

TicTacToe

Generated by Doxygen 1.9.6

1 Hierarchical Index	1
1.1 Class Hierarchy	1
2 Class Index	3
2.1 Class List	3
3 File Index	5
3.1 File List	5
4 Class Documentation	7
4.1 Board Class Reference	7
4.2 EasyStrategy Class Reference	7
4.2.1 Member Function Documentation	8
4.2.1.1 GetDifficulty()	8
4.2.1.2 GetNextMove()	8
4.3 Game Class Reference	9
4.3.1 Member Function Documentation	10
4.3.1.1 AddListeners()	10
4.3.1.2 GetComputer()	10
4.3.1.3 GetGameBoard()	10
4.3.1.4 GetPlayer()	11
4.3.1.5 InitializeGame()	11
4.3.1.6 NotifyAll()	11
4.3.1.7 PlaceSign()	11
4.3.1.8 RemoveListeners()	12
4.3.1.9 RunRound()	12
4.4 GameState Class Reference	12
4.5 HardStrategy Class Reference	13
4.5.1 Member Function Documentation	13
4.5.1.1 GetDifficulty()	13
4.5.1.2 GetNextMove()	13
4.5.1.3 Minimax()	14
4.6 IDifficultyStrategy Class Reference	15
4.6.1 Detailed Description	15
4.6.2 Member Function Documentation	15
4.6.2.1 GetNextMove()	15
4.7 IGame Class Reference	16
4.7.1 Detailed Description	17
4.7.2 Member Function Documentation	17
4.7.2.1 AddListeners()	17
4.7.2.2 GetComputer()	17
4.7.2.3 GetGameBoard()	18
4.7.2.4 GetPlayer()	18

4.7.2.5 InitializeGame()	18
4.7.2.6 NotifyAll()	18
4.7.2.7 PlaceSign()	18
4.7.2.8 Produce()	19
4.7.2.9 RemoveListeners()	19
4.7.2.10 RunRound()	20
4.8 IGameListener Class Reference	20
4.8.1 Detailed Description	20
4.9 IPlayer Class Reference	21
4.9.1 Detailed Description	21
4.9.2 Member Function Documentation	21
4.9.2.1 GetPlayerName()	21
4.9.2.2 GetSign()	22
4.9.2.3 Produce()	22
4.9.2.4 SetPlayerName()	22
4.9.2.5 SetSign()	22
4.10 MediumStrategy Class Reference	23
4.10.1 Member Function Documentation	23
4.10.1.1 GetDifficulty()	23
4.10.1.2 GetNextMove()	23
4.11 Player Class Reference	24
4.11.1 Member Function Documentation	25
4.11.1.1 GetPlayerName()	25
4.11.1.2 GetSign()	25
4.11.1.3 SetPlayerName()	25
4.11.1.4 SetSign()	26
5 File Documentation	27
5.1 Board.h	27
5.2 Difficulty.h	27
5.3 EasyStrategy.h	27
5.4 Game.h	28
5.5 GameState.h	28
5.6 HardStrategy.h	28
5.7 IDifficultyStrategy.h File Reference	29
5.7.1 Detailed Description	29
5.8 IDifficultyStrategy.h	29
5.9 IGame.h File Reference	29
5.9.1 Detailed Description	30
5.10 IGame.h	30
5.11 IGameListener.h File Reference	30
5.11.1 Detailed Description	31

5.12 IGameListener.h	31
5.13 IPlayer.h File Reference	31
5.13.1 Detailed Description	31
5.14 IPlayer.h	32
5.15 MediumStrategy.h	32
5.16 Player.h	32
5.17 Sign.h	33
Index	35

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Board	7
GameState	12
IDifficultyStrategy	15
EasyStrategy	7
HardStrategy	13
MediumStrategy	23
IGame	16
Game	9
IGameListener	20
IPlayer	21
Player	24

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Board	7
EasyStrategy	7
Game	9
GameState	12
HardStrategy	13
IDifficultyStrategy	
Abstract class representing a difficulty strategy for the Tic Tac Toe game. This class defines an interface for different difficulty strategies that can be implemented to provide different levels of challenge for the game. Subclasses must implement the GetNextMove() method to provide a move based on the current state of the board	15
IGame	
An abstract interface for a game class that manages gameplay and player moves	16
IGameListener	
An interface for classes that listen for updates to the game state and display the game board	20
IPlayer	
The interface for a Tic Tac Toe player	21
MediumStrategy	23
Player	24

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Board.h	??
Difficulty.h	??
EasyStrategy.h	??
Game.h	??
GameState.h	??
HardStrategy.h	??
IDifficultyStrategy.h	
Header file for the IDifficultyStrategy abstract class	29
IGame.h	
An abstract interface for a game class that manages gameplay and player moves	29
IGameListener.h	
Defines the IGameListener interface for updating game state and displaying the game board	30
IPlayer.h	
Contains the interface for a Tic Tac Toe player	31
MediumStrategy.h	??
Player.h	??
Sign.h	??

Chapter 4

Class Documentation

4.1 Board Class Reference

Public Member Functions

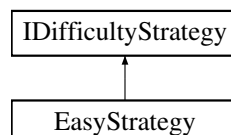
- `std::array< Sign::sign, 9 > GetBoard () const`
- `GameState::gameState CheckGameState ()`
- `void ClearBoard ()`
- `std::vector< uint16_t > GetEmptyCells () const`
- `Sign::sign & operator[] (uint16_t position)`
- `void PrintBoard ()`
- `bool CheckTie ()`
- `bool CheckWin (const Sign::sign &sign)`

The documentation for this class was generated from the following files:

- Board.h
- Board.cpp

4.2 EasyStrategy Class Reference

Inheritance diagram for EasyStrategy:



Public Member Functions

- virtual uint16_t [GetNextMove](#) ([Board](#) &board, const Sign::sign computerSign) override
Virtual method to get the next move based on the current state of the board. This method choses a random empty position from the board.
- virtual Difficulty::Level [GetDifficulty](#) () const override
- virtual uint16_t [GetNextMove](#) ([Board](#) &board, const Sign::sign computerSign)=0
Virtual method to get the next move based on the current state of the board. This method must be implemented by subclasses to return a valid move for the computer player based on the current state of the board and on the strategy used.
- virtual Difficulty::Level [GetDifficulty](#) () const =0

4.2.1 Member Function Documentation

4.2.1.1 GetDifficulty()

Difficulty::Level EasyStrategy::GetDifficulty () const [override], [virtual]

Implements [IDifficultyStrategy](#).

4.2.1.2 GetNextMove()

```
uint16_t EasyStrategy::GetNextMove (
    Board & board,
    const Sign::sign computerSign ) [override], [virtual]
```

Virtual method to get the next move based on the current state of the board. This method choses a random empty position from the board.

Parameters

<i>board</i>	The current state of the game board.
<i>computerSign</i>	The sign used by the computer player.

Returns

The index of the board square where the computer player should place their next move.

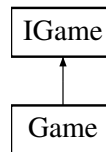
Implements [IDifficultyStrategy](#).

The documentation for this class was generated from the following files:

- EasyStrategy.h
- EasyStrategy.cpp

4.3 Game Class Reference

Inheritance diagram for Game:



Public Member Functions

- **Game** (Difficulty::Level level)
- void **InitializeGame** () override
Initializes the player objects and their order of playing.
- **Board** **GetGameBoard** () override
Gets the current state of the game board.
- **IPlayerPtr** **GetPlayer** () override
Gets a shared pointer to the human player object.
- **IPlayerPtr** **GetComputer** () override
Gets a shared pointer to the computer player object.
- void **AddListeners** (IGameListenerPtr ptr) override
Adds a game listener to the game object.
- void **RemoveListeners** (IGameListenerPtr ptr) override
Removes a game listener from the game object.
- void **NotifyAll** () override
Notifies all registered game listeners of a game event.
- void **RunRound** (uint16_t position) override
Runs a round of the game.
- bool **PlaceSign** (uint16_t position, IPlayerPtr player)
Places the specified player's sign on the game board at the specified position.

Public Member Functions inherited from IGame

- virtual void **AddListeners** (IGameListenerPtr ptr)=0
Adds a game listener to the game object.
- virtual void **RemoveListeners** (IGameListenerPtr ptr)=0
Removes a game listener from the game object.
- virtual void **NotifyAll** ()=0
Notifies all registered game listeners of a game event.
- virtual void **RunRound** (uint16_t position)=0
Runs a round of the game.
- virtual bool **PlaceSign** (uint16_t position, IPlayerPtr player)=0
Places the specified player's sign on the game board at the specified position.
- virtual void **InitializeGame** ()=0
Initializes the player objects and their order of playing.
- virtual **Board** **GetGameBoard** ()=0
Gets the current state of the game board.
- virtual **IPlayerPtr** **GetPlayer** ()=0
Gets a shared pointer to the human player object.
- virtual **IPlayerPtr** **GetComputer** ()=0
Gets a shared pointer to the computer player object.
- virtual **~IGame** ()=default
A virtual destructor for the IGame class.

Additional Inherited Members

Static Public Member Functions inherited from [IGame](#)

- static [IGamePtr](#) [Produce](#) (int difficulty)
Produces a game object with the specified difficulty level.

4.3.1 Member Function Documentation

4.3.1.1 AddListeners()

```
void Game::AddListeners (
    IGameListenerPtr ptr ) [override], [virtual]
```

Adds a game listener to the game object.

Parameters

<i>ptr</i>	A shared pointer to the game listener object.
------------	---

Implements [IGame](#).

4.3.1.2 GetComputer()

```
IPlayerPtr Game::GetComputer ( ) [override], [virtual]
```

Gets a shared pointer to the computer player object.

Returns

A shared pointer to the computer player object.

Implements [IGame](#).

4.3.1.3 GetGameBoard()

```
Board Game::GetGameBoard ( ) [override], [virtual]
```

Gets the current state of the game board.

Returns

A [Board](#) object representing the current state of the game board.

Implements [IGame](#).

4.3.1.4 GetPlayer()

```
IPlayerPtr Game::GetPlayer ( ) [override], [virtual]
```

Gets a shared pointer to the human player object.

Returns

A shared pointer to the human player object.

Implements [IGame](#).

4.3.1.5 InitializeGame()

```
void Game::InitializeGame ( ) [override], [virtual]
```

Initializes the player objects and their order of playing.

Implements [IGame](#).

4.3.1.6 NotifyAll()

```
void Game::NotifyAll ( ) [override], [virtual]
```

Notifies all registered game listeners of a game event.

Implements [IGame](#).

4.3.1.7 PlaceSign()

```
bool Game::PlaceSign (
    uint16_t position,
    IPlayerPtr player ) [virtual]
```

Places the specified player's sign on the game board at the specified position.

Parameters

<i>position</i>	An integer representing the position on the game board where the player wants to place their sign.
<i>player</i>	A shared pointer to the player object that is placing the sign.

Returns

A boolean value indicating whether or not the sign was successfully placed on the game board.

Implements [IGame](#).

4.3.1.8 RemoveListeners()

```
void Game::RemoveListeners (
    IGameListenerPtr ptr ) [override], [virtual]
```

Removes a game listener from the game object.

Parameters

<i>ptr</i>	A shared pointer to the game listener object.
------------	---

Implements [IGame](#).

4.3.1.9 RunRound()

```
void Game::RunRound (
    uint16_t position ) [override], [virtual]
```

Runs a round of the game.

Parameters

<i>position</i>	An integer representing the position on the game board where the player wants to place their sign.
-----------------	--

Implements [IGame](#).

The documentation for this class was generated from the following files:

- Game.h
- Game.cpp

4.4 GameState Class Reference

Public Types

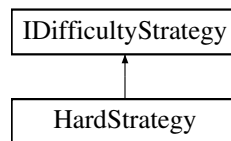
- enum **gameState** { **WonX** , **WonO** , **Tie** , **Undetermined** }

The documentation for this class was generated from the following file:

- GameState.h

4.5 HardStrategy Class Reference

Inheritance diagram for HardStrategy:



Public Member Functions

- virtual uint16_t [GetNextMove](#) ([Board](#) &board, const Sign::sign computerSign) override
Virtual method to get the next move based on the current state of the board. This method uses the Minimax alpha-beta pruning algorithm to find the best possible position to place the sign.
- int [Minimax](#) ([Board](#) &board, int depth, int alpha, int beta, bool maximizingPlayer, const Sign::sign &computer↔ Sign)
Implementation of the minimax algorithm used by the [HardStrategy](#) difficulty strategy. This function recursively applies the minimax algorithm to determine the best move for the computer player. It assigns a score to each possible move based on the resulting state of the board, and returns the maximum or minimum score depending on whether it is maximizing or minimizing the score for the computer player.
- virtual Difficulty::Level [GetDifficulty](#) () const override
- virtual uint16_t [GetNextMove](#) ([Board](#) &board, const Sign::sign computerSign)=0
Virtual method to get the next move based on the current state of the board. This method must be implemented by subclasses to return a valid move for the computer player based on the current state of the board and on the strategy used.
- virtual Difficulty::Level [GetDifficulty](#) () const =0

4.5.1 Member Function Documentation

4.5.1.1 GetDifficulty()

```
Difficulty::Level HardStrategy::GetDifficulty ( ) const [override], [virtual]
```

Implements [IDifficultyStrategy](#).

4.5.1.2 GetNextMove()

```
uint16_t HardStrategy::GetNextMove (
    Board & board,
    const Sign::sign computerSign ) [override], [virtual]
```

Virtual method to get the next move based on the current state of the board. This method uses the Minimax alpha-beta pruning algorithm to find the best possible position to place the sign.

Parameters

<i>board</i>	The current state of the game board.
<i>computerSign</i>	The sign used by the computer player.

Returns

The index of the board square where the computer player should place their next move.

Implements [IDifficultyStrategy](#).

4.5.1.3 Minimax()

```
int HardStrategy::Minimax (
    Board & board,
    int depth,
    int alpha,
    int beta,
    bool maximizingPlayer,
    const Sign::sign & computerSign )
```

Implementation of the minimax algorithm used by the [HardStrategy](#) difficulty strategy. This function recursively applies the minimax algorithm to determine the best move for the computer player. It assigns a score to each possible move based on the resulting state of the board, and returns the maximum or minimum score depending on whether it is maximizing or minimizing the score for the computer player.

Parameters

<i>board</i>	The current state of the game board.
<i>depth</i>	The current depth of the search tree.
<i>alpha</i>	The alpha value for alpha-beta pruning.
<i>beta</i>	The beta value for alpha-beta pruning.
<i>maximizingPlayer</i>	Whether the function is maximizing the score for the computer player.
<i>computerSign</i>	The sign used by the computer player.

Returns

The score assigned to the best possible move for the computer player.

The documentation for this class was generated from the following files:

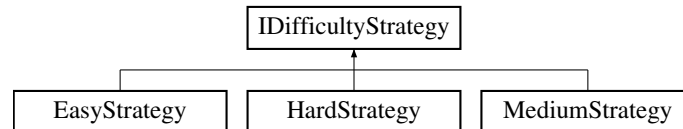
- [HardStrategy.h](#)
- [HardStrategy.cpp](#)

4.6 IDifficultyStrategy Class Reference

Abstract class representing a difficulty strategy for the Tic Tac Toe game. This class defines an interface for different difficulty strategies that can be implemented to provide different levels of challenge for the game. Subclasses must implement the [GetNextMove\(\)](#) method to provide a move based on the current state of the board.

```
#include <IDifficultyStrategy.h>
```

Inheritance diagram for IDifficultyStrategy:



Public Member Functions

- virtual uint16_t [GetNextMove](#) ([Board](#) &board, const Sign::sign computerSign)=0
Virtual method to get the next move based on the current state of the board. This method must be implemented by subclasses to return a valid move for the computer player based on the current state of the board and on the strategy used.
- virtual Difficulty::Level [GetDifficulty](#) () const =0

4.6.1 Detailed Description

Abstract class representing a difficulty strategy for the Tic Tac Toe game. This class defines an interface for different difficulty strategies that can be implemented to provide different levels of challenge for the game. Subclasses must implement the [GetNextMove\(\)](#) method to provide a move based on the current state of the board.

4.6.2 Member Function Documentation

4.6.2.1 GetNextMove()

```
virtual uint16_t IDifficultyStrategy::GetNextMove (
    Board & board,
    const Sign::sign computerSign ) [pure virtual]
```

Virtual method to get the next move based on the current state of the board. This method must be implemented by subclasses to return a valid move for the computer player based on the current state of the board and on the strategy used.

Parameters

<i>board</i>	The current state of the game board.
<i>computerSign</i>	The sign used by the computer player.

Returns

The index of the board square where the computer player should place their next move.

Implemented in [EasyStrategy](#), [HardStrategy](#), and [MediumStrategy](#).

The documentation for this class was generated from the following file:

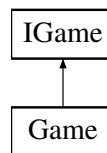
- [IDifficultyStrategy.h](#)

4.7 IGame Class Reference

An abstract interface for a game class that manages gameplay and player moves.

```
#include <IGame.h>
```

Inheritance diagram for IGame:



Public Member Functions

- virtual void [AddListeners](#) ([IGameListenerPtr](#) ptr)=0
Adds a game listener to the game object.
- virtual void [RemoveListeners](#) ([IGameListenerPtr](#) ptr)=0
Removes a game listener from the game object.
- virtual void [NotifyAll](#) ()=0
Notifies all registered game listeners of a game event.
- virtual void [RunRound](#) (uint16_t position)=0
Runs a round of the game.
- virtual bool [PlaceSign](#) (uint16_t position, [IPlayerPtr](#) player)=0
Places the specified player's sign on the game board at the specified position.
- virtual void [InitializeGame](#) ()=0
Initializes the player objects and their order of playing.
- virtual [Board](#) [GetGameBoard](#) ()=0
Gets the current state of the game board.
- virtual [IPlayerPtr](#) [GetPlayer](#) ()=0
Gets a shared pointer to the human player object.
- virtual [IPlayerPtr](#) [GetComputer](#) ()=0
Gets a shared pointer to the computer player object.
- virtual ~**IGame** ()=default
A virtual destructor for the [IGame](#) class.

Static Public Member Functions

- static [IGamePtr Produce](#) (int difficulty)
Produces a game object with the specified difficulty level.

4.7.1 Detailed Description

An abstract interface for a game class that manages gameplay and player moves.

4.7.2 Member Function Documentation

4.7.2.1 AddListeners()

```
virtual void IGame::AddListeners (
    IGameListenerPtr ptr ) [pure virtual]
```

Adds a game listener to the game object.

Parameters

<i>ptr</i>	A shared pointer to the game listener object.
------------	---

Implemented in [Game](#).

4.7.2.2 GetComputer()

```
virtual IPlayerPtr IGame::GetComputer ( ) [pure virtual]
```

Gets a shared pointer to the computer player object.

Returns

A shared pointer to the computer player object.

Implemented in [Game](#).

4.7.2.3 GetGameBoard()

```
virtual Board IGame::GetGameBoard ( ) [pure virtual]
```

Gets the current state of the game board.

Returns

A [Board](#) object representing the current state of the game board.

Implemented in [Game](#).

4.7.2.4 GetPlayer()

```
virtual IPlayerPtr IGame::GetPlayer ( ) [pure virtual]
```

Gets a shared pointer to the human player object.

Returns

A shared pointer to the human player object.

Implemented in [Game](#).

4.7.2.5 InitializeGame()

```
virtual void IGame::InitializeGame ( ) [pure virtual]
```

Initializes the player objects and their order of playing.

Implemented in [Game](#).

4.7.2.6 NotifyAll()

```
virtual void IGame::NotifyAll ( ) [pure virtual]
```

Notifies all registered game listeners of a game event.

Implemented in [Game](#).

4.7.2.7 PlaceSign()

```
virtual bool IGame::PlaceSign (
    uint16_t position,
    IPlayerPtr player ) [pure virtual]
```

Places the specified player's sign on the game board at the specified position.

Parameters

<i>position</i>	An integer representing the position on the game board where the player wants to place their sign.
<i>player</i>	A shared pointer to the player object that is placing the sign.

Returns

A boolean value indicating whether or not the sign was successfully placed on the game board.

Implemented in [Game](#).

4.7.2.8 Produce()

```
IGamePtr IGame::Produce (
    int difficulty ) [static]
```

Produces a game object with the specified difficulty level.

Parameters

<i>difficulty</i>	An integer representing the desired difficulty level of the game.
-------------------	---

Returns

A shared pointer to the produced game object.

Note

If an invalid difficulty level is provided, the function will default to easy mode.

4.7.2.9 RemoveListeners()

```
virtual void IGame::RemoveListeners (
    IGameListenerPtr ptr ) [pure virtual]
```

Removes a game listener from the game object.

Parameters

<i>ptr</i>	A shared pointer to the game listener object.
------------	---

Implemented in [Game](#).

4.7.2.10 RunRound()

```
virtual void IGame::RunRound (
    uint16_t position ) [pure virtual]
```

Runs a round of the game.

Parameters

<i>position</i>	An integer representing the position on the game board where the player wants to place their sign.
-----------------	--

Implemented in [Game](#).

The documentation for this class was generated from the following files:

- [IGame.h](#)
- [IGame.cpp](#)

4.8 IGameListener Class Reference

An interface for classes that listen for updates to the game state and display the game board.

```
#include <IGameListener.h>
```

Public Member Functions

- virtual void **Update** ()=0
Displays the current state of the board.
- virtual void **ShowGameState** ()=0
Displays the current game state.
- virtual ~**IGameListener** ()=default
Virtual destructor for the interface.

4.8.1 Detailed Description

An interface for classes that listen for updates to the game state and display the game board.

The documentation for this class was generated from the following file:

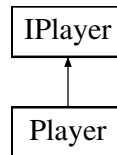
- [IGameListener.h](#)

4.9 IPlayer Class Reference

The interface for a Tic Tac Toe player.

```
#include <IPlayer.h>
```

Inheritance diagram for IPlayer:



Public Member Functions

- virtual std::string [GetPlayerName](#) () const =0
Returns the name of the player.
- virtual void [SetPlayerName](#) (const std::string &name)=0
Sets the name of the player.
- virtual void [SetSign](#) (const Sign::sign &sign)=0
Sets the sign used by the player.
- virtual Sign::sign [GetSign](#) () const =0
Returns the sign used by the player.
- virtual ~**IPlayer** ()=default
Destroys the [IPlayer](#) instance.

Static Public Member Functions

- static [IPlayerPtr](#) [Produce](#) ()
Creates a new [IPlayer](#) instance.

4.9.1 Detailed Description

The interface for a Tic Tac Toe player.

4.9.2 Member Function Documentation

4.9.2.1 GetPlayerName()

```
virtual std::string IPlayer::GetPlayerName ( ) const [pure virtual]
```

Returns the name of the player.

Returns

The name of the player.

Implemented in [Player](#).

4.9.2.2 GetSign()

```
virtual Sign::sign IPlayer::GetSign ( ) const [pure virtual]
```

Returns the sign used by the player.

Returns

The sign used by the player.

Implemented in [Player](#).

4.9.2.3 Produce()

```
IPlayerPtr IPlayer::Produce ( ) [static]
```

Creates a new [IPlayer](#) instance.

Returns

A shared pointer to the new [IPlayer](#) instance.

4.9.2.4 SetPlayerName()

```
virtual void IPlayer::SetPlayerName (
    const std::string & name ) [pure virtual]
```

Sets the name of the player.

Parameters

<i>name</i>	The name of the player.
-------------	-------------------------

Implemented in [Player](#).

4.9.2.5 SetSign()

```
virtual void IPlayer::SetSign (
    const Sign::sign & sign ) [pure virtual]
```

Sets the sign used by the player.

Parameters

<i>sign</i>	The sign used by the player.
-------------	------------------------------

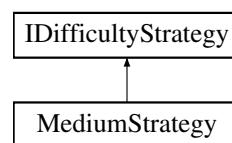
Implemented in [Player](#).

The documentation for this class was generated from the following files:

- [IPlayer.h](#)
- [IPlayer.cpp](#)

4.10 MediumStrategy Class Reference

Inheritance diagram for MediumStrategy:



Public Member Functions

- virtual uint16_t [GetNextMove](#) ([Board](#) &board, const Sign::sign computerSign) override
Virtual method to get the next move based on the current state of the board. This method initialises the queue of order and uses the first strategy from the queue, then adds it to the back of the queue. This function uses at a time [EasyStrategy](#) and [HardStrategy](#).
- virtual Difficulty::Level [GetDifficulty](#) () const override
- virtual uint16_t [GetNextMove](#) ([Board](#) &board, const Sign::sign computerSign)=0
Virtual method to get the next move based on the current state of the board. This method must be implemented by subclasses to return a valid move for the computer player based on the current state of the board and on the strategy used.
- virtual Difficulty::Level [GetDifficulty](#) () const =0

4.10.1 Member Function Documentation

4.10.1.1 GetDifficulty()

```
Difficulty::Level MediumStrategy::GetDifficulty ( ) const [override], [virtual]
```

Implements [IDifficultyStrategy](#).

4.10.1.2 GetNextMove()

```
uint16_t MediumStrategy::GetNextMove (
    Board & board,
    const Sign::sign computerSign ) [override], [virtual]
```

Virtual method to get the next move based on the current state of the board. This method initialises the queue of order and uses the first strategy from the queue, then adds it to the back of the queue. This function uses at a time [EasyStrategy](#) and [HardStrategy](#).

Parameters

<i>board</i>	The current state of the game board.
<i>computerSign</i>	The sign used by the computer player.

Returns

The index of the board square where the computer player should place their next move.

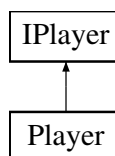
Implements [IDifficultyStrategy](#).

The documentation for this class was generated from the following files:

- MediumStrategy.h
- MediumStrategy.cpp

4.11 Player Class Reference

Inheritance diagram for Player:



Public Member Functions

- `std::string GetPlayerName () const`
Returns the name of the player.
- `void SetPlayerName (const std::string &name)`
Sets the name of the player.
- `void SetSign (const Sign::sign &sign)`
Sets the sign used by the player.
- `Sign::sign GetSign () const`
Returns the sign used by the player.

Public Member Functions inherited from [IPlayer](#)

- `virtual std::string GetPlayerName () const =0`
Returns the name of the player.
- `virtual void SetPlayerName (const std::string &name)=0`
Sets the name of the player.
- `virtual void SetSign (const Sign::sign &sign)=0`
Sets the sign used by the player.
- `virtual Sign::sign GetSign () const =0`
Returns the sign used by the player.
- `virtual ~IPlayer ()=default`
Destroys the [IPlayer](#) instance.

Additional Inherited Members

Static Public Member Functions inherited from [IPlayer](#)

- static [IPlayerPtr](#) [Produce](#) ()
Creates a new [IPlayer](#) instance.

4.11.1 Member Function Documentation

4.11.1.1 [GetPlayerName\(\)](#)

```
std::string Player::GetPlayerName ( ) const [virtual]
```

Returns the name of the player.

Returns

The name of the player.

Implements [IPlayer](#).

4.11.1.2 [GetSign\(\)](#)

```
Sign::sign Player::GetSign ( ) const [virtual]
```

Returns the sign used by the player.

Returns

The sign used by the player.

Implements [IPlayer](#).

4.11.1.3 [SetPlayerName\(\)](#)

```
void Player::SetPlayerName (  
    const std::string & name ) [virtual]
```

Sets the name of the player.

Parameters

<i>name</i>	The name of the player.
-------------	-------------------------

Implements [IPlayer](#).

4.11.1.4 SetSign()

```
void Player::SetSign (
    const Sign::sign & sign ) [virtual]
```

Sets the sign used by the player.

Parameters

<i>sign</i>	The sign used by the player.
-------------	------------------------------

Implements [IPlayer](#).

The documentation for this class was generated from the following files:

- Player.h
- Player.cpp

Chapter 5

File Documentation

5.1 Board.h

```
00001 #pragma once
00002 #include <array>
00003 #include <vector>
00004 #include <iostream>
00005 #include <algorithm>
00006
00007 #include "Sign.h"
00008 #include "GameState.h"
00009
00010 class Board
00011 {
00012
00013 public:
00014     Board();
00015     std::array<Sign::sign, 9> GetBoard() const;
00016     GameState::gameState CheckGameState();
00017     void ClearBoard();
00018     std::vector<uint16_t> GetEmptyCells() const;
00019     Sign::sign& operator[](uint16_t position);
00020     void PrintBoard();
00021     bool CheckTie();
00022     bool CheckWin(const Sign::sign& sign);
00023
00024 private:
00025     Sign::sign GetSign(uint16_t position) const;
00026
00027 private:
00028     std::array<Sign::sign, 9> m_board;
00029 };
00030
```

5.2 Difficulty.h

```
00001 #pragma once
00002 static class Difficulty {
00003 public:
00004     enum class Level {
00005         Easy,
00006         Medium,
00007         Hard
00008     };
00009 };

```

5.3 EasyStrategy.h

```
00001 #pragma once
00002
00003 #include "IDifficultyStrategy.h"
00004
```

```

00005 class EasyStrategy : public IDifficultyStrategy
00006 {
00007 public:
00015     virtual uint16_t GetNextMove(Board& board, const Sign::sign computerSign) override;
00016     virtual Difficulty::Level GetDifficulty() const override;
00017 };
00018

```

5.4 Game.h

```

00001 #pragma once
00002 #include <cstdlib>
00003 #include <iostream>
00004 #include <queue>
00005
00006 #include "IGame.h"
00007 #include "EasyStrategy.h"
00008 #include "MediumStrategy.h"
00009 #include "HardStrategy.h"
00010
00011 class Game: public IGame
00012 {
00013     Board m_board;
00014     IPlayerPtr m_player;
00015     IPlayerPtr m_computer;
00016     std::queue<IPlayerPtr> m_order;
00017     std::vector<IGameListenerPtr> m_listeners;
00018     std::shared_ptr<IDifficultyStrategy> m_strategy;
00019
00020 public:
00021     Game(Difficulty::Level level);
00022     void InitializeGame() override;
00023     Board GetGameBoard() override;
00024     IPlayerPtr GetPlayer() override;
00025     IPlayerPtr GetComputer() override;
00026     void AddListeners(IGameListenerPtr ptr) override;
00027     void RemoveListeners(IGameListenerPtr ptr) override;
00028     void NotifyAll() override;
00029     void RunRound(uint16_t position) override;
00030     bool PlaceSign(uint16_t position, IPlayerPtr player);
00031 };
00032

```

5.5 GameState.h

```

00001 #pragma once
00002
00003 class GameState
00004 {
00005 public:
00006     enum gameState {
00007         WonX,
00008         WonO,
00009         Tie,
00010         Undetermined
00011     };
00012 };

```

5.6 HardStrategy.h

```

00001 #pragma once
00002
00003 #include "IDifficultyStrategy.h"
00004
00005 class HardStrategy : public IDifficultyStrategy
00006 {
00007 public:
00016     virtual uint16_t GetNextMove(Board& board, const Sign::sign computerSign) override;
00017
00031     int Minimax(Board& board, int depth, int alpha, int beta, bool maximizingPlayer, const Sign::sign&
computerSign);
00032     virtual Difficulty::Level GetDifficulty() const override;
00033 };
00034

```

5.7 IDifficultyStrategy.h File Reference

Header file for the [IDifficultyStrategy](#) abstract class.

```
#include <cstdlib>
#include "Board.h"
#include "Difficulty.h"
```

Classes

- class [IDifficultyStrategy](#)

Abstract class representing a difficulty strategy for the Tic Tac Toe game. This class defines an interface for different difficulty strategies that can be implemented to provide different levels of challenge for the game. Subclasses must implement the [GetNextMove\(\)](#) method to provide a move based on the current state of the board.

5.7.1 Detailed Description

Header file for the [IDifficultyStrategy](#) abstract class.

5.8 IDifficultyStrategy.h

[Go to the documentation of this file.](#)

```
00001
00006 #pragma once
00007
00008 #include <cstdlib>
00009
00010 #include "Board.h"
00011 #include "Difficulty.h"
00012
00020 class IDifficultyStrategy {
00021 public:
00030     virtual uint16_t GetNextMove(Board& board, const Sign::sign computerSign) = 0;
00031     virtual Difficulty::Level GetDifficulty() const = 0;
00032 };
```

5.9 IGame.h File Reference

An abstract interface for a game class that manages gameplay and player moves.

```
#include <memory>
#include "IGameListener.h"
#include "IPlayer.h"
#include "Board.h"
#include "Difficulty.h"
```

Classes

- class [IGame](#)

An abstract interface for a game class that manages gameplay and player moves.

Typedefs

- using **IGamePtr** = std::shared_ptr< class [IGame](#) >
A shared pointer to an [IGame](#) object.

5.9.1 Detailed Description

An abstract interface for a game class that manages gameplay and player moves.

5.10 IGame.h

[Go to the documentation of this file.](#)

```
00001
00006 #pragma once
00007
00008 #include <memory>
00009
00010 #include "IGameListener.h"
00011 #include "IPlayer.h"
00012 #include "Board.h"
00013 #include "Difficulty.h"
00014
00019 using IGamePtr = std::shared_ptr<class IGame>;
00024 class IGame
00025 {
00026 public:
00031     virtual void AddListeners(IGameListenerPtr ptr) = 0;
00032
00037     virtual void RemoveListeners(IGameListenerPtr ptr) = 0;
00038
00042     virtual void NotifyAll() = 0;
00043
00049     static IGamePtr Produce(int difficulty);
00050
00055     virtual void RunRound(uint16_t position) = 0;
00056
00063     virtual bool PlaceSign(uint16_t position, IPlayerPtr player) = 0;
00064
00068     virtual void InitializeGame() = 0;
00069
00074     virtual Board GetGameBoard() = 0;
00075
00080     virtual IPlayerPtr GetPlayer() = 0;
00081
00086     virtual IPlayerPtr GetComputer() = 0;
00087
00091     virtual ~IGame() = default;
00092 };
```

5.11 IGameListener.h File Reference

Defines the [IGameListener](#) interface for updating game state and displaying the game board.

```
#include <memory>
```

Classes

- class [IGameListener](#)
An interface for classes that listen for updates to the game state and display the game board.

Typedefs

- using **IGameListenerPtr** = std::shared_ptr< class [IGameListener](#) >
A shared pointer to an [IGameListener](#) object.

5.11.1 Detailed Description

Defines the [IGameListener](#) interface for updating game state and displaying the game board.

5.12 IGameListener.h

[Go to the documentation of this file.](#)

```
00001
00006 #pragma once
00007 #include <memory>
00008
00013 using IGameListenerPtr = std::shared_ptr<class IGameListener>;
00014
00021 class IGameListener
00022 {
00023 public:
00028     virtual void Update() = 0;
00029
00034     virtual void ShowGameState() = 0;
00035
00040     virtual ~IGameListener() = default;
00041 };
```

5.13 IPlayer.h File Reference

Contains the interface for a Tic Tac Toe player.

```
#include <memory>
#include <string>
#include "Sign.h"
```

Classes

- class [IPlayer](#)
The interface for a Tic Tac Toe player.

Typedefs

- using **IPlayerPtr** = std::shared_ptr< class [IPlayer](#) >
A shared pointer alias for the [IPlayer](#) class.

5.13.1 Detailed Description

Contains the interface for a Tic Tac Toe player.

5.14 IPlayer.h

[Go to the documentation of this file.](#)

```

00001
00006 #pragma once
00007
00008 #include <memory>
00009 #include <string>
00010
00011 #include "Sign.h"
00012
00017 using IPlayerPtr = std::shared_ptr<class IPlayer>;
00018
00023 class IPlayer
00024 {
00025 public:
00031     static IPlayerPtr Produce();
00032
00038     virtual std::string GetPlayerName() const = 0;
00039
00045     virtual void SetPlayerName(const std::string& name) = 0;
00046
00052     virtual void SetSign(const Sign::sign& sign) = 0;
00053
00059     virtual Sign::sign GetSign() const = 0;
00060
00065     virtual ~IPlayer() = default;
00066 };

```

5.15 MediumStrategy.h

```

00001 #pragma once
00002
00003 #include "IDifficultyStrategy.h"
00004 #include "EasyStrategy.h"
00005 #include "HardStrategy.h"
00006
00007 #include <queue>
00008
00009 class MediumStrategy : public IDifficultyStrategy
00010 {
00011 private:
00012     std::unique_ptr<IDifficultyStrategy> easyMove = std::make_unique<EasyStrategy>();
00013     std::unique_ptr<IDifficultyStrategy> hardMove = std::make_unique<HardStrategy>();
00014     std::queue<std::unique_ptr<IDifficultyStrategy>> order;
00015     bool initialized = false;
00016
00017 public:
00027     virtual uint16_t GetNextMove( Board& board, const Sign::sign computerSign) override;
00028     virtual Difficulty::Level GetDifficulty() const override;
00029 };
00030
00031

```

5.16 Player.h

```

00001 #pragma once
00002 #include <iostream>
00003 #include <string>
00004
00005 #include "Sign.h"
00006 #include "Board.h"
00007 #include "IPlayer.h"
00008
00009 class Player: public IPlayer
00010 {
00011 public:
00012     Player() = default;
00013     std::string GetPlayerName() const;
00014     void SetPlayerName(const std::string& name);
00015     void SetSign(const Sign::sign& sign);
00016     Sign::sign GetSign() const;
00017 private:
00018     Sign::sign m_sign;
00019     std::string m_playerName;
00020 };
00021

```

5.17 Sign.h

```
00001 #pragma once
00002 static class Sign
00003 {
00004 public:
00005     static enum class sign
00006     {
00007         X,
00008         O,
00009         None
00010     };
00011 };
```


Index

- AddListeners
 - Game, [10](#)
 - IGame, [17](#)
- Board, [7](#)
- EasyStrategy, [7](#)
 - GetDifficulty, [8](#)
 - GetNextMove, [8](#)
- Game, [9](#)
 - AddListeners, [10](#)
 - GetComputer, [10](#)
 - GetGameBoard, [10](#)
 - GetPlayer, [10](#)
 - InitializeGame, [11](#)
 - NotifyAll, [11](#)
 - PlaceSign, [11](#)
 - RemoveListeners, [12](#)
 - RunRound, [12](#)
- GameState, [12](#)
- GetComputer
 - Game, [10](#)
 - IGame, [17](#)
- GetDifficulty
 - EasyStrategy, [8](#)
 - HardStrategy, [13](#)
 - MediumStrategy, [23](#)
- GetGameBoard
 - Game, [10](#)
 - IGame, [17](#)
- GetNextMove
 - EasyStrategy, [8](#)
 - HardStrategy, [13](#)
 - IDifficultyStrategy, [15](#)
 - MediumStrategy, [23](#)
- GetPlayer
 - Game, [10](#)
 - IGame, [18](#)
- GetPlayerName
 - IPlayer, [21](#)
 - Player, [25](#)
- GetSign
 - IPlayer, [21](#)
 - Player, [25](#)
- HardStrategy, [13](#)
 - GetDifficulty, [13](#)
 - GetNextMove, [13](#)
 - Minimax, [14](#)
- IDifficultyStrategy, [15](#)
 - GetNextMove, [15](#)
- IDifficultyStrategy.h, [29](#)
- IGame, [16](#)
 - AddListeners, [17](#)
 - GetComputer, [17](#)
 - GetGameBoard, [17](#)
 - GetPlayer, [18](#)
 - InitializeGame, [18](#)
 - NotifyAll, [18](#)
 - PlaceSign, [18](#)
 - Produce, [19](#)
 - RemoveListeners, [19](#)
 - RunRound, [19](#)
- IGame.h, [29](#)
- IGameListener, [20](#)
- IGameListener.h, [30](#)
- InitializeGame
 - Game, [11](#)
 - IGame, [18](#)
- IPlayer, [21](#)
 - GetPlayerName, [21](#)
 - GetSign, [21](#)
 - Produce, [22](#)
 - SetPlayerName, [22](#)
 - SetSign, [22](#)
- IPlayer.h, [31](#)
- MediumStrategy, [23](#)
 - GetDifficulty, [23](#)
 - GetNextMove, [23](#)
- Minimax
 - HardStrategy, [14](#)
- NotifyAll
 - Game, [11](#)
 - IGame, [18](#)
- PlaceSign
 - Game, [11](#)
 - IGame, [18](#)
- Player, [24](#)
 - GetPlayerName, [25](#)
 - GetSign, [25](#)
 - SetPlayerName, [25](#)
 - SetSign, [26](#)
- Produce
 - IGame, [19](#)
 - IPlayer, [22](#)
- RemoveListeners

- Game, [12](#)
- IGame, [19](#)
- RunRound
 - Game, [12](#)
 - IGame, [19](#)
- SetPlayerName
 - IPlayer, [22](#)
 - Player, [25](#)
- SetSign
 - IPlayer, [22](#)
 - Player, [26](#)