Trabajo Práctico 2 — Java

[7507/9502] Algoritmos y Programación III
Curso 2
Primer cuatrimestre de 2018

Alumno:	Padron:	Email:
CORREA, Andres	102310	correaandres700@gmail.com
LINEIRA SAAVEDRA, Ignacio	103174	ilineira@hotmail.com
LATORRE, Ignacio	101305	ilatorre@fi.uba.ar
SMITH, Alejandro Nicolas	101730	smithalejandro96@gmail.com

Índice

1. Introducción

El presente informe reune la documentación de la solución del segundo trabajo práctico de la materia Algoritmos y Programación III que consiste en desarrollar una version simplificada del juego Minecraft utilizando los conceptos del paradigma de la programacion orientada a objetos vistos hasta ahora en el curso.

2. Supuestos

Entre los supuestos que tomamos a la hora de desarrollar el tp se encuentran:

- La mesa de crafteo que le permite construir herramientas al jugador, viene incorporada con el mismo.

- Las herramientas pico y pico fino pueden golpear madera pero la herramienta se debilita y el material no se rompe.

-Para crear un pico fino, se espera material piedra o metal, con madera se llama a una excepcion.

-En el enunciado dice "Se puede usar un X-Herramienta", se implemento que cada material puede ser roto exclusivamente por esa herramienta.

-La durabilidad y el uso de una herramienta solo se modificara en el caso en que debilite a el material, asi como en el test de que un picoFino solo reduce su durabilidad con el diamante.

-En el caso del descenso "durabilidad -= fuerza/1,5", se devuelve el numero redondeado en entero, ya que asi luego para su durabilidad no sera por ejemplo "0,333za que no tendria sentido.

-La posicion inicial de los materiales, y la posicion inicial del jugador son vienen por defecto, no depende del usuario.

-El personaje no puede excederse del limite de la pantalla, en caso de querer excederce, se queda en la posicion limite previa.

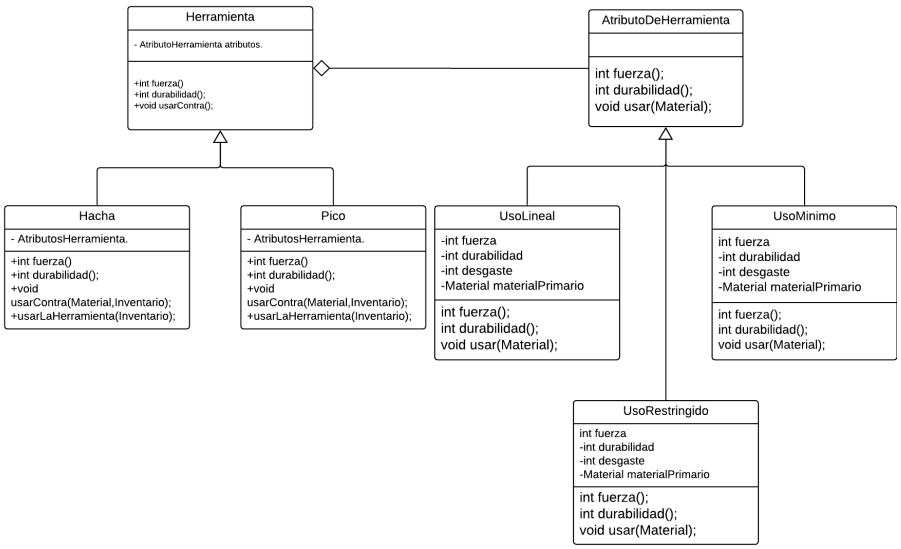
-El personaje tiene una direccion frontal, a donde apunta su vision, en caso de querer romper un material frente a el, debe ubicar su vision contra el, y utilizar su herramienta contra el material.

-Aquellos elementos dropeados en el mapa, en vez de en una zona valida para el elemento como la casilla de construccion, sera perdido.

-Para el desarrollo del juego se utiliza los botones y el mouse.

-Si se crea una herramienta que en este momento tenes en el inventario, esta herramienta renueva sus usos, ahora esta nueva. Es decir, no es valido tener dos herramientas del mismo tipo en el inventario.

3. Diagramas de clase



UML (3).png

Figura 1: Diagrama de clase Herramienta.

El diagrama mostrado, muestra la relacion de herencia entre la clase Herramienta y Hacha,Pico. Esta relacion se dio, por el hecho de que, "Hacha es una herramienta", "Pico es una herramienta". Estas herramientas re implementan los metodos heredados, salvo en fuerza, durabilidad, De esta manera con su estado (atributos), al ser usada va a delegar su uso, etc.

Tambien se muestra a la clase AtributoHerramienta, que es la que contiene los distintos estados de herramientas aplicando el patron state reciben los mismos mensajes, sin embargo, los aplican de manera distinta. Como se puede apreciar en el diagrama, la parte de atributo de herramienta es la que tiene toda la implementacion respecto al tipo de herramienta que es, los materiales que puede destruir, la durabilidad propia, etc.

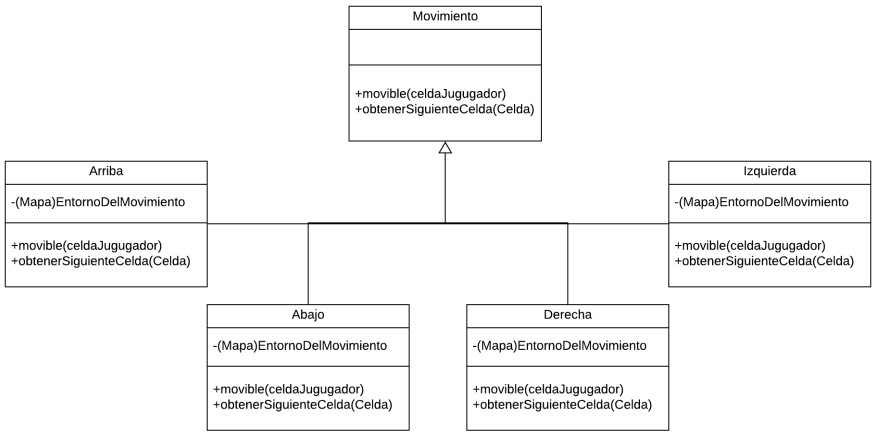


Diagram (1).png

Figura 2: Diagrama de clase Movimiento.

El diagrama mostrado muestra las relaciones en la clase Movimiento, la cual resuelve con Double Dispatch de esta manera:

```
public void moverJugador(Movimiento movimiento){
        movimiento.obtenerEntorno(this);
        this.celdaConJugador=movimiento.movible(this.celdaConJugador);
        this.celdaConJugador.jugador.nuevaPosicionFrontal(movimiento);
}
```

Entonces, el mapa del juego recibe un mensaje de "mover el jugador hacia (movimiento) z, entonces el mapa envia el mensaje al movimiento de moverlo, y me devuelve la nueva posicion. Entonces todos los movimientos interpretan el mismo mensaje "movible ", sin embargo, mueven al personaje para distinto lugar.

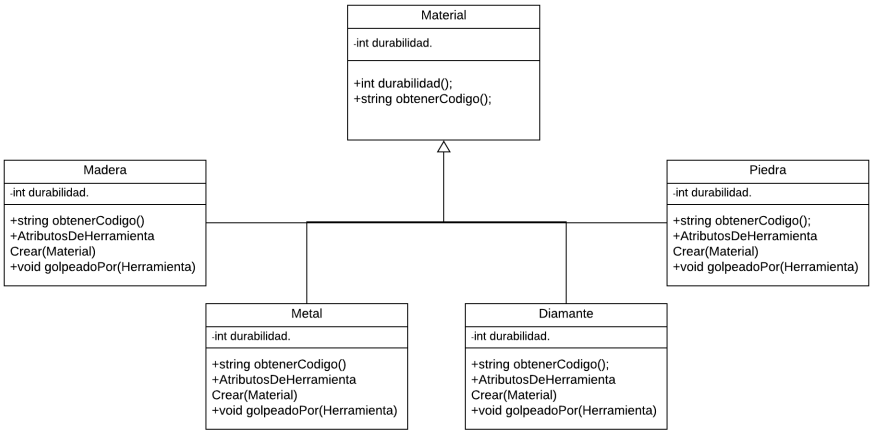
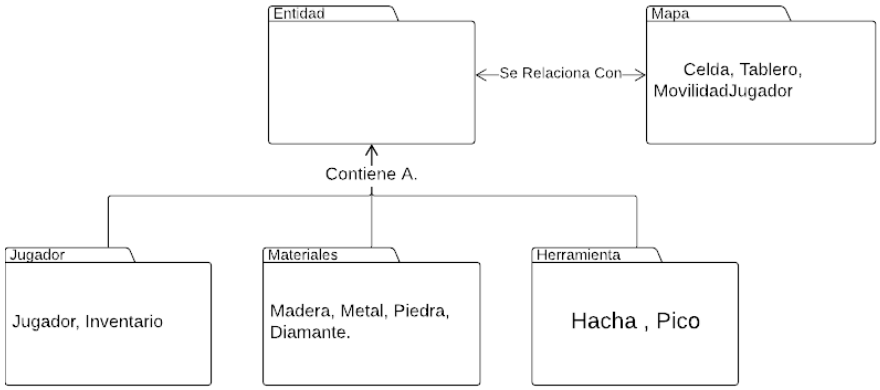


Diagram.png

Figura 3: Diagrama de clase Material.

En el diagrama mostrado se muestran las relaciones entre los materiales que heredan de la clase material, En el diagrama se puede ver como el material crea una herramienta distinta para cada "si mismo ", es decir, el metal recibe el mensaje crear (Hacha), y crea un Hacha de metal, la madera recibe el mismo mensaje y parametro y creara un Hacha de madera.

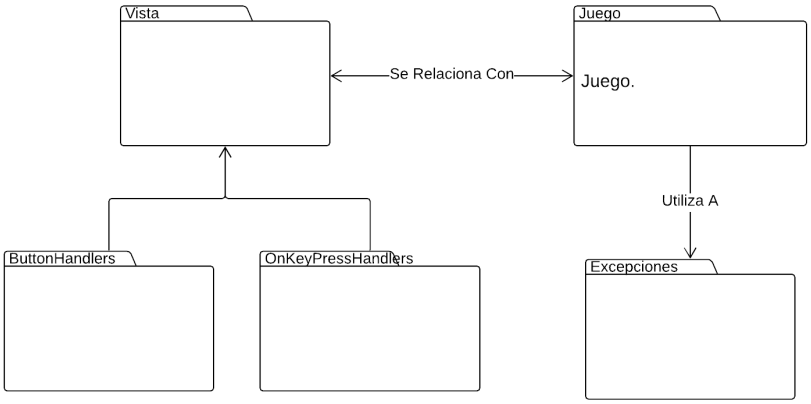
4. Diagramas de paquetes



UML.png

Figura 4: Diagrama de Jugador-Entorno.

El diagrama de paquete presentado, es el mas importante, la relacion presentada en el diagrama es la que da forma a todas las interacciones entre el Jugador(personaje), y su entorno, con esto queremos decir que implica el movimiento del jugador en el mapa, implica la creacion de herramientas, interaccion entre jugador y materiales, asi se logra el uso, sea para romper o para crear, de las mismas.



UML (1).png

Figura 5: Diagrama de paquetes,interaccion grafica.

El diagrama de paquete presentado, muestra la relacion de dependencia entre la interfaz grafica del juego, y el mismo juego. Siendo el juego quien tiene los metodos para mover jugador, iniciar el mapa, etc. La parte visual, interactua constantemente para en manejo correcto de los controladores.

5. Diagramas de secuencia

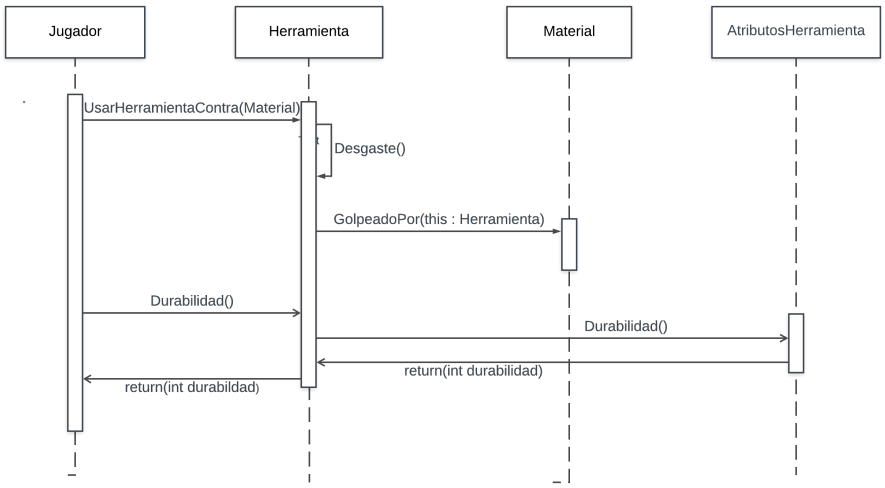


Figura 6: Jugador Usa Herramienta y obtiene su durabilidad

Este diagrama de secuencia muestra la sucesión de métodos mediante los cuales se comprueba que la herramienta, que es genérica,(puede ser Hacha, Pico o PicoFino) tiene la durabilidad correspondiente según los materiales que la componga

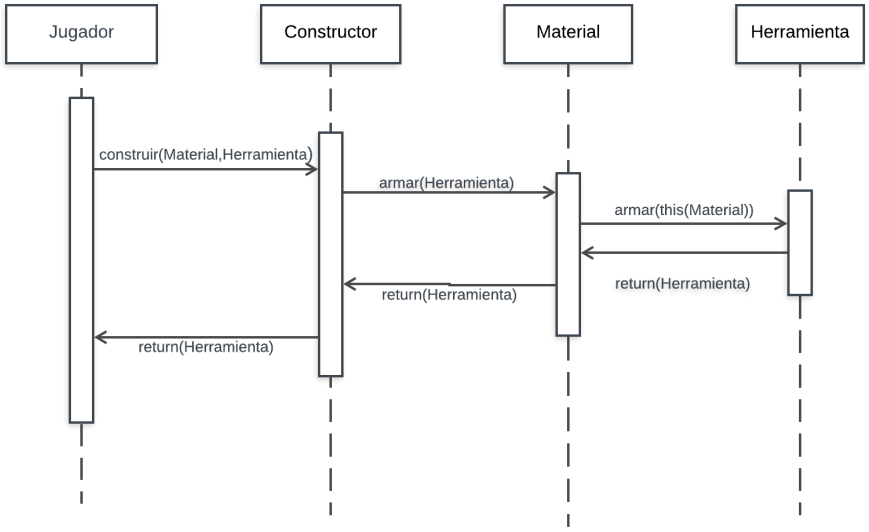


Figura 7: Construccion de Herramienta, solicita datos correctos.

6. Detalles de implementación

Podemos dividir el trabajo practico en diversas entregas, y asi explicar el procedimiento aplicado para resolver partes del todo.

Inicialmente para el uso de herramientas, y para la construccion de cada una de manera correcta, se Aplico Double-Dispatch en la implementacion de construccion, de esta manera cada

material sabe que tipo de herramienta, y que atributos se van a construir. Lo que nos lleva a otra implementacion interesante, que es la aplicacion del patron State, este patron se aplica ya que las herramientas tienen distintos estados, por ejemplo, podriamos explicar el 'Hacha de madera' , 'Hacha de metal' como dos estados de la misma clase Hacha.

Para el mapa del juego, se vio util aplicar el Patron singleton, de esta manera con la instacia de mapa permanente fue mas sencillo manipular las entidades(jugador,material), en el juego. Un problema interesante que tuvimos en la implementacion de test del juego, fue que al utilizar el patron singleton el mapa perduraba para todos los test, lo que violaria un principio de prueba unitaria, para esto se utilizo un metodo que crea un mapa residuo que sirve para test, asimismo se dispuso de metodos para crear mapas sin materiales, o con ellos dependiendo de la necesidad.

Una vez mas otro patron que se rehutilizo fue el de Double-Dispatch para el movimiento del personaje, se considero que el movimiento del jugador debe ser resuelto con polimorfismo y no con cuatro metodos para cada movimiento. De esta manera se creo la interface Movimiento de la cual heredan (Derecha,Izquierda,Arriba,Abajo), estas aplican el movimiento de manera distina para el personaje, sin embargo todas entienden el mensaje "movible ".

La interfaz fue algo que nos costo bastante, se utilizo algo parecido a lo de Pablo como guia y se implementarion Handlers para las teclas del teclado.

Para las divesas implementaciones explicadas, se utilizo herencia en los casos de Hacha,Pico, Que heredan de la clase Herramienta. Se utilizo herencia en los casos de Madera,Metal,Piedra,Diamante, que heredan de Material.

7. Excepciones

Las excepciones son:
Crear una herramienta de diamante: Ninguna herramienta puede ser de este material. Crear pico fino de madera: El pico fino es de metal y piedra solamente. RecetaIntroducidaNoExisteEnElJuegoException; la cual refiere a intentar crear una herramienta inexistente.

PonerMaterialEnCasilleroOcupadoPorOtroTipoDeMaterialException.
SacarMaterialDeCasilleroVacioException
Estas dos, ambas aplican a la construccion de herramientas.

Usar herramienta Rota: No se puede usar una herramienta con durabilidad 0.

<Actualmente el try-catch estan en el método donde se tira la excepción y en el catch se imprime por consola un mensaje describiendo que una excepción ha sido lanzada y el porque.
Ej: 'EXCEPCION LANZADA: se intento usar una herramienta rota'.
Cuando el inventario y la creación por tablero esten implementados ésto va a ser modificado.>