INF-253 Lenguajes de Programación
Tarea 3: Java

Profesores: José Luis Martí – Esteban Daines
Ayudante Cátedras: Juan Pablo Escalona
Ayudantes Tareas: Cristian Vallejos – Javier Echiburú

21 de septiembre de 2016

1. SansaStone™ 2.0

La conocida empresa de videojuegos Stezzard ha decidido llevar un peldaño más alto al mejor juego de cartas virtual ambientado en la Universidad Técnica Federico Santa María. Debido a que los alumnos de Lenguajes de Programación fueron quienes desarrollaron la versión beta de este gran juego, Stezzard ha decidido solicitarles que trabajen en la versión definitiva.

Esta vez, se ha tomado la determinación de implementar más tipos de cartas en el mazo de cada uno de los jugadores, además de habilitar la opción *Jugar con un amigo*.

Debido a lo anterior, habrá algunos cambios en las reglas del juego.

2. Reglas del juego

- El programa a implementar consiste en un juego de cartas virtual versión sansana.
- Cada jugador tendrá disponible su propio mazo, el cual contará con 30 cartas.
- Cada vez que se comience una nueva partida el mazo debe estar ordenado aleatoriamente.
- Los jugadores serán de tipo Sansano.
- Los puntos de vida de los Sansanos están dados por la prioridad.
- Cada jugador tendrá 3000 puntos de prioridad inicialmente.
- La prioridad **máxima** es 3000.
- Gana el jugador que deje al rival con prioridad menor o igual a cero.
- Cada inicio de turno se roba una carta del mazo.
- Las cartas se activan justo después de sacarlas del mazo.
- Cuando ambos mazos se queden sin cartas, el juego termina y gana el Sansano que posea una mayor prioridad.
- En esta version, existen 3 tipos de cartas en el mazo de los jugadores:

- **Curso:**
  - Existen 2 tipos de jugadas para cada carta tipo Curso:
    - Reprobar: Usa su poder de ataque para quitarle prioridad al jugador objetivo.
    - Aprobar: Usa su poder de defensa para aumentar la prioridad del jugador objetivo.
  - A continuación irán las cartas tipo Curso del juego con su descripción correspondiente, además de la cantidad de veces que se encuentra dicha carta en el mazo:
    - Matemáticas: Ataca 550/Cura 200 puntos de prioridad. Cantidad: 1
    - Física: Ataca 450/Cura 150 puntos de prioridad. Cantidad: 4
    - LP: Ataca 510/Cura 180 puntos de prioridad. Cantidad: 2
    - Programación: Ataca 110/Cura 300 puntos de prioridad. Cantidad: 6
    - ED: Ataca 470/Cura 160 puntos de prioridad. Cantidad: 3
    - EDD: Ataca 430/Cura 120 puntos de prioridad. Cantidad: 4
- **Profesor:**
  - Sólo se hallan en el mazo para auto-inflingir daño a la prioridad del jugador.
  - Las cartas de tipo Profesor son únicas en cada mazo, siendo éstas:
    - Bahamondes: Reduce 420 puntos de prioridad.
    - MaxAraya: Reduce 350 puntos de prioridad.
    - Cifuentes: Reduce 390 puntos de prioridad.
    - MaxRivera: Reduce 280 puntos de prioridad.
- **Carrete:**
  - Su propósito en el mazo es curar, en bajas cantidades, la prioridad del jugador.
  - Las cartas de tipo Carrete son únicas en cada mazo, siendo éstas:
    - Cerrito: Recupera 55 puntos de prioridad.
    - InterMechón: Recupera 80 puntos de prioridad.
    - Sansafonda: Recupera 100 puntos de prioridad.
    - SemanaSansana: Recupera 150 puntos de prioridad.
    - BloqueLibre: Recupera 30 puntos de prioridad.
    - FiestaOmbligo: Recupera 125 puntos de prioridad.

En cuanto a la modalidad de juego, existen dos tipos:

2.1. Modo *1 Jugador*

- Al comenzar, el jugador debe elegir un nombre para dar inicio a la partida.
- El enemigo será el computador, cuyas jugadas deben decidirse de manera aleatoria.
- Ambos mazos tienen las mismas cartas. Sin embargo, el mazo del jugador debe estar ordenado aleatoriamente, mientras que el de su rival deberá ser seleccionado entre dos tipos de mazo:
  - *Agresivo*: Las cartas de tipo Curso deben ser ingresadas al mazo en orden descendente, según ataque. Las cartas de tipo Profesor y Carrete pueden ser ingresadas en cualquier orden.
  - *Defensivo*: Las cartas de tipo Curso deben ser ingresadas al mazo en orden descendente, según defensa. Las cartas de tipo Profesor y Carrete pueden ser ingresadas en cualquier orden.

2.2. Modo *Jugar con un amigo*

- Al comenzar, ambos jugadores deben elegir su nombre para luego iniciar la partida.
- El mazo de cada jugador tiene las mismas cartas que el mazo rival. Sin embargo, el orden de éstas es aleatorio en ambos casos.

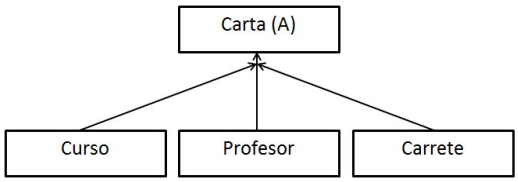
3. Requerimientos

- Se debe implementar un programa que permita jugar el juego descrito en la sección anterior.
- El juego debe tener una interfaz gráfica con la que interactuará el usuario. Se debe entregar feedback de lo que ocurre durante la partida (daño/curación realizad@, prioridad actual, etc.) y las instrucciones de uso al jugador, utilizando botones para permitirle a éste interactuar y cuadros de texto para el feedback.
- Se debe tener una clase **Duelo**, la cual debe almacenar la cantidad de turnos que tomó la partida actual y el nombre del ganador de ésta.
- Debe existir la clase **Sansano**, la cual debe almacenar el nombre del jugador (en el caso de la computadora, el nombre por defecto es *Oponente*), la prioridad de éste y su respectivo *Mazo*.
- Las cartas deben formar una jerarquía de clases, donde la clase **Carta** es considerada como padre y los tres tipos como hijos. Carta debe tener un método *activar()*, el cual dispara la habilidad de la carta respectiva sobre el Sansano objetivo (*la implementación de activar() y sus parámetros quedan a criterio del programador*), y los atributos nombre y descripcion.
- **Curso** debe tener los atributos ataque y defensa, los cuales representan los puntos de daño/curación que este tipo de carta realiza. Además, debe tener los métodos *Aprobar()* y *Reprobar()*, los cuales reciben como parámetro el objeto (clase) Sansano al cual se desea dirigir la habilidad de la carta. Curso sólo puede activar una de sus habilidades a la vez.
- **Profesor** y **Carrete** deben tener su respectivo atributo daño/curación y los métodos *Recorregir()* y *Carretear()*, respectivamente, los cuales también reciben como parámetro al Sansano objetivo.
- Las cartas de cada jugador deben estar en una lista única en cada clase Sansano, la cual se recorre para el desarrollo normal de la partida.
- Para una mejor idea del sistema de clases, ver la siguiente sección.
- Se debe utilizar una lista de (java.Util.List) para el manejo de listas.
- Cada clase debe tener su propio archivo .java asociado.
- Cada atributo debe tener un *getter* y *setter* asociado si es manipulado de forma externa. Los *getter* y *setter* deben ser utilizados en el programa.

4. Herencia

Los siguientes son los esquemas de herencia que se deben implementar en el programa.

- (A): Clase abstracta



5. Archivos a Entregar

Resumiendo lo anterior, los archivos mínimos a entregar son:

- Duelo.java
- Sansano.java
- Carta.java
- Curso.java
- Profesor.java
- Carrete.java
- Juego.java (contiene función main)
- MAKEFILE (Archivo de compilación automática).
- README.txt (Archivo manual con instrucciones de uso del programa).

Los alumnos pueden crear más archivos si lo estiman necesario, mientras tengan los nombrados anteriormente entre los enviados.

6. Sobre Entrega

- El programa debe correr en el Sistema operativo presente en los computadores de la Universidad.
- El programa debe incluir un **MAKEFILE** y **README con instrucciones** para su compilado. Por favor, se debe explicar todo. El readme es como un manual para el usuario.
- El código debe venir identado y sin warnings.
- Cada función debe llevar una descripción según lo establecido por el siguiente ejemplo:

```
/******** Funcion: Suma_Enteros ********************
Descripcion: suma dos enteros positivos

Parametros:
n1 entero
n2 entero

Retorno: resultado de la operacion aritmetica de la suma entero
***********************************************/
```

- Se debe trabajar en grupos de dos personas, las cuales deben pertenecer al mismo curso de malla.
- **La entrega debe realizarse en tarball (tar.gz) y debe llevar el nombre: Tarea3LP_RolIntegrante-1_RolIntegrante-2.tar.gz**
- **El archivo README.txt debe contener nombre y rol de los integrantes del grupo e instrucciones detalladas para la compilación y utilización de su programa.**
- **El no cumplir con las reglas de entrega conllevará -30 puntos en su tarea.**
- La entrega será via moodle y el plazo máximo de ésta es hasta el **lunes 17 de octubre a las 23:55 hora moodle**.
- Por cada día de atraso se descontarán 30 puntos.
- Las copias serán evaluadas con nota 0 y se informarán a las respectivas autoridades.

7. Calificación

La escala a utilizar para la revisión de esta tarea es la siguiente:

- Reglas de juego implementadas (Ejecutable) (40 puntos)
- Clases bien definidas (Encapsulamiento) (20 puntos)
- Código (Orden, implementación y comentarios) (40 puntos)

Los descuentos que pueden ocurrir son:

- No respetar reglas de entrega (100 puntos)
- Warnings (5 puntos cada uno, con un máximo de 25 puntos)
- Código no compila (100 puntos)
- No MAKEFILE (10 puntos)
- No jar (100 puntos)

En caso de existir nota negativa, ésta será reemplazada por un 0.