# Trabalho prático: Jogo da Memória

**GRUPO**

Luca Machado Mendonça
Laura Barbosa de Campos
Leonardo Saracino de Almeida
Igor de Paula Siqueira Monárdez

# DOCUMENTAÇÃO DO CÓDIGO

Funções do arquivo servidor.py:

`limpaTela()`: Limpa a tela (terminal).

`fechaPeca(tabuleiro, i, j)`: Fecha a peça na posição (i, j). Se a posição já está fechada ou se já foi removida, retorna False. Caso contrário, retorna True.

`abrePeca(tabuleiro, i, j)`: Revela a peça na posição (i, j). Se a posição já está aberta ou se ja foi removida, retorna False. Caso contrário, retorna True.

`novoTabuleiro(dim)`: Cria um novo tabuleiro de dimensão dim x dim com peças aleatórias.

`imprimeTabuleiro(tabuleiro)`: Imprime o estado atual do tabuleiro.

`fechaPeca(tabuleiro, i, j)`: Fecha a peça na posição (i, j). Se a posição já está fechada ou se já foi removida, retorna False. Caso contrário, retorna True.

`removePeca(tabuleiro, i, j)`: Remove a peça na posição (i, j). Se a posição já está removida, retorna False. Caso contrário, retorna True.

`novoPlacar(nJogadores)`: Cria um novo placar zerado.

`incrementaPlacar(placar, jogador)`: Adiciona um ponto no placar para o jogador especificado.

`imprimePlacar(placar)`: Imprime o placar atual.

`imprimeStatus(tabuleiro, placar, vez)`: Imprime informações básicas sobre o estado atual da partida.

`leCoordenada(coord)`: Lê as coordenadas de uma peça (recebidas do programa cliente).

Funções do arquivo cliente.py:

`limpaTela()`: Limpa a tela (terminal).

`leCoordenada(dim)`: Lê as coordenadas de uma peça (input do jogador).

`recebeDadosNumericos(msg)`:  Extrai os dados de mensagens recebidas.

Funções do arquivo utils.py:

`envia_mensagem(sock, data)`: Envia uma mensagem para um socket com cabeçalho simples contendo o tamanho da mensagem.

`recebe_mensagem(sock)`: Lê o tamanho do cabeçalho para saber o tamanho exato da mensagem a ser recebida.

`recvall(sock, count)`: Função auxiliar de `recebe_mensagem(sock)`. Cria um buffer para garantir que a mensagem seja recebida sem que nenhum bit seja perdido no processo.

## EXECUÇÃO DOS PROGRAMAS CLIENTE E SERVIDOR

O jogo foi implementado em Python 3. Logo, é necessário ter uma versão instalada para executá-lo.

Para executar o programa servidor utilize o seguinte comando:

```
# python server.py
```

Em seguida, serão feitas duas perguntas. A primeira será:

```
# Insira a dimensão do tabuleiro:
```

Nesse momento, você deve escolher um número par, maior que 0 e menor que 10, que será a dimensão do tabuleiro do jogo. Digite o número e aperte a tecla <ENTER> para prosseguir.

Feito isso, a segunda pergunta aparecerá:

```
# Insira o número de jogadores:
```

Você deverá digitar a quantidade de jogadores da partida e apertar a tecla <ENTER> para prosseguir.

O programa executará localmente, na porta 10040.

A partir desse momento, o programa entrará em modo de espera pela conexão dos jogadores.

Para executar o programa cliente utilize o seguinte comando:

```
# python client.py
```

Serão solicitadas algumas informações sobre o programa servidor para que a conexão possa ser estabelecida. A primeira será:

```
# Insira o endereço de IP do servidor:
```

Você deverá digitar o endereço de IP da máquina onde o programa servidor está executando — “localhost” (sem aspas) caso o programa servidor esteja rodando na mesma máquina — e apertar a tecla <ENTER> para prosseguir.

A próxima pergunta será:

```
# Insira o número da porta:
```

Digite “10040” (sem aspas) e aperte a tecla <ENTER> para prosseguir.

E pronto, a conexão foi feita! Caso a quantidade de jogadores estabelecida no servidor tenha sido atingida, a partida começará. Caso contrário, o programa exibirá a mensagem “Aguardando outros jogadores…” até que todos os jogadores estejam conectados, e então a partida começará.

## DECISÕES E MELHORIAS

A fim de modernizar e facilitar a implementação do jogo, optamos por atualizar o código original do jogo. Sendo assim, o código escrito em python 2 foi modificado para ser executado em python 3.

Para evitar que o usuário escolha um valor inadequado para a dimensão do tabuleiro e cause um mau funcionamento do jogo, foi implementado um tratamento no input da dimensão. Sendo assim, inputs que não cumpram todos os requisitos (número par, maior que 0 e menor que 10) não serão aceitos.

Um problema enfrentado pelo grupo durante o trabalho foi que muitas vezes as mensagens apareciam cortadas ou, mais comumente, um mesmo socket recebia duas mensagens de uma vez. Nossa solução foi criar um cabeçalho simples no começo das mensagens enviadas para especificar seu tamanho exato. Para tal, temos funções para gerar e para ler este cabeçalho. Além disso, também temos uma função auxiliar que cria um buffer para garantir que toda a mensagem era exibida sem que houvesse perda parcial de bits. Essas funções estão todas contidas no arquivo utils.py.

## DIVISÃO DE TAREFAS

A divisão de tarefas adotada pelo grupo foi feita de forma que cada um pudesse trabalhar com o que se sente mais confortável. Montamos uma lista de tarefas com todos os passos da implementação do jogo e outras tarefas necessárias para a entrega do trabalho, e cada membro ficou responsável por uma parte. A seguir está uma lista com mais detalhes sobre as contribuições de cada um:

Luca foi responsável pela lógica de transição entre rodadas e correção do erro das mensagens se mesclando, além da correção de pequenos bugs e ajustes gerais.

Laura foi responsável pela atualização do código original para Python 3, pelo tratamento do input da dimensão do tabuleiro, pela revisão e ajustes finais do código e pelo relatório.

Leonardo foi responsável por parte da lógica de recepção de mensagens, lógica do fim do jogo e do placar final.

Igor foi responsável pela criação dos soquetes, separação das funções de servidor e de cliente, lógica de entrada no servidor e recuperação de seus dados.