

Un **Torneo de Juegos** es una actividad en la que participan cierta cantidad de jugadores o equipos. Tiene tres etapas:

En la primera etapa se inicializan los jugadores o equipos, en dependencia de si el juego o el tipo de torneo los admite. Una segunda etapa es cuando empieza el torneo. En este proceso se van generando partidas, cuya organización es particular de cada tipo de torneo. Hasta que no se haya terminado una partida, no se puede comenzar otra. Una vez finalizada esta etapa, empieza la tercera, el torneo termina y se determina un ganador.

Entre los tipos concretos de torneo implementados están:

- **Torneo de Título:** Torneo que se juega individualmente y se caracteriza por la existencia de un ganador, el cual es retado por los otros jugadores. Al inicio del torneo se elige un jugador, el cual va a ser el ganador actual. La organización de las partidas, está dada de la siguiente forma: el jugador en la posición i , juega con el jugador de la posición $i+1$; el ganador de dicha partida, juega con el jugador en la posición $i+2$ y así sucesivamente. Con la posición del jugador hago referencia al orden en que fueron añadidos al torneo. El ganador de la partida pasa a ser el nuevo ganador al que van a retar. En caso de empate se realizará una partida de desempate. El torneo finaliza cuando haya jugado el último jugador con el ganador actual. Quien gane esta última partida es el ganador del torneo.
- **Torneo Dos a Dos:** Torneo que solo se realiza con juegos que requieran más de dos jugadores o equipos. Al empezar el torneo, se establece una determinada cantidad de partidas entre los jugadores. La organización de las partidas está dada por una combinación de los jugadores. Al final de cada partida se le asigna una puntuación a los jugadores o equipos en dependencia del estado en que haya terminado dicha partida. El torneo termina cuando se hayan realizado todas las partidas establecidas. En

esta etapa se muestra una tabla con los resultados de todos los participantes del torneo.

Los **Jugadores** son entidades fundamentales de los juegos. Su función principal es tomar las decisiones en los juegos. Un jugador sabe jugar cierto juego si implementa una interfaz asociada a dicho juego. En este proyecto se verán dos implementaciones de tipos distintos de jugadores.

- **Jugador Aleatorio:** Jugador que no tiene una forma específica de actuar en el juego, ni se rige por determinada lógica. Juega al azar.
- **Jugador Goloso:** Jugador cuya estrategia es jugar la mejor jugada del juego.

Flujo para crear un nuevo tipo de Torneo:

- 1) Se añade una nueva clase en la solución, cuyo nombre es el torneo que se desea crear.
- 2) Todos los torneos implementan la interface **ITournament** la cual implementa, a su vez, **IEnumerator**. Otras clases e interface relacionadas con los torneos son **Player**, **Match**, **IGame**.

```
namespace Project
{
    7 references
    public interface ITournament : IEnumerator<Match> //represents a Tournament
    {
        4 references
        bool AddPlayer(Player[] player); //returns true if the player or team was succesfully added to Tournament, false otherwise
        3 references
        bool Teams { get; } //returns true if tournament can be playd in teams
        4 references
        bool End { get; } //returns true if the Tournament ended
        3 references
        Player[] ChWinner { get; } // returns the winner/s of the Tournament
    }
}
```

- 3) Los torneos deben recibir como parámetros necesarios en su construcción: un tipo que implemente **IGame**, que sería el juego del cual se haría el torneo; un tipo que implemente **ILogger**, con este objeto se podrán "loggear" todos los sucesos que ocurran en el torneo; un entero con la cantidad máxima de puntos que alcanza el

ganador de las partidas, y un boolean que represente al tipo de partida que se realizará en este torneo.

- 4) Los métodos y propiedades de la interface son comunes a todos los torneos, pero es quien crea el torneo el que decide si le es suficiente con la implementación de la interface o desea añadir nuevas funcionalidades.

Pasos para ejecutar un Torneo:

- 1) Para integrar el torneo al emulador, se debe añadir al diccionario `Dictionary<string, object> Dicc` que se encuentra en la clase `WelcomeForm`, una instancia del tipo de torneo como valor y el nombre de la clase instanciada como llave.

```
namespace Project
{
    31 references
    public partial class WelcomeForm : Form
    {
        IGame Game;
        public ITournament Tournament;
        ILogger logger;
        Form loggingForm;
        Player[] players;
        int countTeam;
        int numberTeam;
        int countPlayers;
        Dictionary<string, object> Dicc;
        bool singlePlayer;
        List<string> PlayersNames;

        1 reference
        public WelcomeForm()
        {
            InitializeComponent();
            countTeam = 0;
            countPlayers = 0;
            Dicc = new Dictionary<string, object>();
            singlePlayer = false;
        }
    }
}
```

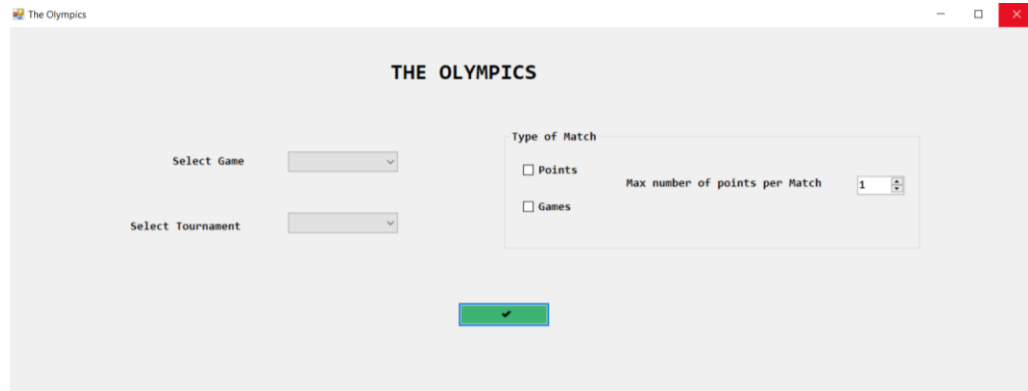
- 2) Luego se deberá implementar en la clase `Factory`, método `TournamentFactory`, un nuevo caso en el switch, que corresponda con el Torneo creado.

```
1 reference
public static ITournament TournamentFactory(string champName, ILogger logger, IGame game, int maxPoints, bool typeMatch) //creates a type I
{
    ITournament champ = null;
    switch (champName)
    {
        case "Title": champ = new TitleTournament(game, maxPoints, typeMatch, logger); break;
        case "Two To Two": champ = new TwoToTwoTournament(game, maxPoints, typeMatch, logger); break;
    }
    return champ;
}

7 references
```

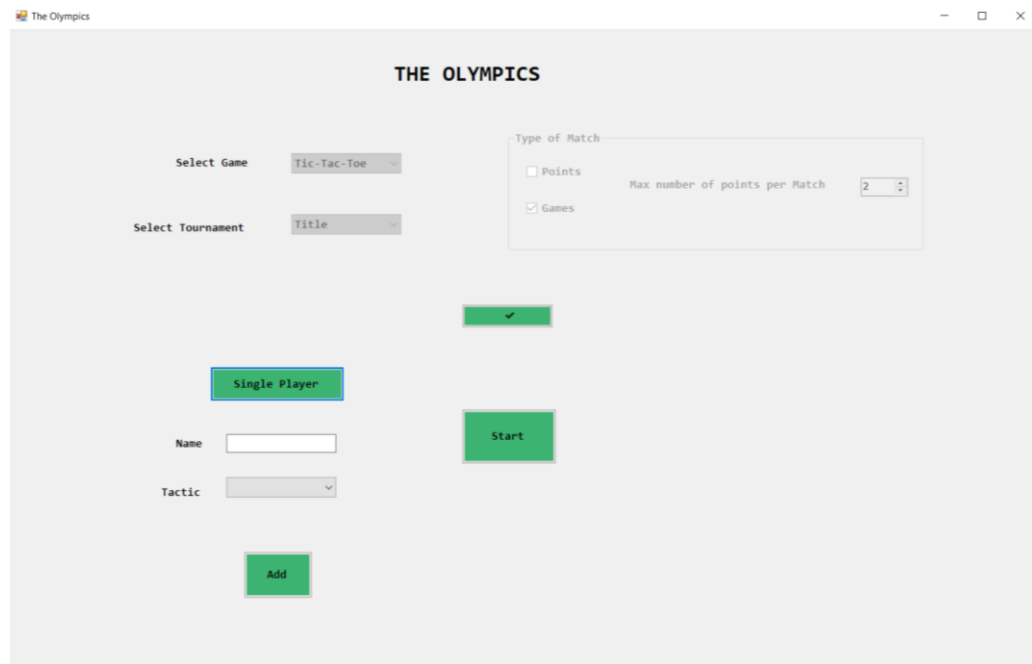
- 3) Finalizado este proceso se ejecuta la aplicación.

- 4) En una primera vista de la interfaz veremos una serie de elementos que debemos seleccionar con el objetivo de ejecutar el torneo.



The screenshot shows a web application window titled "The Olympics". The main heading is "THE OLYMPICS". Below the heading, there are two dropdown menus: "Select Game" and "Select Tournament". To the right of these menus is a "Type of Match" section with two checkboxes: "Points" and "Games". The "Games" checkbox is selected. Next to the "Games" checkbox is a label "Max number of points per Match" followed by a numeric input field containing the value "1". At the bottom center of the form is a green button with a white checkmark icon.

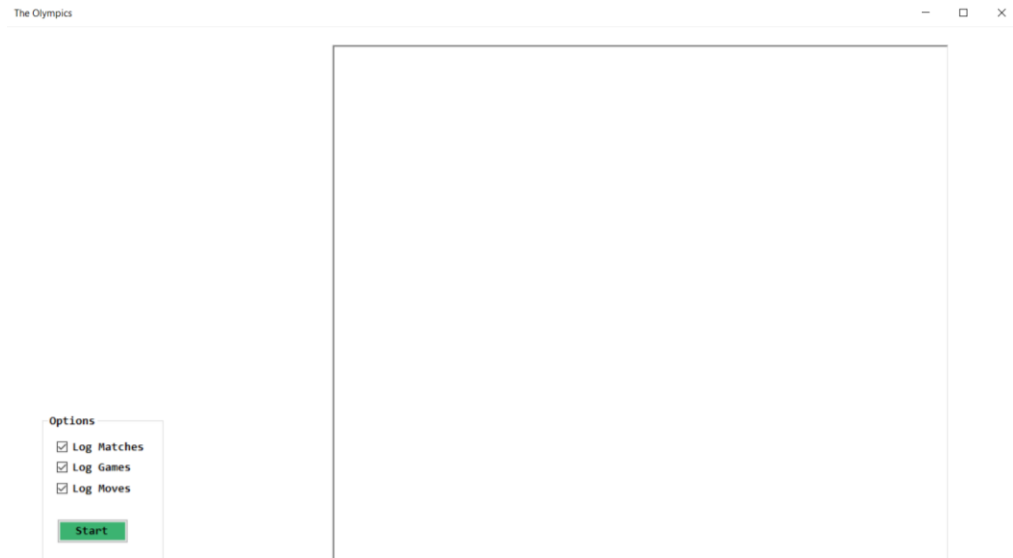
- 5) Primero se selecciona el juego sobre el cual se va a realizar el torneo, luego el tipo de torneo, y los parámetros necesarios para la creación de partidas. Luego se presiona en el botón ✓;
- 6) Después aparecerán una o dos opciones, en dependencia del tipo de juego o el tipo de torneo seleccionado, sobre la forma de los jugadores.
- 7) Se elige una y aparecerán las condiciones para añadir jugadores al torneo.



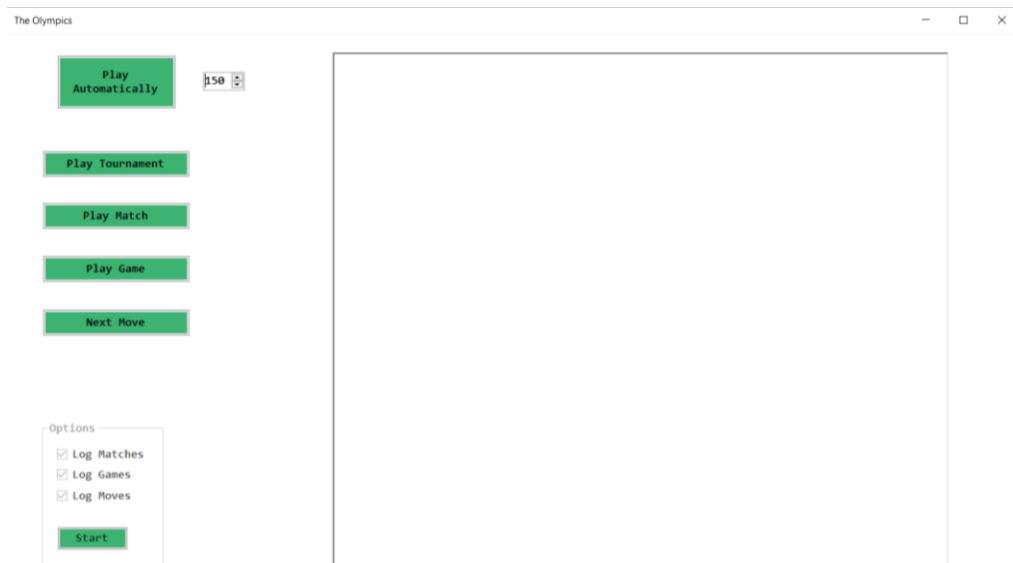
The screenshot shows the same web application window, but with more options visible. The "Select Game" dropdown now shows "Tic-Tac-Toe" and the "Select Tournament" dropdown shows "Title". The "Type of Match" section remains the same with "Games" selected and "Max number of points per Match" set to "2". Below the green checkmark button, there is a green button labeled "Single Player". To the right of the "Single Player" button is a green button labeled "Start". Below the "Single Player" button, there is a text input field labeled "Name" and a dropdown menu labeled "Tactic". At the bottom center is a green button labeled "Add".

- 8) Una vez añadidos los jugadores, se presiona el botón Start.

- 9) Aparecerá una vista con un panel que mostrará todos los acontecimientos durante la ejecución del torneo.



- 10) En la esquina inferior izquierda se escogerá la categoría de los sucesos que se desean mostrar. Después de seleccionadas las opciones se presiona el botón Start.



- 11) Luego el usuario elegirá la forma de ejecutar el torneo y aparecerá en el panel.

Paso para añadir **Jugadores** al emulador:

- 1) Se añade una nueva clase a la solución, cuyo nombre es el tipo de jugador que se implementara.
- 2) Todos los nuevos tipos de jugadores heredan de la clase abstracta **Player**.

```
58 references
public abstract class Player
{
    0 references
    public Player() { }
    2 references
    public Player(string name) { Name = name; } //constructor parameters: name of the player
    6 references
    public string Name { get; set; } //name of the player
}
}
```

- 3) Además, el nuevo jugador deberá implementar las interfaces asociadas a los juegos que "saben" jugar.
- 4) Para integrar el jugador al emulador, se debe añadir al diccionario **Dictionary<string, object> Dicc** que se encuentra en la clase **WelcomeForm**, una instancia del tipo de jugador como valor y el nombre de la clase como llave.
- 5) Luego se deberá implementar en la clase **Factory**, método **PlayersFactory**, un nuevo caso en el switch, que corresponda con el Jugador creado.

```
2 references
public static Player PlayersFactory(string playerName, string playerTactic) //creates a type player
{
    Player player = null;

    switch (playerTactic)
    {
        case "Random": player = new RandomPlayer(playerName); break;
        case "Greedy": player = new GreedyPlayer(playerName); break;
    }

    return player;
}
```

6) Finalizado este proceso el jugador estará añadido al emulador.

Pasos para añadir **Juegos** al emulador:

- 1) Se añade una nueva clase a la solución, cuyo nombre es el tipo de juego que se implementara.
- 2) Los juegos implementan a la interface **IGame**.

```
namespace Project
{
    17 references
    public interface IGame : IEnumerator //represents a game
    {
        12 references
        int NumberPlayers { get; } //number of players needed
        3 references
        int NumberTeams { get; } //number of team allowed
        5 references
        int NumberOfPlayersPerTeam { get; } //number of player allowed per team
        4 references
        bool Teams { get; } //returns true if the game can be played in teams, false otherwise
        4 references
        bool AddPlayer(Player[] g); //returns true if the player or team was succesfully added to Game, false otherwise
        5 references
        bool GameOver { get; } //returns true if the game is over, false otherwise
        3 references
        int WinnerPoints { get; } //number of points the winner of a game gets
        6 references
        Player[] GameWinner { get; } //returns th player or team that won the game
        3 references
        string GetInterface { get; } //returns the name of the interface associated with the game
    }
}
```

- 3) Estos tienen como parámetros de construcción un tipo **ILogger**, que se encargara de mostrar los sucesos del juego.
- 4) Cada juego tiene una interface asocia que implementa al menos un método **Play()**. Se deberá crear esta interface. De no crearse ningún jugador podrá jugar este juego.
- 5) Para integrar el juego al emulador, se debe añadir al diccionario **Dictionary<string, object> Dicc** que se encuentra en la clase **WelcomeForm**, una instancia del tipo de juego como valor y el nombre de la clase como llave.
- 6) Luego se deberá implementar en la clase **Factory**, método **GamesFactory**, un nuevo caso en el switch, que corresponda con el Juego creado.

```
1 reference
public static IGame GamesFactory(string gameName, ILogger logger)//creates a type IGame
{
    IGame game = null;
    switch (gameName)
    {
        case "Tic-Tac-Toe": game = new TicTacToe(logger); break;
        case "Juego Prueba": game = new Class1(); break;
    }
    return game;
}
```

7) Finalizado este proceso el juego estará integrado al emulador.