Laboratorio 5 - Programación orientada a objetos (POO)

INS125 - Lenguajes de Programación

Universidad Andrés Bello

10 de junio de 2020

1. Monopoly

Monopoly es un juego de mesa, el cual consta de cuatro jugadores, los cuales deben avanzar por un tablero para poder comprar propiedades, pagar arriendos o fianzas en el caso inoportuno de caer en una cárcel. En este trabajo, se le pide hacer una pequeña simulación acotada a algunas acciones de este juego, utilizando el paradigma de programación orientado a objeto.

Para esto, se deberá programar una clase **Jugador**, la cual debe contener toda la información vital, tal como el nombre, figura que este utilizará, el dinero con el que cuenta, su turno, casilla y propiedades adquitidas. Esta clase también deberá contar con un método de **tirarDado**, el cual permita obtener un valor aleatorio entre 1 y 6.

Además, se deberá contar con una clase **Casilla**, la cual debe contener el nombre de la propiedad y la casilla en la que está ubicada. De esta clase base heredan dos clases, las cuales son **Propiedad** y **Cárcel**.

La clase **Propiedad** deberá contar con la información del dueño y el precio de compra.

La clase **Cárcel** deberá contener un monto de fianza y un prontuario (es decir, un contador de cuántas veces los jugadores han caído en la cárcel).

Por último, se le pide confeccionar una clase **Partida**, la cual deberá tener como objeto una lista de jugadores (recuerde que el juego exije que sean entre dos a cuatro participantes) relacionado a partida a traves de una *asociación*. Como métodos de la clase partida debe incorporar:

- **Constructor:** Debe leer un archivo de texto de entrada con los datos de la partida, formato el cual está detallado en el punto 3 y una lista de jugadores. En este método se inicializará el tipo de cada casilla (entre propiedad y carcel) de manera aleatoria.

- **elegirJugador:** Debe retornar el turno del jugador correspondiente. Al iniciar el juego debe asignarse un orden de los jugadores y almacenarlo en la propiedad Turnos.
- **Jugar:** Debe retornar una lista, con el turno del jugador y su celda avanzada según el número obtenido por el dado.
- **accionCasilla:** Debe ingresar como entrada el jugador y la lista de todas las casillas del mapa y, según el número obtenido por los dados, debe ver si cae en una casilla de propiedad o de cárcel. Si cae en una casilla de propiedad, y se cuenta con dinero suficiente, el jugador compra la propiedad, retornando una lista con la propiedad adquirida y el dinero restante. En el caso de que la casilla sea cárcel, el jugador deberá restar el dinero de la fianza y retornar el dinero restante.
- **Simulación:** Debe generar una simulación del juego desde que se inicia hasta que termina, de acuerdo a los datos de entrada, y escribir un archivo con el juego simulado. Debe recibir un nombre para el archivo de salida como argumento.
- **finalizarPartida:** Debe mostrar un mensaje de que la partida está terminada y mostrar al ganador. En el caso de empate, el ganador será el jugador a quien le toque primero.
- **partidaTerminada (@property):** Este método retorna `True` o `False` de acuerdo a si la partida está terminada o no. Debe implementarlo utilizando el decorador `@property`.

Figura 1 presenta el diagrama de clases de la tarea, con sus correspondientes propiedades y métodos.

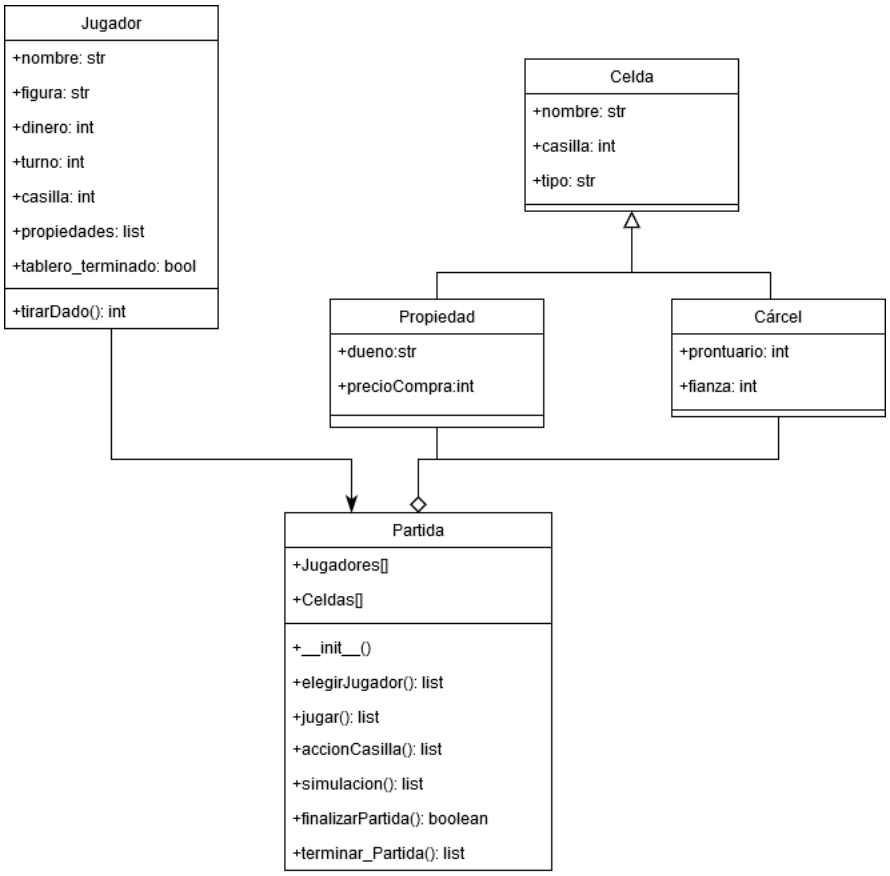

Figura 1: Diagrama de Clases solicitado para el juego Monopoly

2. Tarea

En esta tarea, se le solicita implementar un programa que permita simular una partida simplificada de Monopoly. Se deberan crear todas las clases mencionadas anteriormente (es decir, Jugador, Celda, Propiedad, Cárcel y Partida) junto con sus correspondientes propiedades y métodos. Usted recibirá un archivo de entrada, el cual deberá contener en su primera línea:

```
N_jugadores X_celdas SEED Costo_propiedades Costo_fianza
```

Donde `N_jugadores` corresponde al número de jugadores, el cual varía entre dos y cuatro, `X_celdas` es el número de celdas con las que cuenta el tablero, `SEED` es la semilla que fijará los números aleatorios que puede ser cualquier número entero, `Costo_propiedades` el cual es el costo para todas las propiedades (se asume que todas valen lo mismo) y `Costo_fianza`, el que corresponde al costo que se incurre por caer en la cárcel.

En las líneas posteriores deberá tener el detalle para cada jugador, el cual se entrega de la siguiente manera.

```
Nombre_jugador Figura
```

Como salida, su programa deberá escribir un archivo de la simulación automática del juego que incluya todos los . Se le recuerda que el método tirar dado corresponde

a la generación de un número aleatorio entre 1 y 6, y el juego termina una vez que todos los jugadores terminaron el tablero. Aquel jugador que ya completó el tablero, no puede seguir tirando dados.

```
Nombre_jugador Turno Numero_dado Numero_celda Tipo_celda Dinero_restante
```

donde `Nombre_jugador` corresponde al nombre del jugador, `Numero_dado` corresponde al número que éste obtuvo al lanzar el dado, `Numero_celda` corresponde a la celda la cual este arribó, `Tipo_celda` indica el tipo de la celda a la que el jugador cayó (puede ser propiedad o carcel) y `Dinero_restante` corresponde al dinero que le queda al jugador luego de pagar por la propiedad o caer en la cárcel de acuerdo a la celda correspondiente.

2.1. Entrada y salida del programa

2.1.1. Ejemplos de entrada

```
4 6 1 3000 5000
Samantha Gato
Matias Perro
Marco Sombrero
Nicolas Pelota
```

```
2 8 2 2000 7500
Hulk Dinosaurio
Thor Martillo
```

Figura 2: Dos ejemplos de archivos de entrada

2.1.2. Ejemplos de salida

```
Samantha 1 4 4 Propiedad 7000
Matias 1 3 3 Propiedad 7000
Marco 1 4 4 Propiedad 7000
Nicolas 1 4 4 Propiedad 7000
Samantha 2 2 6 Carcel 2000
Matias 2 1 5 Propiedad 4000
Marco 2 1 5 Propiedad 4000
Nicolas 2 2 6 Carcel 2000
Samantha 3 4 FinJuego
Matias 3 4 FinJuego
Marco 3 2 FinJuego
Nicolas 3 3 FinJuego
Ganador Marco
```

Figura 3: Ejemplo salida de archivos

3. Instrucciones

- Fecha de entrega: Jueves 18 de Junio a las 23:59 hrs.
- Método de entrega: su repositorio privado creado a través del link de la tarea. No se aceptarán entregas en repositorios no creados a través del link de la tarea.
- Trabajo personal hecho en lenguaje Python 3.
- Su programa debe recibir y generar los archivos entrada y salida que indique la entrada por parámetros.
- Sólo puede trabajar con librerías estándar.
- El proceso de revisión será automatizado. Es importante respetar el formato establecido para los archivos de entrada y salida. Este formato no es modificable. Por lo tanto, si su archivo de salida no corresponde al formato preestablecido, su calificación será deficiente.
- Este enunciado podría sufrir modificaciones, las cuales serán anunciadas en el repositorio del laboratorio.
- Las preguntas sobre la tarea deben ser formuladas como un Issue en el repositorio del laboratorio ubicado en el siguiente link https://github.com/INS125/Laboratorio/issues

4. Código de Honor

Toda persona inscrita en este curso se compromete a:

- Actuar con honestidad, rectitud y buena fe frente a sus profesores y compañeros.
- No presentar trabajos o citas de otras personas como propias o sin su correspondiente citación, ya sea de algún compañero, libro o extraídos de internet como también a no reutilizar trabajos presentados en semestres anteriores como trabajos originales.
- No copiar a compañeros ni hacer uso de ayudas o comunicaciones fuera de lo permitido durante las evaluaciones.
- Cualquier alumno o alumna que no respete el código de honor durante una evaluación (sea este la entrega de una tarea o el desarrollo de una prueba o control tanto durante la cátedra como el laboratorio) será evaluado con la nota mínima y será virtud de profesor, de acuerdo con la gravedad de la falta, las acciones siguientes a tomar.