Trabajo Práctico 2 — AlgoCraft

[7507/9502] Algoritmos y Programación III
Curso 1
Primer cuatrimestre de 2019

Alumnos: PALAZON, ALDREY, PAGURA, FRITZ
Número de padrón: 102679, 102731, 102649, 102320
Email: aajoaco@gmail.com, sebastianpagura@gmail.com, mpalazon3@gmail.com,
lautarofritz2@gmail.com

Índice

1. Introducción

El presente informe reune la documentación de la solución de AlgoCraft, trabajo práctico de la materia Algoritmos y Programación III, que consiste en desarrollar una aplicación de un juego parecido al Minecraft en Java utilizando los conceptos del paradigma de la orientación a objetos vistos hasta ahora en el curso.

2. Supuestos

Ya que la consigna no especificaba el comportamiento de las herramientas en algunos casos particulares como usar el hacha contra piedra, o usar el pico contra madera, decidimos que en estos casos que la herramienta no es capzaz de romper el material, la herramienta termina siendo desgastada, pero el material no. Otro supuesto que tuvimos fue que el personaje tenga una herramienta de su inventario equipada, ya que en el caso de tener mas de una herramienta, se debe especificar de alguna manera, cual es la herramienta que el personaje va a usar. El tamaño del mapa y la disposición de materiales tampoco estaba definido por la consigna, por lo tanto decidimos hacer un mapa de 8x8 casilleros, y le insertamos materiales a gusto.

3. Diagramas de clase

Los diagramas de clases muestran todas las relaciones de nuestro modelo implementadas hasta el momento.

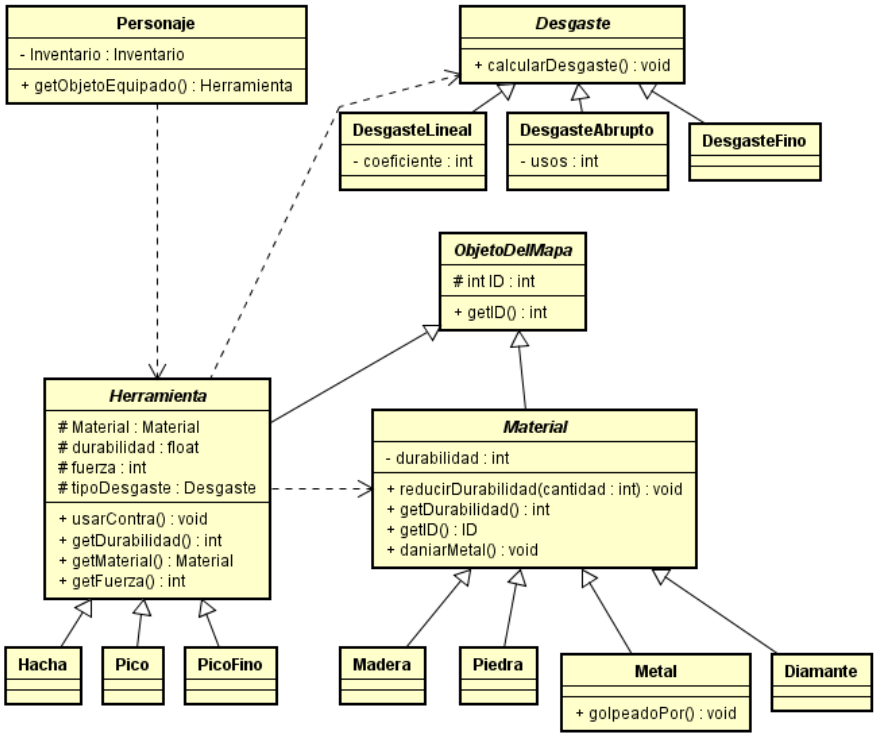


Diagrama de las Herramientas y los Materiales.

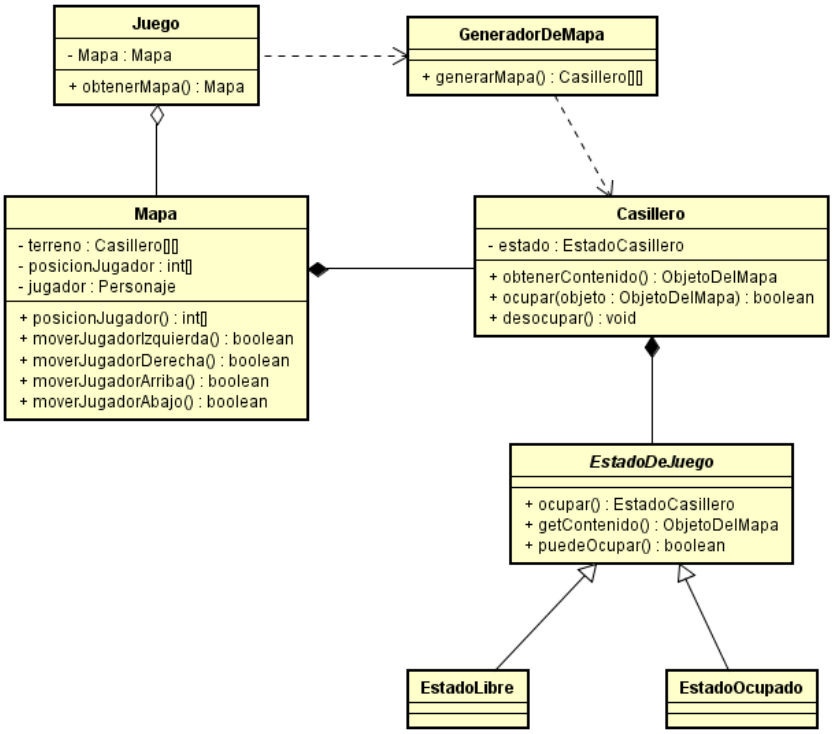


Diagrama del Mapa y Casilleros.

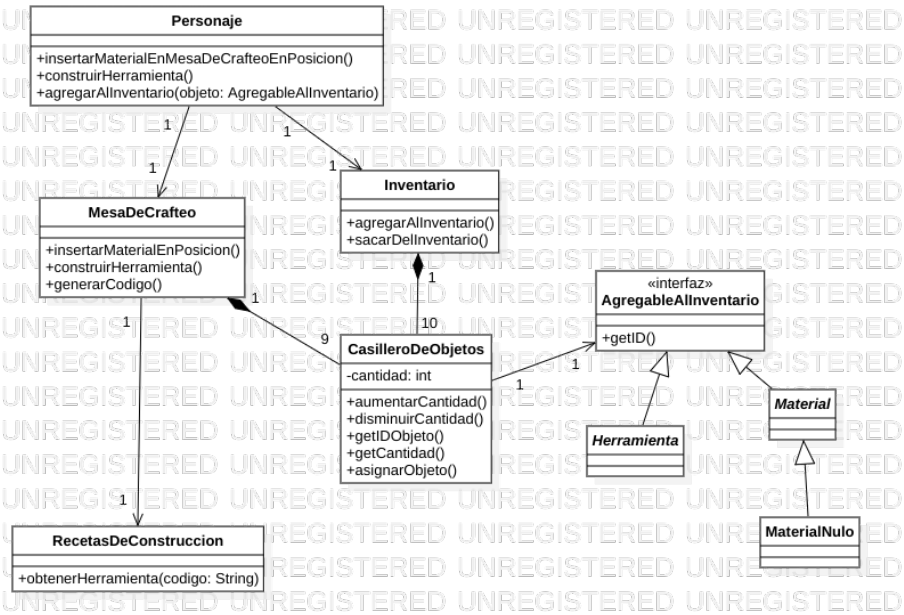


Diagrama de Crafteo e Inventario.

4. Detalles de implementación

4.1. Desgaste de las Herramientas

Debido a que el desgaste de las herramientas dependia no solo del tipo de herramienta, sino que tambien del material de dicha herramienta, decidimos que la herramienta tenga un atributo tipoDesgaste, una instancia de la clase Desgaste, de la cual heredan distintas clases representando las distintas maneras en las que las herrmaientas se pueden desgastar. El tipo de desgaste lo recibe la herramienta de parte de su material, al ser inicializado. De esta manera la herramienta delega la responsabiidad de caluclar su desgaste a su atributo tipoDesgaste. En este caso, estamos usando el patron Strategy, ya que estamos delegando el comportamiento de la clase herramienta, a la clase Desgaste.

4.2. usarContra

Otro punto importante en nuestra implementacion fue como controlar el uso de las herramientas evitando condicionales. Para ello, decidimos sobrecargar el metodo usarContra de cada herramienta, lo que nos permitio que cada herramienta tenga una manera distinta de usarse contra cada material. Esto nos permitio lograr implementar el uso de la herramienta con polimorfismo.

4.3. CasilleroDeObjetos

Tanto la clase Inventario como MesaDeCrafteo están compuestas por instancias de la clase CasilleroDeObjetos. Cada instancia almacena el objeto que contiene en el momento y la cantidad del mismo. Por defecto, el contenido del atributo objeto es una instancia de la clase MaterialNulo, que hereda de Material, pero no se puede construir herramientas con él y tiene un ID de 0. Los objetos almacenados pueden ser herramientas o materiales, ya que estos implementan la interfaz AgregableAInventario.

4.4. Construcción de las Herramientas

La construcción de las herramientas está a cargo de dos clases: MesaDeCrafteo, la cual contiene una matriz de CasillerosDeObjeto de 3x3 en la cual se insertan los materiales, y RecetasDeConstruccion, que posee un diccionario en el cual figuran los códigos como clave y la respectiva herramienta que generan como valor. El proceso es el siguiente:

- Se insertan los materiales en la posición deseada de la tabla. Una vez terminado, se llama al método generarCodigo(). El método concatena los ID de los materiales insertados en la matriz y pasa el resultado a String.
- El código es luego pasado a la instancia de la clase RecetasDeConstruccion, quien compara el código recibido con los que posee en el diccionario. Si encuentra una coincidencia retorna la herramienta correspondiente, caso contrario, dispara una excepción.

4.5. Inventario

La clase Inventario posee un vector de 10 posiciones de CasillerosDeObjeto. Al agregar un objeto, se fija primero si no hay alguna instancia de la misma clase en el inventario. Si esto ocurre, no lo agrega, sino que incrementa en uno la cantidad de dicho objeto. Si no está, lo agrega. La quita de un elemento es similar: si la cantidad es mayor a uno disminuye la cantidad en uno, caso contrario, elimina el objeto en cuestión.

4.6. Casillero

Para implementar la dinámica de ocupación y desocupación de los casilleros del Mapa, utilizamos el patrón State. De esta manera, el estado del casillero y su comportamiento se define de manera polimórfica y dinámica. El casillero tiene un atributo: estado, que es del tipo EstadoCasillero, este puede ser EstadoOcupado o EstadoLibre, el casillero delega la responsabilidad de manejar su ocupación a este atributo.

4.7. Juego

Juego es la clase encargada de manejar los objetos a gran escala del juego, por el momento lo único que hace es crear y usar el mapa para la correcta inicializacion de el mismo.

4.8. Controlador del Mapa

Para quitarle un poco de responsabilidad al mapa, de manera que funcione como un simple display de posiciones, decidimos implementar un controlador del mapa, este se encarga de mover el jugador a traves del mapa y calcular su próxima posición.

5. Excepciones

A esta altura del desarrollo, tenemos las siguientes excepciones:

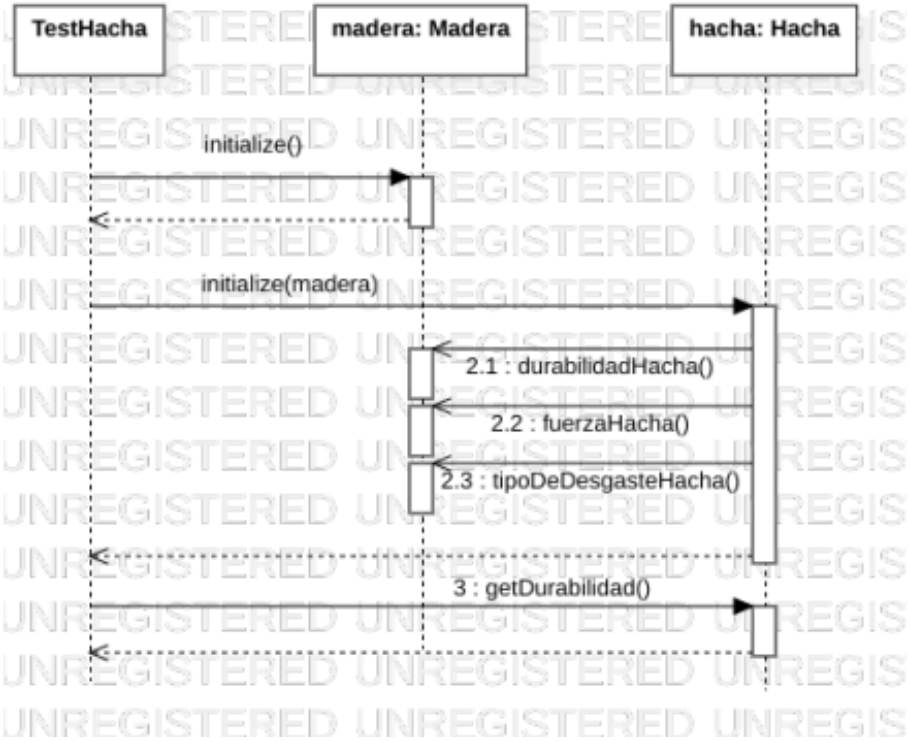
NoHayContenidoExcepcion Esta excepción aparece al querer ver el contenido de un casillero no ocupado.

NoSePuedeOcuparExcepcion Esta excepción aparece al querer ocupar un casillero ya ocupado.

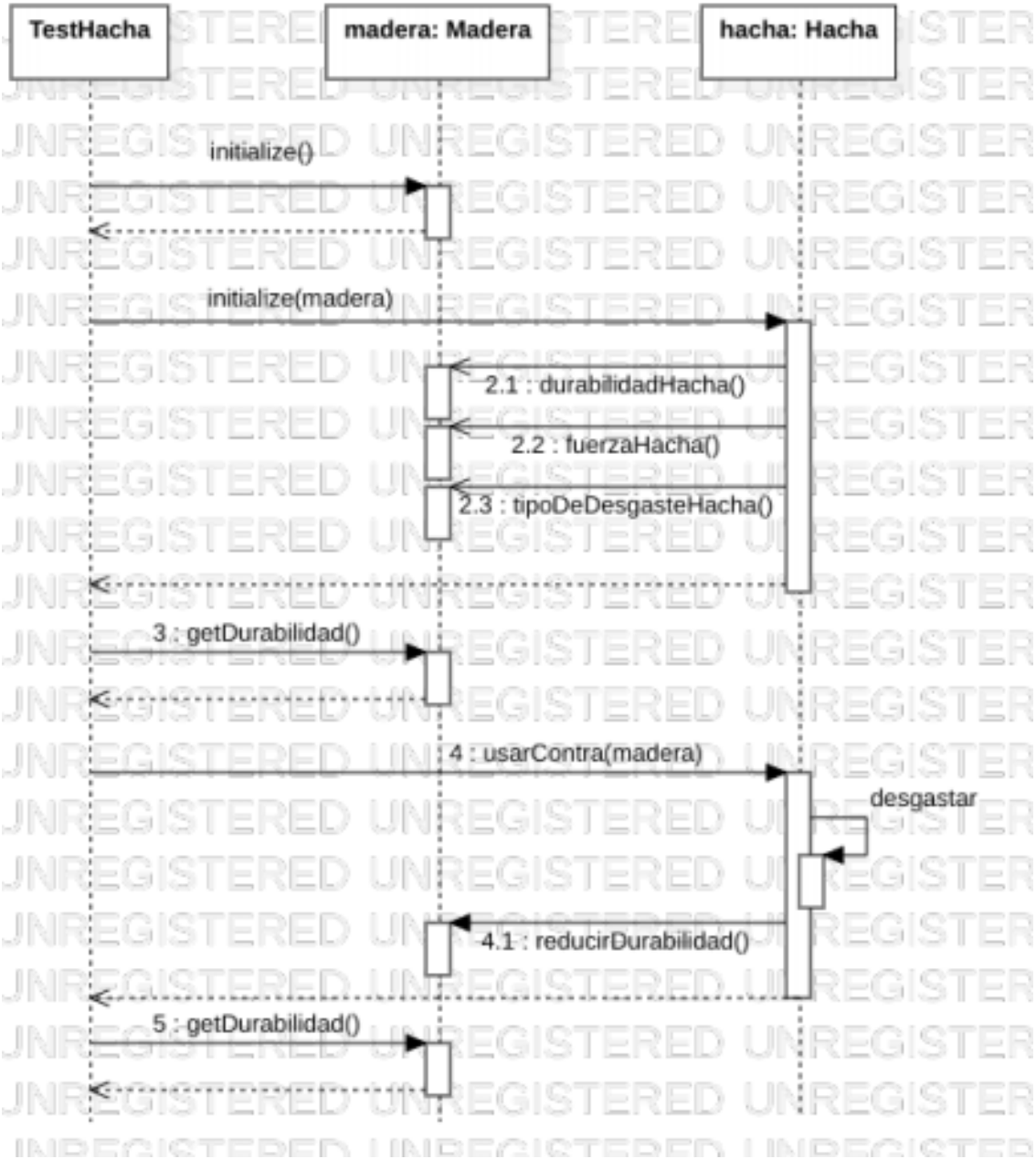
CodigoDeHerramientaInvalidoError Esta excepción se dispara cuando el usuario trata de construir una herramienta pero inserta los materiales en alguna forma no reconocida por la aplicación.

6. Diagramas de secuencia

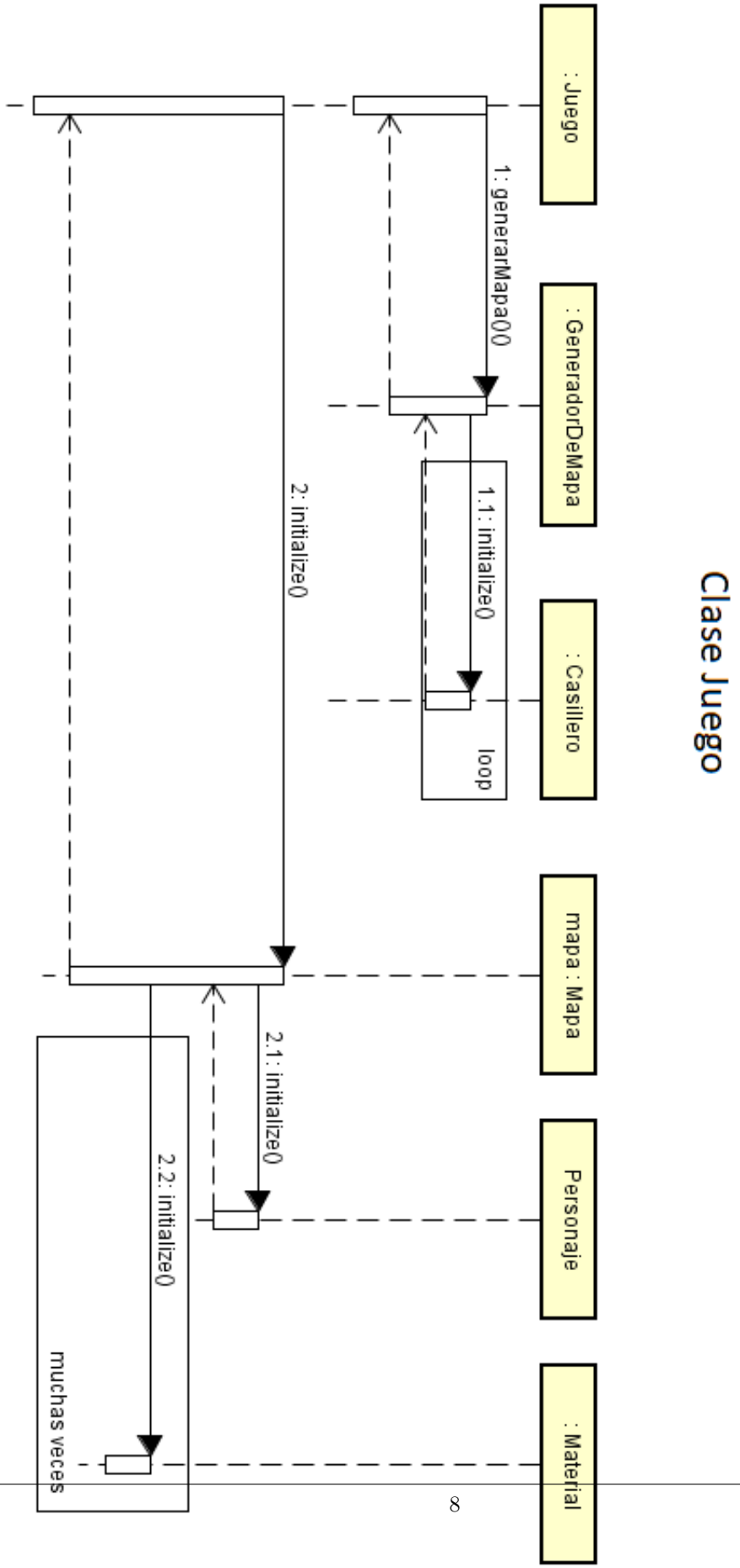
Estos diagramas muestran como se crea un hacha, como se usa contra el material madera y cómo se crea el mapa con la clase Juego.



TestHachaDeMaderaSeCreaConDurabilidadCorrespondiente



TestMaderaGolpeadaPorHachaDeMaderaReduceDurabilidad.



Inicializacion de juego.