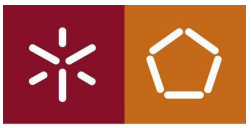
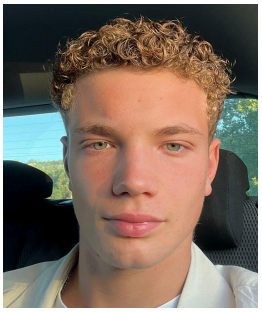
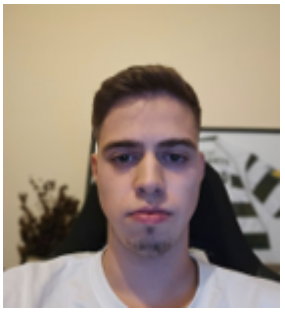


University of Minho
School of Engineering

Programação Com Objetos – LEGSI, 2º Ano, 2024/25

PL3 - Grupo 5

Gestão de uma Liga de e-Sports

	
Afonso Martim Carvalho Leite a108552 (a108552@alunos.uminho.pt)	Sérgio Paulo Vieira Carvalho a108274 (a108274@alunos.uminho.pt)

Índice

Funcionalidades implementadas

Classe Jogador

Funcionalidade	Descrição
Jogador (String NomeCompleto, String nickname, String passe)	Construtor da classe, inicializa um jogador com o nome completo, o nickname e a palavra-passe. Configura contadores e listas de torneios a participar pelo Jogador.
Inserção dos getters e setters das respetivas variáveis	Métodos para obter e modificar valores das variáveis como, Id, NomeCompleto, Nickname, passe, PartidasJogadas, Vitórias, Derrotas, TorneiosAParticipar, DadosPessoais. Os setters incluem validações que lançam exceções para casos em que os dados inválidos.
autenticar(String nickname, String senha)	Utilizado para verificar se um par de credenciais (um `nickname` e uma `senha`) fornecidos como parâmetros correspondem aos armazenados no objeto.
adicionarTorneio (Torneio torneio)	Adiciona um torneio à lista de torneios do jogador, caso ainda não esteja inscrito, e guarda a informação.
registarPartida(boolean venceu)	Regista uma partida jogada, incrementando o total de partidas e ajusta o número de vitórias ou derrotas. Guarda a informação.
editarDadosPessoais(String novoNome, String novoNickname)	Edita o nome completo e/ou o nickname do jogador, se válidos, guarda as informações e indica se houve alterações.
exibirDadosPessoais()	Retorna uma string com os dados pessoais do jogador (ID, nomeCompleto, nickname).
exibirEstatisticas()	Retorna uma string com as estatísticas do jogador, incluindo total de partidas, vitórias e derrotas.
exibirTorneios()	Exibe os torneios em que o jogador participa ou informa que não há torneios registrados.

toString()	Método sobrescrito para exibir uma visão simplificada do jogador contento o ID, o nome e o nickname do mesmo.

Classe Jogador Efootball

Funcionalidade	Descrição
JogadorEfootball (String nomeCompleto, String nickname, String senha, String posicao, int golosMarcados, int golosDefendidos, int assistencias)	Constrói um jogador de eFootball com nome, nickname, senha, posição, golos marcados, golos defendidos e assistências, validando os parâmetros.
Getters e Setters das respetivas variáveis	Obter e modificar os valores das variáveis como posicao, golosMarcados, golosDefendidos, e assistencias. Os setters incluem validações para garantir que os valores sejam coerentes.
atualizarEstatisticas(int novosGolosMarcados, int novosGolosDefendidos, int novasAssistencias)	Atualizar as estatísticas do jogador (golos marcados, defendidos e assistências).
adicionarTorneio(Torneio torneio)	Adicionar um torneio ao jogador, utilizando a lógica da classe base
exibirEstatisticas()	Retornar uma string com as estatísticas completas do jogador.
toString()	Método sobrescrito para exibir uma visão detalhada do jogador, incluindo ID, nome, nickname, posição e o desempenho.

Classe Jogador FPS

Funcionalidade	Descrição
ogadorFPS(String nomeCompleto, String nickname, String senha, double precisao, int headshots)	Constrói um jogador de FPS com nome, nickname, senha, precisão, headshots e inicializa os tiros a 0, validando os parâmetros.
Getters e Setters das respetivas variáveis	Métodos para obter e modificar os valores das variáveis como TotalTiros, TirosAcertados, Precisão e Headshots. Os setters incluem validações para garantir que os valores sejam coerentes.
adicionarTorneio(Torneio torneio)	Adicionar um torneio ao jogador, utilizando a lógica da classe base.
atualizarPrecisao()	Atualiza a precisão do jogador com base na proporção de tiros acertados em relação ao total de tiros.
atualizarEstatisticasFPS(double tirosAcertados, double tirosTotais, int headshots)	Atualiza a precisão e os headshots do jogador de FPS, considerando a média ponderada com partidas jogadas, e salva o jogador.
exibirEstatisticas()	Sobrescreve o método para exibir as estatísticas do jogador, incluindo tiros, acertos, precisão e headshots.
toString()	Retornar uma representação textual do jogador com ID, nome, nickname, tiros, acertos, precisão e headshots..

Classe Jogador MOBA

Funcionalidade	Descrição
JogadorMOBA(String nomeCompleto, String nickname, String senha, String personagem, int kills, int deaths, int assists)	Constrói um jogador de MOBA com nome, nickname, senha, personagem principal, kills, deaths e assists, validando os parâmetros.
Getters e Setters das variáveis	Métodos para obter e modificar valores como personagemPrincipal, kills, deaths e assists. Os setters incluem validações para evitar valores inválidos, como negativos.
getKDA()	Calcula e retorna o KDA (kills + assists dividido por deaths), considerando deaths 0 como caso especial.
atualizarEstatisticasMOBA(int novasKills, int novasDeaths, int novasAssists)	Atualiza as estatísticas de kills, assists e deaths do jogador de MOBA e guarda o jogador.
adicionarTorneio(Torneio torneio)	Sobrescreve o método para adicionar um torneio, reutilizando a lógica da classe base sem alterações adicionais.
exibirEstatisticas()	Exibe as estatísticas do jogador com informações do personagem principal, kills, deaths, assists e KDA, adicionando à exibição da classe base.
toString()	Retornar uma representação textual do jogador com ID, nome, nickname, personagem principal, kills, deaths, assists e KDA.

Classe Equipa

Funcionalidade	Descrição
Equipa(String nome, Treinador treinador)	Constrói uma equipa com nome, treinador, listas vazias de jogadores e torneios, e inicializa o ID, vitórias e derrotas.
Getters e Setters das variáveis	Métodos para obter e modificar atributos como nome, treinador, vitórias, e derrotas, listas de jogadores e de torneios. Os setters incluem validações para evitar valores inválidos.
adicionarJogador(Jogador jogador)	Adiciona um jogador à equipa, sincroniza os torneios da equipa com o jogador e salva a equipa, retornando se a adição foi bem-sucedida.
removerJogador(Jogador jogador)	Remove um jogador da equipa, salva a equipa após a remoção e retorna se a operação foi bem-sucedida.
inscreverEmTorneio(Torneio torneio)	Inscreve a equipa em um torneio, sincroniza o torneio com os jogadores e salva a equipa, retornando se a inscrição foi bem-sucedida.
exibirTorneios()	Exibe os torneios em que a equipa está inscrita ou informa que não há nenhum torneio associado.
registarVitoria()	Regista uma vitória para a equipa, incrementando o total e exibindo a atualização.
registarDerrota()	Regista uma derrota para a equipa, incrementando o total e exibindo a atualização.
exibirEstatisticas()	Exibe as estatísticas da equipa, incluindo vitórias, derrotas e a lista de jogadores.
exibirJogadores()	Exibe a lista de jogadores da equipa ou informa que não há jogadores na equipa.

guardarEquipa()	Guarda a equipa no sistema, atualizando ou adicionando à lista existente de equipas, e exibe uma mensagem de sucesso ou erro.
toString()	Retornar uma descrição textual da equipa, incluindo nome, jogadores, torneios, treinador, vitórias e derrotas.

Classe Gestor de Ficheiros

Funcionalidade	Descrição
guardarEmFicheiro(Object objeto, String caminhoFicheiro)	Guarda um objeto num ficheiro no caminho especificado, sobrescrevendo os dados, e exibe uma mensagem de erro em caso de falha.
lerListaDeFicheiro(String caminhoFicheiro, Class<T> tipoClasse)	Lê uma lista de objetos de um ficheiro, filtra pelos objetos compatíveis com o tipo especificado e retorna a lista, exibindo mensagens de erro ou aviso, conforme necessário.
guardarJogadores(List<Jogador> jogadores)	Guarda a lista de jogadores no ficheiro jogadores.dat.
lerJogadores()	Lê a lista de jogadores do ficheiro jogadores.dat.
guardarTreinadores(List<Treinador> treinadores)	Guarda a lista de treinadores no ficheiro treinadores.dat.
lerTreinadores()	Lê a lista de treinadores do ficheiro treinadores.dat e retorna os objetos do tipo Treinador.
guardarAdministradores(List<Administrador> administradores)	Guarda a lista de administradores no ficheiro administradores.dat.
lerAdministradores()	Lê a lista de administradores do ficheiro administradores.dat e retorna os objetos do tipo Administrador.
guardarTorneios(List<Torneio> torneios)	Guarda a lista de torneios no ficheiro torneios.dat.
lerTorneios()	Lê a lista de torneios do ficheiro torneios.dat e retorna os objetos do tipo Torneio.
guardarPartidas(List<Partida> jogos)	Guarda a lista de partidas no ficheiro “partidas.dat”
lerPartidas()	Lê a lista de partidas do ficheiro partidas.dat e retorna os objetos do tipo Partida.

guardarEquipas(List<Equipa> equipas)	Guarda a lista de equipas em equipas.dat.
lerEquipas()	Lê equipas do ficheiro equipas.dat.
visualizarDadosDeFicheiro(String caminhoFicheiro, Class<?> tipoClasse)	Lê e exibe os dados de um ficheiro filtrados pelo tipo especificado, ou informa se não há dados.

Classe Partida

Funcionalidade	Descrição
Partida(Equipa equipaA, Equipa equipaB, int pontosEquipaA, int pontosEquipaB, LocalDate data)	Constrói uma partida entre duas equipas diferentes com os pontos de cada equipa, data e torneio, validando os parâmetros e atribuindo um ID único.
Getters e Setters das variáveis	Métodos para obter e modificar os atributos como equipaA, equipaB, pontosEquipaA, pontosEquipaB, Data, Vencedor. Os setters incluem validações de coerência.
isEmpate()	Verifica se a partida terminou empatada, comparando os pontos das duas equipas.
obterResultado()	Retorna uma descrição do resultado da partida, indicando a equipa vencedora ou se houve empate, com os pontos de ambas as equipas.
exibirDetalhes()	Exibe os detalhes da partida, incluindo data, equipas, pontos e o resultado.
salvarPartida()	Salva a partida no torneio associado, atualizando ou adicionando o torneio na lista existente, e exibe uma mensagem de sucesso ou erro.
toString()	Sobrescreve o método toString para retornar uma representação textual da partida com equipas, resultado, data e o vencedor.

Classe TipoTreinador

TipoTreinador(String descricao)	O construtor TipoTreinador(String descricao) inicializa a descrição associada a cada constante do enum TipoTreinador.
TipoTreinador fromString(String valor)	Converte uma string para o correspondente TipoTreinador, ignorando maiúsculas/minúsculas, ou lança uma exceção se o valor não corresponder a nenhum tipo válido.
toString()	Sobrescreve o método toString para retornar a descrição associada ao tipo de treinador.

Classe Torneio

Funcionalidade	Descrição
Torneio(String nome, String jogo)	Constrói um torneio com nome, jogo, inicializando listas de equipas e partidas, a classificação e atribuindo um ID único, validando os parâmetros.
Getters das variáveis	Métodos para obter valores como id, designacao, tipoJogo, listaEquipas, listaPartida e tabelaClassificacao.
adicionarEquipa(Equipa equipa)	Adiciona uma equipa ao torneio, inicializa a sua pontuação na classificação, salva o torneio e evita duplicatas.
adicionarPartida(Partida partida)	Adiciona uma partida ao torneio, valida se as equipas estão inscritas, atualiza a classificação, salva o torneio e exibe uma mensagem de confirmação.
atualizarClassificacao(Partida partida)	Atualiza a classificação do torneio atribuindo pontos: 3 para a equipa vencedora ou 1 para cada equipa em caso de empate, e salva o torneio.
exibirClassificacao()	Exibe a classificação atual do torneio, ordenando as equipas por pontos de forma decrescente.
exibirPartidas()	Exibe todas as partidas realizadas no torneio, listando o ID e o resultado de cada partida, ou informa se nenhuma foi realizada.
getEstatisticas()	Gera um relatório com as estatísticas do torneio, incluindo nome, tipo de jogo, classificação ordenada por pontos e detalhes das partidas realizadas.
salvarTorneio()	Salva o torneio atual, atualizando ou adicionando-o à lista de torneios existentes, e exibe uma mensagem de sucesso ou erro.

toString()	Sobrescreve o método toString para retornar uma descrição textual do torneio, incluindo nome, tipo de jogo, número de equipas e número de partidas.

Classe Treinador

Funcionalidade	Descrição
Treinador(String nomeCompleto, String email, String senha, TipoTreinador tipo)	Constrói um treinador com nome completo, email, senha e tipo, validando os parâmetros e atribuindo um ID único.
Getters e Setters das variáveis	Métodos para obter e modificar valores como, id nome, email, tipo, equipa e pass. Os setters incluem validações para assegurar que os dados sejam coerentes e seguros.
adicionarJogadorEquipa(Jogador jogador)	Adiciona um jogador à equipa do treinador, validando se a equipa e o jogador são válidos, e vincula os torneios da equipa ao jogador, persistindo a alteração.
removerJogadorEquipa(Jogador jogador)	Remove um jogador da equipa do treinador, validando a existência da equipa e do jogador, e exibe uma mensagem de sucesso ou erro conforme o caso.
editarNomeEquipa(String novoNome)	Altera o nome da equipa associada ao treinador, validando se o novo nome é válido e não vazio.
inscreverEquipa(Torneio torneio)	Inscreve a equipa do treinador em um torneio, validando se a equipa existe e não está já inscrita no torneio.
acompanharResultadosTorneio(Torneio torneio)	Exibe os resultados das partidas da equipa do treinador em um torneio específico, ou informa se ainda não houve partidas envolvendo a equipa.
toString()	Sobrescreve o método toString para retornar uma descrição textual do treinador, incluindo ID, nome completo e email.

Classe Administrador

Funcionalidade	Descrição
Administrador(String nomeCompleto, String email, String senha)	Constrói um administrador com nome completo, email e senha, validando os parâmetros e atribuindo um ID único.
Getters e Setters das variáveis	Métodos para obter e modificar atributos como nomeCompleto, e-mail, passe, lista de torneios. Os setters incluem validações para garantir dados válidos e coerentes.
autenticar(String email, String senha)	Autentica o administrador verificando se o email e a senha fornecidos coincidem com os armazenados.
criarTorneio(String nome, String jogo)	Cria um novo torneio com nome e tipo de jogo válidos (FPS, MOBA ou eFootball), adiciona à lista de torneios e retorna o torneio criado.
removerTorneio(Torneio torneio)	Remove um torneio da lista de torneios, validando se o torneio é válido e existe na lista, e exibe uma mensagem de sucesso.
adicionarJogador(List<Jogador> jogadores, Jogador jogador)	Adiciona um jogador a uma lista de jogadores, validando se o jogador é válido e se ainda não está na lista, exibindo uma mensagem de sucesso.
removerJogador(List<Jogador> jogadores, Jogador jogador)	Remove um jogador de uma lista de jogadores, validando se o jogador existe na lista, e exibe uma mensagem de sucesso.
adicionarTreinador(List<Treinador> treinadores, Treinador treinador)	Adiciona um treinador a uma lista de treinadores, validando se o treinador não é nulo e ainda não está na lista, exibindo uma mensagem de sucesso.
agendarPartida(Torneio torneio, Equipa equipaA, Equipa equipaB, int pontosEquipaA, int pontosEquipaB)	Valida parâmetros de um torneio e duas equipas, garantindo que as equipas não sejam iguais, e agenda uma partida entre elas no torneio, exibindo uma mensagem de confirmação.

registarResultado(Torneio torneio, Partida partida, int pontosEquipaA, int pontosEquipaB)	Regista o resultado de uma partida, configurando os pontos das equipas, atualiza as estatísticas das equipas e exibe uma mensagem de sucesso.
atualizarEstatisticasAleatorias(Equipa equipa)	Atualiza as estatísticas de cada jogador de uma equipa de forma aleatória, com base no tipo de jogador (FPS, MOBA ou eFootball).
acompanharEstatisticasTorneio(Torneio torneio)	Exibe as estatísticas de um torneio, incluindo o nome do torneio, tipo de jogo, número de equipas, partidas realizadas e a classificação atual.
toString()	Sobrescreve o método toString para retornar uma descrição textual do administrador, incluindo nome completo e email.

Classe Autenticação

Funcionalidade	Descrição
Autenticacao(List<JogadorFPS> jogadoresFps, List<JogadorMOBA> jogadoresMoba, List<JogadorEfootball> jogadoresEfootball, List<Treinador> treinadores, List<Administrador> administradores, Consola consola)	Constrói uma instância de Autenticacao, validando que nenhuma das listas de usuários nem a consola sejam nulas.
Getters e Setters	Métodos para obter e modificar lista de administradores, treinadores e jogadores.
autenticarJogadorFps(String nickname, String senha)	Valida as credenciais e autentica um jogador de FPS, retornando o jogador se as credenciais corresponderem. Caso contrário, retorna null.
autenticarJogadorMoba(String nickname, String senha)	Valida as credenciais e autentica um jogador de MOBA, retornando o jogador se as credenciais corresponderem. Caso contrário, retorna null.
autenticarJogadorEfootball(String nickname, String senha)	Valida as credenciais e autentica um jogador de eFootball, retornando o jogador se as credenciais corresponderem. Caso contrário, retorna null.
autenticarTreinador(String email, String senha)	Valida as credenciais e autentica um treinador, retornando o treinador se as credenciais corresponderem. Caso contrário, retorna null.
autenticarAdministrador(String email, String senha)	Valida as credenciais e autentica um administrador, retornando o administrador se as credenciais corresponderem. Caso contrário, retorna null.
criarConta(String tipo, String nome, String emailNick, String senha)	Cria uma nova conta dependendo do tipo (Jogador FPS, Jogador MOBA, Jogador Efootball, Treinador ou Administrador), validando os parâmetros e exibindo uma mensagem de sucesso ou erro.

adicionarJogador(Jogador jogador)	Adicionar um jogador à lista de jogadores.
adicionarTreinador(Treinador treinador)	Adicionar um treinador à lista de treinadores.
adicionarAdministrador (Administrador administrador)	Adicionar um administrador à lista de administradores.
validarCredenciais(String identificador, String senha)	Valida se o identificador (e-mail ou nickname) e a senha não estão vazios, lançando exceções em caso de dados inválidos.

Classe Consola

Funcionalidade	Descrição
Consola()	Construtor da classe Consola, inicializando o objeto teclado com uma nova instância de Scanner.
escreverTexto(String texto)	Exibe o texto fornecido na tela utilizando System.out.println.
escreverErro(String texto)	Exibe uma mensagem de erro na consola utilizando System.err.println, prefixada com "[ERRO]".
lerString(String mensagem)	Exibe uma mensagem, lê a entrada do utilizador e garante que não seja vazia, solicitando uma nova tentativa em caso de entrada inválida.
lerInteiro(String mensagem)	Exibe uma mensagem, lê a entrada do utilizador e tenta convertê-la para um número inteiro, pedindo uma nova tentativa caso o valor seja inválido.
lerDouble(String mensagem)	Exibe uma mensagem, lê a entrada do utilizador e tenta convertê-la para um número decimal, pedindo uma nova tentativa caso o valor seja inválido.
lerOpcaoMenu(String[] opcoes)	Exibe as opções de um menu, lê a escolha do usuário, valida a opção e pede nova entrada se a opção for inválida.
fecharScanner()	Fecha o objeto Scanner utilizado para a leitura de entradas do usuário.

Classe Principal

Funcionalidade	Descrição
Principal()	Construtor da classe `Principal`, inicializa as listas de equipas, administradores, treinadores, jogadores e torneios, e cria uma instância de `Autenticacao`.
iniciar()	Exibe o menu inicial, lê a escolha do usuário e executa a opção escolhida (login, criar conta, visualizar ficheiro ou sair).
criarConta()	Lê os dados do utilizador, cria uma nova conta de acordo com o tipo (Jogador FPS, Jogador MOBA, Jogador Efootball, Treinador ou Administrador) e a adiciona à lista correspondente, atualizando o ficheiro de dados.
fazerLogin()	Solicita ao utilizador o tipo de conta, identificador e senha, e processa o login com base no tipo de conta selecionado, chamando o método específico para cada tipo de conta.
processarLoginJogadorFPS(String nickname, String senha)	Procura um jogador FPS pelo nickname, verifica a senha e, se a autenticação for bem-sucedida, exibe o menu do jogador FPS. Se o jogador não for encontrado, exibe uma mensagem de erro.
processarLoginJogadorMOBA(String nickname, String senha)	Procura um jogador MOBA pelo nickname, verifica a senha e, se a autenticação for bem-sucedida, exibe o menu do jogador MOBA. Se o jogador não for encontrado, exibe uma mensagem de erro.
processarLoginJogadorEfootball(String nickname, String senha)	Procura um jogador de eFootball pelo nickname, verifica a senha e, se a autenticação for bem-sucedida, exibe o menu do jogador eFootball. Se o jogador não for encontrado, exibe uma mensagem de erro.

processarLoginTreinador(String email, String senha)	Procura um treinador pelo email, verifica a senha e, se a autenticação for bem-sucedida, exibe o menu do treinador. Se o treinador não for encontrado, exibe uma mensagem de erro.
processarLoginAdministrador(String email, String senha)	Procura um administrador pelo email, verifica a senha e, se a autenticação for bem-sucedida, exibe uma mensagem de boas-vindas e o menu do administrador. Se a senha estiver incorreta ou o administrador não for encontrado, exibe mensagens de erro.
verificarSenha(Object usuario, String senha)	Verifica a senha do usuário (Jogador ou Treinador). Se for o primeiro login, solicita a configuração de uma nova senha. Caso a senha seja incorreta, exibe uma mensagem de erro. Se a senha for correta, exibe uma mensagem de boas-vindas.
alterarSenha(Object usuario)	Solicita ao utilizador (Jogador ou Treinador) uma nova senha, garantindo que não seja vazia, e altera a senha do usuário. Exibe uma mensagem de sucesso após a alteração.
menuJogadorFPS(JogadorFPS jogador)	Apresenta as opções disponíveis para utilizadores do tipo jogador FPS.
menuJogadorMOBA(JogadorMOBA jogador)	Apresenta as opções disponíveis para utilizadores do tipo jogador MOBA.
menuJogadorEfootball(JogadorEfootball jogador)	Apresenta as opções disponíveis para utilizadores do tipo jogador Efootball.
menuTreinador(Treinador treinador)	Apresenta as opções disponíveis para utilizadores do tipo treinador.
menuAdministrador(Administrador administrador)	Apresenta as opções disponíveis para utilizadores do tipo administrador.
procurarJogador(String nomeCompleto, String nickname)	Procura um jogador pelo nome completo e nickname nas listas de jogadores FPS, MOBA e Efootball, retornando o jogador se encontrado, ou null caso contrário.

procurarJogadorPorNickname(String nickname)	Procurar Jogador por nickname.
procurarAdministradorPorEmail(String email)	Procurar Administrador pelo email.
procurarTreinadorPorEmail(String email)	Procurar Treinador pelo email.
procurarTorneio(String nome)	Procurar torneio pelo nome.
procurarEquipa(String nome)	Procurar equipa pelo nome.
procurarPartidaPorId(Torneio torneio, int id)	Procurar partir pelo id.
nicknameJaExiste(String nickname)	Verifica se o nickname dos Jogadores já existe.
carregarDados()	Permite carregar os dados dos ficheiros.
visualizarFicheiro()	Permite vizualizar os dados dos ficheiros.
salvarDados()	Permite guardar os dados dos ficheiros.
main(String[] args)	Ponto de entrada da aplicação, que inicia o sistema.

Workflow

Primeiro, desenvolvemos um conjunto de classes essenciais para o sistema, incluindo **Jogador**, **Administrador**, **Treinador**, **Torneio** e **Equipa**. Além disso, criámos subclasses de **Jogador**, como **JogadorFPS**, **JogadorMOBA** e **JogadorEFootball**, que herdam da classe base **Jogador**. Estas subclasses possuem atributos específicos para cada tipo de jogador, além de getters e setters necessários para a manipulação dos atributos.

A seguir, implementámos métodos nessas classes para fornecer as funcionalidades essenciais. Em paralelo, desenvolvemos a classe **Consola**, situada no Frontend, com uma série de métodos que facilitam a interação com o utilizador. Com as funcionalidades básicas em funcionamento, introduzimos a classe **Autenticacao**, que tem como responsabilidade a gestão da autenticação de utilizadores. Esta classe permite realizar o registo e o log in e validar os acessos dos utilizadores ao sistema, assegurando que apenas os utilizadores autorizados possam interagir com o sistema.

Em seguida, desenvolvemos a classe **Principal**, que serve como ponto de entrada para o programa. A **Principal** apresenta ao utilizador um menu principal que dá acesso a vários submenus, cada um responsável por ativar funcionalidades específicas de outras classes, facilitando a interação com o sistema de forma organizada e intuitiva.

Finalmente, criamos a classe **GestorDeFicheiros**, que permite guardar as informações dos utilizadores, equipas e torneios em cinco ficheiros distintos. Esta classe foi acompanhada por alterações em alguns métodos existentes, com o objetivo de possibilitar a leitura desses ficheiros e a recuperação das informações armazenadas, garantindo que o sistema possa aceder e atualizar os dados conforme necessário. Assegura a persistência e integridade das informações ao longo do tempo.

Essas etapas culminaram num sistema bem estruturado, com funcionalidades robustas e capacidade de armazenar dados de forma eficiente, permitindo a interação entre diferentes tipos de utilizadores e operações dentro do sistema.