Tarea #2
Lenguajes de Programación INF-253
¿¡Juguemos Elepé!?

Sven von Brand
svbrand@inf.utfsm.cl

Rodolfo Castillo
rodolfo.castillo.13@sansano.usm.cll

Marcelo Jara
marcelo.jara.13@sansano.usm.cl

Segundo semestre académico, 2015

1 Objetivos

- Que el alumno entienda y aplique el paradigma de programación orientada a objetos con el lenguaje insignia Java.
- Que aplique nociones de herencia e interfaces.
- Que aprenda a interactuar con librerías no estándar usando documentación.

2 Introducción

La economía de la república de Elepé esta cayendo a pedazos, por lo que los analistas estatales están barajando varias alternativas para poder sacar a flote la nación.

Los estudios arrojaron como conclusión que el mercado de los vídeojuegos es un camino factible y conveniente para el desarrollo, por lo que se tomó la decisión de aprovechar la mano de obra barata (gratis - recordemos que promueve la ideología esclavista) para implementar la iniciativa.

3 La Tarea

3.1 Descripción

En esta tarea deberán ser capaces de implementar un diseño de juego usando una librería minimalista escrita en Java para dibujar en una pantalla, las reglas del juego siguen la lógica del juego de *Damas*, pero con algunas variaciones para hacerlo algo más especial.

3.1.1 Reglas

Existen muchas variaciones para el juego[1], por lo que se pedirá que implementen la versión de Damas Inglesas (Angloamericanas) sin considerar el *soplido* (que será considerado bonus). Se entiende que pueden surgir confusiones respecto a las reglas, por lo que pueden preguntar a los ayudantes mediante los múltiples canales que existen respecto a si cierta implementación está correcta.

3.1.2 Algo más especial

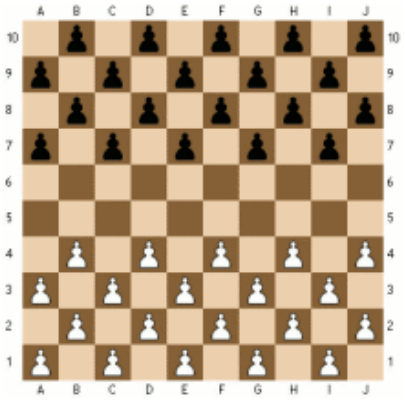
Algunas diferenciaciones del juego original:

- Las fichas puede ser de tres elementos: agua, fuego, hierba, los cuales cumplen las reglas normales de debilidades, es decir: fuego daña a hierba, hierba daña a agua, agua daña a fuego. En caso de que una ficha desee comer a una ficha del tipo más débil o bien del mismo tipo, esta gana, pero si intenta comer una pieza más fuerte, su pieza pierde.
- Existirán PowerUps que modificarán las habilidades de una ficha o del jugador, entre las posibilidades existen:
  - Poder retroceder algun espacio.
  - Poder avanzar 3 espacios.
  - Poder obligar al enemigo a mover alguna pieza.
  - Cambiar alguna pieza de elemento. Para esto siga el orden: Agua → Fuego → Hierba → Agua ..., es decir, por ejemplo, si se cliquea una pieza del tipo Fuego, entonces esta pasa a ser tipo Hierba.
  - Transformar alguna pieza del enemigo en suya.

  Deberá distribuir estos powerups en el tablero de juego de forma equitativa.

[1] `https://es.wikipedia.org/wiki/Damas#Damas_espa.C3.B1olas`

- El tablero será de 10x10 y las fichas se distribuirán como se muestra a continuación:



- La manera en que represente los powerups está a su libre disposición (Especificar en el README).
- La manera en que asigna inicialmente los elementos de cada ficha (i.e. aleatorio, manual, automático balanceado) está a su libre disposición (Especificar en el README).

**** Cabe destacar que el juego es de dos jugadores, presenciales. ****

3.1.3 Sobre la implementación

Como se mencionó anteriormente, usted usará una librería minimalista para poder comenzar a crear su juego, sin embargo, recibirá como argumento, a la hora de dibujar, un componente llamado `Graphics`, debe investigar para qué sirve y cómo poder usarlo para dibujar imágenes. (Este componente pertenece a la librería AWT).

Además, la librería contrendrá una interfaz de **Elemento**, la cual debe ser usada para poder hacer preguntas respecto a cual elemento vence a cual.

Se adjuntará documentación JavaDoc sobre la librería para que consulte sus dudas. No se pueden hacer consultas a los ayudantes si estas están respondidas en la documentación.

Considere que la tarea fue pensada en que usted utilice herencia, por lo que sea específicamente cuidadoso al identificar estos casos.

4 Consideraciones adicionales

- Usted puede realizar supuestos respecto a su tarea en el archivo README, serán considerados.
- El código debe ser completamente modular, no se permite repetir secciones de código si son posibles de evitar usando **herencia**.
- Debe poder **identificar** claramente los casos donde la **herencia** es pertinente y tambien los casos donde crear una **interfaz** es una ventaja.
- Se recomienda trabajar con un IDE para su desarrollo, puede elegir cual guste, pero recomendamos: IntelliJ IDEA o Eclipse.

  PD: Prefiera la primera.
- La tarea se realizará en los mismos grupos formados para la tarea1. Aquellos estudiantes que hayan trabajado solos, pueden seguir en la misma condición, o formar un grupo. (Si no queda opción, se formaría a lo más un grupo de 3 integrantes)

5 Evaluación del código

- Uso correcto del lenguaje (herencia, interfaces y otros).
- Uso general del paradigma orientado a objetos.
- Debe escribir un código ordenado y consistente, por ésta vez, se exigirá seguir las convenciones de Java para escribir código y se será riguroso en su uso.
- Sea cuidadoso con los modificadores de acceso.

6 Bonificaciones

- Documentación: debe documentar su código utilizando JavaDoc (5 puntos).
- ¡Soplada!: Como se indicó anteriormente, implementar esta funcionalidad será recompensado (5 puntos).

7 Sobre la entrega

- La tarea debe entregarse antes de las 23:55 hrs. del día **4 de diciembre del 2015**.

- Para consultas sobre el reglamento de tareas y/o el desarrollo de este enunciado, por favor dirigirse a la sección correspondiente en la plataforma Moodle.
- La entrega será en el siguiente formato:

  `tarea2-<user1>-[<user2>]-[<user3>].tar.gz`

  donde `userX` es su usuario del `DI`.
- En el README debe indicar que IDE utilizó y si necesita alguna configuración en particular.
- Al ser la implementación de un juego, usted puede ser creativo y agregar más funcionalidades siempre y cuando cumpla con las exigidas. Es por esto, que de darse tal caso, debe especificar qué cosas fueron agregadas en el archivo README.
- Adicionalmente, desde este momento, se pide que se explique la estrategia usada para resolver el problema.