**TP2 ALGOFORMERS**

**GRUPO N1**

Franco Bragantini 97190 [franco-xk94@hotmail.com](mailto:franco-xk94@hotmail.com)

Federico Rossini 97161 [federicorossini09@gmail.com](mailto:federicorossini09@gmail.com)

Mariano Baffetti 93791 [marianobaffetti@hotmail.com](mailto:marianobaffetti@hotmail.com)

Pedro Martinez Peck 97123 [pedro\_peck\_8@hotmail.com](mailto:pedro_peck_8@hotmail.com)

**Supuestos**

**Modelo de dominio**

Como se puede ver en los diagramas de clases, hay una clase principal llamada Juego que contiene a ambos jugadores (clase Jugador) y a un “tablero” (un HashMap con los casilleros).

Cada jugador tiene sus 3 Algoformers correspondientes y se encarga de elegir a que algoformer quiere mover o utilizar en cada turno y también tiene la responsabilidad de avisarle a los 3 algoformers cuando el turno acaba.

De la clase Algoformer heredan dos clases Autobot y Decepticon, de los cuales heredan los 8 algoformers correspondientes (Optimus, Bumblebee, Megatron, etc y las dos combinaciones).

Un algoformer tiene un EstadoAlgoformer (humanoide o alterno) y dependiendo de las características dadas por el enunciado, cada uno implementa una EstrategiaDeMovimiento (ej: MovimientoAlternoAereo) además de tener 3 clases Piernas, Arma y Armadura que se encargan de modificar los atributos cuando se aplica un bonus

Cada algoformer tiene una referencia al Casillero que ocupa y este mismo esta compuesto por una Zona (aérea o terrestre) que puede ser un Pantano, Nube, Rocosa, etc y un Bonus (Flash, DobleCanion, BurbujaInmaculada)

Se adjuntan los Diagramas de clase y de secuencia en el repositorio junto con el programa utilizado para crear el tablero por medio del Json

**Excepciones**

## **AlgoFormerAtrapadoEnNebulosaNoSePuedeMoverException** : se creó con el fin de avisar al usuario que quiera mover a un algoformer en estado alterno aéreo que se encuentra atrapado en una nebulosa. Esta excepción se lanza antes de realizar la mecánica del movimiento del algoformer (clase Jugador, línea 131) y se atrapa en el evento que llama al método que hace mover al algoformer (clase BotonMoverHandler, línea 51). Una vez capturada la excepción se un mensaje al usuario a través de la vista.

* **AlgoFormerFueraDeAlcanceException** : fue creada para notificar al usuario que el algoformer al que quiere atacar se encuentra fuera del alcance del algoformer que seleccionó como atacante. Se lanza luego de verificar si el algoformer enemigo se encuentra en rango (clases ArmaAutobot y ArmaDecepticon, línea 26 en ambas) y se atrapa en el evento que llama al método que efectúa al ataque (BotónAtacarHandler, línea 46). Una vez capturada se lanza un mensaje al usuario a través de la vista.
* **MovimientoInvalidoException** :se creó con el fin de notificar al usuario que quiera mover a un algoformer a un casillero que no se encuentra a su alcance. Se lanza luego de verificar que el movimiento es valido (clase EstrategiaDeMovimiento, línea 55) y se captura en el evento de la vista que llama al método para mover al algoformer (BotonMoverHandler, línea 44). Al capturarla se muestra un mensaje al usuario.
* **HumanoideNoSePuedeMoverSiEstaEnPantanoException**: fue creada para alertar al usuario que el algoformer en estado humanoide que seleccionó se encuentra atrapado en el pantano y por lo tanto no lo puede mover. Se lanza en la verificación anterior al llamado al metodo del movimiento (Clase Jugador, línea 133) y se captura en el evento que dispara el movimiento de un algoformer (Clase BotonMoverHandler, línea 59). También se muestra un mensaje al usuario.

Las excepciones anteriores son todas las que capturamos en nuestra implementación. Las excepciones creadas restantes no son capturadas o lanzadas ya que nuestra interfaz gráfica contempla los casos excepcionales, por ejemplo, deshabilitando botones de acciones que el usuario podría realizar. Fueron creadas con el objetivo de dejar constatado lo que no debería ocurrir durante la ejecución del programa.