Relatório - Projeto Final - Programação Orientada a Objetos

Professor: Marcio Delamaro

Nomes dos integrantes e NUSP:
Cauê Paiva Lira - 14675416
Luiz Felipe Diniz Costa - 13782032
João Pedro Alves Notari Godoy - 14582076

1. Manipulação de arquivos em C

Esta parte do programa foi desenvolvida durante a disciplina de Arquivos, utilizando linguagem C para a manipulação de um arquivo CSV contendo os jogadores da FIFA nos anos 2017-23 e que foi disponibilizado para os alunos no começo da disciplina para ser usado nos trabalhos práticos da mesma. Esse CSV é convertido em binário para serem realizadas operações como busca, adição e remoção de registros. Também foi utilizado o index para maior eficiência na busca de elementos dentro dos arquivos.

2. Socket em Python

2.1. Arquivo ClientMessagesHandler.py

O arquivo ClientMessagesHandler.py contém a classe ClientMessagesHandler, que é responsável por lidar com as mensagens recebidas do cliente. Esta classe possui um manipulador de arquivos (TrabalhoArquivosHandler) que realiza as operações de manipulação dos arquivos binários. Quando uma mensagem é recebida, a classe divide a mensagem em parâmetros e executa a funcionalidade solicitada no manipulador de arquivos. Após a execução, o resultado é enviado de volta ao cliente.

2.2. Arquivo SocketsHandler.py

O arquivo SocketsHandler.py define a classe SocketsHandler, que gerencia o servidor de sockets. Esta classe cria um socket do tipo TCP, configura-o para reutilizar endereços, e o vincula a um endereço IP e porta específicos. O servidor então entra em modo de escuta para aceitar conexões de clientes.

Quando um cliente se conecta, o servidor aceita a conexão e cria um manipulador de mensagens (ClientMessagesHandler) específico para o cliente. O servidor então entra em um loop onde ele recebe mensagens do cliente, passa essas mensagens para o manipulador de mensagens para que a funcionalidade apropriada seja executada, e envia de volta o resultado ao cliente. Se ocorrer um erro ou o cliente se desconectar, o servidor fecha a conexão e remove os dados do cliente.

2.3. Arquivo TrabalhoArquivosHandler.py

O arquivo TrabalhoArquivosHandler.py contém a classe TrabalhoArquivosHandler, que é responsável por realizar todas as operações de manipulação de arquivos binários. Esta classe lida com os comandos no terminal, criados com a biblioteca subprocess do python, sendo esses comandos aqueles especificados no trabalho de arquivos, bem como a conversão de arquivos CSV para binários.

2.3.1. Inicialização:
A classe determina o sistema operacional em uso (Linux ou Windows) e compila o código C que manipula os arquivos binários. Comandos Específicos do Sistema Operacional: Dependendo do sistema operacional, a classe define os comandos apropriados para executar o código C e deletar arquivos.

2.3.2. Execução de Comandos:
Métodos como __run_command são usados para executar comandos específicos, passando parâmetros através do stdin para o programa C.

2.3.3. Manipulação de Arquivos:
Métodos como csv_to_binary, show_all_players, search_players, delete_players, insert_player e update_player lidam com a conversão de arquivos CSV para binários e a manipulação dos registros dentro desses arquivos.

3. Comunicação do Python com o JAVA (FIFAFetch.java)

3.1. O arquivo FIFAFetch.java define uma classe responsável pela comunicação entre a interface gráfica do usuário (GUI) e o servidor de sockets. Esta classe estabelece uma conexão com o servidor, envia solicitações e processa as respostas recebidas. Aqui estão os principais pontos:

3.1.1. Construtores:
Existem dois construtores: um que utiliza valores padrão para endereço do servidor e porta, e outro que permite especificar esses valores.

3.1.2. Método sendRequest:
Este método estabelece uma conexão com o servidor, envia a solicitação e aguarda a resposta. Se o servidor ainda não tem o ID do cliente, ele envia o ID primeiro.

3.1.3. Métodos getPlayers, removePlayer e updatePlayer:
Estes métodos utilizam sendRequest para enviar solicitações específicas ao servidor, como buscar, remover ou atualizar jogadores. Eles processam a resposta do servidor e retornam os resultados apropriados.

4. GUI em Java (FIFAFetchGUI.java)

4.1. O arquivo FIFAFetchGUI.java define a interface gráfica que os usuários interagem para buscar, remover e atualizar informações de jogadores de futebol. Ele utiliza a classe FIFAFetch para se comunicar com o servidor. Aqui estão os principais componentes e funcionalidades:

4.1.1. Criação da GUI: O método createAndShowGUI configura a janela principal da aplicação, incluindo campos de texto para inserir critérios de busca e um botão para iniciar a busca.

4.1.2. Método showAllPlayers: Quando o botão Show All Players é clicado, ele transcreve todo o conteúdo do arquivo (todos os registros de jogadores) na interface.

4.1.3. Método searchPlayer: Quando o botão de busca é clicado, este método coleta os critérios de busca dos campos de texto, constrói a solicitação, envia-a ao servidor e atualiza a interface com os resultados.

4.1.4. Método openPlayerWindow: Este método cria uma nova janela com detalhes de um jogador específico e botões para remover ou atualizar o jogador.

4.1.5. Método showPlayersButtons:
Atualiza a interface para exibir botões para cada jogador encontrado na busca.

4.1.6. Método removePlayer:
Envia uma solicitação ao servidor para remover um jogador e atualiza a interface com base no resultado.

4.1.7. Método updatePlayer:
Abre uma nova janela para atualizar os dados de um jogador e envia a solicitação de atualização ao servidor.

4.1.8. Método updatePlayersList:
Atualiza a lista de jogadores na interface com as novas informações após uma atualização bem-sucedida.

5. Funcionamento da Comunicação GUI-Servidor

5.1. Inicialização da GUI:
Quando a aplicação é iniciada, a GUI é configurada e a conexão com o

servidor é estabelecida usando FIFAFetch.

5.2. Envio de Solicitações:
Quando o usuário interage com a GUI (por exemplo, clicando no botão de busca), a solicitação é enviada ao servidor através de métodos como sendRequest.

5.3. Recebimento e Processamento de Respostas:
As respostas do servidor são recebidas, processadas e a interface é atualizada de acordo.

5.4. Operações de Manipulação de Dados:
A GUI permite operações como busca, remoção e atualização de jogadores, que são executadas pelo servidor e refletidas na interface.

6. Como inicializar o programa

6.1. Linux

6.1.1. Instalações

Para utilizar o programa em um sistema operacional linux, não é preciso baixar nenhuma dependência prévia.

6.1.2. Rodar o programa

Para compilar, dentro da pasta out os arquivos java utilize o comando:

```
make compile
```

Feito isso, utilize esse comando para inicializar o programa.

```
make run
```

6.2. Windows

6.2.1. Instalações

Para a utilização no Windows, será necessário que a dependência Make seja previamente instalada, além do compilador gcc - g++.

6.2.2. Rodar o programa (powershell)

Para rodar, limpe builds prévias que possam estar disponíveis, utilize esse comando no powershell:

```
make clean
```

Após isso, para compilar o código em java, utilize:

```
make compile
```

Então, utilize esse comando para rodar o server em python

```
make run_server
```

Finalmente, em outro terminal, utilize esse comando para rodar a GUI em java

```
make run_interface
```