FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico 2 -Java
AlgoCraft

[7507/9502] Algoritmos y Programación III
Curso 2
Primer cuatrimestre de 2019

ALUMNO	PADRON
Bisso, Nicolás	101735
Figueroa, Matías Ariel	92498
Haberkon, Brenda Micaela	103156
Jodara, Sabrina Lorena	83826

Índice

1. Introducción

El presente informe reúne la documentación de la solución de la primera entrega del segundo trabajo práctico de la materia Algoritmos y Programación III que consiste en desarrollar una aplicación de manera grupal aplicando todos los conceptos vistos en el curso, utilizando un lenguaje de tipado estático (Java) con un diseño del modelo orientado a objetos y trabajando con las técnicas de TDD e Integración Continua.

La consigna general consiste en desarrollar la aplicación completa, incluyendo el modelo de clases e interfaz gráfica. La aplicación deberá ser acompañada por pruebas unitarias e integrales y documentación de diseño.

En esta primera entrega, se solicita pasar las pruebas unitarias de las clases bases de la aplicación, las cuales se adjuntan en el repositorio de Github y pasan de manera exitosa.

2. Supuestos

La consigna detallada en el enunciado, para las clases base, al ser muy clara no da a lugar a muchos supuestos. Por este motivo el único supuesto que tuvimos en cuenta es para el Pico de Piedra que además de desgastar al metal, también desgasta la piedra. Están las pruebas unitarias que prueban los supuestos correspondientes.

3. Diagramas de clase

En este diagrama se muestra la interacción general de la herramienta con su golpe, el desgaste que tiene y las dependencias.

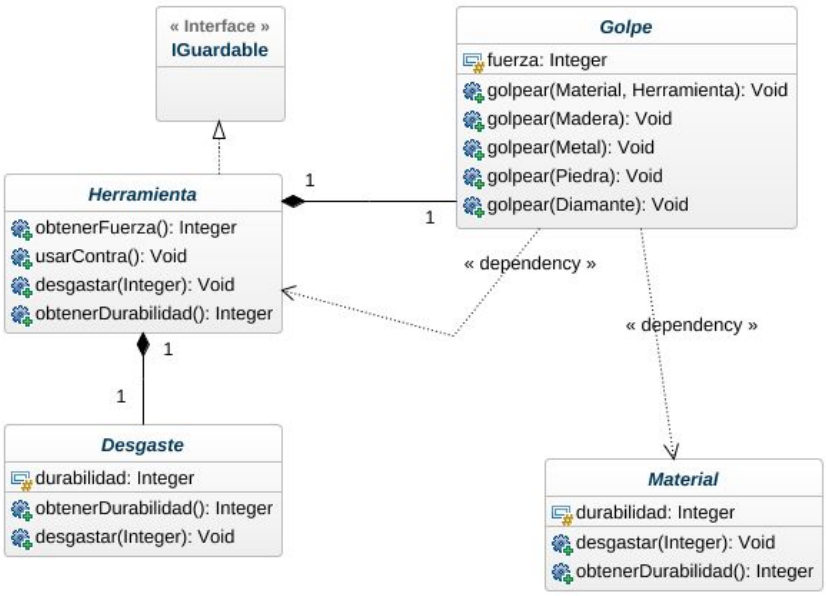

Figura 1: Diagrama de Clase Herramienta general

En este otro diagrama podemos ver cómo el golpe implementa por cada tipo de herramienta un golpe en específico, esto lo vuelve extensible lo que permite, en caso de que se requiera, hacer más herramientas, y poder ir redefiniendo el comportamiento del golpe. Por ejemplo, un hacha por ahora va a tener un único golpe (golpeHacha) pero el pico en cambio tendrá dos golpes, uno es el GolpePico el cual sólo afecta a la piedra y uno que es GolpePicoPiedra que al ser un pico debe picar piedra y al ser de piedra también afecta al metal.

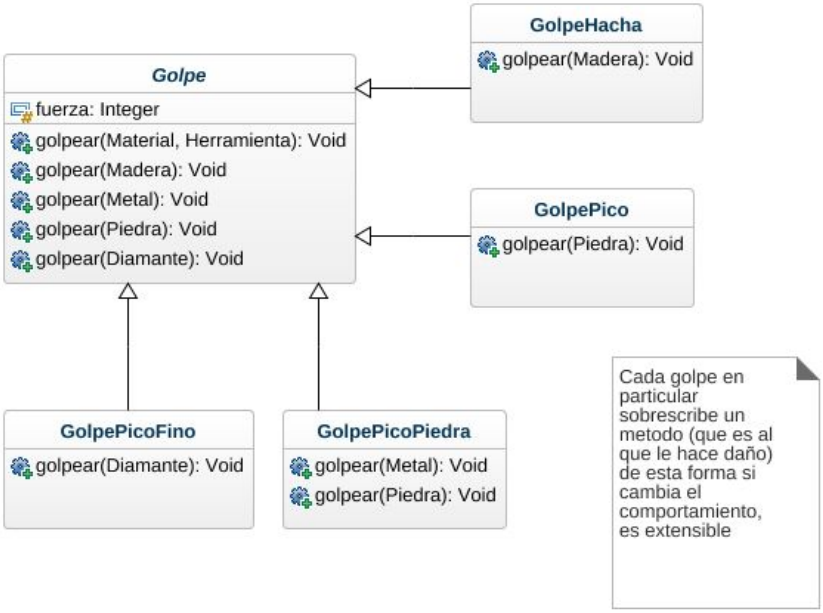

Figura 2: Diagrama de los golpes.

4. Diagramas de secuencia

Un diagrama importante es el comportamiento del golpe según la herramienta y material que golpee. Siempre la herramienta se termina desgastando, pero puede que el material lo haga o no, según el golpe de la herramienta.

En este caso, un pico de metal se usa contra el metal y no le hace daño (en base a los supuestos) podemos ver cómo se usa el patrón de diseño 'Visitor' y cómo no termina de concretar el daño a causa de que no está override el método que debería tener ese comportamiento.

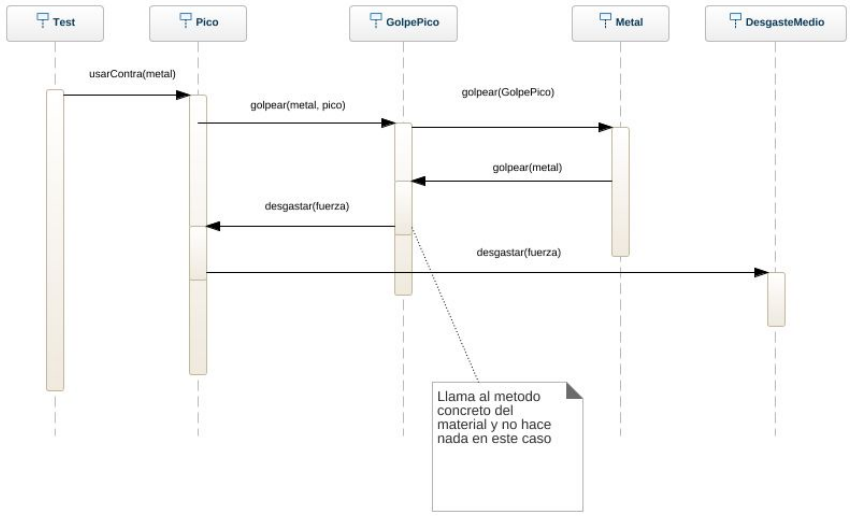

Figura 3: picoDeMetalChochaContraMetalYNolerestaPuntosDeDurabilidad

En cambio si el Pico es de piedra y le pega a un metal, en el caso de que consiga concretar el daño, al estar override, llama al material para que se haga daño y cierra el ciclo.

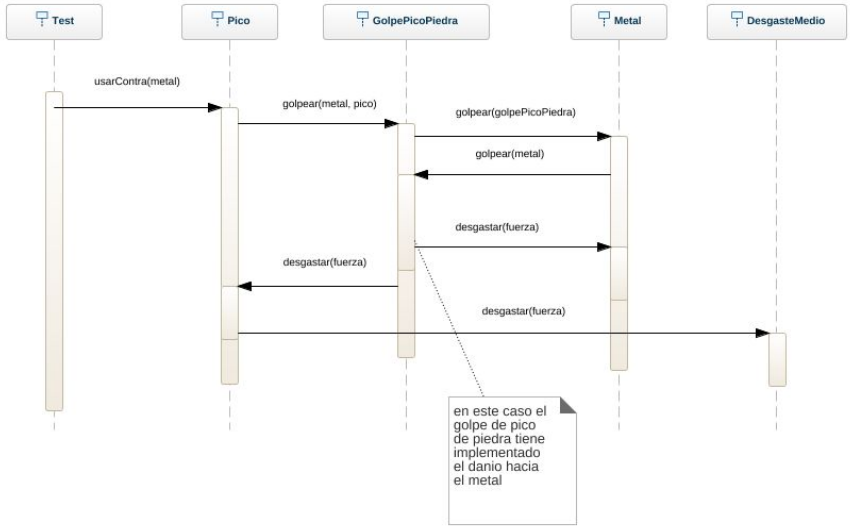

Figura 4: picoDePiedraChochaContraMetalYLeDesgasta4PuntosDeDurabilidad

5. Detalles de implementación

Al leer el enunciado del trabajo práctico se pudo observar que había objetos que poseían un mismo comportamiento y estado. Estos objetos decidimos instanciarlos en clases, agrupándolos en clases abstractas e interfaces.

Luego, para la implementación de la construcción de las herramientas nos basamos en el patrón de diseño Factory en el cual se genera una clase constructora para que pueda crear las clases concretas de la clase abstracta Herramienta. De esta manera se delega la resposabilidad a la clase constructora y asípueda crear diferentes tipos de herramientas según su material que lo compone, con su respectivo tipo de golpe, y tipo de desgaste.

En cuanto a la implementación de cómo se relacionarían estas clases decidimos utilizar el patrón Visitor debido a que se desean realizar operaciones que dependen de las clases concretas de las interfaces implementadas.

Todas estas implementaciones fueron evaluadas en forma grupal como para lograr que cada clase cumpla con el 'Sigle Responbility' ya que solo tienen una sola responsabilidad y que la aplicación pueda ser extensible lo que cumpliría con el 'Open-closed' de los principios SOLID.