TicTacToe

Generated by Doxygen 1.9.6

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

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

Chapter 4

Class Documentation

4.1 Board Class Reference

Public Member Functions

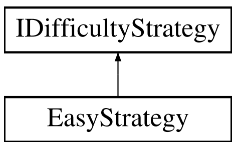
- std::array< Sign::sign, 9 > **GetBoard** () const
- GameState::gameState **CheckGameState** ()
- void **ClearBoard** ()
- std::vector< uint16_t > **GetEmptyCells** () const
- Sign::sign & **operator[ ]** (uint16_t position)
- void **PrintBoard** ()
- bool **CheckTie** ()
- bool **CheckWin** (const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

board	The current state of the game board.
computerSign	The sign used by the computer player.

Returns

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

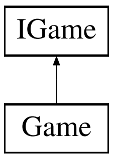
Implements IDifficultyStrategy.

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

ptr	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

position	An integer representing the position on the game board where the player wants to place their sign.
player	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

ptr	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

position	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

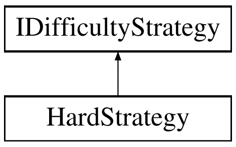
- enum **gameState** { **WonX** , **WonO** , **Tie** , **Undetermined** }

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

board	The current state of the game board.
computerSign	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

board	The current state of the game board.
depth	The current depth of the search tree.
alpha	The alpha value for alpha-beta pruning.
beta	The beta value for alpha-beta pruning.
maximizingPlayer	Whether the function is maximizing the score for the computer player.
computerSign	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:

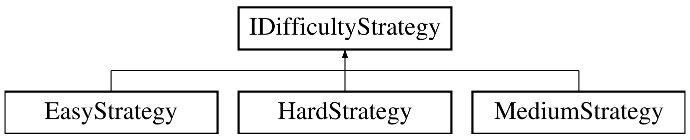
- HardStrategy.h
- HardStrategy.cpp

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

board	The current state of the game board.
computerSign	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:

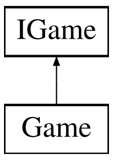
- IDifficultyStrategy.h

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

ptr	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

position	An integer representing the position on the game board where the player wants to place their sign.
player	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

difficulty	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

ptr	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

position	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:

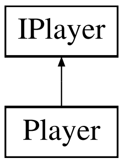
- IGameListener.h

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

name	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

sign	The sign used by the player.

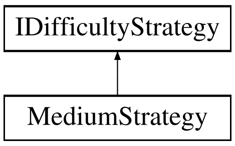
Implemented in Player.

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

board	The current state of the game board.
computerSign	The sign used by the computer player.

Returns

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

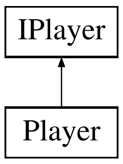
Implements IDifficultyStrategy.

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

name	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

sign	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

```
#pragma once
#include <array>
#include <vector>
#include <iostream>
#include <algorithm>

#include "Sign.h"
#include "GameState.h"

class Board
{

public:
    Board();
    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);

private:
    Sign::sign GetSign(uint16_t position) const;

private:
    std::array<Sign::sign, 9> m_board;
};

```

5.2 Difficulty.h

```
#pragma once
static class Difficulty {
public:
    enum class Level {
        Easy,
        Medium,
        Hard
    };
};
```

5.3 EasyStrategy.h

```
#pragma once

#include "IDifficultyStrategy.h"

```

```
class EasyStrategy : public IDifficultyStrategy
{
public:
    virtual uint16_t GetNextMove(Board& board, const Sign::sign computerSign) override;
    virtual Difficulty::Level GetDifficulty() const override;
};

```

5.4 Game.h

```
#pragma once
#include <cstdlib>
#include <iostream>
#include <queue>

#include "IGame.h"
#include "EasyStrategy.h"
#include "MediumStrategy.h"
#include "HardStrategy.h"

class Game: public IGame
{
    Board m_board;
    IPlayerPtr m_player;
    IPlayerPtr m_computer;
    std::queue<IPlayerPtr> m_order;
    std::vector<IGameListenerPtr> m_listeners;
    std::shared_ptr<IDifficultyStrategy> m_strategy;

public:
    Game(Difficulty::Level level);
    void InitializeGame() override;
    Board GetGameBoard() override;
    IPlayerPtr GetPlayer() override;
    IPlayerPtr GetComputer() override;
    void AddListeners(IGameListenerPtr ptr) override;
    void RemoveListeners(IGameListenerPtr ptr) override;
    void NotifyAll() override;
    void RunRound(uint16_t position) override;
    bool PlaceSign(uint16_t position, IPlayerPtr player);
};

```

5.5 GameState.h

```
#pragma once

class GameState
{
public:
    enum gameState {
        WonX,
        WonO,
        Tie,
        Undetermined
    };
};
```

5.6 HardStrategy.h

```
#pragma once

#include "IDifficultyStrategy.h"

class HardStrategy : public IDifficultyStrategy
{
public:
    virtual uint16_t GetNextMove(Board& board, const Sign::sign computerSign) override;

    int Minimax(Board& board, int depth, int alpha, int beta, bool maximizingPlayer, const Sign::sign& computerSign);
    virtual Difficulty::Level GetDifficulty() const override;
};

```

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.

```

#pragma once

#include <cstdlib>

#include "Board.h"
#include "Difficulty.h"

class IDifficultyStrategy {
public:
    virtual uint16_t GetNextMove(Board& board, const Sign::sign computerSign) = 0;
    virtual Difficulty::Level GetDifficulty() const = 0;
};
```

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.

```

#pragma once

#include <memory>

#include "IGameListener.h"
#include "IPlayer.h"
#include "Board.h"
#include "Difficulty.h"

using IGamePtr = std::shared_ptr<class IGame>;
class IGame
{
public:
    virtual void AddListeners(IGameListenerPtr ptr) = 0;

    virtual void RemoveListeners(IGameListenerPtr ptr) = 0;

    virtual void NotifyAll() = 0;

    static IGamePtr Produce(int difficulty);

    virtual void RunRound(uint16_t position) = 0;

    virtual bool PlaceSign(uint16_t position, IPlayerPtr player) = 0;

    virtual void InitializeGame() = 0;

    virtual Board GetGameBoard() = 0;

    virtual IPlayerPtr GetPlayer() = 0;

    virtual IPlayerPtr GetComputer() = 0;

    virtual ~IGame() = default;
};
```

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.

```
#pragma once
#include <memory>

using IGameListenerPtr = std::shared_ptr<class IGameListener>;

class IGameListener
{
public:
    virtual void Update() = 0;

    virtual void ShowGameState() = 0;

    virtual ~IGameListener() = default;
};
```

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.

```

#pragma once

#include <memory>
#include <string>

#include "Sign.h"

using IPlayerPtr = std::shared_ptr<class IPlayer>;

class IPlayer
{
public:
    static IPlayerPtr Produce();

    virtual std::string GetPlayerName() const = 0;

    virtual void SetPlayerName(const std::string& name) = 0;

    virtual void SetSign(const Sign::sign& sign) = 0;

    virtual Sign::sign GetSign() const = 0;

    virtual ~IPlayer() = default;
};
```

5.15 MediumStrategy.h

```
#pragma once

#include "IDifficultyStrategy.h"
#include "EasyStrategy.h"
#include "HardStrategy.h"

#include <queue>

class MediumStrategy : public IDifficultyStrategy
{
private:
    std::unique_ptr<IDifficultyStrategy> easyMove = std::make_unique<EasyStrategy>();
    std::unique_ptr<IDifficultyStrategy> hardMove = std::make_unique<HardStrategy>();
    std::queue<std::unique_ptr<IDifficultyStrategy» order;
    bool initialized = false;

public:
    virtual uint16_t GetNextMove( Board& board, const Sign::sign computerSign) override;
    virtual Difficulty::Level GetDifficulty() const override;
};

```

5.16 Player.h

```
#pragma once
#include <iostream>
#include <string>

#include "Sign.h"
#include "Board.h"
#include "IPlayer.h"

class Player: public IPlayer
{
public:
    Player() = default;
    std::string GetPlayerName() const;
    void SetPlayerName(const std::string& name);
    void SetSign(const Sign::sign& sign);
    Sign::sign GetSign() const;
private:
    Sign::sign m_sign;
    std::string m_playerName;
};

```

5.17 Sign.h

```
#pragma once
static class Sign
{
public:
    static enum class sign
    {
        X,
        O,
        None
    };
};
```

Index