TRABAJO PRÁCTICO FINAL - SEGUNDO CUATRIMESTRE

MATERIA: PROGRAMACIÓN

TEMA: PROGRAMACIÓN ORIENTADA A OBJETOS.

CONSIGNA: Generar un programa en el cual se puedan realizar apuestas dentro de un casino.

Requerimientos:

Deberá tener 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 deberá proveer funcionalidades para elegir un juego y realizar una apuesta, obteniendo el resultado de la misma (ganó X plata o perdió). Incorporar 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.

PROFESORES:

- Marcelo Bettini.
- Braian Aued.

TUTORES:

- Rafaela Ruggeri.
- Carolina Vasconcello

NOMBRE DEL GRUPO: MURCIÉLAGOS
INTEGRANTES:

Andres Coda
Juan Tomas Lacave
Ezequiel Calderon
Juan José Curutchet

INTRODUCCIÓN:

El proyecto fue realizado reuniéndonos mediante la plataforma meet e intercambiando ideas mediante slack.

Inicialmente nos reunimos mediante meet y planteamos una idea conceptual de lo que queríamos realizar, construyendo un diagrama de clases, plasmando en él todas las clases con sus propiedades, funcionalidades y las interacciones entre ellas. Una vez realizado esto, dividimos las tareas entre los integrantes del grupo y comenzamos a trabajar en ellas de forma individual intercambiando ideas mediante slack. A medida que íbamos avanzando nos fuimos reuniendo para unificar todo lo que cada uno iba haciendo, ya que al momento no teníamos demasiado conocimiento sobre github.

Con el avance del proyecto fueron apareciendo diferentes ideas que hacen al mejoramiento del programa, que hicieron que el UML realizado en un principio tuviera que ser modificado. Más adelante obtuvimos conocimientos sobre github y subimos el proyecto a un repositorio para seguir trabajando desde ahí.

DESCRIPCIÓN:

Nuestro proyecto de Casino “La Viruleta” consta con diez clases -**Casino**, **Jugador**, **Pantalla**, **Cartas**, **Frutas**, **Tragamonedas**(abstracta), **TragamonedasFrutas**, **TragamonedasCartas**, **Dados**, **MayorOmenor**, una interfaz (**IFrutas**) y un archivo **index.ts** para correr el programa.

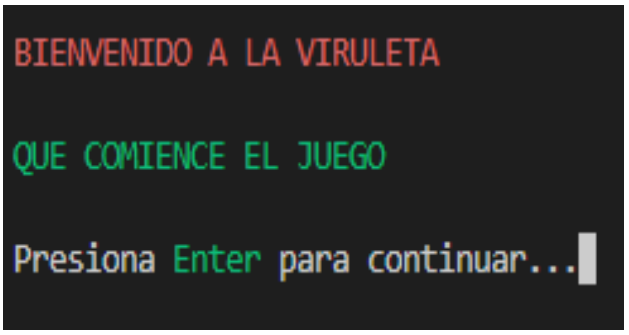
Paquetes utilizados:

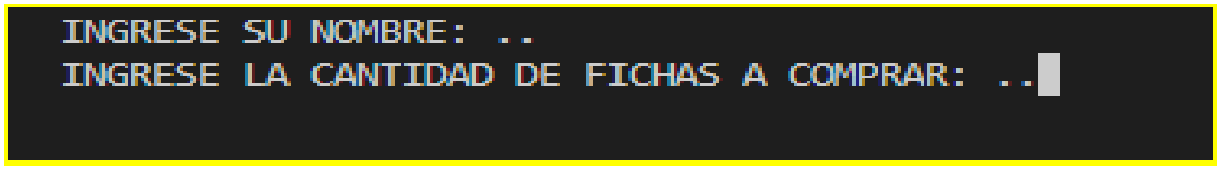
- readline-sync
- colors

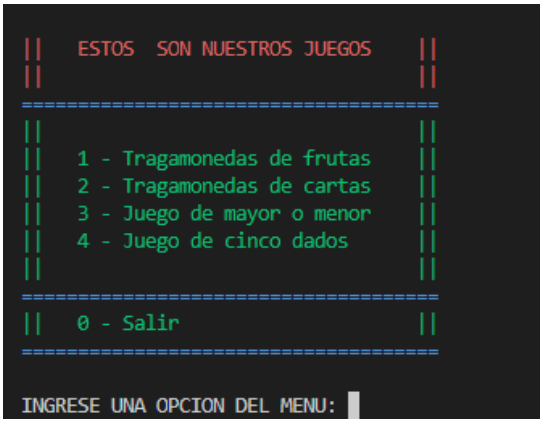
Descripción de las Clases:

- **Clase Casino**: la clase Casino es la entrada al casino, como la mesa de entrada, en ella se encuentra el nombre del casino, con los métodos para leerlo y modificarlo y a su vez está compuesta por los juegos. Cuenta con un método para inscribirse mediante el cual utilizando el paquete readlink-sync, el jugador ingresa su nombre, las fichas que desea comprar y comienza a jugar, luego este método también es el encargado de despedir al jugador. También cuenta con un método para que el jugador elija a través de un índice el juego al que quiera jugar. Esta clase a su vez utiliza las clases **Jugador** para setear el dinero y el nombre, y **Pantalla** para mostrar los mensajes e interactuar con el jugador.
- **Clase Pantalla**: Es la encargada de mostrar todos los mensajes y el entorno visual por consola.

Se puede ver en el ejemplo siguiente que la clase Pantalla es responsable de mostrar mensajes e interactuar con el jugador en la consola.







Se utiliza el paquete "colors" para agregar color a los mensajes en consola. En resumen, la clase Pantalla proporciona una forma clara y organizada de mostrar información al usuario y obtener información del usuario a través de la consola.

- **Clase Tragamonedas:** esta es una clase abstracta de la cual heredan las clases **TragamonedasCartas** y **TragamonedasFrutas** está compuesta por la clase **Jugador,** contiene la propiedad nombre y los métodos: entregarPremios, que es el encargado de generar los mensajes de entrega de premios y llamar a los métodos para saber si gano o perdio y luego mostrar en pantalla la cantidad de fichas que quedan; y juego que es el encargado de llevar adelante el juego interactuando con el jugador a través de la pantalla. Además posee los métodos abstractos mostrarEnPantalla,

setTirada, reglamentoJuego, calcularPremio y cargarGuias que son implementados por las subclases.

- **Clase TragamonedasCartas**: extiende de la clase **Tragamonedas** que simula un juego de tragamonedas con cartas. Esta clase está compuesta por Mazo y tiene propiedades como la tirada actual del juego (tirada), y una serie de guías que contienen combinaciones de cartas (guía). También tiene métodos para cargar las guías, establecer la tirada actual, mostrar las cartas en pantalla y para verificar si se han ganado premios (verificar Cuatro Iguales y verificar Línea).El método calcularPremio aprovecha estas verificaciones para determinar cuánto ganó el jugador en la última tirada.En resumen, esta subclase extiende la funcionalidad del Tragamonedas para simular un juego de tragamonedas con cartas y verificar las combinaciones ganadoras.
- **Clase Tragamonedas Frutas:** extiende de la **clase Tragamonedas** y agrega la funcionalidad del juego de tragamonedas con frutas. Tiene un array, llamado guía que contiene las diferentes frutas y un array, llamado tirada que contiene las frutas que se mostrarán en la pantalla al momento de realizar la jugada. También cuenta con los métodos necesarios para cargar la guía de frutas, generar una tirada de frutas aleatoria, mostrar la tirada en pantalla y verificar si hay premio según las combinaciones de frutas obtenidas en la tirada. En caso de haber premio, se devuelve un valor numérico que representa el premio obtenido según las combinaciones obtenidas en la tirada.
- **Clase Dados**: esta clase tiene las propiedades dados y nombre, a su vez está compuesta por la clase **Jugador** y contiene los métodos: **calcularPremio** que es el encargado de calcular el premio obtenido según la combinación de los dados; **reglamentoJuego** que se encarga de pushear el reglamento a un arreglo para luego mostrarlo por pantalla; **jugar** que es el encargado de llevar adelante el juego interactuando con el jugador a través de la pantalla; **calcularProbabilidadCuatroIguales**, **calcularProbabilidadEscalera**, **calcularProbabilidadCincoIguales**, **calcularProbabilidadTresDosIguales**, que son los encargados de calcular las probabilidades según la combinación de dados obtenida; **probabilidad** que se encarga de pushear los datos obtenidos de los métodos anteriores a un arreglo para luego mostrarlos por pantalla; **tirarDados** que carga un arreglo con los cinco dados de forma aleatoria; verificarGenerala, **verificarPoker**, **verificarEscalera** y **verificarFull** que se encargan de realizar las verificaciones para comprobar si se obtuvo alguna combinación ganadora y son utilizados por el método **calcularPremio** para calcular el premio obtenido.
- **Clase MayorOmenor:** esta clase tiene las propiedades título, carta1, carta2 y está compuesta por **Mazo** y **Jugador**. Cuenta con los métodos: **reglamentoJuego** que se encarga de pushear el reglamento a un arreglo para luego mostrarlo por pantalla; **cantidadCartasMazo** que se encarga de calcular las cartas en el mazo restando al mismo las cartas del descarte;

probabilidadMayor que calcula la probabilidad que salga una carta mayor a la que está en la mesa; **probabilidadMenor** que calcula la probabilidad que salga una carta menor a la de la mesa; **probabilidadComodin** que calcula la probabilidad de obtener un comodín; **verificaMayor** devuelve true si la segunda carta es mayor que la primera que está en la mesa y false en cualquier otro caso; **verificaMenor** devuelve true si la segunda carta es menor que la primera y false en cualquier otro caso; **verificaComodin** devuelve true si la segunda carta es un comodín y false en cualquier otro caso; **calcularPremio** verifica que la segunda carta no sea un comodín, si lo es da una nueva carta porque el jugador perdió y verifica si la apuesta fue a mayor o menor y retorna el premio en caso de ganar; **entregarPremio** genera el mensaje para entregar premio y llama a los métodos para saber si perdió; **probabilidad** se encarga de pushear los datos obtenidos de los métodos anteriores a un arreglo para luego mostrarlos por pantalla; **juego** es el encargado de llevar adelante el juego interactuando con el jugador a través de la pantalla.

- **Clase Frutas**: tiene como propiedad a nombre y los métodos **getNombre** y **setNombre** e implementa la interfaz **IFrutas** y es utilizada por la clase **TragamonedasFrutas** para cargar las guías con los nombres de las frutas.
- **Clase Jugador**: compone las clases de todos los juegos,se encarga de manejar las apuestas, el dinero del jugador y de elegir el juego al que desea jugar.Tiene como atributos el nombre, el dinero y la apuesta que va a realizar
- **Clase Mazo:** compone las clases **TragamonedasCartas** y **MayorMenor,** crea un mazo para ambos juegos.
- **Clase Cartas:** compone la clase **Mazo,** setea cartas para construir el mazo.

Consta además de una archivo index.ts que instancia al jugador sin datos, o datos vacíos, e instancia al casino, pasandole como parametro el jugador, recién instanciado, y es al archivo con el cual corre el programa.