INF-253 Lenguajes de Programación
Tarea 3: Java

Profesor: Roberto Diaz
Ayudante Cátedras: Anastasiia Fedovora
Ayudante Tareas: Héctor Larrañaga - Sebastián Campos - Axel Reyes

31 de octubre de 2020

1. Lore

Despiertas desorientado y aturdido, te das cuenta que te encuentras arropado con una clase en una pequeña tienda hecha de cuero de puntero, miras a tu alrededor en busca de algo familiar que te haga recordar lo que sucedió en la nave de refugiados cuando, al intentar alcanzar un vaso con agua cerca tuyo, notas un tatuaje en forma de * en tu mano. Súbitamente, comienzas a recordar los horribles sucesos ocurridos en la nave. Resulta que en medio de tu misión para salvarte a ti y a los refugiados, fuiste emboscado por un ente desconocido el cual te transformó en un puntero y te liberó, lo cual te envió al plano de los void* donde sólo existía el texto genérico llamado "Lorem Ipsum".

Aún no tienes la certeza de quién fue, pero de alguna forma alguien debió haberte casteado a tu forma humana, salvándote así de tu estado vacío. Con determinación, sales de la tienda encontrándote con un panorama aterrador, la nave se estrelló en un planeta confuso, verboso y muy POO. Antes que te dieras cuenta, se te acercaron unas criatura salvajes conocidas como Klazes, Avstraktos e Ingterfazes, las cuales están en busca de su próxima presa. Con miedo, pero a la vez lleno de determinación para sobrevivir, te preparas para domar a estas bestias temidas en toda la galaxia.

2. Idea General

La idea es implementar un pequeño juego de cartas que representará nuestra querida vida universitaria. Este juego tendrá tres tipos de cartas: ramos, estudios y eventos. Cada carta de ramo tendrá un nivel de dificultad (reflejado en los créditos del ramo) y un área (Matemática, Humanista o Informática). Para poder pasar los ramos se deberán jugar inteligentemente las cartas de estudio y cartas de evento. Las cartas estudio tienen dos cualidades: rareza y área. Una carta de estudio es más efectiva al aplicarse sobre un ramo de la misma área. Por otra parte, la rareza de una carta define su efectividad en pasar ramos en general. Por último, las cartas eventos aplican efectos que afectarán a alguna carta o al tablero.

En el tablero existirán en todo momento dos mazos: el mazo de carrera, que contendrá las cartas de ramos, y el mazo universitario, desde donde se robarán las cartas de estudio y eventos que podrán ser jugadas.

El flujo del juego será el siguiente:

1. Al inicio de cada turno se colocarán en el tablero 2 cartas de ramos desde el mazo de carrera y robará cartas del mazo universitario hasta tener 6 cartas en su mano.

2. El jugador podrá jugar sus cartas. Las jugadas que tendrá disponibles son:

- Jugar cartas de estudio sobre los ramos, siempre y cuando la cantidad de horas disponibles sea mayor o igual al costo de horas de la carta a jugar. Al inicio de cada turno el jugador tendrá 12 horas disponibles y se restarán las horas correspondientes cada vez que se juegue una carta.
- Jugar directamente sus cartas de eventos y aplicar su efecto al tablero.

Si el jugador no juega alguna de sus cartas, podrá conservarla para el próximo turno. Sin embargo, la mano del jugador nunca podrá exceder las 6 cartas.

3. Luego de finalizar su jugada, se calculan las notas obtenidas en cada uno de los ramos, declarándose como aprobados si su nota es mayor o igual a 55, y reprobado si son menores que 55. Las cartas de ramo y estudio son luego retiradas del tablero.

4. El juego continua y lo anterior se repite hasta que se aprueben cuatro ramos o se reprueben dos, resultando en victoria o derrota respectivamente.

3. Estructura

El programa estará compuesto por las clases que se muestran en las Figuras 1 y 2 descritas a continuación junto con un listado de sus métodos. Los atributos y métodos indicados son los mínimos que **deben ser implementados y usados** en su programa, pero se pueden incluir más según se estime conveniente.

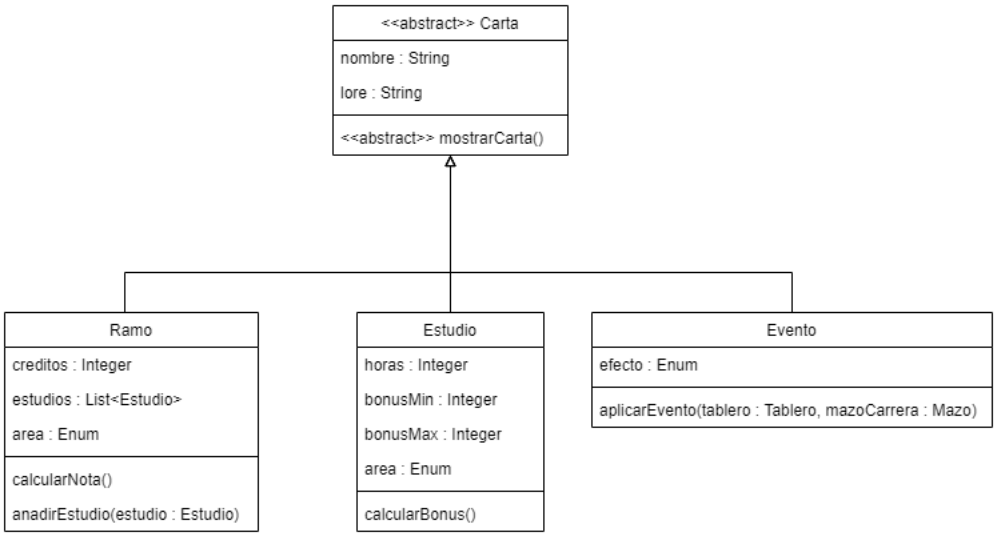

Figura 1: Diagrama de clases tipo Carta

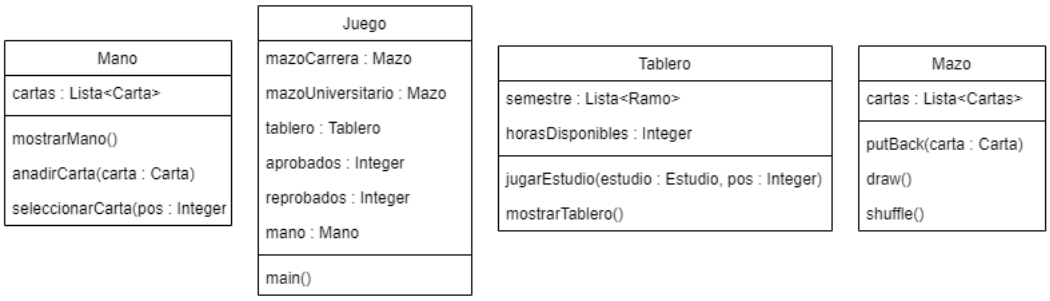

Figura 2: Diagrama de otras clases

3.1. Carta

Corresponde a la clase abstracta para la estructura básica de las cartas, la cual posee como atributos el nombre y su lore (una breve reseña que será escrita por usted con temática libre).

- `abstract void mostrarCarta()`: Corresponde al método que mostrará la carta por pantalla con todas sus características (nombre, lore, área, créditos, etc). En el caso de una carta de Ramo, se deben mostrar las cartas de Estudio jugadas encima de ésta.

Esta clase abstracta la extienden las siguientes 3 clases.

3.2. Ramos

Corresponde a la clase que representa las cartas de Ramos, las cuales poseen como atributos la cantidad de créditos, el área (Matemática, Humanista o Informática) y una lista con las cartas de Estudio jugadas sobre el ramo. Notar que área deberá estar definida con una enumeración.

- `int calcularNota()`: que calcula la nota final del ramo, sumando los puntajes de todos sus estudios. Si un ramo y estudio tienen la misma área la nota aumenta en un 25 %, por otro lado, siempre se resta el doble de la cantidad de créditos del ramo.

$$NF = (\sum_{i}^{|Estudios|} Bonus_i * BonusArea_i) - 2 \cdot Creditos$$

- `void anadirEstudio(Estudio estudio)`: agrega una carta de estudio sobre este ramo al ser jugada.

Los ramos Humanistas tendrán 2 créditos, los ramos de Informática tendrán 5 créditos y los de Matemática 7 créditos.

3.3. Estudio

Corresponde a la clase que representa las cartas de Estudio, las cuales se pueden jugar sobre una carta de Ramo y sumarle puntos a su nota. Estos poseen como atributos la cantidad de horas a gastar para usar la carta, área (Matemática, Humanista o Informática), el mínimo y máximo puntaje que podrá sumarle a la nota de un ramo.

- `int calcularBonus()`: calcula el puntaje a sumar a la nota del ramo correspondiente, retornando un entero aleatorio entre el puntaje mínimo y máximo.

Habrán 3 variantes de cartas Estudio:

1. *Comunes*: que tendrán un puntaje entre 18 y 26, y costarán 2 horas de estudio
2. *Raras*: que tendrán un puntaje entre 28 y 40, y costarán 3 horas de estudio
3. *Épicas*: que tendrán un puntaje entre 20 y 90, y costarán 4 horas de estudio.

3.4. Eventos

Corresponde a la clase que representa las cartas de Eventos, las cuales poseen como atributo el efecto que realiza sobre el tablero definido en una enumeración.

- `void aplicarEvento(Tablero tablero):` que realizará un cambio sobre el tablero dependiendo del efecto que tengan.

Las cartas de evento pueden tener uno de tres posibles efectos:

- **RAV**: Permite al jugador seleccionar un ramo y mezclarlo con el mazo de carrera.
- **Buff (Área)**: Aumenta el puntaje máximo que pueden entregar todas las cartas de estudio a un ramo de esa área en el tablero en un 25 % durante este turno.
- **Cambio de coordinación**: Hay 50 % de probabilidad que durante el turno actual los créditos de un ramo a elección del jugador aumenten en 3 y 50 % de probabilidad de que disminuyan en 3, pudiendo quedar un valor de créditos negativo.

3.5. Mano

Corresponde a la clase que representa las cartas en la mano del jugador, las cuales son mantenidas en una lista de cartas.

- `void mostrarMano()`: que permita mostrar por pantalla las cartas de la mano actual del jugador
- `void anadirCarta(Carta carta)`: que permite añadir una carta a la mano del jugador
- `Carta seleccionarCarta(int pos)`: retorna la carta de la mano indicada por el parámetro `pos`

3.6. Mazo

Corresponde a la clase que representa los mazos del jugador, este se representa como una lista de cartas.

- Carta draw(): retorna la carta en el tope del mazo
- void shuffle(): revuelve las cartas del mazo de manera aleatoria
- void putBack(Carta carta): devuelve la carta al fondo del mazo

El mazo universitario comenzará con 25 cartas, de las cuales 20 serán de estudio (10 comunes, 7 raras y 3 épicas) y 5 de evento (Una de cada tipo). El número de cartas de estudio en el mazo, por área y rareza, es la siguiente:

- Humanista: 3 Comunes, 2 Raras y 1 Épica.
- Informática: 4 Comunes, 3 Raras y 1 Épica.
- Matemática: 3 Comunes, 2 Raras y 1 Épica.

El mazo de carrera tendrá 10 cartas al iniciar el juego.

3.7. Tablero

Corresponde a la clase que representa el tablero del juego, este consta de una lista de ramos y una cantidad de horas disponibles (en un principio 12 horas).

- `void jugarEstudio(Estudio estudio, int pos)`: aplica la carta `estudio` a la carta de ramo en la posición `pos`, descontando la cantidad de horas disponibles indicadas en la carta
- `void mostrarTablero()`: muestra el tablero por pantalla con toda la información relevante para el jugador (cartas ramo en juego y cartas estudio puestas en cada ramo)

3.8. Juego

Aquí ustedes plantearán el flujo del programa, tendrán que guardar el estado del juego con ambos mazos, una mano y un tablero, además deberá llevar el conteo de el numero de ramos aprobados y reprobados.

4. Jugabilidad

La tarea debe permitir al usuario jugar según lo descrito anteriormente negando cualquier jugada inválida. El input puede ser por consola o con una interfaz gráfica (válido por puntos extra). Por lo tanto, el input debe ser la acción que realizará el usuario en cada momento (jugar cartas, seleccionar carta por algún efecto, etc), permitiendo cumplir con todas las condiciones y restricciones del juego. Además, como output debe informar de lo que está ocurriendo en el juego para que el usuario pueda entender el contexto y pensar la siguiente jugada o saber si la partida ya terminó junto con su resultado. El formato del input y output es libre. En el README.txt deben incluirse instrucciones detalladas del método de interacción con el juego.

5. A entregar

- Juego.java
- Carta.java
- Ramo.java, Estudio.java, Evento.java
- Mano.java, Tablero.java, Mazo.java
- makefile
- README.txt

6. Sobre Entrega

- El código debe venir ordenado.
- Se asignará un bono de 15 pts por la implementación de una interfaz gráfica, esto será totalmente opcional.
- Se asignará un bono de 8 pts si el input se procesa interactivamente en lugar de ser procesado por entrada estándar, esto será totalmente opcional y no acumulable con el bono anterior.
- Cada método no especificado en la tarea debe llevar un comentario que indique nombre de la función, parámetros que recibe, una breve descripción de lo que hace y retorna. Deben utilizar el siguiente formato de **Javadoc**

```
/**
* NombreDeLaFuncion:
* DescripcionDeLaFuncion
*
* @param NombreDelParametro Tipo:Descripción.
* @param NombreDelParametro Tipo:Descripción.
* @return Tipo:Descripción.
*/
```

- Debe estar presente el archivo **makefile** para que se efectúe la revisión, este debe compilar **TODOS** los archivos. Se puede incluir un comando que permita la ejecución del programa, luego de la compilación. Tarea sin makefile no será revisada.
- El trabajo es individual.
- **La entrega debe realizarse en zip y debe llevar el nombre: Tarea3LP_RolIntegrante.zip**
- **El archivo README.txt debe contener nombre y rol del alumno e instrucciones detalladas para la compilación y utilización de su programa. SE DESCONTARÁ SI FALTA ALGO DE LO SEÑALADO.**
- **El no cumplir con las reglas de entrega conllevará un descuento máximo de 30 puntos en su tarea.**
- La entrega será vía aula y el plazo máximo de entrega es hasta el **Miércoles 18 de Noviembre a las 23:55 hora aula**.
- Solo se contestarán dudas hasta 3 días antes de la fecha de entrega original.
- Por cada día de atraso se descontarán 20 puntos (10 dentro de la primera hora).
- Las copias serán evaluadas con nota 0 y se informarán a las respectivas autoridades.

7. Calificación

- Carta (30 pts)
  - Estudio (10 pts)
  - Ramos (10 pts)

    - Evento (10 pts)
- Tablero (20 pts)
- Mano (15 pts)
- Mazo (10 pts)
- Juego (25 pts)
- No compila: Recorrección