Trabajo Final

Gaming House Casino

24 ABRIL 2023

Integrantes:
Eric Schenone
Josefina Irigoyen
Nahuelquir Laura

Introduccion

Programacion Orientada a objetos

En este segundo cuatrimestre en la carrera Full stack comenzamos a conocer y aprender la programación orientada a objetos (POO).
Es un paradigma de programación que se enfoca en la creación de objetos que contienen datos y métodos para manipular esos datos. Es un modelo de programación que se basa en la idea de que un programa de computadora se construye a partir de objetos que interactúan entre sí para realizar una tarea.
En POO, los objetos son instancias de una clase, que es una plantilla que define las propiedades y métodos que todos los objetos de esa clase tendrán. Los objetos interactúan entre sí mediante el intercambio de mensajes, que son llamadas a métodos de otros objetos.
La programación en objetos se basa en cuatro conceptos fundamentales: encapsulamiento, herencia, polimorfismo y abstracción.
El encapsulamiento se refiere a la capacidad de ocultar los detalles internos de un objeto, protegiéndolo de modificaciones externas.
La herencia permite que una clase pueda heredar propiedades y métodos de otra clase, lo que facilita la reutilización de código.
El polimorfismo se refiere a la capacidad de un objeto para tomar diferentes formas o comportarse de manera diferente en función del contexto.
Por último, la abstracción se refiere a la capacidad de representar conceptos complejos de manera simplificada, lo que permite que el programador pueda enfocarse en los aspectos esenciales del problema. Este ultimo no fue requerido para el trabajo.

“La POO es ampliamente utilizada en la programación moderna, ya que permite la creación de programas más modulares, flexibles y escalables.”.

Pautas del Trabajo:

Crear un Casino que contenga:

• Tiene un juego tragamonedas con distintas variantes siempre cumpliendo la misma funcionalidad, que dependen de la temática y del valor mínimo de apuesta. Deberán hacer 2 variantes (herencia).

• Además debe tener otros 2 juegos, como mínimo, elegidos por ustedes, cada uno con los valores de apuesta y la probabilidad de ganar.

• El programa debera proveer funcionalidades para elegir un juego y realizar una apuesta, obteniendo el resultado de la misma (ganó X plata o perdió).

• Incoporar en el trabajo práctico una clase abstracta o una interfaz (o ambas, a elección del grupo)

• Generar el UML (diagrama de clases) correspondiente al programa que van a desarrollar

• Pueden agregar todo lo que consideren necesario para el correcto funcionamiento del programa, siempre que se cumplan los requerimientos mínimos exigidos.

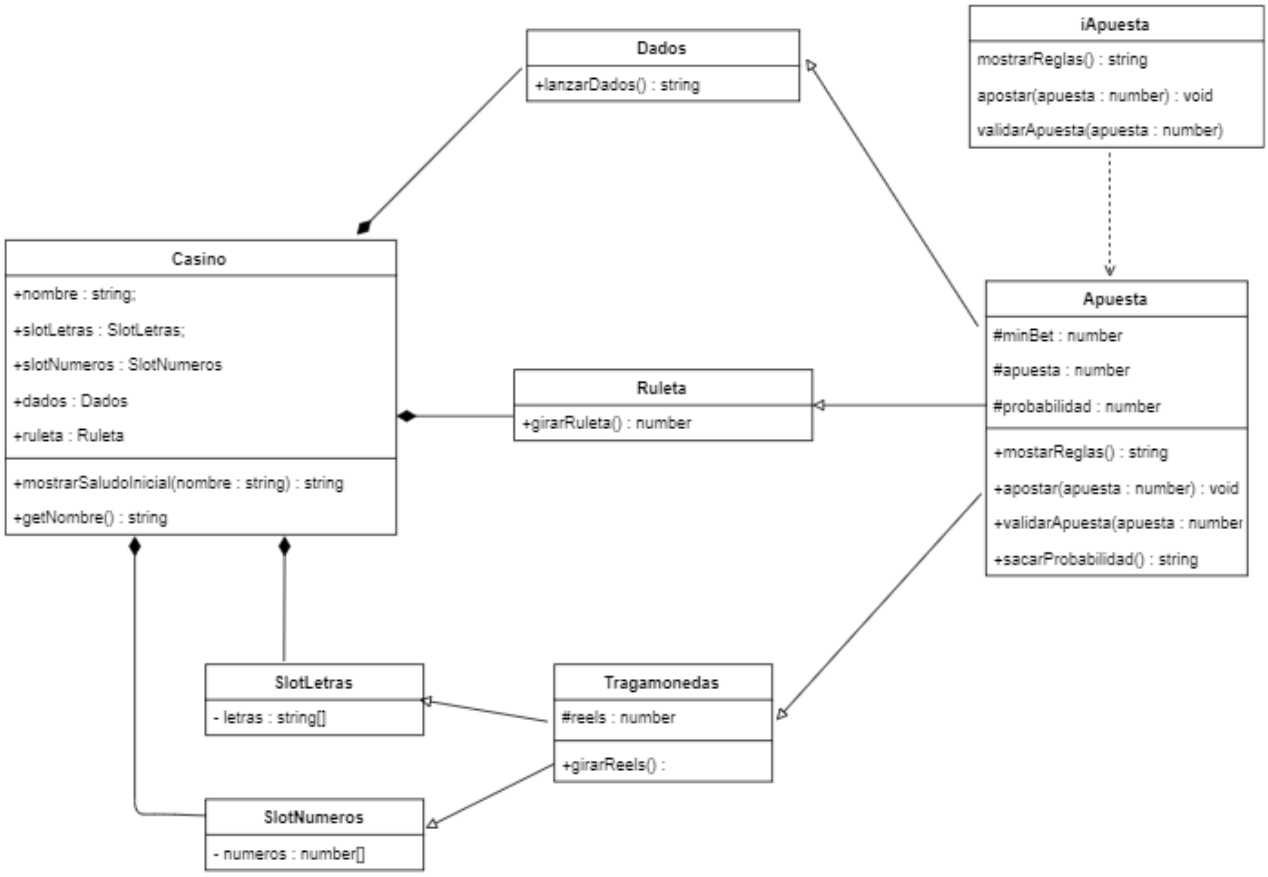
• Se debe realizar en un Proyecto NPM Nuevo.

• Grupos de 4 integrantes como máximo

• Fecha límite de entrega: 24 de ABRIL 2023

• Modo de entrega: el trabajo debe ser subido a un repositorio nuevo en github (un repositorio porequipo)

Presentacion del UML



Desarrollo

En dicha actividad desarrollamos la propuesta de la siguiente manera: En primera instancia desarrollamos el UML de manera que pueda demostrar la estructura de nuestro trabajo.

Para ello nos reunimos via meet y comenzamos a organizarlo .

Luego comenzamos a desarrollar el codigo utilizando Visual Code.

Denominamos al Casino “Gaming House” Casino y el mismo cuenta con 9 carpetas, las cuales desarrollaremos:

- **iApuesta.ts:** El archivo es una definición de una interfaz . Esta interfaz describe los métodos que un objeto debe implementar para poder ser considerado una apuesta en un juego de casino. La interfaz define tres métodos que deben ser implementados:
  "apostar()": Este método toma un argumento "apuesta" de tipo numérico que representa la cantidad de dinero que el jugador quiere apostar en el juego. Este método no devuelve nada, pero se utiliza para registrar la apuesta del jugador y actualizar el saldo en consecuencia.
  “mostrarReglas()”: Muestra las reglas correspondientes a cada juego.
  “validarApuesta()”: evalua si la apuesta ingresada por el usuario es menor a la apuesta minima.

- **apuesta.ts:** Este archivo es una super clase que se utiliza para implementar la funcionalidad relacionada con las apuestas. Tiene tres atributos: apuesta, minbet (minimo de apuesta) y probalidad de ganar.
- **dado.ts:** Este es un archivo que define una clase llamada "Dados". Esta clase extiende la clase "Apuesta" y tiene tres métodos: "jugar()", "apostar()", y "probabilidadDeGanar()". El método "jugar()" genera dos números aleatorios que representan los resultados de dos dados. El método "apostar()" toma un parámetro de número que representa la cantidad de la apuesta. Ademas hereda el metodo de “validarapuesta()” de la clase Apuesta. El método "probabilidadDeGanar()" calcula la probabilidad de ganar en el juego de dados y muestra el resultado en la consola.

- **ruleta.ts:** Este es un archivo que define una clase llamada "Ruleta" y extiende de clase Apuesta. Hereda todos los metodos de la super clase. Ademas tiene un metodo “girarRuleta()” que simula girar la rueda de una ruleta y devuelve un numero aleatorio.

- **tragamoneda.ts:** Este es un archivo que define una clase Tragamonedas que extiende la clase Apuesta. La clase Tragamonedas tiene un atributo propio que es “reels”. El método “girarReels()” no tiene implementación, ya que en cada subclase se aplica polimorfismo.

- **slotLetras.ts:** Este es un archivo que define una clase llamada "SlotLetras". Esta clase extiende la clase "Tragamonedas" y tiene dos atributos propios: "letras" y "reels". La propiedad "letras" se inicializa con un arreglo que contiene las letras "A", "B" y "C". El método "girarReels()" genera un resultado aleatorio seleccionando una letra aleatoria de la propiedad "letras" para cada reel en el juego. El número de reels está determinado por el valor del argumento "reels". Los resultados se almacenan en un arreglo y se muestran en la consola.

- **slotNumeros.ts:** Este archivo es una clase hija de “Tragamoneda” y se denomina SlotNumeros. SlotNumeros representa una máquina tragamonedas que tiene un número determinado de reels (dado por el argumento reels en su constructor) y en cada carrete se pueden mostrar cuatro números (1, 2, 3 y 4). Al igual que SlotLetras hereda los metodos de la clase padre.

- **casino.ts:** Es un archivo que define una clase llamada "Casino", que se compone de 4 clases: "Dados", "Ruleta", "SlotLetras" y "SlotNumeros" y un atributo privado que es nombre del casino.

- **app.ts:** Este archivo importa otros archivos, como "casino.ts", "dados.ts", "ruleta.ts", "slotLetras.ts" y "slotNumeros.ts", para crear instancias de diferentes juegos de casino. El programa primero solicita al usuario que ingrese su nombre y lo saluda. Luego, presenta una lista de juegos a elegir ("Dados", "Ruleta", "SlotLetras" y "SlotNumeros") y solicita al usuario que elija uno. Después de elegir un juego, el programa solicita al usuario que ingrese una apuest. Si la apuesta es suficientemente grande, el juego se

ejecuta y se muestra el resultado. Si la apuesta es demasiado pequeña, se muestra un mensaje de error.

Conclusion:

Consideramos que el codigo se puede mejorar en un monton de funcionalidades que por falta de tiempo y experiencia no pudimos hacer.

Se podria agregar al codigo cuestiones como:

- Crear funcion que calcule el saldo total.
- Crear Funcion para retirar el dinero.
- Mejorar el juego de Ruleta agregando la eleccion del color, apostar a mas de un numero, apostar a cuartos, etc.
- Crear un menu dentro del mismo juego que pregunte si volver a apostar o salir del juego.
- Mejorar el codigo para que sea mas legible.
- Crear una clase usuario.
- Crear el HTML correspondiente para visualizar el programa.
- Crear otros juegos posibles.