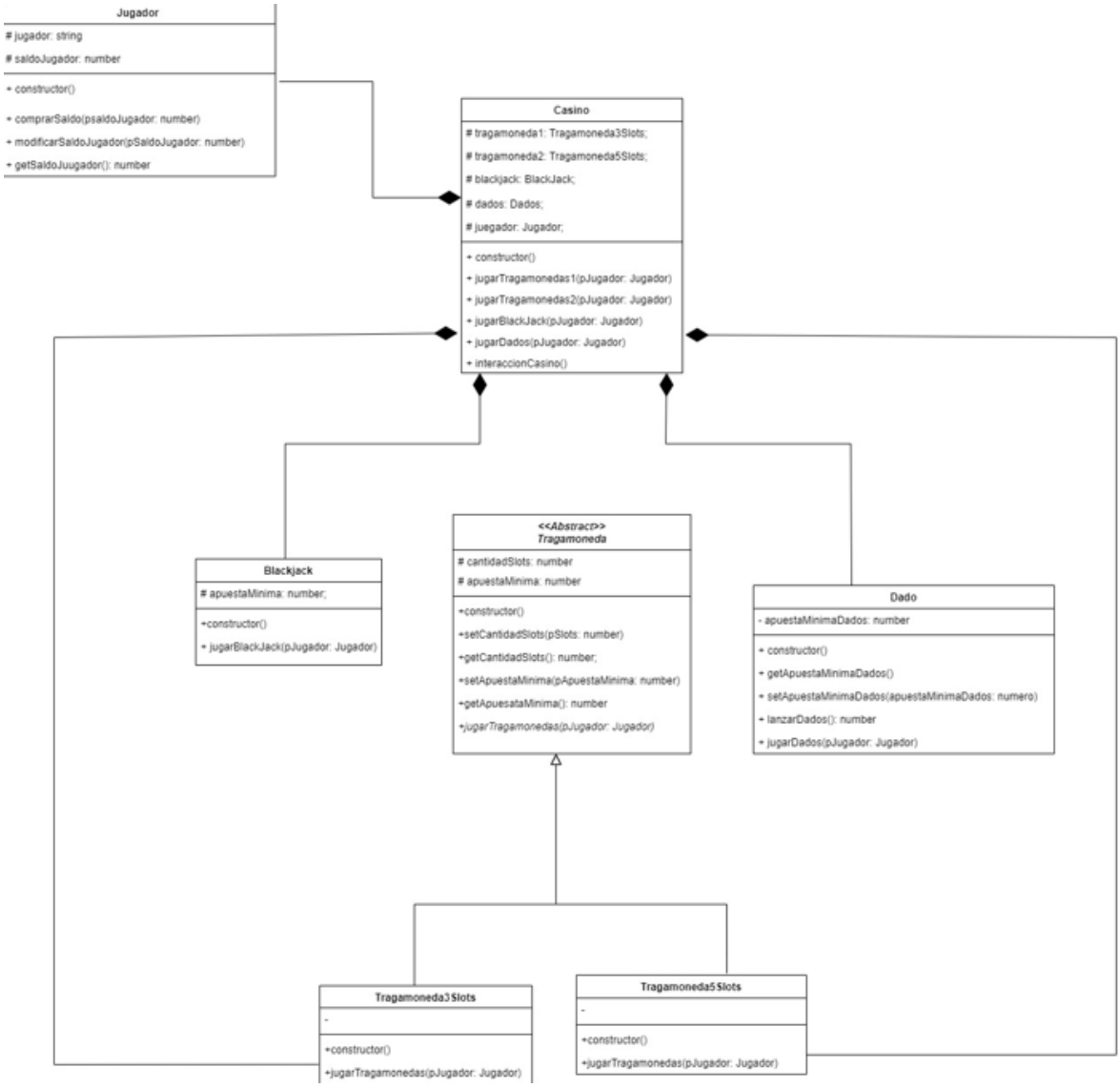
Trabajo Final EVALUATORIO de MODULO POO

Fecha de entrega MAXIMA: miércoles 9 de noviembre de 2022 a las 23.59

1. Integrantes

- Fernando Frias Gastaldi
- Kevin Santander Max
- Fabiola Horas Stevenson

2. Diagrama UML



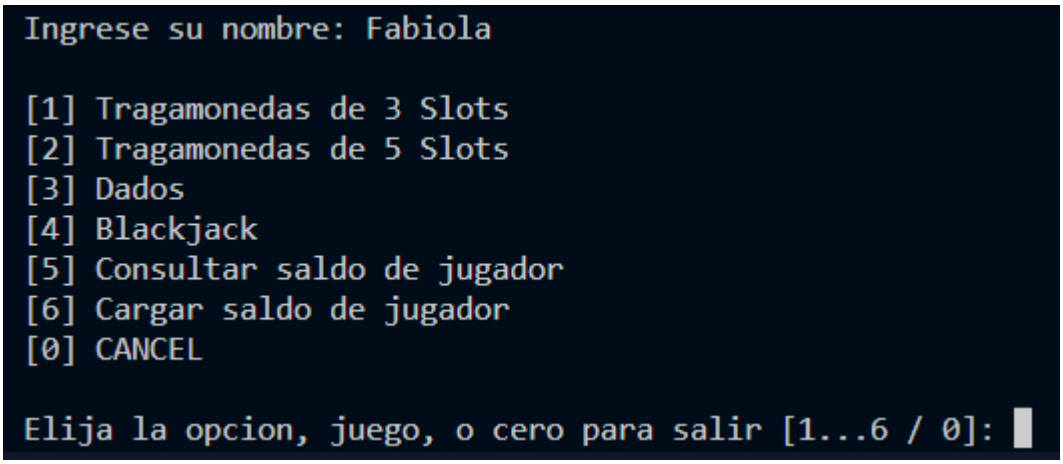
El UML se puede consultar en el siguiente link https://github.com/fabiolahorasstevenson/casino-poo/tree/CasinoPoo (Casino UML.jpg)

3. Desarrollo de trabajo integrador CASINO POO

El trabajo práctico realizado consiste en un programa que ofrece 4 tipos de juegos:

- Tragamonedas 3 Slots
- Tragamonedas 5 Slots
- Dados
- BlackJack

A continuación se puede apreciar el menú diseñado.



Además el sistema ofrece opciones de consultar saldo, cargar saldo de jugador, o cancelar en caso de que quiera salir del juego y no seguir con el mismo.

4. Librerías utilizadas

- **ReadlineSync**: es una librería que se utiliza para que el usuario pueda ingresar datos y valores por teclado. En este trabajo la mencionada librería se usó para distintas acciones de interacción con el usuario:
  - Al comienzo del programa para ingresar el nombre del usuario que jugará en los juegos del casino.
  - Durante el programa para elegir las distintas opciones que ofrece el casino, detalladas en el apartado 3.
  - Dentro de cada opción de juego elegida, si el usuario (jugador) tiene la posibilidad de definir si juega de nuevo o no.

- **@types/node**: es una librería que se utiliza para poder insertar valores al programa desde un archivo de tipo txt. Esta librería se utilizó para grabar un archivo de tipo log de cada console.log de cada interacción con el jugador.

5. Descripción del proyecto

El programa solicita al jugador en primer lugar el nombre, luego le muestra las opciones disponibles para interactuar con el casino, detallando cada uno de los juegos disponibles.

- ✓ Si elige Tragamonedas 3 Slots:
  - ➢ Se implementa el método JugarTragamonedas() el que pregunta en primer instancia si la apuesta es la mínima requerida por el juego, al momento de generar el objeto de tipo Tragamoneda3Slots.
  - ➢ Una vez validado esto el sistema ejecuta un método para generar un random entre 1 y 9, getRandomInt(min,max), recibe dos atributos uno que refleja el mínimo (1) y otro el máximo (9).
  - ➢ Luego compara si los valores son iguales, si son igual el jugador gana y suma la apuesta inicial. Si los valores no son iguales, el jugador pierde, pero si tiene saldo, puede seguir jugando.
- ✓ Si elige Tragamonedas 5 Slots:
  - ➢ El tipo de juego implementa el mismo método JugarTragamonedas(), solo que aplica polimorfismo y se redefine para 5 Slots.
  - ➢ Valida nuevamente el saldo y ejecuta también método para generar un random entre 1 y 9, getRandomInt(min,max).
  - ➢ Luego compara si los valores son iguales, si son igual el jugador gana y suma la apuesta inicial. Si los valores no son iguales, el jugador pierde, pero si tiene saldo, puede seguir jugando.
- ✓ Si elige Dados:
  - ➢ El tipo de juego implementa el método lanzarDados(), que usa el método getRandom(1,7) para emular el lanzamiento de un dado y que saque un valor entre 1 y 6. Los valores arrojados, se suman y se comparan con valores pre definidos:
    - Si la suma es igual a 7 y 11 el jugador gana.
    - Si la suma es igual a 2, 3 y 12 el jugador pierde.
    - Los valores que caen fuera de estos, pueden seguir jugando.
  - ➢ Si gana se ejecuta el método modificarSaldoJugador() que incremente el saldo del jugador inicial.
  - ➢ Si pierde, se ejecuta el mismo método.
- ✓ Si elige BlackJack:
  - ➢ El tipo de juego implementa el método el mismo getRandom(1,14) representando cada valor una carta. En cada vuelta del while se suma cada carta y no debe superar el valor de

21. Una vez que se suman las cartas generadas con el random, el juego consulta si el jugador quiere sacar una carta nueva, y vuelve a comparar con el valor de 21.
    - Si suma más de 21 el jugador pierde.
    - Si la suma es menor o igual a 21 y elige quedarse. El juego analiza en qué valor se quedó el jugador. El casino siempre intentará superar este valor.

- Si gana se ejecuta el método modificarSaldoJugador() que incremente el saldo del jugador inicial.
- Si pierde, se ejecuta el mismo método, que resta el valor apostado.