Projeto Final de PDS II

Generated by Doxygen 1.9.8

1 README 1

2 Hierarchical Index 3

2.1 Class Hierarchy . . . 3

3 Class Index 5

3.1 Class List . . . 5

4 File Index 7

4.1 File List . . . 7

5 Class Documentation 9

5.1 Estatisticas Class Reference . . . 9

5.1.1 Detailed Description . . . 9

5.1.2 Constructor & Destructor Documentation . . . 10

5.1.2.1 Estatisticas() . . . 10

5.1.3 Member Function Documentation . . . 10

5.1.3.1 desserializar() . . . 10

5.1.3.2 getDerrotas() . . . 10

5.1.3.3 getVitorias() . . . 10

5.1.3.4 imprimirEstatisticas() . . . 11

5.1.3.5 incrementarDerrota() . . . 11

5.1.3.6 incrementarVitoria() . . . 11

5.1.3.7 serializar() . . . 11

5.2 GerenciadorDeJogos Class Reference . . . 12

5.2.1 Detailed Description . . . 12

5.2.2 Member Function Documentation . . . 12

5.2.2.1 executarPartida() . . . 12

5.3 Jogador Class Reference . . . 13

5.3.1 Detailed Description . . . 14

5.3.2 Constructor & Destructor Documentation . . . 14

5.3.2.1 Jogador() . . . 14

5.3.3 Member Function Documentation . . . 14

5.3.3.1 desserializar() . . . 14

5.3.3.2 getApelido() . . . 14

5.3.3.3 getNome() . . . 15

5.3.3.4 imprimirEstatisticas() . . . 15

5.3.3.5 incrementarDerrota() . . . 15

5.3.3.6 incrementarVitoria() . . . 15

5.3.3.7 serializar() . . . 15

5.4 JogoDaVelha Class Reference . . . 16

5.5 JogoDeTabuleiro Class Reference . . . 16

5.5.1 Detailed Description . . . 17

5.5.2 Constructor & Destructor Documentation . . . 17

5.5.2.1 ~JogoDeTabuleiro() . . . 17
5.5.3 Member Function Documentation . . . 17
5.5.3.1 imprimirTabuleiro() . . . 17
5.5.3.2 iniciarJogo() . . . 17
5.5.3.3 jogadaValida() . . . 17
5.5.3.4 realizarJogada() . . . 18
5.5.3.5 tabuleiroCheio() . . . 18
5.5.3.6 verificarVitoria() . . . 19
5.5.4 Member Data Documentation . . . 19
5.5.4.1 tabuleiro . . . 19
5.6 Lig4 Class Reference . . . 19
5.6.1 Detailed Description . . . 20
5.6.2 Constructor & Destructor Documentation . . . 20
5.6.2.1 Lig4() . . . 20
5.6.3 Member Function Documentation . . . 20
5.6.3.1 imprimirTabuleiro() . . . 20
5.6.3.2 iniciarJogo() . . . 21
5.6.3.3 jogadaValida() . . . 21
5.6.3.4 realizarJogada() . . . 21
5.6.3.5 tabuleiroCheio() . . . 22
5.6.3.6 verificarVitoria() . . . 22
5.7 ListaDeJogadores Class Reference . . . 22
5.7.1 Detailed Description . . . 23
5.7.2 Constructor & Destructor Documentation . . . 23
5.7.2.1 ListaDeJogadores() . . . 23
5.7.2.2 ~ListaDeJogadores() . . . 23
5.7.3 Member Function Documentation . . . 23
5.7.3.1 adicionarJogador() . . . 23
5.7.3.2 buscarJogador() . . . 24
5.7.3.3 carregarDeArquivo() . . . 24
5.7.3.4 listarJogadores() . . . 24
5.7.3.5 removerJogador() . . . 25
5.7.3.6 salvarEmArquivo() . . . 25
5.8 Reversi Class Reference . . . 25
5.8.1 Detailed Description . . . 26
5.8.2 Constructor & Destructor Documentation . . . 27
5.8.2.1 Reversi() . . . 27
5.8.3 Member Function Documentation . . . 27
5.8.3.1 contarPecas() . . . 27
5.8.3.2 imprimirTabuleiro() . . . 27
5.8.3.3 iniciarJogo() . . . 27
5.8.3.4 jogadaValida() . . . 28

5.8.3.5 realizarJogada() 28
5.8.3.6 tabuleiroCheio() 28
5.8.3.7 verificarVitoria() 29

6 File Documentation **31**

6.1 estatisticas.hpp 31
6.2 gerenciador_de_jogos.hpp 31
6.3 jogador.hpp 32
6.4 jogo_da_velha.hpp 32
6.5 jogos_tabuleiro.hpp 32
6.6 lig4.hpp 33
6.7 lista_de_jogadores.hpp 33
6.8 reversi.hpp 34

Index **35**

Chapter 1

README

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

- Estatisticas . . . 9
- GerenciadorDeJogos . . . 12
- Jogador . . . 13
- JogoDeTabuleiro . . . 16
  - JogoDaVelha . . . 16
  - Lig4 . . . 19
  - Reversi . . . 25
- ListaDeJogadores . . . 22

Chapter 3

Class Index

3.1 Class List

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

Estatisticas
: Classe para gerenciar estatísticas de vitórias e derrotas de diferentes jogos 9

GerenciadorDeJogos
: Classe responsável por gerenciar e executar partidas de jogos de tabuleiro 12

Jogador
: Classe que representa um jogador com informações pessoais e estatísticas de desempenho . 13

JogoDaVelha
: Classe que implementa o jogo da velha como um jogo de tabuleiro 16

JogoDeTabuleiro
: Classe abstrata que define a interface para jogos de tabuleiro 16

Lig4
: Classe que implementa o jogo Lig4 (Conecta 4) 19

ListaDeJogadores
: Classe que representa uma lista de jogadores 22

Reversi
: Classe que representa o jogo Reversi (também conhecido como Othello) 25

Chapter 4

File Index

4.1 File List

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

include/estatisticas.hpp 31
include/gerenciador_de_jogos.hpp 31
include/jogador.hpp 32
include/jogo_da_velha.hpp 32
include/jogos_tabuleiro.hpp 32
include/lig4.hpp 33
include/lista_de_jogadores.hpp 33
include/reversi.hpp 34

Chapter 5

Class Documentation

5.1 Estatisticas Class Reference

Classe para gerenciar estatísticas de vitórias e derrotas de diferentes jogos.

`#include <estatisticas.hpp>`

Public Member Functions

- Estatisticas ()
  Construtor padrão.
- void incrementarVitoria (char jogo)
  Incrementa o número de vitórias para o jogo especificado.
- void incrementarDerrota (char jogo)
  Incrementa o número de derrotas para o jogo especificado.
- int getVitorias (char jogo) const
  Retorna o número de vitórias para o jogo especificado.
- int getDerrotas (char jogo) const
  Retorna o número de derrotas para o jogo especificado.
- void imprimirEstatisticas () const
  Imprime as estatísticas de vitórias e derrotas para todos os jogos.
- std::string serializar () const
  Serializa os dados de vitórias e derrotas em uma string.
- void desserializar (const std::string &dados)
  Desserializa uma string e preenche os mapas de vitórias e derrotas.

5.1.1 Detailed Description

Classe para gerenciar estatísticas de vitórias e derrotas de diferentes jogos.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 Estatisticas()

```
Estatisticas::Estatisticas ( )
```

Construtor padrão.

Inicializa os mapas de vitórias e derrotas com valores zerados para os jogos padrão ('R', 'L', 'V').

5.1.3 Member Function Documentation

5.1.3.1 desserializar()

```
void Estatisticas::desserializar (
            const std::string & dados )
```

Desserializa uma string e preenche os mapas de vitórias e derrotas.

Parameters

dados	String no formato "<jogo>,<vitórias>,<derrotas>;".

5.1.3.2 getDerrotas()

```
int Estatisticas::getDerrotas (
            char jogo ) const
```

Retorna o número de derrotas para o jogo especificado.

Parameters

jogo	Caractere que identifica o jogo.

Returns

Número de derrotas para o jogo ou 0 se o jogo não existir no mapa.

5.1.3.3 getVitorias()

```
int Estatisticas::getVitorias (
            char jogo ) const
```

Retorna o número de vitórias para o jogo especificado.

Parameters

jogo	Caractere que identifica o jogo.

Returns

Número de vitórias para o jogo ou 0 se o jogo não existir no mapa.

5.1.3.4 imprimirEstatisticas()

```
void Estatisticas::imprimirEstatisticas ( ) const
```

Imprime as estatísticas de vitórias e derrotas para todos os jogos.

O formato de saída é: "<jogo> - V: <vitórias> D: <derrotas>".

5.1.3.5 incrementarDerrota()

```
void Estatisticas::incrementarDerrota (
            char jogo )
```

Incrementa o número de derrotas para o jogo especificado.

Parameters

jogo	Caractere que identifica o jogo.

5.1.3.6 incrementarVitoria()

```
void Estatisticas::incrementarVitoria (
            char jogo )
```

Incrementa o número de vitórias para o jogo especificado.

Parameters

jogo	Caractere que identifica o jogo.

5.1.3.7 serializar()

```
std::string Estatisticas::serializar ( ) const
```

Serializa os dados de vitórias e derrotas em uma string.

O formato da string é: "<jogo>,<vitórias>,<derrotas>;" para cada jogo.

Returns

String serializada contendo os dados.

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

- include/estatisticas.hpp
- src/estatisticas.cpp

5.2 GerenciadorDeJogos Class Reference

Classe responsável por gerenciar e executar partidas de jogos de tabuleiro.

`#include <gerenciador_de_jogos.hpp>`

Static Public Member Functions

- static void executarPartida (char tipoJogo, Jogador &jogador1, Jogador &jogador2)
  Executa uma partida de um jogo de tabuleiro especificado.

5.2.1 Detailed Description

Classe responsável por gerenciar e executar partidas de jogos de tabuleiro.

Suporta os seguintes jogos:

- 'R': Reversi.
- 'L': Lig4.
- 'V': Jogo da Velha.

5.2.2 Member Function Documentation

5.2.2.1 executarPartida()

```
void GerenciadorDeJogos::executarPartida (
            char tipoJogo,
            Jogador & jogador1,
            Jogador & jogador2 )  [static]
```

Executa uma partida de um jogo de tabuleiro especificado.

Este método gerencia o fluxo de uma partida entre dois jogadores, alternando turnos, verificando condições de vitória ou empate e registrando os resultados.

Parameters

tipoJogo	Caractere que identifica o tipo de jogo a ser jogado. • 'R': Reversi. • 'L': Lig4. • 'V': Jogo da Velha.
jogador1	Referência ao primeiro jogador.
jogador2	Referência ao segundo jogador.

Exceptions

std::invalid_argument	Se o tipo de jogo for inválido.
std::runtime_error	Se ocorrerem erros na entrada do usuário.

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

- include/gerenciador_de_jogos.hpp
- src/gerenciador_de_jogos.cpp

5.3 Jogador Class Reference

Classe que representa um jogador com informações pessoais e estatísticas de desempenho.

`#include <jogador.hpp>`

Public Member Functions

- Jogador (const std::string &apelido, const std::string &nome)
  Construtor da classe Jogador.
- const std::string & getApelido () const
  Obtém o apelido do jogador.
- const std::string & getNome () const
  Obtém o nome completo do jogador.
- void incrementarVitoria (char jogo)
  Incrementa o número de vitórias do jogador para um tipo específico de jogo.
- void incrementarDerrota (char jogo)
  Incrementa o número de derrotas do jogador para um tipo específico de jogo.
- void imprimirEstatisticas () const
  Imprime as estatísticas de desempenho do jogador.
- std::string serializar () const
  Serializa os dados do jogador em uma string.

Static Public Member Functions

- static Jogador desserializar (const std::string &linha)
  Desserializa uma string para criar um objeto Jogador.

5.3.1 Detailed Description

Classe que representa um jogador com informações pessoais e estatísticas de desempenho.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 Jogador()

```
Jogador::Jogador (
            const std::string & apelido,
            const std::string & nome )
```

Construtor da classe Jogador.

Inicializa o jogador com apelido e nome.

Parameters

apelido	Apelido do jogador.
nome	Nome completo do jogador.

5.3.3 Member Function Documentation

5.3.3.1 desserializar()

```
Jogador Jogador::desserializar (
            const std::string & linha )  [static]
```

Desserializa uma string para criar um objeto Jogador.

A string deve estar no formato: "<apelido>|<nome>|<estatísticas>".

Parameters

linha	String contendo os dados serializados.

Returns

Objeto Jogador inicializado com os dados desserializados.

5.3.3.2 getApelido()

```
const std::string & Jogador::getApelido ( ) const
```

Obtém o apelido do jogador.

Returns

Referência constante para a string do apelido.

5.3.3.3 getNome()

```
const std::string & Jogador::getNome ( ) const
```

Obtém o nome completo do jogador.

Returns

Referência constante para a string do nome.

5.3.3.4 imprimirEstatisticas()

```
void Jogador::imprimirEstatisticas ( ) const
```

Imprime as estatísticas de desempenho do jogador.

Exibe o apelido, nome e estatísticas de vitórias e derrotas em um formato legível.

5.3.3.5 incrementarDerrota()

```
void Jogador::incrementarDerrota (
            char jogo )
```

Incrementa o número de derrotas do jogador para um tipo específico de jogo.

Parameters

jogo	Caractere que identifica o tipo de jogo (ex.: 'R', 'L', 'V').

5.3.3.6 incrementarVitoria()

```
void Jogador::incrementarVitoria (
            char jogo )
```

Incrementa o número de vitórias do jogador para um tipo específico de jogo.

Parameters

jogo	Caractere que identifica o tipo de jogo (ex.: 'R', 'L', 'V').

5.3.3.7 serializar()

```
std::string Jogador::serializar ( ) const
```

Serializa os dados do jogador em uma string.

O formato da string é: "<apelido>|<nome>|<estatísticas>".

Returns

Uma string contendo os dados serializados do jogador.

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

- include/jogador.hpp
- src/jogador.cpp

5.4 JogoDaVelha Class Reference

Classe que implementa o jogo da velha como um jogo de tabuleiro.

`#include <jogo_da_velha.hpp>`

Inheritance diagram for JogoDaVelha:

5.5 JogoDeTabuleiro Class Reference

Classe abstrata que define a interface para jogos de tabuleiro.

`#include <jogos_tabuleiro.hpp>`

Inheritance diagram for JogoDeTabuleiro:

Public Member Functions

- virtual ~JogoDeTabuleiro ()
  Destrutor virtual.
- virtual void iniciarJogo ()=0
  Inicializa o estado do jogo.
- virtual bool realizarJogada (int linha, int coluna)=0
  Realiza uma jogada no tabuleiro.
- virtual bool verificarVitoria () const =0
  Verifica se houve uma vitória no jogo.
- virtual void imprimirTabuleiro () const =0
  Imprime o tabuleiro no console.
- virtual bool jogadaValida (int linha, int coluna) const =0
  Verifica se uma jogada é válida.
- virtual bool tabuleiroCheio () const =0
  Verifica se o tabuleiro está cheio.

Protected Attributes

- int **linhas**
  Número de linhas do tabuleiro.
- int **colunas**
  Número de colunas do tabuleiro.
- std::vector< std::vector< char > > tabuleiro
  Representação do tabuleiro como uma matriz 2D de caracteres.

5.5.1 Detailed Description

Classe abstrata que define a interface para jogos de tabuleiro.

Esta classe fornece a estrutura base para implementar diferentes jogos de tabuleiro. Contém métodos virtuais puros que devem ser implementados pelas classes derivadas.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 ~JogoDeTabuleiro()

```
virtual JogoDeTabuleiro::~JogoDeTabuleiro ( )  [inline], [virtual]
```

Destrutor virtual.

Garante que as subclasses possam realizar limpeza de recursos adequadamente.

5.5.3 Member Function Documentation

5.5.3.1 imprimirTabuleiro()

```
virtual void JogoDeTabuleiro::imprimirTabuleiro ( ) const  [pure virtual]
```

Imprime o tabuleiro no console.

Deve ser implementado para exibir o estado atual do tabuleiro de maneira visualmente clara.

Implemented in JogoDaVelha, Lig4, and Reversi.

5.5.3.2 iniciarJogo()

```
virtual void JogoDeTabuleiro::iniciarJogo ( )  [pure virtual]
```

Inicializa o estado do jogo.

Deve ser implementado para configurar o tabuleiro e quaisquer variáveis necessárias antes do início do jogo.

Implemented in JogoDaVelha, Lig4, and Reversi.

5.5.3.3 jogadaValida()

```
virtual bool JogoDeTabuleiro::jogadaValida (
            int linha,
            int coluna ) const  [pure virtual]
```

Verifica se uma jogada é válida.

Este método deve validar se a posição especificada está dentro dos limites do tabuleiro e se a célula correspondente está disponível.

Parameters

linha	Índice da linha a ser verificada.
coluna	Índice da coluna a ser verificada.

Returns

true Se a jogada é válida.

false Caso contrário.

Implemented in JogoDaVelha, Lig4, and Reversi.

5.5.3.4 realizarJogada()

```
virtual bool JogoDeTabuleiro::realizarJogada (
            int linha,
            int coluna )  [pure virtual]
```

Realiza uma jogada no tabuleiro.

O método deve ser implementado para marcar o tabuleiro com base nos índices fornecidos de linha e coluna, se a jogada for válida.

Parameters

linha	Índice da linha onde a jogada será realizada.
coluna	Índice da coluna onde a jogada será realizada.

Returns

true Se a jogada foi realizada com sucesso.

false Se a jogada for inválida.

Implemented in JogoDaVelha, Lig4, and Reversi.

5.5.3.5 tabuleiroCheio()

```
virtual bool JogoDeTabuleiro::tabuleiroCheio ( ) const  [pure virtual]
```

Verifica se o tabuleiro está cheio.

Deve ser implementado para determinar se todas as células do tabuleiro estão ocupadas.

Returns

true Se o tabuleiro está completamente preenchido.

false Caso contrário.

Implemented in JogoDaVelha, Lig4, and Reversi.

5.5.3.6 verificarVitoria()

```
virtual bool JogoDeTabuleiro::verificarVitoria ( ) const  [pure virtual]
```

Verifica se houve uma vitória no jogo.

Este método deve verificar as condições de vitória específicas para o jogo.

Returns

true Se um jogador venceu o jogo.

false Caso contrário.

Implemented in JogoDaVelha, Lig4, and Reversi.

5.5.4 Member Data Documentation

5.5.4.1 tabuleiro

```
std::vector<std::vector<char> > JogoDeTabuleiro::tabuleiro  [protected]
```

Representação do tabuleiro como uma matriz 2D de caracteres.

Cada célula do tabuleiro é representada por um caractere.

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

- include/jogos_tabuleiro.hpp

5.6 Lig4 Class Reference

Classe que implementa o jogo Lig4 (Conecta 4).

```
#include <lig4.hpp>
```

Inheritance diagram for Lig4:

Collaboration diagram for Lig4:

Public Member Functions

- Lig4 ()
  Construtor da classe Lig4.
- void iniciarJogo () override
  Inicializa o tabuleiro do jogo.
- bool realizarJogada (int linha, int coluna) override
  Realiza uma jogada no tabuleiro.
- bool verificarVitoria () const override
  Verifica se houve vitória no jogo.
- void imprimirTabuleiro () const override
  Imprime o tabuleiro no console.
- bool jogadaValida (int linha, int coluna) const override
  Verifica se uma jogada é válida.
- bool tabuleiroCheio () const override
  Verifica se o tabuleiro está cheio.

Public Member Functions inherited from JogoDeTabuleiro

- virtual ~JogoDeTabuleiro ()

  Destrutor virtual.

Additional Inherited Members

Protected Attributes inherited from JogoDeTabuleiro

- int **linhas**

  Número de linhas do tabuleiro.
- int **colunas**

  Número de colunas do tabuleiro.
- std::vector< std::vector< char > > tabuleiro

  Representação do tabuleiro como uma matriz 2D de caracteres.

5.6.1 Detailed Description

Classe que implementa o jogo Lig4 (Conecta 4).

A classe representa a lógica e o estado do jogo Lig4, herda da classe abstrata `JogoDeTabuleiro`. O objetivo do jogo é conectar quatro peças consecutivas em linha, coluna ou diagonal.

5.6.2 Constructor & Destructor Documentation

5.6.2.1 Lig4()

```
Lig4::Lig4 ( )
```

Construtor da classe Lig4.

Inicializa o tabuleiro com 6 linhas e 7 colunas, preenchido com espaços em branco, e define o jogador inicial como 'X'.

5.6.3 Member Function Documentation

5.6.3.1 imprimirTabuleiro()

```
void Lig4::imprimirTabuleiro ( ) const  [override], [virtual]
```

Imprime o tabuleiro no console.

Exibe o estado atual do tabuleiro, substituindo células vazias por pontos para maior clareza.

Implements JogoDeTabuleiro.

5.6.3.2 iniciarJogo()

```
void Lig4::iniciarJogo ( )  [override], [virtual]
```

Inicializa o tabuleiro do jogo.

Configura o tabuleiro com células vazias antes do início da partida.

Implements JogoDeTabuleiro.

5.6.3.3 jogadaValida()

```
bool Lig4::jogadaValida (
            int linha,
            int coluna ) const  [override], [virtual]
```

Verifica se uma jogada é válida.

Uma jogada é válida se a coluna especificada está dentro do intervalo permitido e se ainda há espaço disponível na coluna.

Parameters

linha	Índice da linha (não utilizado diretamente neste jogo).
coluna	Índice da coluna a ser verificada.

Returns

true Se a jogada é válida.

false Caso contrário.

Implements JogoDeTabuleiro.

5.6.3.4 realizarJogada()

```
bool Lig4::realizarJogada (
            int linha,
            int coluna )  [override], [virtual]
```

Realiza uma jogada no tabuleiro.

Coloca a peça do jogador atual na coluna especificada, na linha disponível mais baixa. Alterna para o próximo jogador ao final de uma jogada válida.

Parameters

linha	Índice da linha (não utilizado diretamente neste jogo).
coluna	Índice da coluna onde a peça será colocada.

Returns

true Se a jogada foi realizada com sucesso.

false Se a jogada foi inválida (coluna cheia ou índice fora do intervalo).

Implements JogoDeTabuleiro.

5.6.3.5 tabuleiroCheio()

```
bool Lig4::tabuleiroCheio ( ) const  [override], [virtual]
```

Verifica se o tabuleiro está cheio.

O tabuleiro é considerado cheio se a linha superior (primeira linha) estiver completamente ocupada.

Returns

true Se o tabuleiro está cheio.

false Caso contrário.

Implements JogoDeTabuleiro.

5.6.3.6 verificarVitoria()

```
bool Lig4::verificarVitoria ( ) const  [override], [virtual]
```

Verifica se houve vitória no jogo.

Examina todas as posições do tabuleiro em busca de quatro peças consecutivas do mesmo jogador em linha, coluna ou diagonal.

Returns

true Se um jogador conseguiu conectar quatro peças consecutivas.

false Caso contrário.

Implements JogoDeTabuleiro.

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

- include/lig4.hpp
- src/lig4.cpp

5.7 ListaDeJogadores Class Reference

Classe que representa uma lista de jogadores.

```
#include <lista_de_jogadores.hpp>
```

Public Member Functions

- ListaDeJogadores ()
  Construtor da classe ListaDeJogadores.
- ~ListaDeJogadores ()
  Destruidor da classe ListaDeJogadores.
- void adicionarJogador (const Jogador &jogador)
  Adiciona um jogador à lista de jogadores.
- void removerJogador (const std::string &apelido)
  Remove um jogador da lista de jogadores pelo apelido.
- Jogador * buscarJogador (const std::string &apelido)
  Busca um jogador na lista pelo apelido.
- void salvarEmArquivo (const std::string &nomeArquivo) const
  Salva a lista de jogadores em um arquivo.
- void carregarDeArquivo (const std::string &nomeArquivo)
  Carrega os jogadores de um arquivo.
- void listarJogadores (char criterio) const
  Lista todos os jogadores, ordenados por um critério.

5.7.1 Detailed Description

Classe que representa uma lista de jogadores.

Esta classe gerencia uma lista de objetos do tipo `Jogador`. Ela permite a adição, remoção, busca, e listagem de jogadores. Além disso, também realiza a persistência dos jogadores em arquivos de texto para garantir que os dados sejam mantidos entre diferentes execuções do programa.

5.7.2 Constructor & Destructor Documentation

5.7.2.1 ListaDeJogadores()

```
ListaDeJogadores::ListaDeJogadores ( )
```

Construtor da classe ListaDeJogadores.

O construtor verifica a existência de um diretório para armazenar o arquivo de jogadores e cria o arquivo `jogadores.txt` caso ele não exista. Caso o arquivo exista, os dados são carregados dele.

5.7.2.2 ~ListaDeJogadores()

```
ListaDeJogadores::~ListaDeJogadores ( )
```

Destruidor da classe ListaDeJogadores.

O destruidor verifica se o arquivo `jogadores.txt` está vazio. Caso esteja, o arquivo é removido. Caso contrário, os dados da lista de jogadores são salvos no arquivo.

5.7.3 Member Function Documentation

5.7.3.1 adicionarJogador()

```
void ListaDeJogadores::adicionarJogador (
            const Jogador & jogador )
```

Adiciona um jogador à lista de jogadores.

Este método verifica se já existe um jogador com o mesmo apelido antes de adicionar o novo jogador. Caso o apelido já exista, um erro é exibido.

Parameters

jogador	O objeto do tipo `Jogador` a ser adicionado à lista.

5.7.3.2 buscarJogador()

```
Jogador * ListaDeJogadores::buscarJogador (
            const std::string & apelido )
```

Busca um jogador na lista pelo apelido.

Este método busca um jogador pelo seu apelido. Caso o jogador seja encontrado, um ponteiro para o objeto `Jogador` correspondente é retornado. Caso contrário, o método retorna um ponteiro nulo.

Parameters

apelido	O apelido do jogador a ser buscado.

Returns

Ponteiro para o jogador encontrado, ou `nullptr` se não encontrado.

5.7.3.3 carregarDeArquivo()

```
void ListaDeJogadores::carregarDeArquivo (
            const std::string & nomeArquivo )
```

Carrega os jogadores de um arquivo.

Este método lê os dados de um arquivo e desserializa os jogadores, populando a lista de jogadores com os dados lidos do arquivo.

Parameters

nomeArquivo	O nome do arquivo de onde os jogadores serão carregados.

5.7.3.4 listarJogadores()

```
void ListaDeJogadores::listarJogadores (
            char criterio ) const
```

Lista todos os jogadores, ordenados por um critério.

Este método permite listar os jogadores da lista, ordenados por um critério especificado. O critério pode ser o nome ou apelido do jogador.

Parameters

criterio	O critério de ordenação: 'A' para apelido e 'N' para nome.

5.7.3.5 removerJogador()

```
void ListaDeJogadores::removerJogador (
            const std::string & apelido )
```

Remove um jogador da lista de jogadores pelo apelido.

Este método busca o jogador pelo seu apelido. Caso o jogador seja encontrado, ele é removido da lista. Caso contrário, um erro é exibido.

Parameters

apelido	O apelido do jogador a ser removido.

5.7.3.6 salvarEmArquivo()

```
void ListaDeJogadores::salvarEmArquivo (
            const std::string & nomeArquivo ) const
```

Salva a lista de jogadores em um arquivo.

Este método grava todos os jogadores da lista no arquivo especificado. Cada jogador é serializado como uma linha no arquivo.

Parameters

nomeArquivo	O nome do arquivo onde os jogadores serão salvos.

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

- include/lista_de_jogadores.hpp
- src/lista_de_jogadores.cpp

5.8 Reversi Class Reference

Classe que representa o jogo Reversi (também conhecido como Othello).

```
#include <reversi.hpp>
```

Inheritance diagram for Reversi:

Collaboration diagram for Reversi:

Public Member Functions

- Reversi ()

  Construtor da classe Reversi.
- void iniciarJogo () override

  Inicia o jogo configurando as peças iniciais no tabuleiro.
- bool realizarJogada (int linha, int coluna) override

  Realiza uma jogada no tabuleiro.
- bool verificarVitoria () const override

  Verifica se há um vencedor no jogo.
- void imprimirTabuleiro () const override

  Imprime o tabuleiro na tela.
- bool jogadaValida (int linha, int coluna) const override

  Verifica se uma jogada é válida.
- bool tabuleiroCheio () const override

  Verifica se o tabuleiro está cheio.
- int contarPecas (char jogador) const

  Conta o número de peças de um jogador no tabuleiro.

Public Member Functions inherited from JogoDeTabuleiro

- virtual ~JogoDeTabuleiro ()

  Destrutor virtual.

Additional Inherited Members

Protected Attributes inherited from JogoDeTabuleiro

- int **linhas**

  Número de linhas do tabuleiro.
- int **colunas**

  Número de colunas do tabuleiro.
- std::vector< std::vector< char > > tabuleiro

  Representação do tabuleiro como uma matriz 2D de caracteres.

5.8.1 Detailed Description

Classe que representa o jogo Reversi (também conhecido como Othello).

Esta classe é uma implementação do jogo de tabuleiro Reversi, que é jogado em um tabuleiro 8x8. O objetivo do jogo é capturar as peças do adversário ao cercá-las com suas próprias peças. A classe gerencia as jogadas, verificações de vitória, e captura de peças durante o jogo.

Note

Esta classe herda de `JogoDeTabuleiro`, que define métodos genéricos para jogos de tabuleiro.

5.8.2 Constructor & Destructor Documentation

5.8.2.1 Reversi()

```
Reversi::Reversi ( )
```

Construtor da classe Reversi.

Inicializa um tabuleiro 8x8 e define o jogador atual como 'X'. As peças iniciais são posicionadas nas posições (3, 3), (3, 4), (4, 3) e (4, 4).

5.8.3 Member Function Documentation

5.8.3.1 contarPecas()

```
int Reversi::contarPecas (
            char jogador ) const
```

Conta o número de peças de um jogador no tabuleiro.

Este método percorre o tabuleiro e conta quantas peças do jogador especificado estão presentes.

Parameters

jogador	O caractere que representa o jogador ('X' ou 'O').

Returns

O número de peças do jogador no tabuleiro.

5.8.3.2 imprimirTabuleiro()

```
void Reversi::imprimirTabuleiro ( ) const  [override], [virtual]
```

Imprime o tabuleiro na tela.

Este método exibe o estado atual do tabuleiro, substituindo espaços vazios por pontos ('.') para facilitar a visualização.

Implements JogoDeTabuleiro.

5.8.3.3 iniciarJogo()

```
void Reversi::iniciarJogo ( )  [override], [virtual]
```

Inicia o jogo configurando as peças iniciais no tabuleiro.

Este método coloca as 4 peças iniciais no centro do tabuleiro.

Implements JogoDeTabuleiro.

5.8.3.4 jogadaValida()

```
bool Reversi::jogadaValida (
            int linha,
            int coluna ) const  [override], [virtual]
```

Verifica se uma jogada é válida.

Este método verifica se a posição informada é válida para a realização de uma jogada. Para que a jogada seja válida, a célula precisa estar vazia e deve ser possível capturar pelo menos uma peça adversária.

Parameters

linha	A linha onde a jogada será realizada.
coluna	A coluna onde a jogada será realizada.

Returns

`true` se a jogada for válida, `false` caso contrário.

Implements JogoDeTabuleiro.

5.8.3.5 realizarJogada()

```
bool Reversi::realizarJogada (
            int linha,
            int coluna )  [override], [virtual]
```

Realiza uma jogada no tabuleiro.

Este método verifica se a jogada é válida e, se for, coloca a peça do jogador na posição especificada e captura as peças do adversário. Após a jogada, o jogador atual é alternado.

Parameters

linha	A linha onde a jogada será feita.
coluna	A coluna onde a jogada será feita.

Returns

`true` se a jogada for realizada com sucesso, `false` caso contrário.

Implements JogoDeTabuleiro.

5.8.3.6 tabuleiroCheio()

```
bool Reversi::tabuleiroCheio ( ) const  [override], [virtual]
```

Verifica se o tabuleiro está cheio.

Este método verifica se o tabuleiro está completamente preenchido com peças.

Returns

`true` se o tabuleiro estiver cheio, `false` caso contrário.

Implements JogoDeTabuleiro.

5.8.3.7 verificarVitoria()

```
bool Reversi::verificarVitoria ( ) const  [override], [virtual]
```

Verifica se há um vencedor no jogo.

Este método verifica se o tabuleiro está cheio ou se um dos jogadores não tem mais peças no tabuleiro, indicando que o jogo terminou.

Returns

`true` se houver um vencedor, `false` caso contrário.

Implements JogoDeTabuleiro.

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

- include/reversi.hpp
- src/reversi.cpp

Chapter 6

File Documentation

6.1 estatisticas.hpp

```
00001 #ifndef ESTATISTICAS_HPP
00002 #define ESTATISTICAS_HPP
00003
00004 #include <map>
00005 #include <string>
00006 #include <iostream>
00007 #include <sstream>
00008
00013 class Estatisticas
00014 {
00015 private:
00022     std::map<char, int> vitorias;
00023
00030     std::map<char, int> derrotas;
00031
00032 public:
00038     Estatisticas();
00039
00045     void incrementarVitoria(char jogo);
00046
00052     void incrementarDerrota(char jogo);
00053
00060     int getVitorias(char jogo) const;
00061
00068     int getDerrotas(char jogo) const;
00069
00075     void imprimirEstatisticas() const;
00076
00084     std::string serializar() const;
00085
00091     void desserializar(const std::string &dados);
00092 };
00093
00094 #endif
```

6.2 gerenciador_de_jogos.hpp

```
00001 #ifndef GERENCIADOR_DE_JOGOS_HPP
00002 #define GERENCIADOR_DE_JOGOS_HPP
00003
00004 #include <memory>
00005 #include "jogo_da_velha.hpp"
00006 #include "lig4.hpp"
00007 #include "reversi.hpp"
00008 #include "jogador.hpp"
00009
00019 class GerenciadorDeJogos
00020 {
00021 public:
00039     static void executarPartida(char tipoJogo, Jogador &jogador1, Jogador &jogador2);
00040 };
00041
00042 #endif
```

6.3 jogador.hpp

```
00001 #ifndef JOGADOR_HPP
00002 #define JOGADOR_HPP
00003
00004 #include <string>
00005 #include "estatisticas.hpp"
00006
00011 class Jogador
00012 {
00013 private:
00019     std::string apelido;
00020
00026     std::string nome;
00027
00033     Estatisticas estatisticas;
00034
00035 public:
00044     Jogador(const std::string &apelido, const std::string &nome);
00045
00051     const std::string &getApelido() const;
00052
00058     const std::string &getNome() const;
00059
00065     void incrementarVitoria(char jogo);
00066
00072     void incrementarDerrota(char jogo);
00073
00079     void imprimirEstatisticas() const;
00080
00088     std::string serializar() const;
00089
00098     static Jogador desserializar(const std::string &linha);
00099 };
00100
00101 #endif
```

6.4 jogo_da_velha.hpp

```
00001 #ifndef JOGO_DA_VELHA_HPP
00002 #define JOGO_DA_VELHA_HPP
00003
00004 #include "jogos_tabuleiro.hpp"
00005
00013 class JogoDaVelha : public JogoDeTabuleiro
00014 {
00015 public:
00021     JogoDaVelha();
00022
00028     void iniciarJogo() override;
00029
00041     bool realizarJogada(int linha, int coluna) override;
00042
00051     bool verificarVitoria() const override;
00052
00058     void imprimirTabuleiro() const override;
00059
00071     bool jogadaValida(int linha, int coluna) const override;
00072
00081     bool tabuleiroCheio() const override;
00082
00083 private:
00089     char jogadorAtual;
00090
00096     void alternarJogador();
00097 };
00098
00099 #endif
```

6.5 jogos_tabuleiro.hpp

```
00001 #ifndef JOGOS_TABULEIRO_HPP
00002 #define JOGOS_TABULEIRO_HPP
00003
00004 #include <iostream>
00005 #include <vector>
00006 #include <string>
00007
00015 class JogoDeTabuleiro
```

```
00016 {
00017 public:
00023     virtual ~JogoDeTabuleiro() {}
00024
00031     virtual void iniciarJogo() = 0;
00032
00044     virtual bool realizarJogada(int linha, int coluna) = 0;
00045
00054     virtual bool verificarVitoria() const = 0;
00055
00061     virtual void imprimirTabuleiro() const = 0;
00062
00074     virtual bool jogadaValida(int linha, int coluna) const = 0;
00075
00084     virtual bool tabuleiroCheio() const = 0;
00085
00086 protected:
00090     int linhas;
00091
00095     int colunas;
00096
00102     std::vector<std::vector<char> tabuleiro;
00103 };
00104
00105 #endif
```

6.6 lig4.hpp

```
00001 #ifndef LIG4_HPP
00002 #define LIG4_HPP
00003
00004 #include "jogos_tabuleiro.hpp"
00005
00013 class Lig4 : public JogoDeTabuleiro
00014 {
00015 public:
00022     Lig4();
00023
00029     void iniciarJogo() override;
00030
00042     bool realizarJogada(int linha, int coluna) override;
00043
00053     bool verificarVitoria() const override;
00054
00060     void imprimirTabuleiro() const override;
00061
00073     bool jogadaValida(int linha, int coluna) const override;
00074
00083     bool tabuleiroCheio() const override;
00084
00085 private:
00089     char jogadorAtual;
00090
00096     void alternarJogador();
00097
00111     bool verificarSequencia(int linha, int coluna, int deltaLinha, int deltaColuna) const;
00112 };
00113
00114 #endif
```

6.7 lista_de_jogadores.hpp

```
00001 #ifndef LISTADEJOGADORES_HPP
00002 #define LISTADEJOGADORES_HPP
00003
00004 #include <vector>
00005 #include <string>
00006 #include "jogador.hpp"
00007
00016 class ListaDeJogadores
00017 {
00018 private:
00026     std::vector<Jogador> jogadores;
00027
00028 public:
00036     ListaDeJogadores();
00037
00045     ~ListaDeJogadores();
00046
```

```
00055     void adicionarJogador(const Jogador &jogador);
00056
00065     void removerJogador(const std::string &apelido);
00066
00078     Jogador *buscarJogador(const std::string &apelido);
00079
00088     void salvarEmArquivo(const std::string &nomeArquivo) const;
00089
00098     void carregarDeArquivo(const std::string &nomeArquivo);
00099
00108     void listarJogadores(char criterio) const;
00109 };
00110
00111 #endif
```

6.8 reversi.hpp

```
00001 #ifndef REVERSI_HPP
00002 #define REVERSI_HPP
00003
00004 #include "jogos_tabuleiro.hpp"
00005
00017 class Reversi : public JogoDeTabuleiro
00018 {
00019 public:
00026     Reversi();
00027
00033     void iniciarJogo() override;
00034
00047     bool realizarJogada(int linha, int coluna) override;
00048
00057     bool verificarVitoria() const override;
00058
00065     void imprimirTabuleiro() const override;
00066
00079     bool jogadaValida(int linha, int coluna) const override;
00080
00088     bool tabuleiroCheio() const override;
00089
00100     int contarPecas(char jogador) const;
00101
00102 private:
00109     char jogadorAtual;
00110
00116     void alternarJogador();
00117
00130     bool capturaPecas(int linha, int coluna);
00131
00146     bool capturaDirecao(int linha, int coluna, int deltaLinha, int deltaColuna) const;
00147 };
00148
00149 #endif
```

Index

~JogoDeTabuleiro
    JogoDeTabuleiro, 17
~ListaDeJogadores
    ListaDeJogadores, 23

adicionarJogador
    ListaDeJogadores, 23

buscarJogador
    ListaDeJogadores, 24

carregarDeArquivo
    ListaDeJogadores, 24
contarPecas
    Reversi, 27

desserializar
    Estatisticas, 10
    Jogador, 14

Estatisticas, 9
    desserializar, 10
    Estatisticas, 10
    getDerrotas, 10
    getVitorias, 10
    imprimirEstatisticas, 11
    incrementarDerrota, 11
    incrementarVitoria, 11
    serializar, 11
executarPartida
    GerenciadorDeJogos, 12

GerenciadorDeJogos, 12
    executarPartida, 12
getApelido
    Jogador, 14
getDerrotas
    Estatisticas, 10
getNome
    Jogador, 14
getVitorias
    Estatisticas, 10

imprimirEstatisticas
    Estatisticas, 11
    Jogador, 15
imprimirTabuleiro
    JogoDeTabuleiro, 17
    Lig4, 20
    Reversi, 27
include/estatisticas.hpp, 31
include/gerenciador_de_jogos.hpp, 31
include/jogador.hpp, 32
include/jogo_da_velha.hpp, 32
include/jogos_tabuleiro.hpp, 32
include/lig4.hpp, 33
include/lista_de_jogadores.hpp, 33
include/reversi.hpp, 34
incrementarDerrota
    Estatisticas, 11
    Jogador, 15
incrementarVitoria
    Estatisticas, 11
    Jogador, 15
iniciarJogo
    JogoDeTabuleiro, 17
    Lig4, 20
    Reversi, 27

jogadaValida
    JogoDeTabuleiro, 17
    Lig4, 21
    Reversi, 27
Jogador, 13
    desserializar, 14
    getApelido, 14
    getNome, 14
    imprimirEstatisticas, 15
    incrementarDerrota, 15
    incrementarVitoria, 15
    Jogador, 14
    serializar, 15
JogoDaVelha, 16
JogoDeTabuleiro, 16
    ~JogoDeTabuleiro, 17
    imprimirTabuleiro, 17
    iniciarJogo, 17
    jogadaValida, 17
    realizarJogada, 18
    tabuleiro, 19
    tabuleiroCheio, 18
    verificarVitoria, 18

Lig4, 19
    imprimirTabuleiro, 20
    iniciarJogo, 20
    jogadaValida, 21
    Lig4, 20
    realizarJogada, 21
    tabuleiroCheio, 22
    verificarVitoria, 22

ListaDeJogadores, 22
~ListaDeJogadores, 23
adicionarJogador, 23
buscarJogador, 24
carregarDeArquivo, 24
ListaDeJogadores, 23
listarJogadores, 24
removerJogador, 25
salvarEmArquivo, 25
listarJogadores
ListaDeJogadores, 24

README, 1
realizarJogada
JogoDeTabuleiro, 18
Lig4, 21
Reversi, 28
removerJogador
ListaDeJogadores, 25
Reversi, 25
contarPecas, 27
imprimirTabuleiro, 27
iniciarJogo, 27
jogadaValida, 27
realizarJogada, 28
Reversi, 27
tabuleiroCheio, 28
verificarVitoria, 29

salvarEmArquivo
ListaDeJogadores, 25
serializar
Estatisticas, 11
Jogador, 15

tabuleiro
JogoDeTabuleiro, 19
tabuleiroCheio
JogoDeTabuleiro, 18
Lig4, 22
Reversi, 28

verificarVitoria
JogoDeTabuleiro, 18
Lig4, 22
Reversi, 29