Proyecto de Programación
>Breve Recuento de la Creación del Trabajo

Con el objetivo de completar nuestro proyecto de domino nos propusimos como paso inicial ,la creación de un programa funcional y simple que simulara un partido de domino, una vez creado el programa se utilizo como estructura principal para nuestro proyecto. Durante el proceso de creación fuimos notando que existían partes del programa que representaban similitudes en sus objetivos por lo que podían se agrupadas en un mismo cojunto que representara estas similitud diferenciados nada mas por sus peculiaridades. Uno de nuestro primero cambios al primer programa fue la creación de una de nuestra clases principales del proyecto principal "Game Rules" con esta case tuvimos el objetivos de lograr un punto de control de todo el desarrollo de las futuras interfaces que implementamos al tener todo este proceso de implementación por solo una entidad excluyendo asi futuros problemas a la hora de modificación y adaptación del proyecto. Dentro de esta clase se fue implementando paulatinamente diversas interfaces que describían un comportamiento en particular de nuestro simulador de domino el cual podia ser variado , modificado o alterado en fechas posteriores. Entre ellos podemos mencionar "IValidateMove" como su nombre lo indica se encargará de definir cuando se puede considerar validada una jugada, al inicio pensábamos en mantener variable esta información pero después de dialogar entre nosotros y consultar la de otros compañeros pensamos que no siempre podemos tener nuestra perspectiva en lo que es comúnmente conocido a cambiar pues al igual que nuestro actual condición de vida que nos da sorpresas en cada momento debemos poder adaptarnos y reaccionar estos cambios constantes por ellos decimos incluir esta variación ademas de otras mas peculiares como la posibilidad de definir la caras de las fichas con las que se jugara pues aunque se conoce que el domino se juega con números quien dice que alguien no desea probar jugar uno con los nombres de su familia como caras jejeje.
Aunque debo de aclarar que el proyecto final aun si acepta la edición de la cara de las fichas este seguirá trabajando bajo un sistema de numero por lo que se le asignará valor 0, esto como muchos se darían cuenta provocaría problemas pero solo, dando un informe de aviso al informar de la situación, claro esta que aunque actualmente no acepta un sistema para definir un valor particular a palabras puede ser incluido en caso que fuera deseado. Continuando con la historia de nuestro proyecto se incluyeron ademas interfaces que permiten variar detalles como el resultado final de la simulación al definir diversas maneras en el que este puede ser declarado , dando como ejemplo el modo normal de suma de puntos de cada equipo. También esta el método para declarar la finalización del partido , el método que define el orden en que juegan los participantes y el que declara el valor de cada ficha. Estos fueron los diversos comportamiento que definimos como posibles información mutable. También esta la creación de nuestros jugadores estos lo hicimos de tal modo que funcionen si la necesidad de saber nada sobre el funcionamiento de las reglas del juego. Siendo la única información que los relaciona con la mismas la cantidad de jugadores que existen. Entre otras de las clases importantes esta "EstadoBase" que rige el desarrollo de la simulación al cambiar las diferentes acciones de cada jugar o ficha en dependencia de la circunstancias, actualiza el tablero y cambia las fichas que tiene cada jugador. regulando constantemente el juego y guardando cada jugada o suceso en un registro el cual puede ser usado posteriormente en cualquier necesidad. Tenemos nuestra clase encargada de regular todo el comportamiento de las fichas esta en un principio se creo como una interface de la cual heredaban tres clases que definían definían jugada , un pase o la jugada inicial. pero después de ver que existían incongruencia en este método , pues su implementación era ineficiente . se decidió la creación de una clase con tres constructores que regulan los cambios antes mencionados. Ademas del desarrollo de la parte logica o backend, También estuvo el desarrollo del frontEnd o interfaz gráfica , aquí se debe hacer énfasis que se paso mucho trabajo en un principio para lograr el solo hecho de seleccionar una plataforma visual que permitiera desarrollar lo deseado pues en un principio solo teníamos conocimiento de windowForm pero al trabajar en un sistema operativo incompatible con la app tuvimos que indagar en diferentes soluciones. se nos recomendó plataformas de paginas web pero para poder aprender de las misma solo podíamos descargar

videos instructivos de Internet detalle el cual no imposibilito usar dicha via. Al final se uso una plataforma llamada Raylib.Csl. la cual aunque no es de las mejores era relativamente mas facil de aprender u obtener que otras opciones. Para la parte visual al tener pocas herramientas en nuestra manos en este nuevo campo de desarrollo inexplorado, creamos una simple app para visualizar el proyecto. Aquí usamos una máquina de stack el cual en el apoyo de una clase abstracta que definía la acción que puede realizarse en pantalla se creo varias clases interconectadas ,las cuales cada una agrupaba un tipo de acción con la que se podia interactuar, ejemplo de esto es "Setting" en el cual esta definidos los diversos aspectos generales que se podían cambiar, en este caso los players y las GameRules. estos dos antes mencionados También definían clases que permitían definir las especificaciones de cada jugador o regla del juego. se uso cuadros interactivos donde se podia escribir información especificada o botones predefinidos para interacturar con algunas opciones. se creo una clase encargada de controlar las opciones elegidas y permitir la inicialización de la simulación y visual al enviar todos los datos a un método que comprobaba que se pudiera iniciar el juego con las opciones elegidas después de todo es imposible declarar un total de 50 fichas y 6 jugadores e iniciar cada jugador con 10 fichas cada uno. Por ultimo esta la clase "Game" esta es la que recibe el registro de la simulación y la visualiza en pantalla , se debe especificar nuevamente que la parte visual fue el aspecto mas desafiante por la falta de herramientas de aprendizaje y la dificultad de comprensión de la usada para aprender. al tener datos predefinidos para el uso online , obligando a dedicar buen tiempo a redefinirlos en offline para su comprensión.
Finalmente luego de nuestra primera exposición fueron se 锟絞 lados algunas deficiencias y se hicieron varias sugerencias, estas fueron en gran parte acatadas haciendo cambios como permitir
al al programa adaptarse a nuevas implementaciones de las interfaces sin tener que cambiar ningún aspecto de la interfaz gráfica , aumentar la cantidad de información que la interfaz gráfica
ofrece a los usuarios, crear clases de tipo estrategia separando esto de los jugadores, entre otros.
> Jerarquía de clase

GameRules: Clase que contiene todas las propiedades necesarias para hacer funcionar el programas cuyo valor se asigna en su constructor dichas propiedades son de los tipos siguientes:

* IvalitateMove: Interface que contiene un método denominado Valitate el cual recibe una jugada y el estado actual del juego y regresa un bool si la jugada realizada es valida o no.
* BaseState: Clase que contiene las caras activas en la mesa , un registro de jugada y un método actualizar que recibe una jugada y al jugador que la realizo y actualiza las dos propiedades anteriores
* List(Token): Una lista de fichas
* IOrder: Interface que contiene un método Next que recibe una lista de jugadores y devuelve un int indicando el indice del próximo jugador.\
* controladorDeDesarrollo: Clase que controla los jugadores que se pasan y las caras que no poseen.El método no lleva detectado, es el único usado activamente en el programa , tiene otro método que actúan sobre sus propiedades de quien no lleva y que es lo que no llevan que podrían ser útiles en el futuro.
* IvalorateToken: Interface que contiene un método que recibe una ficha y regresa su valor.
* IgameFinisher: Interface cuyo método recibe información del estado actual del juego y regresa un bool indicando si este ya termino o no.
* Iresult: Interface cuyo método indica el ganador del juego.

IDescribable: En general todas las interfaces la usan. Permite obtener información de las implementaciones para proporcionarles al usuario

IStrategy: interface que modifica un array de enteros en dependencia de otro array de moves del mismo tama 锟給

Player: Clase abstracta que contiene dos propiedades un entero que actúa como nombre del jugador y una lista de fichas. Ademas poseen un método asignar que le da valor a la lista de fichas y otro método abstracto MoveActual en el que se regresan una jugada de entre una lista de jugada validas. Este método MoveActual funciona con una lista de IStategy que actúan sobre un array numérico quien indica cual es la jugada a regresar. Por tanto la única diferencia entre un jugador y otro es que elementos contiene la lista de IStrategy.

Token: Clase que contiene un array genérico que indica las caras, si bien no fue posible crear un juego con mas de dos caras mantiene esa posibilidad futura.

GetGameRules: Clase que contiene dos métodos que regresan objetos GameRules y ProgramedRules respectivamente. Ademas poder otros métodos cuya función es proporcionar listas de las implementaciones de cada una de las interfaces, por tal motivo al crear una nueva clase que implemente estas interfaces se debe agregar a su método correspondiente en GetGameRules.

ProgramedRules: Contiene las listas de las implementaciones de las interfaces asi como cual de ellas se va a usar.

ProgramedPlayers: Contiene una lista de los jugadores que se han implementado.

TokenCreator: Clase que usa Un método recursivo para generar una listas de fichas en dependencia de las posibles caras introducidas previamente en la interface gráfica , se usa dentro de getgamerule,

Playercreator: Clase que devuelve una lista de jugadores en dependencia de la lista que indica las ia usada y el equipo seleccionado. También permite la creación de jugadores de IA Random.

Auxiliares: Clase que se dividió en dos debido a su funciones tan diferentes
* la primera parte como su nombre indica contiene métodos auxiliares que se usan en diferentes partes del programa, el mas importantes de estos es el método mezclar que permiten alterar el orden de una lista de fichas de forma random el resto son en su mayoría métodos para imprimir en consola que si bien no son útiles para el funcionamiento del programa si lo son para el mantenimiento y detección de errores.
* la segunda parte contiene el método validate setting que indica si la configuración introducida en la interface gráfica es valida de serlo llama a los métodos de las clases getgamerules y players creators. esto se realizo para evitar ejecutar el método recursivo dos veces y es posible trasladarla a una clase particular en el futuro.

Game: Clase que contiene un método single con un bucle white donde todo el proceso del juego se corre siguiendo las regulaciones establecida por un objeto Gamerules y siguiendo las indicaciones de un arreglos de jugadores.

> Los 5 Puntos variables

* #1 Caras de las fichas: El programa completo se establece de tipo genérico, si bien actualmente solo trabaja con string ofrece todo tipo de posibilidades futuras
* #2 Condición para que una jugada sea valida: Se pueden establecer la condición normal , la condición de ser mayor en uno , mas otras variaciones
* #3 Orden variable: El jugador en turno se indica por un método que para nada esta obligado a seguir un orden predefinido.
* #4 Valor de las fichas : El valor de cada ficha imprescindible en la actuación de los jugadores y en la determinación del ganador También se puede cambiar según se desee
* #5 Condición de finalización: Cada vez que un jugador termina su jugada se analiza el estado

dell juego para comprobar si ya termino, y su condición para hacerlo es variable.
* #6 Forma de determinar el ganador: por ejemplo la puntuación se saca individualmente y la victoria de un jugador indica la victoria de su equipo o cada equipo tiene una puntuación general.

> Descripción de la interfaz gráfica

* Pantalla principal:

* Botón Siguiente: Permite agregar el contenido del TextBox a su izquierda a una lista de caras validas de las fichas.
* Botones D/9 y D/6: Permiten establecer el conjunto de caras validas de 0 a 9 y de 0 a 6 respectivamente. Estos no son necesarios para el programa ya que lo mismo puede hacerse con el botón siguiente, solo se crearon para agilizar el proceso de probar el programa.
* Botones + y -: Permiten aumentar o disminuir la cantidad de fichas que se le asignan a cada jugador
* Botones de las Interfaces: Estos 5 botones dispuestos en columna permiten cambiar el aspecto del juego que se indica a la izquierda de cada uno entre las diferentes implementaciones proporcionadas por el programa.
* Botones ?: Abren un MessageBox que brinda al usuario información sobre la funcionalidad a la izquierda del botón .
* Botones + (lado derecho): Permiten cambiar la IA (tipo de jugador) y el equipo al que este pertenece. Ambas cosas se indican en el TextBox de la izquierda de cada botón .
* Botones Incluir y Eliminar: Permiten incluir al un nuevo jugador y eliminar al ultimo introducido respectivamente. La cantidad actual de jugadores se muestra debajo.
* Botón Aceptar: Permite, si las opciones seleccionadas son validas, abrir una nueva ventana donde se ejecutara el partido, si las opciones son incompatibles muestra un mensaje de advertencia.

* Pantalla de ejecución del partido:
En la parte superior se muestran mensajes de los sucesos del juego.
En esta pantalla hay botones que no funcionan como tal sino que se usan para representar caras de una ficha. Estos son:
* Estáticos: Dos botones en el centro de la pantalla que indican las caras disponibles en la mesa, es decir los extremos.
* Móviles: Dos botones que se van moviendo por la pantalla y representan una nueva ficha siendo colocada por un jugador.
El movimiento de los botones anteriores se logra mediante un timer que va modificando la posición de estos según valla transcurriendo el juego. Es este También quien controla a los mensajes que se van mostrando.

* Botones Comenzar y Regresar: Ubicados en la parte inferior de la pantalla, permiten comenzar el partido y cerrar la pantalla respectivamente.

> Descripción de las opciones de cada Interface

Ivalitator:

* Opción #1: La usual en el domino
* Opción #2(Solo funciona correctamente para data numéricas):Una ficha se puede jugar por una cara si es mayor en uno que dicha cara, la cara 0 se juega sobre la ficha de mayor valor en la mesa.

Ivalorator:

* Opción #1: EL valor de una ficha es la suma del valor de sus caras.
* Opción #2: El Valor de una Ficha es el MCD de sus caras.

IOrder:
* Opción #1: Los jugadores juegan en orden ascendente.
* Opción #2: Cuando un jugador se pasa se invierte el orden, permanece asi un ciclo completo y cuando otro jugador se pasa se vuelve a invertir.

IFinisher:

* Opción #1: El juego finaliza cuando un jugador se pegue o cuando ninguno lleve.
* Opción #2: El juego finaliza cuando cada jugador se ha pasado al menos una vez o se cumpla la opción #1.

IResult:

* Opción #1: La puntuación es individual.y la victoria de un jugador indica la victoria de su equipo.
* Opción #2: La puntuación es por equipo.(En caso de no haber equipo ambas opciones son equivalentes)