Ejercicio 01 - Avispas y Abejas

Nombre y apellido de los integrantes:
Manuel Sanchez Fernandez la Vega (107951)
Ian Shih (108349)

Fecha de entrega: 25/04/2022

Ian Shih y Manuel Sanchez Fernandez de la Vega

Sobre implementar funcionalidad

Los tests 01, 02 y 03 demuestran la funcionalidad de cómo se incrementa la cantidad de huevos de avispas a medida que los van dejando. Cuando los implementaste, ¿esos tests pasaron (los tres) de una? ¿podrías haber implementado esta funcionalidad de a partes, haciendo que pase el 01, luego el 01 y el 02 y por último el 01, 02 y 03? ¿se te ocurre cómo? Y si lograste hacerlo, ¿qué pensas de implementar esa funcionalidad de esa forma?

Como vimos en clase, los tests 01, 02 y 03 los implementamos de a partes. Primero logramos que pase el test 01, y cuando intentamos hacer pasar el test 02, empezó a fallar el primero. En consecuencia, debimos buscar una solución más general que logre pasar todos los tests en vez de “hardcodear” para que pase cada uno individualmente. En nuestra opinión, puede llegar a ser útil en problemas más complejos. Al ser este, un problema más bien sencillo, la implementación deseada era relativamente fácil de comprender desde un principio. Pero logramos ver como esta filosofía y forma de tomar los desafíos en el desarrollo puede llegar a ser útil.

Sobre código repetido

¿Les quedó código repetido? ¿Dónde? ¿Se animan a adivinar qué cosa del dominio les faltó representar (y por eso tienen código repetido)? Responsabilidad de dejar un huevo consumiendo otro insecto ¿Quién les quedó, en su modelo, que es el responsable de ver si hay suficientes polillas u orugas y entonces dejar un huevo? ¿el insecto (Polly, Oriana, etc) o el hábitat? ¿por qué? ¿por qué tendría sentido que fuera de la otra forma? ¿con cuál nos quedamos?

Nos quedó código repetido en la categoría de recursos del hábitat, ya que nos faltó representar a un recurso como idea general, ya que podrían llegar a haber más recursos en un hábitat que se comportan de forma similar. Este problema se extendió en las avispas. el código repetido entre Ornella y Oriana lo resolvimos creando a Ornella como hija de Oriana, sin embargo, no podíamos hacer lo mismo con Polly y Lara ya que su comportamiento es distinto (al consumir otro recurso) por lo que tuvimos que cambiar el método de para intentar reproducirse.

En nuestro modelo, el responsable para ver si hay suficientes polillas u orugas y dejar un huevo es la avispa, quien le envía un mensaje al hábitat diciéndole que puso un huevo. El hábitat en sí es quien mantiene un registro de cuántos huevos hay puestos y a que firma genética corresponden. Para nosotros no tiene sentido que fuera de la otra forma ya que la que se reproduce es la avispa, no el hábitat (esto es análogo a que un auto sea capaz de guardarse como JSON) por lo que quien se debería encargar de verificar las condiciones para su propia reproducción, es la avispa. En conclusión, nos quedamos con que la avispa es la encargada de verificar cuántos recursos hay disponibles en el hábitat para reproducirse.

Ian Shih y Manuel Sanchez Fernandez de la Vega

Sobre código repetido 2

Con lo que vimos en la clase del Jueves (en la parte teórica, prototipos vs clases) ¿cómo sacarían este código? Sobre la implementación ¿cómo resolvieron guardar los huevos? ¿Usaron colecciones? ¿Diccionarios? ¿Uno, varios? ¿con qué indexaban? Pero la pregunta más importante: ¿es lo más sencillo que hacía falta? ¿o se podía hacer menos y todo andaba?

Sacariamos este código utilizando diccionarios para representar los recursos y además haciendo que las avispas sean hijas de un objeto que represente la idea de una avispa, el cual tenga los métodos principales que puede llegar a realizar una avispa, como por ejemplo, reproducirse.

Para guardar los huevos, utilizamos un diccionario que contiene como claves las firmas genéticas de las avispas y como valor, la cantidad de huevos con esa firma.

En cuanto a otras opciones, también se podría haber implementado haciendo que el hábitat tenga un colaborador por cada firma genética. Sin embargo, nosotros creemos que esto sería mucho más difícil de comprender, y es más sencillo de manejar con un diccionario que agrupe todos los huevos de distintas firmas genéticas. Lo único que se nos complicó un poco es la implementación del diccionario en Smalltalk, al carecer de experiencia previa. No obstante, no es un problema de diseño sino que es una falta de conocimiento técnico del lenguaje.