

## **Informe para cliente:**

GameEmulator es una estructura que brinda mucha facilidad de ejecución por su simplicidad. Es una aplicación con un uso bastante intuitivo.

La primera pestaña presenta dos opciones:

1- Ir a la pestaña donde se almacenan los juegos incluidos en la plataforma

2- Revisión de ajustes (no desarrollado)

Al seleccionar la primera opción es evidente que presionando el botón específico te inmersionás en el juego seleccionado.

### **\*PointingParty:**

PointingParty es un juego muy sencillo, trata de la ejecución, con la cantidad de equipos conformados por jugadores que se deseen, de un exclusivo torneo por clasificación individual, donde los jugadores (de tipo RandomPlayer y GreedyPlayer) escogerán un número entre 0 y 100 y dado el número escogido se puntuará multiplicando por 1000 su valor (se puede apreciar la tabla una vez finalizada la ejecución) (hecho para poner a prueba el Torneo de Clasificación Individual, además de la extensibilidad d la plataforma Emulator al añadirle otro juego(ver informe para desarrolladores ))

### **\*TicTacToe:**

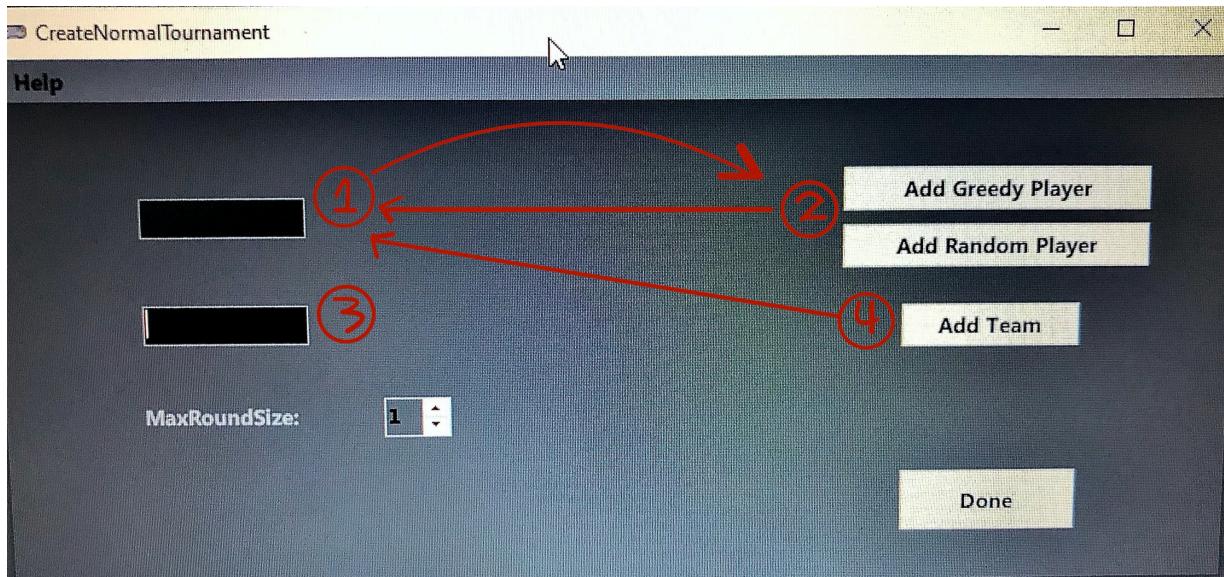
TicTacToe es actualmente el juego "insignia" de la plataforma. Consta de un menú bastante sencillo d 3 pestañas:

#### **1- Game:**

En esta pestaña se podrán añadir las especificaciones de las formas a ejecutar, por ejemplo, la creación de una partida rápida, un torneo en forma de "liguilla" o incluso un torneo "por el Título".

Nota: En la creación de cualquiera de los anteriores serás remitido a un espacio donde añades los equipos con sus respectivos jugadores. En este debes añadir primero el nombre del equipo(consola de color negro de abajo), para luego, añadir todos los jugadores que deseas ( nombre a la izquierda arriba ) ( a la derecha escoger tipo de jugador cuando se añade ). Cuando se haya completado el equipo con la cantidad de jugadores deseados, puede insertar presionando el botón "Add Team" para q se vacíe ese espacio, y por tanto, añadir el siguiente equipo si se desea.

Paso a paso:



- 1- Rellenar nombre primer jugador
  - 2-Añadir primer jugador al conjunto del primer equipo
  - 2 —> 1 para añadir siguiente jugador ( a decisión de usted ) ( repetir el proceso las veces que se estimen )
  - 3- Rellenar nombre primer equipo
  - 4- Añadir primer equipo adjuntándole los jugadores previamente agregados
  - 4 —> 1 para añadir jugadores del siguiente equipo a poner (repetir el proceso las veces que se estimen)
- MaxRoundSize indica la cantidad máxima de rondas q se ejecutarían en cada partida del torneo ( ejemplo : 3 se jugaría el clásico "de 3 ganar 2", o sea, 3 juegos a lo sumo)
- Luego del proceso concluido, presionar Done.

## **2- Execution:**

En este paso, una vez definidos los equipos y forma de ejecutar la competición, se brinda toda la flexibilidad posible para su ejecución, puedes seleccionar más d 5 modalidades, además de los botones "Pause" y "Stop" con la facultad de poder presionar cada opción incluso, durante la ejecución.

## **3- Help:**

Incluye el Guide for users (no desarrollado)

Se puede apreciar dos ventanas de valores modificables:

### **Timer Interval:**

Te permite modificar el tiempo ( en segundos ) que deseas mantener entre una jugada y la siguiente, que, para añadirle más libertades a la plataforma, permite hacerlo incluso durante la ejecución (siempre y cuando sea pausado previamente)

### **Notifications Intensity: ("Bitácora")**

Te permite escoger cuánta intensidad de notificaciones deseas, desde las básicas como lo son las jugadas representadas en una tabla ( expresando posiciones, integrantes del equipo, etc..) ( al igual que los resultados ), hasta las más recónditas, como cuando te notifica si ejecutas una jugada (Play Step en la

pestaña Execution del menú). Con mayor libertad de notificaciones se tiene mejor experiencia en el juego. ( Recomendado nivel máximo)

Estas notificaciones están integradas en las notificaciones del sistema.

Luego, se tiene un espacio reservado para ver quién es el actual campeón, así como, evidentemente, la tabla del TicTacToe a la izquierda.

¡Espero que disfrute su experiencia!

## **Informe para desarrolladores:**

El proyecto Emulator es una plataforma lo más extensible posible, para que, primero q todo, se le puedan añadir todos los juegos que se quieran siempre y cuando se cumplan una serie de parámetros.

La parte "lógica" de Emulator cuenta con 4 carpetas:

### **1- Tournament:**

En esta, se encuentra la clase abstracta "Tournament" que tiene un comportamiento con propiedades y métodos (información, en general) q utilizarán siempre cada uno de sus hijos (como lo son los equipos que participan, su método abstracto de ejecución, y elementos post-torneo como su ganador o los resultados en general). De ella por el momento heredan 3 clases:( 3 torneos implementados hasta el momento )

#### **•LeagueTournament:**

Torneo de "liguilla" descrito en el proyecto, también conocido como Dos a Dos

#### **•IndividualClasificationTournament:**

Torneo por clasificación individual descrito en el proyecto ( empleado en el juego PointingParty exclusivamente)

#### **•KingTitleTournament:**

Torneo por el título, donde todos sus participantes retan al actual campeón, ya sea el campeón por defecto (en un principio) o el ganador del último torneo ejecutado (posterior a ejecutar completamente un torneo)

Para añadir un torneo de algún tipo específico que usted desee basta con heredar la clase Tournament, implementar la clase abstracta, y completar según desee el tipo de torneo utilizando los métodos y propiedades ya establecidos.

### **2- Player:**

En esta carpeta, se encuentra la clase Team, estructura que contiene el conjunto de jugadores (equipo), con propiedades como su Id (string del nickname) y otras que se usan internamente.

También está la clase Player, abstracta, padre de todo tipo de jugadores, en esta se encuentra el método abstracto Play<T> que devuelve un T (jugada, trabajado con genericidad).

Existen dos carpetas más, BasicPlayer y ComplexPlayer ( dentro están sus respectivas clases abstractas, heredando de Player). Con idea de hacer el proyecto lo más extensible posible es que se definen estas nuevas clases

abstractas, con peculiaridades cada una, luego nuestros famosos GreedyPlayer y RandomPlayer heredan de BasicPlayer, que tienen además de lo que tiene un player, un tipo "Random" protected en el padre que implementarán, puesto que en mi concepto de BasicPlayer no existe ningún tipo de decisión sabia, simplemente si hay que decidir algo, se decide "random", o como mucho, lo más urgente en el momento, como es el caso de un "greedy".

La clase ComplexPlayer se agregó con la idea de heredar de ella como unos tipos de jugadores inteligentes (no agregados hasta el momento, pero contando con una extensibilidad más detallada).

Para incluir un tipo de jugador basta con heredar de alguna clase como BasicPlayer o ComplexPlayer siempre que cumpla con sus conceptos. Se pide que si se desea establecer otro patrón de jugadores con características diferentes de las dos anteriores, solo ahí se herede este tipo en una clase abstracta directamente desde Player, y después se defina el tipo específico heredando desde esta nueva (para seguir cumpliendo con la estructura de herencia definida en el proyecto).

### **3- Match:**

Esta carpeta contiene únicamente a esta clase, que es la encargada de al recibir una cantidad N de equipos en la partida, determinar las veces que se necesiten jugar hasta que haya un ganador de la partida.

No incluye una estructura de herencia definida como las anteriores porque bajo mi consideración no hay tipos de partida, simplemente existen cantidades diferentes de rondas que insertar(maxRoundSize incluido en parámetros del constructor).

### **4- Game:**

En esta se encuentra, primero que todo, la clase abstracta Game.

Esta implementa la interfaz I Enumerable<Team>, puesto que se encarga de mover en el juego los turnos del equipo que le toca ( básicamente ). Tiene métodos y propiedades abstractas, que se les hará override una vez tengamos heredada la clase del juego específico a añadir en la dinámica del proyecto ( propiedades abstractas booleanas como GameOver y Draw, entre muchísimas más, que se han podido abstraer del comportamiento general de cada juego, es decir, son comunes en TODOS). Tiene embebida la clase GameEnumerator como parte de la implementación de la interfaz utilizada.

Evidentemente, se puede apreciar que para añadir un juego basta con heredar de la clase anterior (como ha ocurrido en los casos de los dos juegos ya implementados(explicados en el informe para clientes )), pero: ¿ qué hace diferente al resto la "dinámica de herencia" de las clases relacionadas con Game ? R/ El uso de las interfaces.

Para hacer el GameEmulator muchísimo más inteligente, se tiene una carpeta Interfaces, con la idea de que a medida que se vayan añadiendo nuevos juegos, se

vayan agregando interfaces y así, factorizando elementos de juegos específicos, para que cada vez que se añada un hijo de Game, crezca la posibilidad de "hacer más cómodo" el proceso, es decir, se pueda aumentar la posibilidad de tener concebido un concepto parecido en un juego anteriormente agregado. Por ejemplo: el TicTacToe utiliza de esta carpeta la interface IBoard ( creada para juegos con tablero ) con propiedades como los arrays de direcciones, cantidad de filas y columnas, etc..., que son utilizados en cada uno de los juegos que utilizan tableros, por tanto si se quiere añadir el juego Ajedrez, por ejemplo, se implementa esta interfaz, y con además la herencia del padre Game, vas a tener prácticamente el "esqueleto" de tu juego, con una precisión muy alta, que como anteriormente se plantea, crece a medida que crece el emulador.

Muchas gracias por su atención, espero que disfrute el proyecto.