Instituto Politécnico Viana do Castelo
Escola Superior de Tecnologia e Gestão

Laboratório de Programação
2020/2021

Grupo nº7

Leonardo Andrade nº23415

João Miranda nº23416

Miguel Cruzeiro nº23788

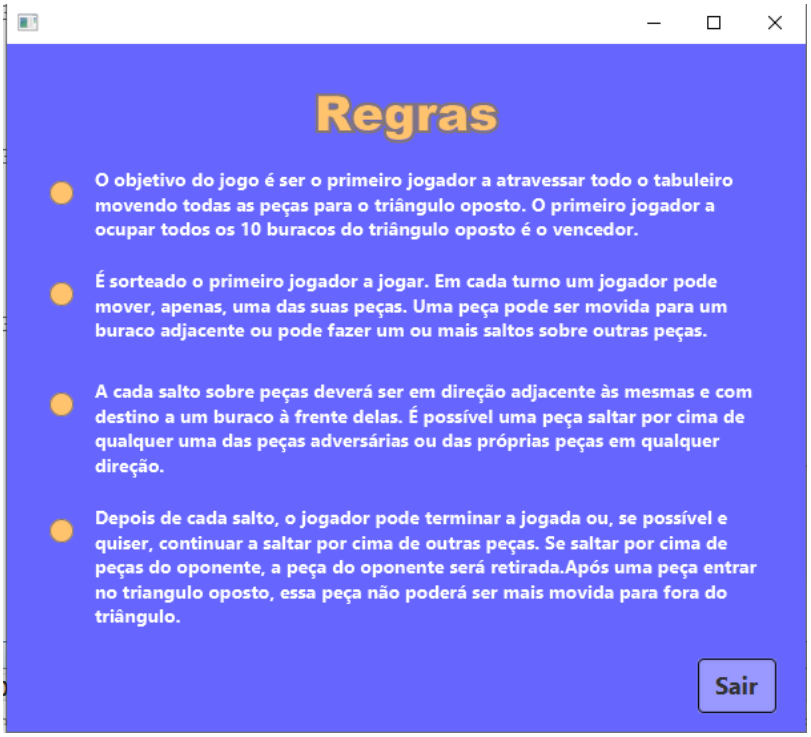
Relatório de Projeto

Junho 2021

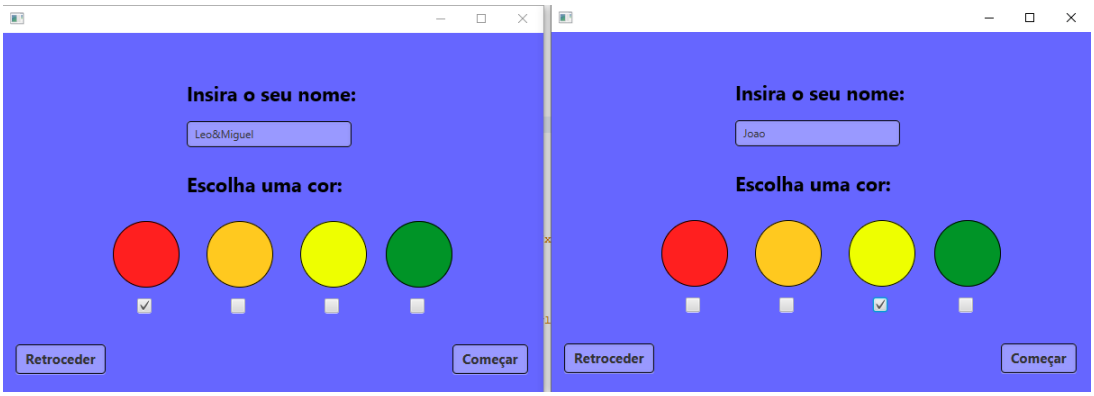
1. Manual de utilização

Primeiramente, o utilizador tem de correr o servidor, para que os 2 jogadores se conectem ao jogo. Dois jogadores conectam-se ao servidor e é lhes apresentado o ecrã inicial do jogo, com as opções “Jogar”, “Regras” e “Sair”.

Caso um jogador selecione a opção de “Sair”, a janela de jogo fecha. Caso selecione a opção de “Regras”, é apresentado ao jogador uma nova janela com as regras principais do jogo.

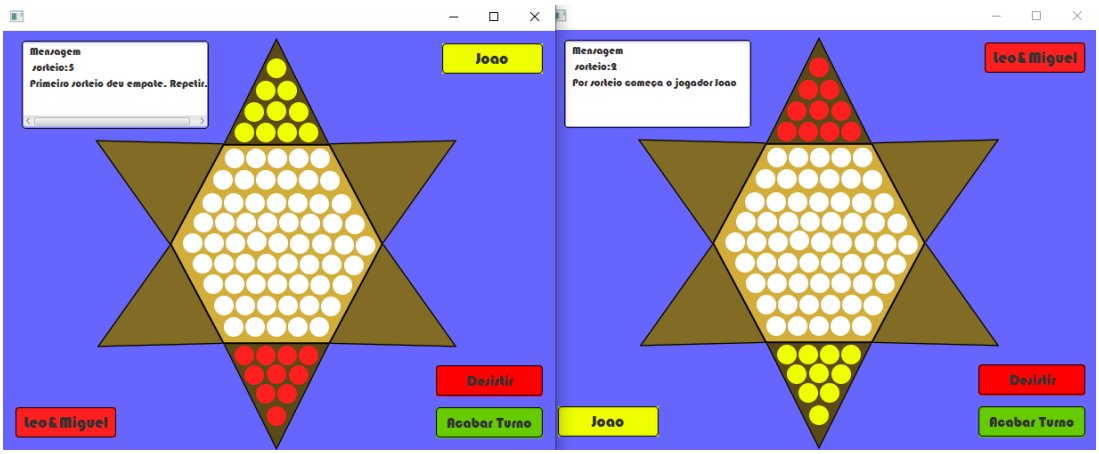


Caso o jogador selecione o botão “Jogar”, vai para outro ecrã onde poderá escrever o seu nome e escolher a cor das suas peças. Para prosseguir para a página seguinte, os jogadores terão de fazer um *click* no botão “Começar”. Caso desejem voltar para a página anterior, fazem *click* no botão de “Retroceder”.

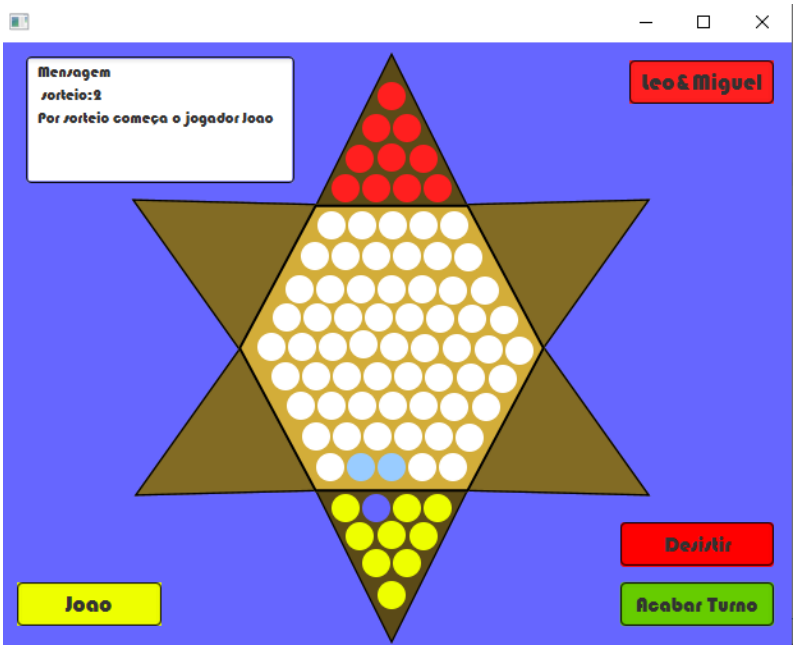


Após escolher a opção de “Começar”, o jogador será direcionado para uma página onde irá aguardar pelo outro jogador. Esta animação de espera aparecerá ao jogador que selecionar o botão de “Começar” em primeiro lugar.

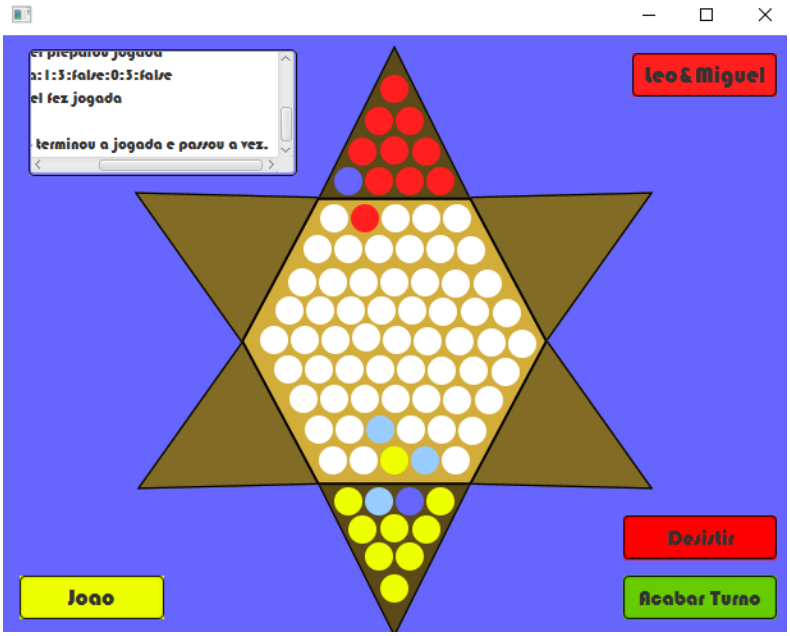
Após os 2 jogadores, introduzirem o seu nome e cor e selecionarem o botão de "Começar", serão direcionados para a página de jogo. Esta página tem o tabuleiro com as peças dos 2 adversários com as respetivas cores.
Antes de começar o jogo é feito um sorteio para definir o primeiro jogador a jogar.
Cada jogador tem 10 peças e pode jogar uma peça para uma posição vazia, em cada turno.



Na figura seguinte está representada uma possível 1ºjogada. Foi selecionada uma bola amarela. Para representar que uma bola foi selecionada, a bola mudará para uma cor azul-escura. Quando selecionada, é apresentada as jogadas possíveis, tendo em conta a posição da bola selecionada. Neste caso, o jogador pode jogar nas 2 casas á frente da bola.
Caso o jogador queira desselecionar a bola tem de fazer um novo *click* na bola selecionada.
Cada jogada feita pelo jogador é guardada e aparecerá na caixa de mensagens, assim como todas as informações adicionais do jogo.

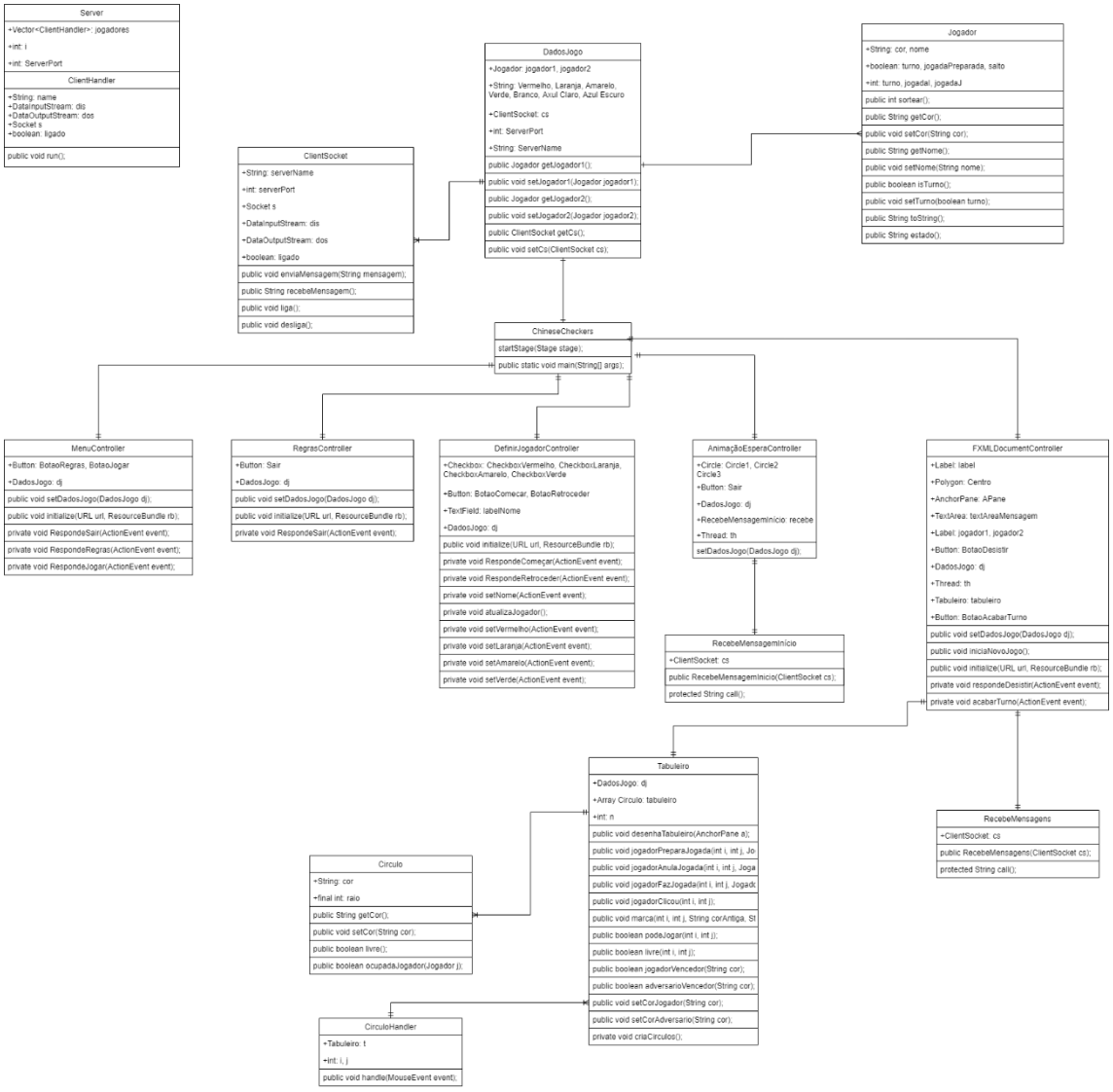


Na figura seguinte está representada a possibilidade de uma peça ter pela frente outra peça, neste caso é possível saltar por cima dessa peça e realizar uma jogada adicional.
O botão “Acabar turno” tem como função passar a vez ao adversário.
O botão “Desistir”, tal como o nome indica, tem como função desistir do jogo. Assim que um jogador faça *click* no botão, o outro jogador receberá uma mensagem a dizer que o adversário desistiu.



O objetivo do jogo é ser o primeiro jogador a atravessar todo o tabuleiro movendo todas as peças para o triângulo oposto. O primeiro jogador a ocupar todos os 10 buracos do triângulo oposto é o vencedor.

2. Diagrama de classes e relações



3. Desenvolvimento e implementação

Classe *Server*

Esta classe contém o servidor do jogo e permite que os jogadores se conectem ao mesmo. O método *main* desta classe contém um ciclo *while* que permitirá adicionar clientes à lista ativa do servidor à medida que estes se conectam. É criado um *ClientHandler* para cada cliente que se conectar, contendo um *DataInputStream* e um *DataOutputStream.*
Dentro da classe *Server* existe a classe *ClientHandler* criada para cada cliente que se ligar e associa ao mesmo um nome, um *DataOutputStream*, um *DataInputStream*, um *Socket* e um *boolean* para verificar a ligação ao servidor. O método *run(),* contido na classe *ClientHandler,* permitirá, enquanto o cliente estiver ligado ao servidor, ler as mensagens dos outros clientes e escrever mensagens para os restantes clientes.

Classe *ClientSocket*

A classe *ClientSocket* permitirá fazer a conexão dos clientes ao servidor.
A classe contém os métodos *enviaMensagem()* e *recebeMensagens()* que permitirem enviar uma mensagem para o servidor e receber e ler mensagens do servidor, respetivamente. Contém também os métodos *liga()* e *desliga()* que permitem fazer a ligação de um cliente com o servidor e desligar o cliente do servidor, respetivamente.

Classe *Jogador*

A classe *Jogador* definirá as propriedades de cada jogador. Tem um método *sortear()* que permite simular um lançamento de dados, de modo, a decidir quem começará a jogar. O método *estado()* é um método utilizado para retornar em formato de mensagem a jogada executada.

Classe *DadosJogo*

A classe *DadosJogo* define as propriedades do jogo, assim como inicializar os jogadores. Nesta classe são definidas as cores, jogadores e as definições de ligação ao servidor.

Classe *ChineseCheckers*

A classe *ChineseCheckers* permitirá fazer a conexão dos clientes ao servidor. A partir desta classe podemos inicializar o jogo. Esta predefinido, nesta classe, que a primeira página a aparecer seja a do menu.

Classe *MenuController*

Nesta classe temos o controlador da página de Menu. Contém um método *setDadosJogo()* que permite passar os atributos dos dados do jogo. Tem um método *RespondeSair()* que permite fechar o jogo. Tem um método *RespondeRegras()* que ao fazer *click* no botão de "Regras" irá mandar o jogador para a página de regras.

Classe *RegrasController*

Nesta classe temos o controlador da página de Regras. Contém um método *setDadosJogo()* que permite passar os atributos dos dados do jogo. Tem um método *RespondeSair()* que permite voltar à página de Menu.

Classe *DefinirJogadorController*

Nesta classe temos o controlador da página de DefinirJogador. Contém um método *setDadosJogo()* que permite passar os atributos dos dados do jogo. Neste método temos um conjunto de ciclos *ifs* para impedir que um jogador escolha mais que uma cor.
Tem um método *RespondeRetroceder()* que permite voltar à página de Menu e um método *RespondeComeçar()* que permite aceder á página de espera pelo outro jogador (AnimaçãoEspera). Tem um método *setNome()* que permite atualizar a *label*, tendo em conta o nome do jogador. Contém o método *atualizaJogador()* que permite atualizar o nome e a cor do jogador, tendo em conta, os inputs. Os métodos *setVermelho(), setLaranja(), setAmarelo() e setVerde()* têm como função selecionar uma *checkbox*, desselecionar as restantes e guardar a cor escolhida.

Classe *RecebeMensagemInicio*

Nesta classe será utilizado um método de nome call(), onde através do *ClientSocket* iremos poder fazer a receção de mensagens em formato String.

Classe *AnimaçãoEsperaController*

Nesta classe temos o controlador da página de AnimaçãoEspera. Contém um método *setDadosJogo()* onde vamos fazer um *Set* dos dados do jogo, ou seja, vamos atualizar os dados de ambos os jogadores tendo em conta os inputs realizados anteriormente, e vamos fazer com que os dados de ambos os jogadores comuniquem, através de mensagens. Assim sendo através de um *Listener* vamos conseguir efetuar a receção dos dados do nosso adversário, e atualizar localmente o jogador2.
No método *initialize()* desta classe está implantada a animação das bolas.
Tem um método *RespondeSair()* que permite voltar à página inicial. Além disso, cancela a tarefa de receber e desliga o *socket.*
O método *IniciaJogo()* permite aceder à página de jogo com o tabuleiro e peças dos jogadores e jogar.

Classe *RecebeMensagens*

Nesta classe será utilizado um método de nome call(), onde através do *ClientSocket* iremos poder fazer a receção de mensagens em formato String.

Classe *Circulo*

Esta classe define as propriedades de cada bola no tabuleiro. Nesta classe temos o método *livre()* que verifica se uma casa está livre, ou seja, se um circulo tem a cor branca. Tem um método *ocupadaJogador()* que verifica se uma casa está ocupada, ou seja, se um circulo tem a cor de um jogador.

Classe *CirculoHandler*

Nesta classe é atribuída propriedades de *click* para as bolas do tabuleiro. Tem um método *handle()* para ativar um evento do *mouse* na bola de coordenadas (i, j).

Classe *Tabuleiro*

A classe Tabuleiro define o tabuleiro do jogo.
A classe contém os seguintes métodos:
No método *desenhaTabuleiro()* são utilizados dois ciclos *for* para adicionar bolas no *AnchorPane* do tabuleiro e para cada que cada uma delas tenha propriedades de *click*.
O método *jogadorPreparaJogada()* é utilizado para pintar as casas livres com a cor Azul claro e a bola que se clicou com Azul Escuro. Este método fornece informações visuais para onde pode jogar.
O método *jogadorAnulaJogada()* é utilizado para anular uma jogada, faz com que as casas anteriormente pintadas a Azul no método *jogadorPreparaJogada()* voltem à cor inicial.
O método *jogadorFazJogada()* é utilizado quando uma jogada é concluída, ou seja, move a bola anteriormente selecionada e coloca as restantes na cor inicial.
O método jogadorClicou() é evocado após o *click* numa bola. Inicialmente, este método é composto por várias validações, sendo a primeira, verificar se é o turno do jogador, se não for, sai do método. A segunda, valida se clicou numa peça da cor do adversário ou branca, saindo também do método nestes casos. Seguidamente, verifica se a jogada estiver preparada e se:

- a cor é azul claro, evocando o método *jogadorFazJogada()* para o local da peça;
- a cor é azul-escura, evocando o método *jogadorAnulaJogada()*;
- a cor é igual à cor de jogador e pode inicializar uma nova jogada (este caso é evocado após um salto).

O método *marca()* marca uma casa do tabuleiro com uma cor.
O método *podeJogar()* indica as casas que é possível jogar.
O método *livre()* verifica se uma casa se encontra livre, ou seja, com a cor branca. Verifica também se a casa está dentro do tabuleiro.
O método *jogadorVencedor()* verifica se o jogador venceu o jogo. Valida se as cores das casas iniciais do adversário forem da cor do jogador.
O método *adversarioVencedor()* verifica se o adversário venceu o jogo. Valida se as cores das casas iniciais forem da cor do adversário.
O método *setCorJogador()* pinta as casas do jogador com a sua cor.
O método *setCorAdversario()* pinta as casas do adversário com a sua cor.
O método *criaCirculos()* adiciona todos os círculos presentes no tabuleiro.

Classe *FXMLDocumentController*

Esta classe é o controlador da página de tabuleiro.
A classe contém os seguintes métodos:
O método *setDadosJogo()* é semelhante ao *setDadosJogo()* localizado na classe *AnimaçãoEsperaControlle*r, ou seja, este método irá atualizar os dados do jogo e o tabuleiro de cada um dos jogadores/clientes(cada jogador receberá por mensagens informações do outro e atualizará localmente o adversário e o tabuleiro tendo em conta estas alterações), através de um sistema de receção e envio de mensagens, além disso este método irá fornecer ao utilizador mensagens personalizadas que aparecerão na *TextArea* no canto superior esquerdo da pagina do tabuleiro.
O método *iniciaNovoJogo()* que permite iniciar um novo jogo, cria um tabuleiro, tendo em conta, as propriedades de dados do jogo e altera as *labels* dos jogadores.
O método *RespondeDesistir()* que permite aceder á página de menu.
O método *acabarTurno()* que permite passar o turno ao adversário.

4. Observações e correções

Tendo em conta as apresentações anteriormente realizadas, consideramos que houve uma alteração drástica, pois não tínhamos o conhecimento devido para o desenvolvimento deste projeto, relativamente à estrutura de classes.
Relativamente à 2ºapresentação realizada (Protótipo de Baixo Nível – Interface e Interação) consideramos que conseguimos representar a interface da maneira planeada inicialmente.

5. Resultados e conclusões

Para concluir, consideramos que este projeto foi essencial na consolidação de conhecimentos obtidos nas aulas. Através do *Scene Builder* conseguimos criar a interface do jogo *Chinese Checkers*. Foi possível conectar o jogo a um servidor, onde os jogadores podem jogar e enviar ao adversário a jogada executada.
Durante a realização do projeto, tivemos bastantes dificuldades, mas, consideramos que os requisitos foram cumpridos na sua maioria.

Link GitHub: https://github.com/MCruzeiro/ChineseCheckers.git

6. Referências

[1] Java Network 200527.pdf
[2] Java Threads 200510.pdfJava Threads 200510.pdf
[3] JavaDoc 210302.pdfJavaDoc 210302.pdf
[4] Javadoc_tags.pdfJavadoc_tags.pdf
[5] JavaFX A01 Intro 210413.pdfJavaFX A01 Intro 210413.pdf
[6] JavaFX A02 PropBindList 210426.pdfJavaFX A02 PropBindList 210426.pdf
[7] JavaFX A03 Graficos 210426.pdfJavaFX A03 Graficos 210426.pdf
[8] JavaFX A04 MenuListFile 200503.pdfJavaFX A04 MenuListFile 200503.pdf
[9] JavaFX A05 Media 200504.pdfJavaFX A05 Media 200504.pdf
[10] https://stackoverflow.com/