dotask()

{

intermediator.tick()

}

public void rest()

{

intermediator.tick()

}

}

//below class uses antFarm objects to show communication between

public class antFarm1{

antFarm ant1= new antFarm(ant1);

antFarm ant2= new antFarm(ant2);

ant1.doTask();

ant2.doTask();

}

**4.Singleton:-** Here as mentioned deal with one meadow, we create a global single instance of meadow through which we access.