Documentación (Java)

Gestor de juegos de mesa

Introdución

Programa de consola en Java que gestiona, mediante un menú, el CRUD (Create, Read, Update, Delete) de jugadores y diferentes juegos de mesa. Utiliza programación orientada a objetos, clases abstractas e interfaces.

Como usar

Al iniciar el programa, se preguntará al usuario si desea definir unos datos por defecto (ya que no incorporamos una base de datos). Esto es recomendable para poder "jugar" con el programa sin necesidad de inicializar manualmente todas las opciones.

Después, mediante el uso repetido del siguiente menú, se pueden realizar diferentes acciones.

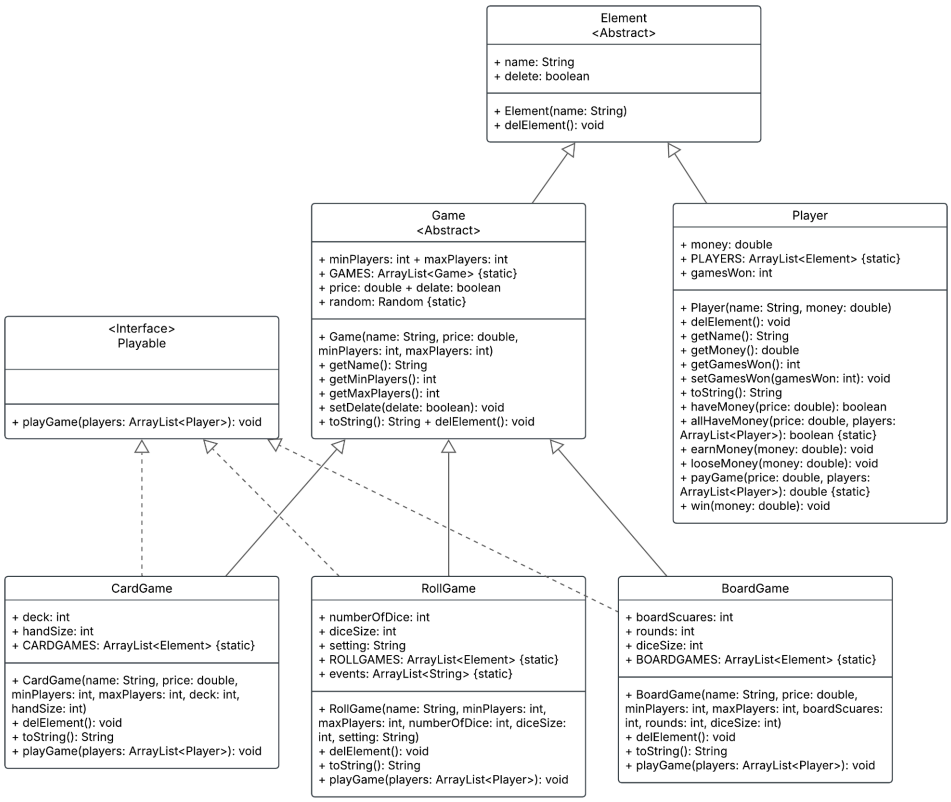
```
.........SELECCIONA UNA OPCIÓN.........
 0 -> Jugar juego de cartas
 1 -> Jugar juego de roll (Lunes no disponible)
 2 -> Jugar jugar juego de tablero
 3 -> Crear
 4 -> Consultar
 5 -> Actualizar
 6 -> Borrar
 7 -> Pasar día
 8 -> Cerrar programa
.......................................
```

Es importante tener en cuenta que algunas opciones estarán bloqueadas según el día de la semana en el que nos encontremos dentro del programa. Si se desea realizar estas acciones, primero se debe seleccionar la opción "Pasar día".

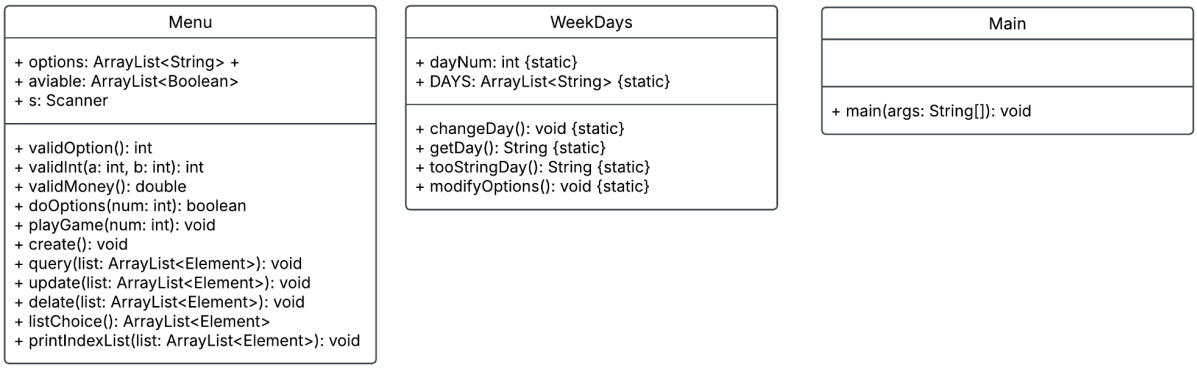
```
7
Ya es Martes!
(Envía cualquier tecla para continuar)
```

Diseño de Clases

A continuación, se muestra el siguiente diagrama UML:



Además, dentro del mismo paquete, tenemos las siguientes clases estáticas:



Estructura y funcionamiento del código

Clase Abstracta Element

Esta clase actúa como base para todos los elementos del programa, tanto jugadores como juegos. Permite recorrer, imprimir y eliminar cualquier elemento de forma genérica.

Atributos:

- String name: Nombre del elemento.
- boolean delete: Si el elemento está eliminado.

Métodos:

- public Element(String name, double price): Constructor
- public abstract void delElement(): Método abstracto para eliminar un elemento.

Clase Abstracta Player

Representa a un jugador dentro del sistema.

Atributos:

- String name: Nombre del jugador.
- double money: Dinero disponible.
- int gamesWon: Número de partidas ganadas.

Métodos:

- public Player(String name, double money): Constructor
- public boolean haveMoney(double amount): Verifica si el jugador tiene suficiente dinero.
- public void looseMoney(double amount): Reduce el dinero del jugador.
- public void earnMoney(double amount): Aumenta el dinero del jugador.
- public void payGame(ArrayList<Player> players, double price): Realiza pagos entre jugadores en una partida.

Clase Abstracta Game

Clase genérica de un juego, que sirve de plantilla para los distintos tipos de juegos dentro del programa. Hereda de Element.

Atributos:

- int minPlayers: Número mínimo de jugadores permitidos.
- int maxPlayers: Número máximo de jugadores permitidos.
- static Random random: Generador de números aleatorios para simular partidas

Métodos:

- public Game(String name, double price, int minPlayers, int maxPlayers): Constructor
- public abstract void playGame(ArrayList<Player> players): Método abstracto para ejecutar una simulación de partida real.

Interfaz Playable

Interfaz que define el método playGame para los juegos de mesa.

Métodos:

- void playGame(ArrayList<Player> players): Método para jugar un juego con una lista de jugadores.

Clase CardGame

Representa un juego de cartas. Hereda de Game e implementa la interfaz Playable.

Atributos:

- int deck: Número total de cartas en la baraja.
- int handSize: Cantidad de cartas iniciales de cada jugador.
- static ArrayList<CardGame> CARDGAMES: Lista de instancias creadas y no eliminadas.

Métodos:

- public CardGame(String name, double price, int minPlayers, int maxPlayers, int deck, int handSize): Constructor
- public void delElement(): Marca el elemento como eliminado y lo elimina de la lista.
- public String toString(): Devuelve una descripción del juego de cartas.

- public void playGame(ArrayList<Player> players): Simula una partida, en la que los jugadores se deshacen de cartas aleatoriamente hasta que uno vacía su mano, proclamándose ganador.

Clase RollGame

Representa un juego de rol. Hereda de Game e implementa la interfaz Playable.

Atributos:

- int numberOfDice: Número de dados utilizados en el juego.
- int diceSize: Tamaño de los dados utilizados en el juego.
- String setting: Lugar de ambientación del juego.
- static ArrayList<Element> ROLLGAMES: Lista de instancias creadas y no eliminadas.
- static ArrayList<String> events: Lista de eventos aleatorios que pueden suceder durante el juego.

Métodos:

- public RollGame(String name, int minPlayers, int maxPlayers, int numberOfDice, int diceSize, String setting): Constructor
- void delElement(): Elimina un juego de rol del programa.
- public String toString(): Devuelve una descripción del juego de rol.
- public void playGame(ArrayList<Player> players): Realiza un juego donde los jugadores hacen jugadas aleatorias hasta que queda uno vivo.

Clase BoardGame

Representa un juego de tablero . Hereda de Game e implementa la interfaz Playable.

Atributos:

- int boardSquares: Cantidad de casillas en el tablero.
- int rounds: Número de rondas que dura la partida.
- int diceSize: Tamaño del dado utilizado para avanzar en el tablero.
- static ArrayList<BoardGame> BOARDGAMES: Lista de instancias creadas y no eliminadas.

Métodos:

- public BoardGame(String name, double price, int minPlayers, int maxPlayers, int boardSquares, int rounds, int diceSize): Constructor.
- public void delElement(): Marca el elemento como eliminado y lo elimina de la lista.
- public String toString(): Devuelve una descripción del juego de tablero.
- public void playGame(ArrayList<Player> players): Simula una partida, donde los jugadores avanzan según el lanzamiento de un dado y gana quien más lejos llegue en el tablero.

Clase WeekDays

Gestiona los días de la semana y modifica las opciones del menú según el día actual.

Atributos:

- static int dayNum: Número del día actual.
- static ArrayList<String> DAYS: Lista de días de la semana en Strings.

Métodos:

- static void changeDay(): Pasa al día siguiente, si es domingo pasa al lunes.
- static String getDay(): Devuelve el nombre del día actual.
- static String tooStringDay(): Imprime el nombre del día actual.
- static void modifyOptions(): Modifica la disponibilidad de las opciones del menú según el día actual.

Clase Menu

Gestiona el menú de opciones del programa.

Atributos:

- static ArrayList<String> options: Lista de opciones del menú.
- static ArrayList<Boolean> aviable: Lista de opciones disponibles.
- static Scanner s: Scanner para la entrada del usuario.

Métodos:

- static void resetAviable(): Restablece la lista de opciones disponibles.

- static void clear(): Limpia la consola creando 50 filas en blanco.
- static Integer displayMenu(): Muestra el menú de opciones y devuelve la opción seleccionada.
- public static int validOption(): Valida la opción del menú en un bucle y la devuelve.
- public static int validInt(int a, int b): Valida la entrada de un número en un rango.
- public static double validMoney(): Valida que un valor de dinero esté en el rango de 0 a 1000.
- static boolean doOptions(int num): Ejecuta las acciones del menú según el número de la opción seleccionada.
- static void playGame(int num): Permite seleccionar y simular la partida de un juego.
- static void create(): Solicita los datos necesarios para crear juegos o jugadores.
- static void query(ArrayList<Element> list): Permite imprimir un elemento a seleccionar por el usuario.
- static void update(ArrayList<Element> list): Permite modificar un elemento, eliminándolo y creando uno de las mismas características.
- static void delate(ArrayList<Element> list): Elimina un elemento del programa.
- static ArrayList<Element> listChoice(): Elige entre juegos y jugadores, devolviendo la lista elegida.
- static void printIndexList(ArrayList<Element> list): Imprime una lista con índices.

Clase Main

Clase principal que ejecuta el programa.

Métodos:

- public static void main(String[] args): Presenta el programa y inicializa (si lo eliges) unos valores por defecto, después que inicia el menú en bucle hasta que se elige la opcion de salir del programa.