

Dominó Project

Facultad : Matcom

Grupo : C212

Integrantes :

- Amanda Noris Hernández
- Juan Miguel Pérez Martínez

Lo tratado:

En este documento trataremos de darle a conocer al lector una idea acerca de cómo funciona nuestro desarrollo del proyecto Dominó , cómo puede interactuar el usuario con nuestra versión y algunas cuestiones que considero de importancia acerca del mismo.

El proyecto consta de dos partes ,una biblioteca de clases que tiene toda la parte lógica del proyecto y una WindowsFormAplication desarrollada en Windows Forms la cual permite al usuario personalizar su juego de dominó y ver las jugadas que se realizan.

Sobre la biblioteca de clases

En esta se encuentran todas las clases que hacen posible que el juego funcione, jugadores, las estructuras del juego como son la mesa, las fichas y las diferentes configuraciones que se le pueden realizar al juego. Estas últimas se encuentran abarcadas en los siguientes aspectos: Calcular Puntuación, Calcular peso de la ficha, Repartir, Siguiente Jugador, Condición de finalización, Validador de Jugada; todos estos aspectos serán tratados con mayor profundidad más adelante.

Jugadores

Aquí decidimos crear una clase abstracta JugadorBasico nos decidimos aquí hacia una clase abstracta y no una interfaz ya que todos los jugadores hacen ciertas funciones que son comunes en todos. En esta clase JugadorBasico se verifica si el jugador lleva comprobando si la jugada es válida, además desarrolla un método que le permite al jugador saber cuáles son las posibles jugadas que tiene permitido hacer y ya las clases que hereden de jugador básico decidan según su implementación del método virtual jugar que jugada hacer según su forma de jugar.

Sobre los tipos de jugadores implementados:

Tenemos el jugador BotaGorda que según la opción de peso de cada ficha seleccionada por el usuario busca entre las posibles jugadas la ficha que más peso tenga y esa es la que juega

Está el jugador inteligente el cuál juega por el lado de la mesa que más fichas disponible tenga para no quedarse sin fichas de ese tipo, en fin un jugador inteligente.

Está el jugador ordenado el cual juega la primera ficha que tenga en el orden que tiene la mano.

Por último tenemos al imprescindible jugador Random que juega una ficha aleatoria de las que tenga en la mano.

Mesa

Esta es una estructura de la mesa que almacena todos los sucesos que han ocurrido durante el juego ya que cuenta con una lista de jugadas, las caras disponibles (en dependencia de las jugadas que se realicen estas se actualizan), también cuenta con propiedades para ver cuál es la ficha que primero se jugó y para ver si no se ha jugado ninguna ficha.

Mano

La mano es un objeto que permite almacenar las fichas que tiene cada jugador. Gracias a esta se puede saber cuántos dobles tiene el jugador, cuántas fichas y permite también agregar y eliminar fichas de la misma.

Ficha

La ficha es la estructura celular del juego. Nuestra implementación de ficha tiene dos caras, tiene además un método que convierte a la ficha en un string.

Jugada

Una jugada es un objeto que almacena 2 variables, siendo estas el jugador que realizó dicha jugada y la ficha que se jugó. También posee una propiedad llamada `jugoDerecha` que devuelve `true` si el jugador jugó por la cara disponible a la derecha.

Las interfaces

Para los elementos que son variables creamos diferentes interfaces con los métodos imprescindibles para el funcionamiento del programa. La interfaz `IFinalizador` que contiene el método booleano `GameOver`, `ICalculador` contiene al método `CalcularPuntos` que retorna un `int`, `IValidador` consta de un método booleano `EsValida` que comprueba si la jugada es válida y una propiedad `ParametroValidacion` que es el parámetro que usa es válida para comprobar si la jugada es válida, `ISiguienteJugador` que consta con el método `NextPlayer` el cuál nos devuelve el jugador que le toca jugar, `IRepartidor` el cual cuenta con el método `void Repartir`, `ICalculadorScore` que tiene un método `CalcularScore` que devuelve la puntuación que se le da a la ficha.

```
using System.Collections.Generic;
namespace Domino
{
    6 references
    public interface IFinalizador
    {
        4 references
        bool GameOver(Mesa mesa, Juego juego, JugadorBasico jugador);
    }
    6 references
    public interface ICalculador
    {
        4 references
        int CalcularPuntos(JugadorBasico jugador, ICalculadorScore calculadorScore);
    }
    7 references
    public interface IValidador
    {
        7 references
        int ParametroValidacion { get; set; }
        8 references
        bool EsValida(int cara, int caraFicha);
    }
    5 references
    public interface ISiguienteJugador
    {
        3 references
        int NextPlayer(List<JugadorBasico> jugadores, Ficha ficha);
    }
}
```

```
6 references
public interface IRepartidor
{
    4 references
    void Repartir(List<JugadorBasico> jugadores, int fichaspormano, Mesa mesa);
}

11 references
public interface ICalculadorScore
{
    7 references
    int CalcularScore(Ficha ficha);
}
```

Finalizadores

Los finalizadores son los que implementan la interfaz IFinalizador y hay tres implementaciones: el GameOver_Usual el cual termina la partida cuando algún jugador se queda sin fichas o si no hay más opciones por donde jugar para ningún jugador, GameOver_DobleBlanco que termina el juego además de con lo usual si a algún jugador le queda una ficha y esta es el doble blanco, GameOver_Balanceado que acaba el juego además de con las condiciones usuales si la cantidad de ficha que tienes es igual a la cantidad de pases que ha tomado y a la cantidad de fichas que el jugador ha jugado.

CalculaPuntos

Los calculadores de puntos son los que implementas la interfaz ICalculador, está el usual el cual suma el peso de cada una de las fichas con las que el jugador se queda en la mano una vez terminado el juego, el CalculaPuntosX2 que calcula la puntuación de la misma manera que el usual pero si el jugador tiene menos de 10 puntos, entonces multiplica por 2 la cantidad de puntos y finalmente el CalculaPuntosPases que divide la cantidad de puntos que tenga el jugador calculados de manera usual entre la cantidad de pases que se haya dado durante el juego.

Validadores

Los validadores son los encargados de decidir si una jugada es válida o no, el usual devuelve true siempre que las 2 caras que se analicen sean iguales, en cuanto al ValidadorParidad y al ValidadorMult ambos hacen uso del parámetro de validación que introduce el usuario. El primero tiene como condición que los valores de ambas caras den el mismo resto al ser divididos entre el parámetro de validación y el segundo que la suma de ambas caras sea múltiplo del parámetro.

SiguienteJugador

La interfaz SiguienteJugador se encarga de decidir a qué jugador le corresponde jugar, como su nombre lo indica, la implementación usual sigue el orden típico en forma de ciclo de menor a mayor y el invierte orden, invierte el orden en el ciclo siempre que un jugador se pase (si se pasan 2 jugadores seguidos con este último caso el juego finaliza).

Repartidor

Los repartidores reparten las fichas a los jugadores cuando se inicializa cada juego, el repartidor usual reparte un número x de fichas (que el usuario introduce previamente) a cada uno de los jugadores. El repartidor 5 dobles se encarga de que si tras la primera vez que se reparte algún jugador posee más de 5 dobles en la mano, entonces desecha todos estos dobles y coge una ficha nueva. El repartidor sin dobles se encarga de que ningún jugador posea dobles en sus manos.

CalculadorScore

Los calculadores de score son los encargados de asignarle un peso a cada una de las fichas. El usual suma los valores en ambas caras, el `CalcularScoreMultiplicar` multiplica dichos valores y el `DifModular` divide entre 2 el módulo del resultado de restar los valores en ambas caras.

Reglas

El objeto reglas es el encargado de agrupar todos los aspectos modificables del juego. En su constructor recibe una implementación de cada una de las interfaces mencionadas anteriormente, además de los parámetros numéricos que introduce el usuario. Este objeto reglas es necesario a la hora de crear el juego.

Juego

El objeto juego de la clase `Juego` es que modela finalmente el juego de dominó. Para su inicialización requiere de un objeto de la clase `Reglas`, así como de una lista de jugadores.

Este objeto además posee una `Mesa`, una propiedad cantidad de jugadores y una lista de ganadores. El método principal de la clase juego `''Jugar''` es el encargado de modular todo lo que ocurre en una partida de dominó habitual.

Clase Auxiliares

En esta clase se encuentran un grupo de métodos secundarios que utiliza el juego en su creación y luego cuando finaliza. Estos son el `CreadorFichas`, que se encarga de generar todas las fichas necesarias para la partida, el `Barajeador` que, como se dice popularmente, `''da agua''` a las fichas ya creadas y el `GetGanadores`, el cual decide el ganador de cada partida.

Sobre la interfaz gráfica

La interfaz gráfica está hecha en la biblioteca de clases de .Net Framework Windows Forms y está diseñada de manera tal que sea de fácil entendimiento y uso para el usuario. A esta se puede acceder de manera muy sencilla a través de un archivo .exe. Está dividida en 5 forms principales que detallaremos a continuación.

Form 1



Es la forma que da la bienvenida al usuario, consta de 2 botones, el primero llamado "Salir" que cierra la aplicación y el segundo "Jugar" que da inicio al proceso de selección de parámetros para poder llevar a cabo la partida.

Form 2

Form2

Escoge las reglas

Condicion de finalizacion

- ☒ Usual
- ☐ Doble_Blanco
- ☐ Balanceado

Validador de Jugada

- ☒ Usual
- ☐ Paridad
- ☐ Mult

Calculador de Puntuación

- ☒ Usual
- ☐ X2
- ☐ Pases

Siguiete Jugador

- ☒ Usual
- ☐ InvierteOrden

Repartidor

- ☒ Usual
- ☐ 5Dobles
- ☐ SinDobles

Calcular Score

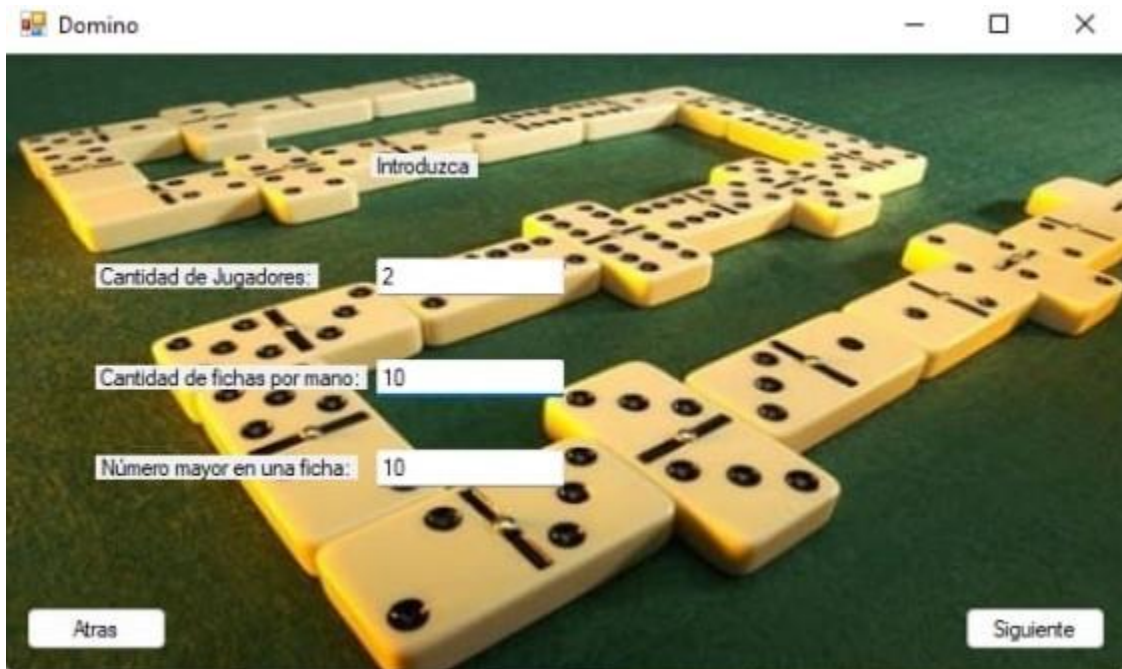
- ☒ Usual
- ☐ Multiplicar
- ☐ DiferenciaModular

Atras

Siguiete

En esta forma el usuario debe seleccionar las reglas de su partida, las cuales están agrupadas por tipo en diferentes groupBoxes para facilitar la comprensión de las selecciones. Si en el caso del validador se selecciona una implementación que deba hacer uso del parámetro de validación, se activa un TextBox para que el usuario introduzca el valor que desee asignarle al parámetro. Nos aseguramos de que solo se seleccione una implementación por cada aspecto a través del uso de RadioButtons y siempre se le asigna a cada aspecto una selección por default la cual es la usual. Así mismo en la forma se controla que el usuario rellene todos los campos necesarios. Posee un botón se siguiente, para avanzar a la siguiente forma y uno de atrás para retroceder.

Form 3



Domino

Introduzca

Cantidad de Jugadores: 2

Cantidad de fichas por mano: 10

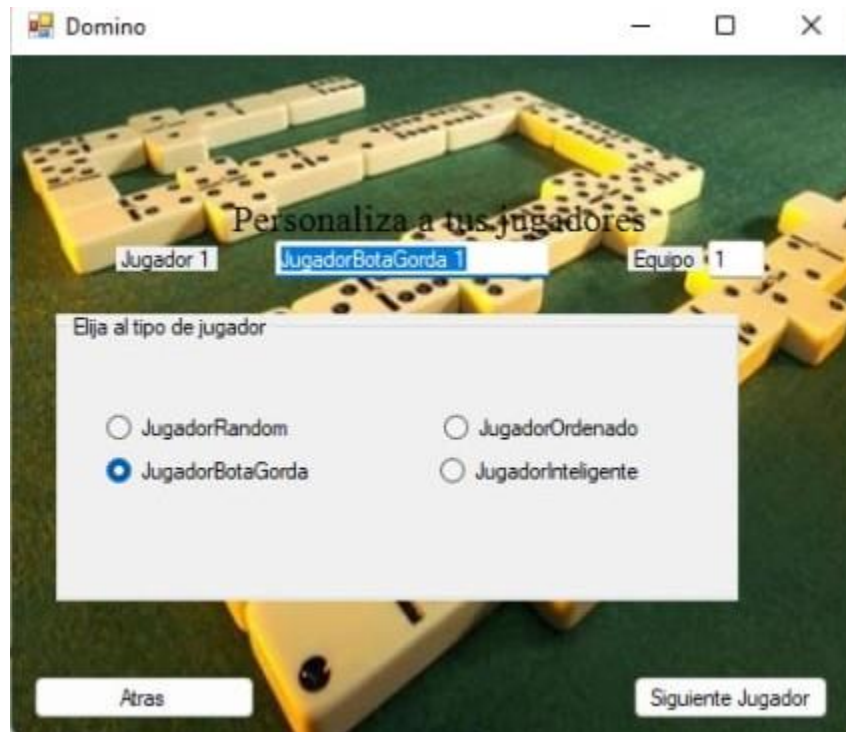
Número mayor en una ficha: 10

Atras

Siguiente

Aquí es donde el usuario introduce los valores numéricos variables(cantidad de fichas por mano, mayor valor que puede tener una cara de una ficha y cantidad de jugadores) y se comprueba que los valores introducidos son correctos, por ejemplo, no se le permite al usuario introducir un número de ficha por mano tal que al multiplicarlo por la cantidad de jugadores sea mayor que la cantidad de fichas que la mesa posee(este último valor depende íntegramente del mayor valor que puede tener una cara de una ficha). Posee también los botones de "Atras" y "Siguiente".

Form 4



The screenshot shows a window titled "Domino" with a background image of dominoes. Overlaid on this is a dialog box titled "Personaliza a tus jugadores". Inside the dialog, there is a label "Elija al tipo de jugador" followed by four radio button options: "JugadorRandom", "JugadorBotasGorda" (which is selected), "JugadorOrdenado", and "JugadorInteligente". Above the dialog, there are three text input fields: "Jugador 1", "JugadorBotasGorda 1", and "Equipo 1". At the bottom of the dialog, there are two buttons: "Atras" and "Siguiente Jugador".

La form 4 es en la que el usuario elige los diferentes jugadores. Una nueva form aparece por cada jugador necesario, permitiéndole al usuario elegir entre las 4 opciones de jugadores y asignar un nombre y equipo a cada jugador. En la parte lógica de esta forma se preparan además todas las condiciones para el inicio del juego cuando se hace click en el botón "Iniciar Partida".

Form 5

The screenshot shows a Java Swing window titled "Form5" with a standard Mac OS X title bar (Archivo, Editar, Ver, Formato, Herramientas, Ayuda). The window has a light green background. In the top-left corner, a status bar with a green background displays the text "JugadorRandom 1 jugó el [7| 0]". In the center of the window, there is a game board represented by two rows of white rectangular boxes with black borders. The first row contains the numbers 6, 8, 8, 0, 0, 7. The second row contains the numbers 6, 4, 4, 2, 2, 0, 0, 10, 10, 7, 7, 9. At the bottom center of the window, there is a small white button with a black border and the text "Siguiente Jugada".

6	8	8	0	0	7						
6	4	4	2	2	0	0	10	10	7	7	9

Siguiente Jugada

Es en el que se visualiza lo que está ocurriendo en la partida a través de una representación de la mesa. Al hacer click en el botón "SiguienteJugada" las fichas que han sido jugadas se observan mediante una imagen que se genera en la clase Fichalmagen método DibujaHorizontal.y que se coloca mediante trabajo con coordenadas en su posición correspondiente en la pantalla. En la esquina superior izquierda un cartel va narrando las jugadas.