Universidad de Chile
Facultad de Ciencias Físicas y Matemáticas
Departamento de Ciencias de la computación
CC3002-1 Metodologías de Diseño y Programación

Tarea n°2

Alpaca Emblem

Integrantes: Beatriz Graboloza

Profesor: Alexandre Bergel
Auxiliares: Juan-Pablo Silva
Ignacio Slater
Ayudantes: Alejandro González
Daniel Soto
Heinich Porro
Marco Caballero
Nicolás Escudero
Rodrigo Llull
Tomas Vallejos

Fecha de entrega: Algún Día 2019
Santiago, Chile

1. Resumen

Datos:

Link del repositorio: https://github.com/BeaNyann/TareasCC3002*

Nombre: Beatriz Graboloza

Rut: 19.665.944-7

Usuario de GitHub: BeaNyann

*Intenté que se pudiera acceder haciendole clic pero me sale que no se encuentra la página, pero si copio el link si funciona.

UMLs:

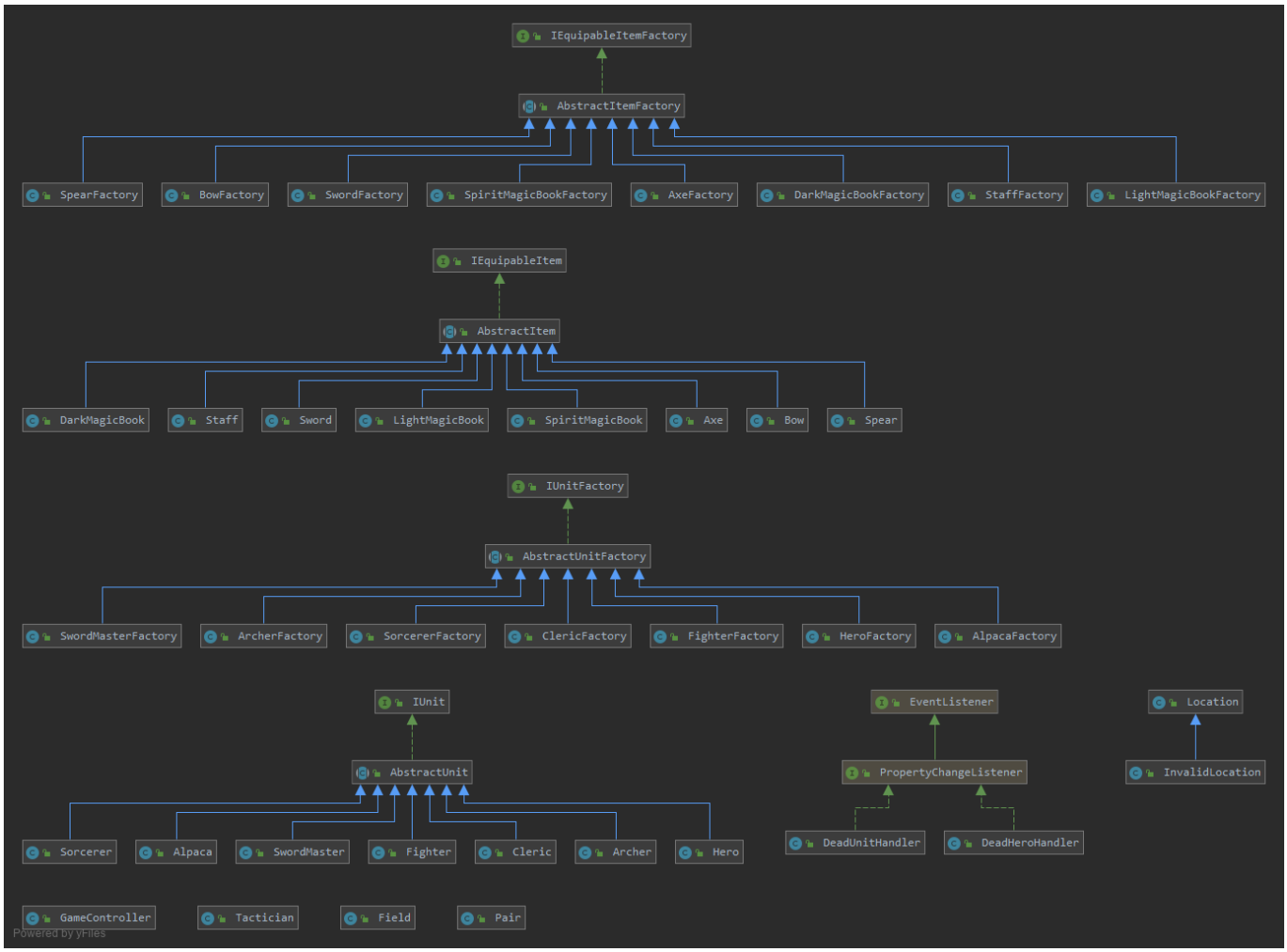

Figura 1: Diagrama UML

En este primer UML sin métodos se ve la organización general incluyendo lo agregado en esta entrega, donde además de lo de la tarea anterior se tienen interfaces para las fatories de items y unidades en donde se encuentran los métodos comunes de todas las subclases.

Luego les sigue una clase abstracta donde se implementan los métodos que deben funcionar de la misma forma para todas las fatories de unidades/items, la manera default de funcionamiento, para luego en cada subclase crear el objeto que corresponda.

Se encuentran también los EventHandlers, el Tactician y el Controller.

Las clases para unit factories y para item factories funcionan de manera parecida por lo que se explicaran juntos:

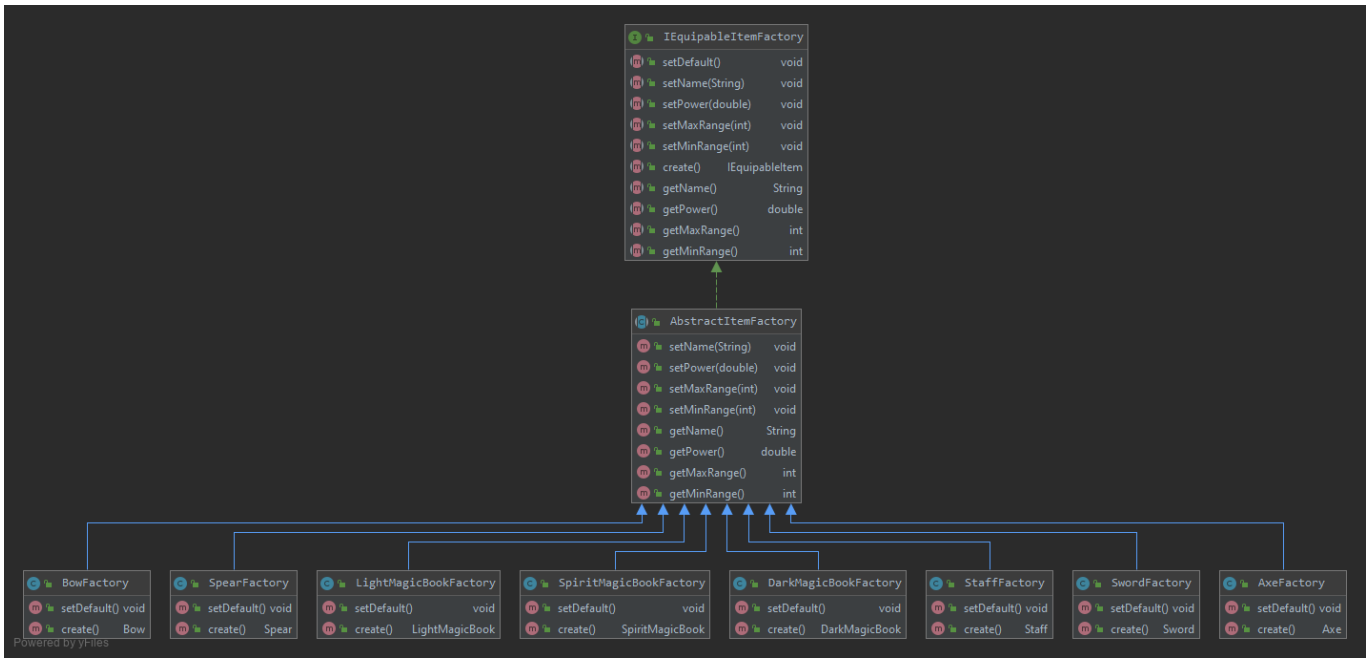

Figura 2: Diagrama UML con metodos de las Item Factories

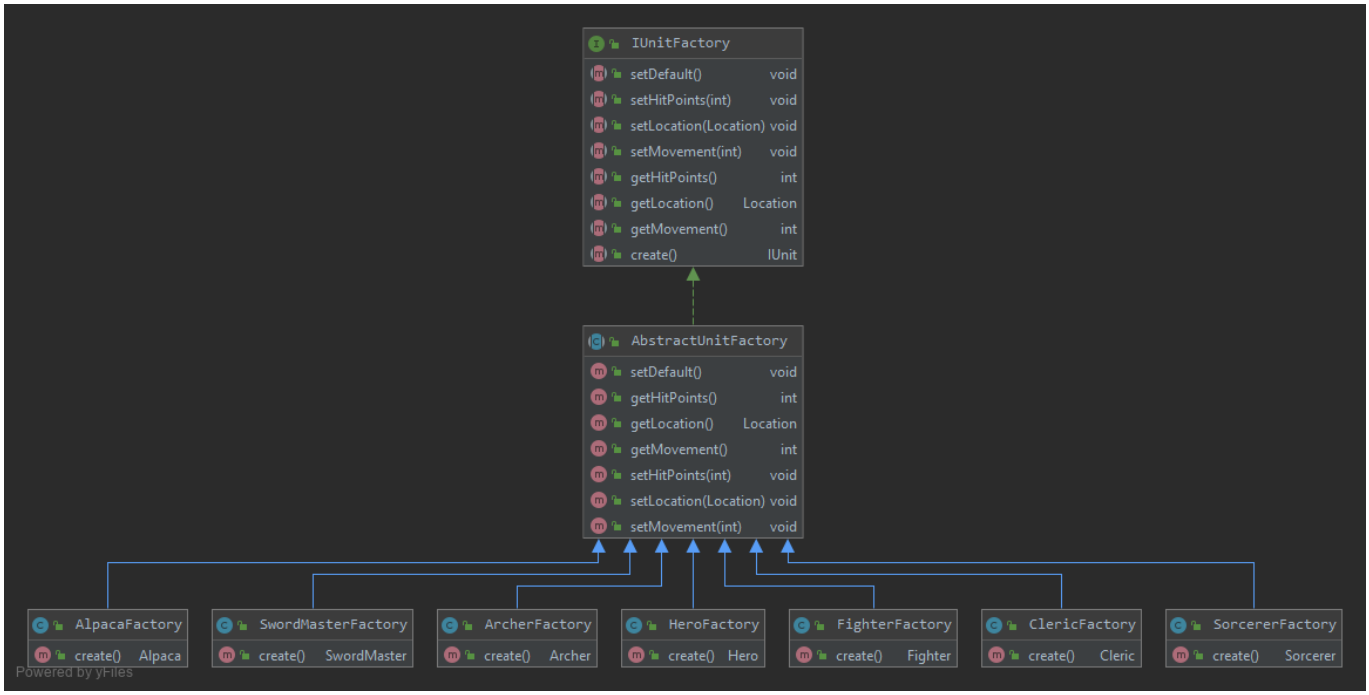

Figura 3: Diagrama UML con metodos de las Unit Factories

Acá se puede ver como la interfaz tiene los métodos para Setear defaults, setear atributos especificos u obtenerlos y el metodo create

En la clase abstracta se implementan todos los de arriba excepto create. En items factory el set de-

fault cambiaba para cada uno de los items por lo que no fue necesario ponerlo en la clase abstracta, en cambio en unit factory todas las units tenían valores default iguales por lo que si se implementó.

Luego cada clase tiene su create, y en las que corresponde su setDefault.

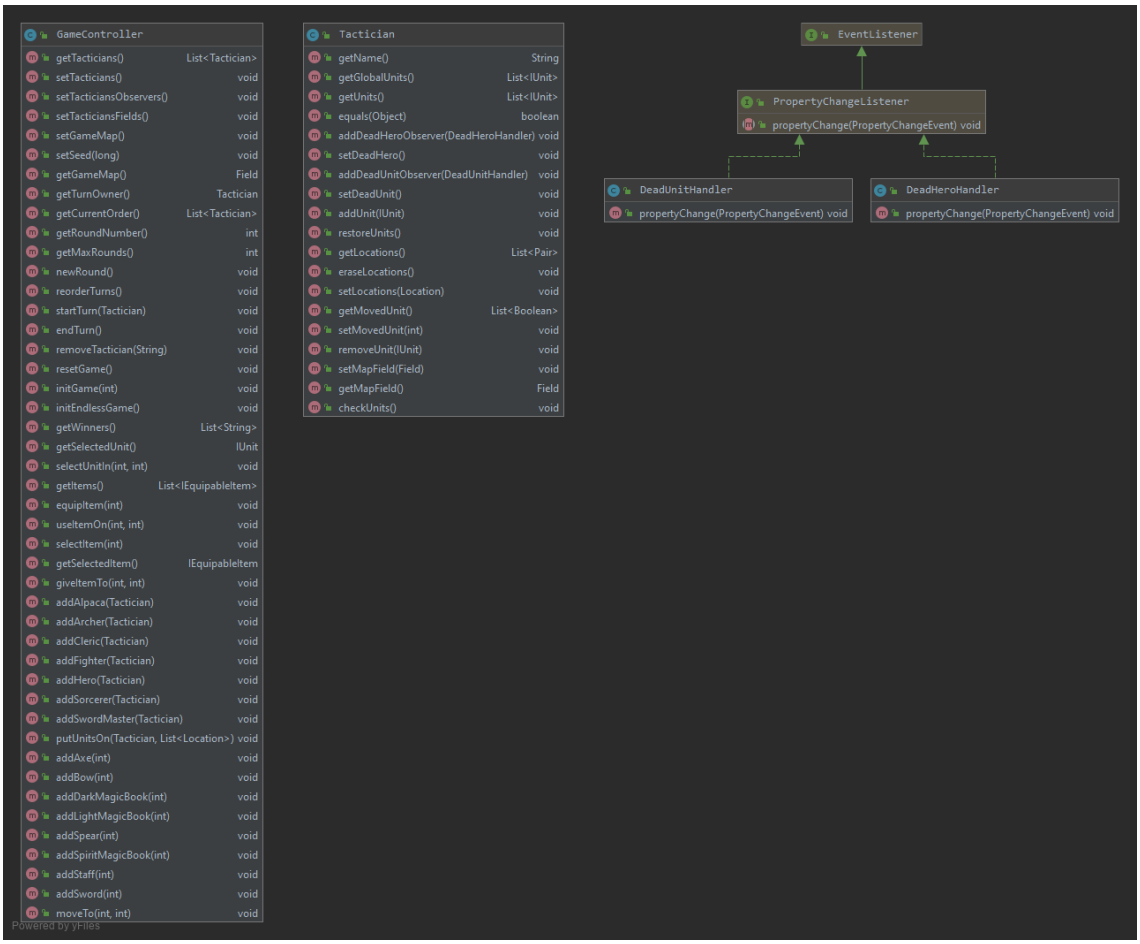

Figura 4: Diagrama UML con metodos del Tactidian, del Game Controller y de los EventHandlers

Como se puede ver, tenemos dos event handlers, uno para manerar la muerte de unidades y otro para la muerte de heroes. Ambos extienden la clase PropertyChangeListener para cumplir correctamente sus funciones y tienen el metodo propertyChange, que es el que realiza las acciones correspondientes luego de que ocurre uno de los eventos.

El tactician tiene métodos para obtener su nombre, sus unidades, sus unidades con vida, un método para comparar tacticians, los métodos para suscribir observer y para indicarles a estos que algo cambió, puede añadirse unidades, restaurar las que ya tiene, obtener las locations de sus unidades y si es que estas se han movido o no, remover unidades, obtener el mapa de juego y por último un método para checkear si alguna de sus unidades esta muerta.

El controller tiene los métodos para setear y obtener los tacticians que participarán, setear el mapa,

manejar los turnos y el orden de estos, manejar las rondas, remover tacticians del juego, obtener los ganadores, reiniciar el juego y también empezarlo. Tiene métodos para añadir items, equipar items, seleccionar una unidad y seleccionar un item, mover una unidad, intercambiar objetos entre unidades, atacar a una unidad y añadirle a un tactician la unidad que desee.