

# Reporte Segundo Proyecto Programacion

MATCOM  
2022 – 2023

## Implementacion de un juego de Domino

Adrian Hernandez Santos – C211  
Alejandro Alvarez Lamazares – C212

# Sobre el juego:

Nuestra implementación del juego de dominó está basada en el juego clásico, tanto doble 9, como doble 7, para la implementación inicial, las reglas, etc. Partiendo de ahí pudimos determinar distintos conceptos base para modificar y crear un juego más extensible que su variante clásica. Apoyándonos en los conceptos básicos de Programación Orientada a Objetos aprendidos en el curso de Programación, obtuvimos las herramientas necesarias para crear un proyecto base en el que cualquiera en un futuro puede llegar e implementar nuevas funcionalidades con un mínimo de código en la mayoría de los casos, siempre que esté definida una abstracción para dicho elemento a extender.

## Aspectos clave:

- Fichas
- Jugadores
- Condición de Ganado
- Ganador
- Interfaz gráfica

## Fichas:

Las fichas en el dominó son la parte más importante, puesto que en ellas se basa todo el sistema de puntaje, los posibles movimientos y demás funcionalidades que permiten el correcto funcionamiento del juego. Para el nuestro, decidimos implementar 3 variantes:

### DominoToken:

Representa la ficha estándar del juego de dominó. Su cálculo de valor está dado por la suma de los valores de cada componente (izquierda y derecha) como en el juego normal. Además de poseer un sistema para generar las fichas dada una numeración de juego (doble 9 por ejemplo), el cual es heredado a cada una de las fichas hijas para que tenga la misma funcionalidad, dependiendo del tipo de ficha en cuestión.

### SixUnvaluableDominoToken:

Representa una ficha en la cual el valor 6 está penalizado. Esto quiere decir que si en alguna de sus componentes aparece un 6, el valor del mismo automáticamente pasa a ser 0, o sea, no tiene valor.

## DoubledValueDominoToken:

Representa una ficha en la cual el valor de cada una de sus componentes es el doble de lo que tuviera si fuera una implementación estándar de dicha ficha. Por ejemplo, el doble 3, al ser sus 2 componentes 3 y 3, se convierte (representativamente) en un doble 6, puesto que el valor de cada una se duplica, pero sigue siendo un 3 a la hora de jugarla.

## Jugadores:

El jugador es una parte clave en el juego de dominó. Sin él, el propio concepto de juego no tendría sentido. La funcionalidad básica del jugador sería jugar una ficha en su turno, dado un estado del juego, dependiendo de su implementación. Dicha funcionalidad tiene 2 momentos clave, por una parte la ficha inicial, si el jugador es el que comienza la partida, y por otro, jugar una ficha en su turno, después de una cierta cantidad de jugadas. Por su importancia en el propio juego fue que decidimos hacer varias implementaciones de estos, en específico 4:

### RandomDominoPlayer:

Un jugador que seleccionara una ficha aleatoria para jugar de las posibles jugadas que pueda hacer, basado en el sistema RNG de .NET.

### GreedyDominoPlayer:

Un jugador que jugará siempre la ficha con mayor valor de entre las posibles jugadas que tenga en mano, independientemente de la forma de calcular el valor del tipo de ficha que se utilice en el juego.

### HumanDominoPlayer:

Implementación de un jugador humano, se le pedirá al usuario que dado un estado del juego y una mano de posibles fichas, realice una jugada en su turno, con una entrada en la terminal de la consola del juego.

### DataDominoPlayer:

Jugador que basará su jugada primeramente seleccionando de una data (número con mayor ocurrencia entre sus fichas), y posteriormente, en caso de no poder jugar por su data, jugará la primera ficha que sea posible.

# Condición de Ganado:

La condición de ganado en el dominó determina cuando un juego dado debe terminar. Independientemente del juego que se juegue, luego de cada jugada de un jugador, debe verificarse el estado de la partida, para decidir si se debe continuar o se debe terminar dicho juego. Hemos realizado 2 implementaciones para esta funcionalidad:

## StandardWinCondition:

Condición de ganado estándar, o sea, las reglas básicas para que una partida termine: que nadie pueda hacer una jugada o que algún jugador se quede sin fichas en la mano.

## FourRoundsWinCondition:

Condición de ganado que consiste en que el juego finaliza luego de 4 rondas, o cuando un jugador se queda sin fichas. Una ronda está definida como la sucesión de jugadas desde un jugador a sí mismo nuevamente, o sea, un ciclo completo de jugadas.

# Ganador:

Como es de esperar, en un juego de varios jugadores, debe haber al menos un ganador (digo al menos porque hasta ahora no hemos tenido en cuenta el concepto de empate, pero podría existir). Dicho ganador, dependiendo de las reglas, puede estar determinado por diferentes factores, siempre luego de que el juego finaliza (ahí entra la condición de ganado). Hemos hecho 2 implementaciones de cómo seleccionar un ganador:

## MinValueWinner:

Esta implementación selecciona al jugador con menor valor entre todas sus fichas para declararlo ganador. Dicho valor viene calculado con la implementación de ficha que se esté utilizando en la partida en cuestión.

## MoreTokensWinner:

Esta implementación selecciona al ganador como el jugador con mayor cantidad de fichas luego de que se acaba la partida, sin tomar en cuenta el valor de las mismas, solo la cantidad en mano.

# Interfaz gráfica:

La interfaz gráfica de nuestro juego esta compuesta por un script bastante simple que ejecuta una aplicación de consola. Dicha aplicación está compuesta por un conjunto de funciones que se encargan de, dada una entrada válida del usuario, crear una configuración previa al juego para luego instanciar este y proceder a dejar que el usuario pueda jugar su partida.

Los métodos encargados de hacer esa “pre-configuración” del estado inicial de la partida son los siguientes:

SetPlayer Count  
SetValueAmount  
SetTokensPerPlayer  
SetPlayers  
SetWinCondition  
SetWinner  
SetToken

Cada uno configura el aspecto del juego especificado en su nombre, retornando un entero, un array de DominoPlayer, o una instancia de una clase que implementa una interfaz, dependiendo el caso, que posteriormente son usados como estado inicial para la partida a ejecutarse.

## PrepareGame:

Este método instancia un nuevo juego, dados los parametros previamente entrados por el usuario.

## GetInput:

Este método auxiliar fue creado con el objetivo de utilizar el patrón DRY (Don't Repeat Yourself), puesto que en cada método anterior se pedía una entrada del usuario explícitamente. Además, se le agregó un parámetro opcional que permite acotar el resultado obtenido a valores con sentido y capturar además, entradas no numéricas, en este caso números enteros.

## Main:

El método Main de dicho script, simplemente se encarga de mostrar una información básica explicando los parámetros a modificar y ejecutar los métodos previamente comentados, para crear un flujo correcto de interacción entre el usuario y la aplicación final.