Sobre implementar funcionalidad

-Los tests fueron pasando de a uno, primero el 1 después el 1 y el 2, y después el 1 2 y 3. En un principio hacerlo de esta forma es bueno para empezar a entender cómo funciona el problema, ir de a poco y no adelantarse. Una vez se tiene una idea del problema agilizamos el funcionamiento de las nuevas funciones porque pudimos predecir que tenían que hacer sin leer los tests, las 4 últimas pruebas pasaron a la vez.

-Sobre código repetido

Los únicos métodos con código repetido son los de agregarDosPolillasMas y agregarDosOrugasMas que simplemente llaman a la función “agregar polilla/oruga” 2 veces. Se puede hacer con un ciclo pero no lo creímos necesario para 2 repeticiones.

En un principio, los insectos eran los responsables de evaluar las condiciones y poner un huevo, pero esto llevaba a la existencia de muchos “Getters” y usábamos “Setters” para agregar el huevo, finalmente nos decidimos por que el hábitat haga el chequeo porque tiene más sentido que el hábitat sepa si se dan las condiciones o no, también logramos borrar varios métodos de ElHabitat.

-Sobre código repetido 2

Utilizamos el paradigma de prototipado ya que no había ningún mensaje al cual todas las avispas respondieran igual, pero si había uno que Ornela respondía igual que Oriana, así que Oriana quedó como “Padre” del resto. Usamos 3 colaboradores internos que funcionaban como contadores para los huevos de c/avispa ya que eran pocas, no utilizamos diccionarios ni colecciones.

Al principio mientras hacíamos que los tests pasen de a uno, la resolución del test 8 no coincidia con el título, ya que Ornella no mataba a todas las orugas, solo a una y sin embargo pasaba el test. Mientras fuimos avanzando, los tests pasaban de a varios, por lo que es posible que haya una solución más sencilla en alguno de los últimos tests, donde la prueba pase sin cumplir los requisitos del título, pero en general creo que nos quedó bastante simple.