



Modelação e Programação

2023/2024

ISEL - DEETC - LEIM

Trabalho Prático 4 - Parte B

Docente: Jorge Branco

Trabalho realizado por:
Guilherme Graça nº: 51827

Turma: LEIM21D

Índice

TP4.....	
Introdução.....	
Atualização dos UML's.....	
UML Simplificado.....	
Diagrama de Classes.....	
UML Completo.....	
Padrão MVC.....	
Interface Visual.....	
Frame Principal.....	
Frame Aposta.....	
Frame Regras.....	
Frame Resultados.....	
Conclusão.....	

Introdução

Este relatório é referente à parte B do TP4 da disciplina de Modelação e programação. Na Parte A, foi apresentada a planificação do jogo de BlackJack single player utilizando a linguagem de programação Java. Nesta segunda parte, detalha-se a implementação da aplicação, com focando se na preparação da interface do utilizador, atualização dos UML e descrição da função de cada classe.

Atualização dos UML's

.Abaixo será mostrado a atualização dos UML's depois da implementação da aplicação.

É de notar que as maiores diferenças relativamente à planificação são relacionadas com a classe da interface gráfica **BlackJackGUI**. Esta agora em vez

de ir buscar as informações de jogo à classe **BlackJack**, vai diretamente às classes **Baralho**, **Jogador**, **Dealer**, **XMLDinheiroJogador** e implementa a sua própria lógica de jogo. Isto foi feito pois no fim de implementada a classe **BlackJack** não era possível reaproveitar a maior parte dos métodos da mesma.

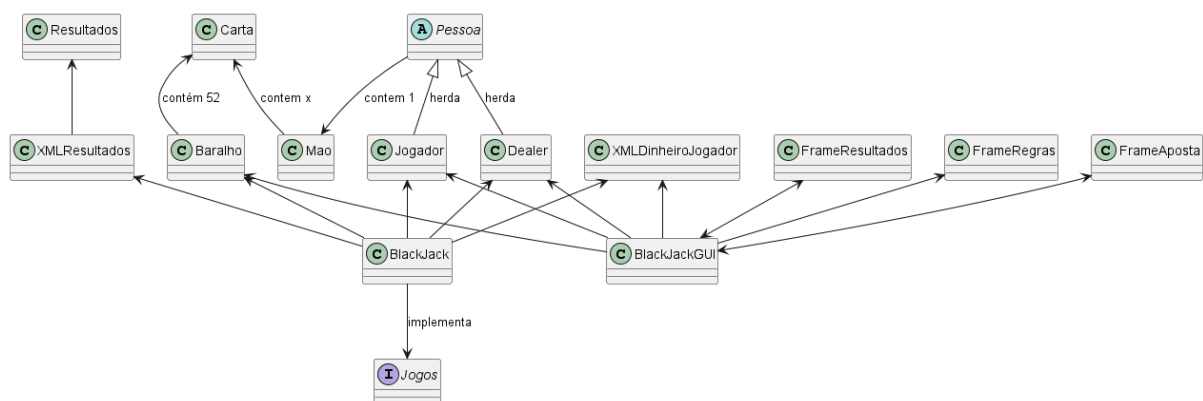
A classe **BlackJackGUI** também se comunica agora com mais 3 classes que representam cada um dos frames secundários que podem ser abertos a partir do frame principal.

Foi também criada uma classe nova chamada resultados, para facilitar o armazenamento dos resultados no fim de cada Jogada.

UML Simplificado

O diagrama UML atualizado abaixo representa as classes principais implementadas na aplicação assim como as suas relações em jogo:

todos os UMLS serão enviados também em anexo com o relatório para melhor visualização



Classes abstratas e com herança:

- **Pessoa** - Classe abstrata que serve como base para a classe **Jogador** e **Dealer**. Contém também uma instância da classe **Mao** (pois cada pessoa tem uma mão em jogo)
- **Jogador** - representa o jogador e herda a classe **Pessoa**.

- **Dealer** - representa o Dealer do jogo e herda a classe **Pessoa** (pois o dealer também é uma pessoa).

Classes relacionadas com as cartas:

- **Carta** - Representa uma carta
- **Baralho** - Contém 52 instancias da classe **Carta** (pois um baralho completo contém 52 cartas)
- **Mao** - Contém um número indeterminado de instâncias da classe **Carta**, no entanto será sempre mais que duas

Classes de lógica do jogo e interface visual

- **BlackJack** - Classe que implementa a lógica principal do jogo, implementa a interface **Jogos** e vai buscar informações às classes: **Jogador**, **Dealer**, **Baralho**, **XMLDinheiroJogador**, e **XMLResultados**.
- **FrameResultado** - representa o frame onde são mostrados os resultados anteriores das jogadas da sessão, comunica se com a classe **BlackJackGUI**;
- **FrameAposta** - representa o frame onde o jogador pode definir a sua aposta e iniciar assim uma nova jogada, comunica-se com a classe **BlackJackGUI**
- **FrameRegras** - representa o frame onde são mostradas as regras do jogo
- **BlackJackGUI** - Classe que também tem uma lógica própria de jogo e onde é implementada a interface gráfica do blackjack e vai buscar informações às classes: **Jogador**, **Dealer**, **Baralho**, **XMLDinheiroJogador**, **FrameAposta**, **FrameResultados** e **FrameRegras**.

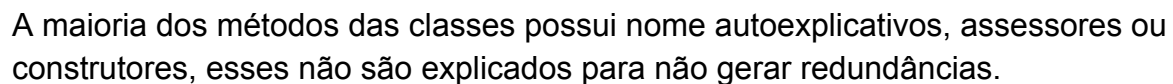
Interface

- **Jogos** - Interface onde estão definidos certos métodos comuns a classes jogo de blackjack, é implementado pela classe **BlackJack**.

XML

- **XMLDinheiroJogador** - Classe com o objetivo de se comunicar com o documento XML que irá guardar o dinheiro Total do Jogador
- **XMLResultados** - Classe com o objetivo de se comunicar com outro documento XML que guarda as jogadas passadas, comunica-se com a classe **Resultados**
- **Resultados** - Classe usada para facilitar o armazenamento dos resultados das partidas, comunica-se com a classe **XMLResultados**

No diagrama de classes abaixo estão representadas as classes usadas no jogo com um maior detalhe nas suas funções, relatando os seus atributos e métodos:



- **Pessoa:**
 - **Atributos:** mao
 - **Métodos:** Pessoa(), adicionarCarta(), getValorMao(), toString()

- **Jogador:**

- **Jogador:**
 - **Atributos:** scanner, dinheiroTotal, apostaAtual

- **Métodos:** Jogador, pedeCarta(), fazerAposta(), ganharAposta(), perderAposta(), empateAposta(), getDinheiroTotal(), getApostaAtual(), injetarDinheiro().

O método injetarDinheiro() Serve para o GUI usar quando o jogador está sem saldo para jogar.

.Os métodos fazerAposta(), ganharAposta(), perderAposta(), empateAposta() definem o que acontece quando com o dinheiro total do jogador dependendo do resultado da jogada

- **Dealer:**
 - **Métodos:** Dealer(), pedeCarta()

Classes relacionadas com as cartas:

- **Carta :**
 - **Atributos:** valor, naipe
 - **Métodos:** Carta() getValor(), getNaipe(), toString(), getValorEmJogo()

O metodo getValor() retorna o valor numérico da carta enquanto o método getValorEmJogo() retorna o quanto essa carta vale em jogo, por exemplo o rei tem como valor de carta 13, no entanto em jogo essa carta, como é uma figura, vale 10.

- **Baralho :**
 - **Atributos:** cartas
 - **Métodos:** Baralho() baralhar(), tirarCarta(), devolverCarta()
- **Mao:**
 - **Atributos:** cartasMao, valorMao
 - **Métodos:** Mao() adicionarCarta(), calcularValorMao(), getValorMao(), mostrarMao().

Nestas classes, tendo em conta que o baralho contem todas as 52 cartas e sempre que se tira uma carta, esta é sempre a de cima do baralho (ultima do array) e ao ir para a mão de um jogador, a mesma sai do objeto baralho. Então estr

XML

Nestas duas classes, para além da utilidade de salvar dados importantes de jogo, é também suposto conseguir aplicar métodos de leitura e para escrever em documentos XML. Para a construção destas classes foi usado como base o exemplo XMLReadWrite do trabalho prático 3.

- **XMLDinheiroJogador :**
 - **Métodos:**carregarDinheiroTotalJogador(),guardarDinheiroTotalJogador()
- **XMLResultados:**
 - **Atributos:** nomeFicheiro
 - **Métodos:** inicializarDocumento(), salvarResultado()
- **Resultados:**
 - **Atributos:** maoJogador, maoDealer
 - **Métodos:** Resultados(), getMaoJogador(), getMaoDealer, getVencedor()

Classes de lógica do jogo e interface visual

É de notar que abaixo estão duas classes que ambas contêm o método main em que uma corre o jogo apenas na consola e a outra corre o jogo com a interface visual ambas implementando a sua propria lógica de jogo.

- **BlackJack:** Ao executar esta classe é possível jogar blackJack através da consola
 - **Atributos:** baralho, jogador, dealer
 - **Métodos:** BlackJack(), main() mostrarRegras(), jogar(), distribuirCartasIniciais(), turnoDoJogador(), turnoDoDealer(), determinarVencedor(), devolverCartasAoBaralho()
- **BlackJackGUI :** Ao executar esta classe é possível jogar BlackJack com acesso a uma interface visual GUI
 - **Atributos:** Labels e Panels com necessidade a serem chamados fora do construtor, baralho, jogador, dealer, listaDResultados
 - **Métodos:** BlackJackGUI(), iniciarJogo(), atualizarLabels(), distribuirCartasIniciais(),turnoDoDealer(), determinarVencedor(), devolverCartasAoBaralho(), atualizarCartas(), getImagemCarta(),getDinheiroDoJogador(), main()

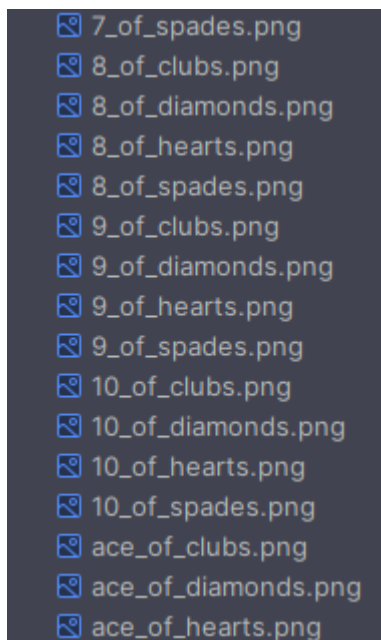
.O método getImagemCarta() pega nos atributos de uma carta e constrói uma string com o nome a localização do ficheiro onde está a imagem dessa carta da seguinte forma:

```

private String getImagemCarta(Carta carta) {
    String valor = "";
    switch (carta.getValor()) {
        case 11: valor = "jack"; break;
        case 12: valor = "queen"; break;
        case 13: valor = "king"; break;
        case 14: valor = "ace"; break;
        default: valor = String.valueOf(carta.getValor()); break;
    }
    String naipe = "";
    switch (carta.getNaipe()) {
        case "Copas": naipe = "hearts"; break;
        case "Espadas": naipe = "spades"; break;
        case "Paus": naipe = "clubs"; break;
        case "Ouros": naipe = "diamonds"; break;
    }
    return valor + "_of_" + naipe;
}

```

tendo em conta que o pacote com as imagens das cartas tem o seguinte aspeto:



Este pacote de imagens foi retirado de um site próprio com assets para jogos chamado Open Game Art. com o seguinte link:
<https://opengameart.org/content/playing-cards-vector-png>

O método atualizarCartas() o Panel onde devem aparecer as Cartas

O método `atualizarPanels()` atualiza todos os panels que têm informações que devem ser atualizadas.

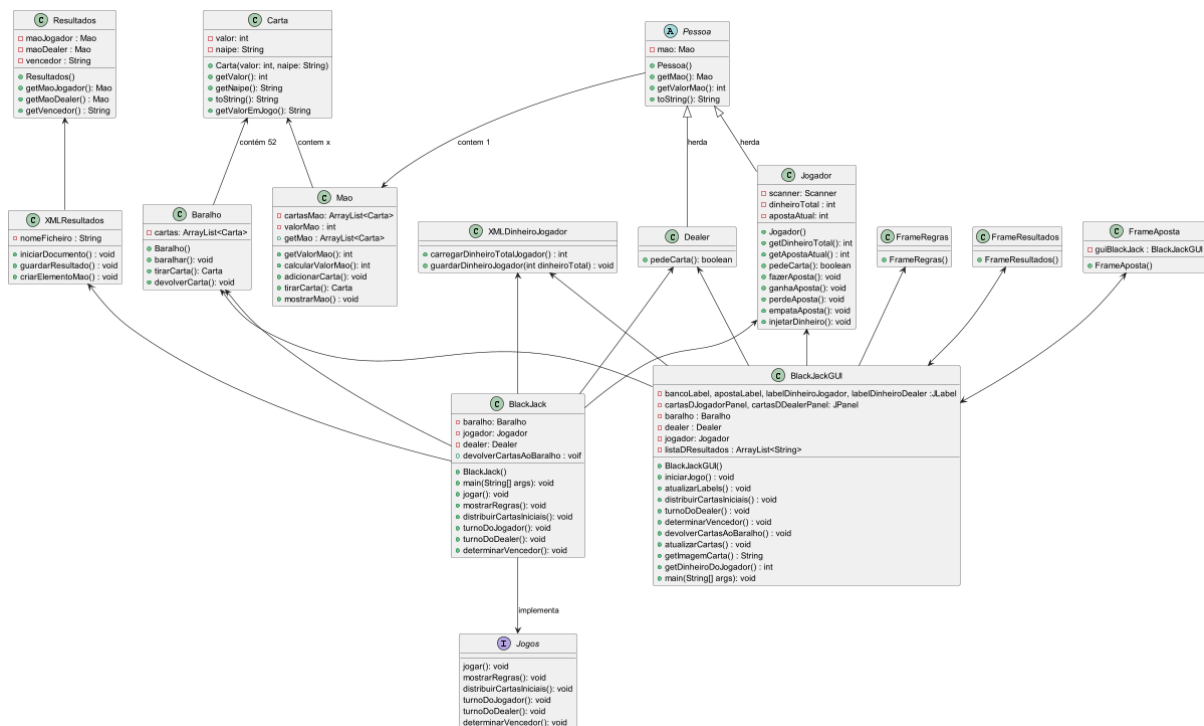
- **FrameResultados:** Janela que exibe os resultados anteriores das jogadas na sessão
 - **Métodos:** `FrameResultados()`

A ideia inicial era que a janela mostrasse todos os resultados anteriores guardados no XML. No entanto, ainda que todos os resultados fiquem guardados no XML, esta janela apenas mostra os resultados da sessão de jogo, guardados em um `ArrayList` nos atributos da classe **BlackJackGUI**.

- **FrameAposta:**
 - **Atributos:** `guiBlackJack()`
 - Tem como atributo uma instância da classe **BlackJackGUI** para conseguir iniciar jogo a partir deste frame
 -
 - **Métodos:** `FrameAposta()`
 -
- **FrameRegras:**
 - **Métodos:** `FrameRegras()`

UML Completo

Abaixo está uma junção do diagrama de classes adicionando a interface **Jogos** que contem diversos métodos que devem ser implementadas em qualquer jogo de blackjack:



Padrão MVC

O padrão MVC (Model-View-Controller) foi utilizado na implementação da aplicação:

Model (Modelo):

- **Classes Usadas:** Carta, Baralho, Mao, Pessoa, Jogador, Dealer, Resultados

View (Visualização):

- **Classes Usadas:** BlackJackGUI, FrameAposta, FrameRegras, FrameResultados

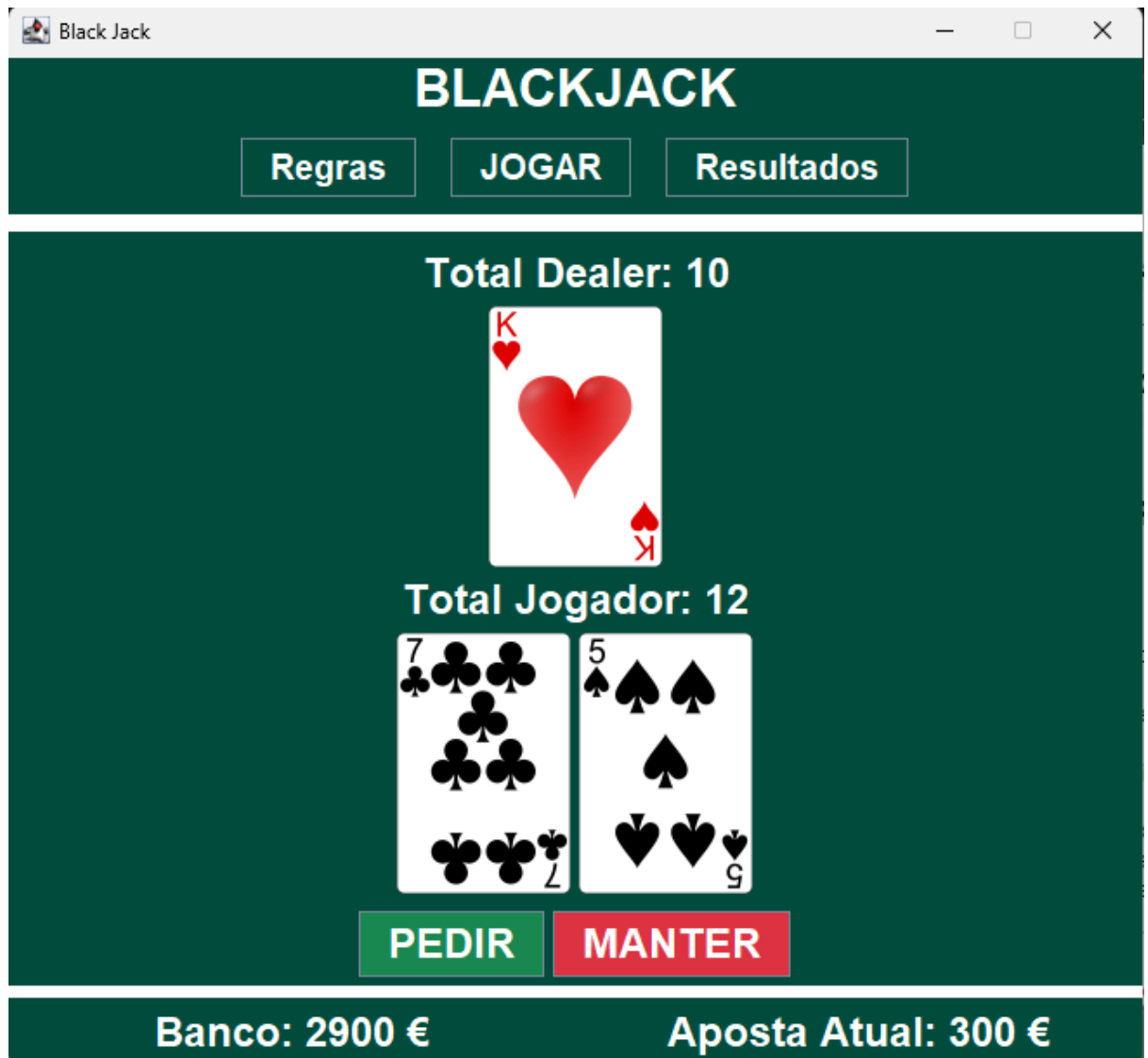
Controller (Controlador):

- **Classes Usadas:** BlackJack, BlackJackGUI

Interface Visual

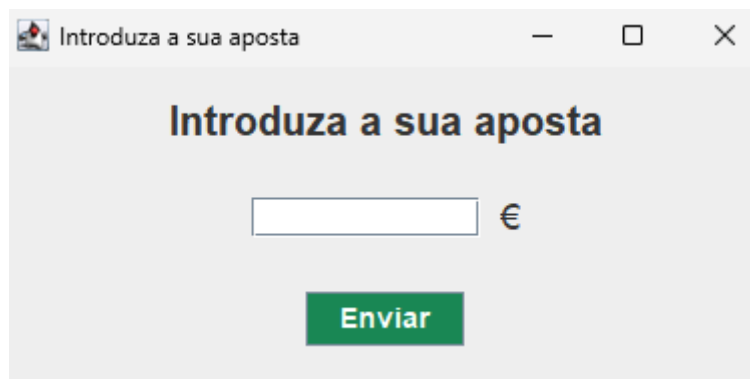
A partir dos esboços criados na parte B deste relatório, foi criada a interface final que tem o seguinte aspeto:

Frame Principal



As principais diferenças entre este frame e o de esboço é que está tudo mais centralizado, os botões de ação passaram para o Panel central e foram feitas linhas divisórias do Panel Central para o Panel do header e do Footer. A maneira de criar estas linhas foi fazendo Panels novos com largura fixa e background branco. A maneira mais simples de criar as cartas foi arranjar um pacote com imagens PNG com todas as cartas de um baralho.

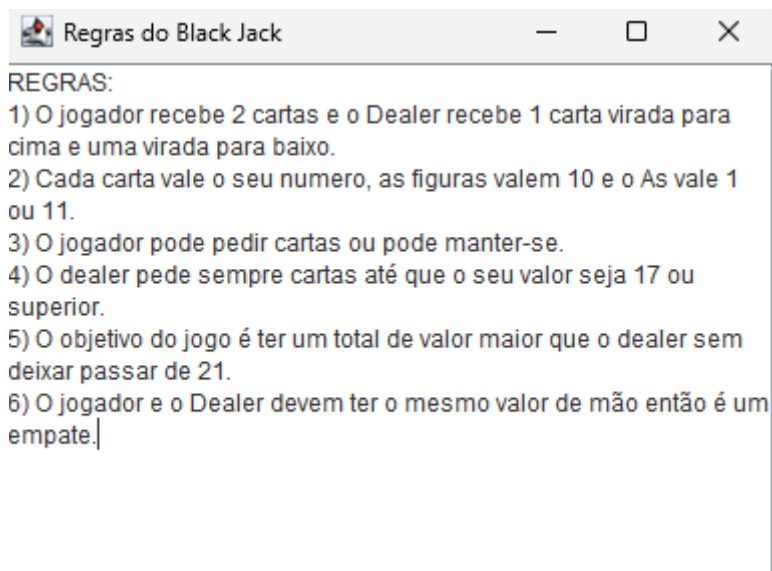
Frame Aposta



A screenshot of a web application window titled 'Introduza a sua aposta'. The window has a standard macOS-style title bar with a red, yellow, and green button on the left, and minus, maximize, and close buttons on the right. The main content area has a light gray background. At the top, the title 'Introduza a sua aposta' is displayed in a bold, black, sans-serif font. Below the title is a white rectangular input field for entering a bet amount, followed by a black Euro symbol (€). At the bottom center of the input area is a green rectangular button with the white text 'Enviar'.

A principal diferença entre este frame é o de esboço é que agora foi adicionado um botão para enviar o valor da aposta que ao pressionar começa uma nova partida.

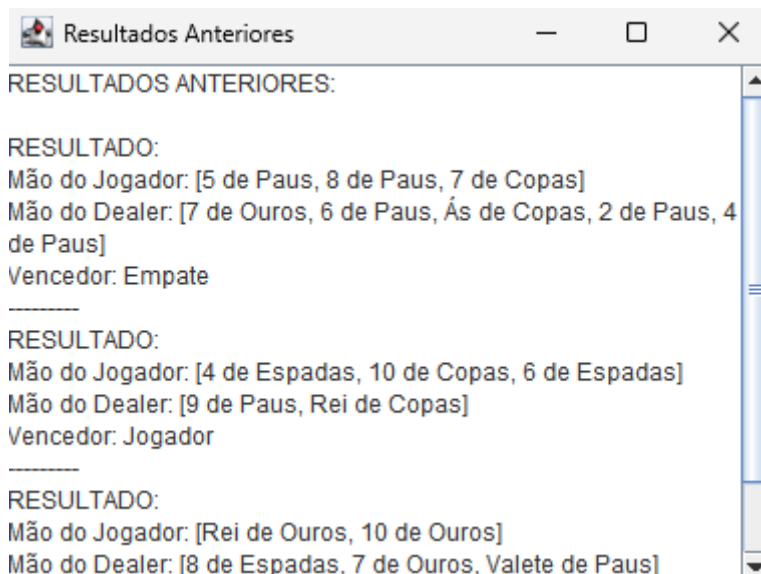
Frame Regras



A screenshot of a web application window titled 'Regras do Black Jack'. The window has a standard macOS-style title bar with a red, yellow, and green button on the left, and minus, maximize, and close buttons on the right. The main content area has a white background. At the top, the title 'Regras do Black Jack' is displayed in a bold, black, sans-serif font. Below the title, the word 'REGRAS:' is written in a bold, black, sans-serif font. Below this, there are six numbered rules listed in a black, sans-serif font: 1) O jogador recebe 2 cartas e o Dealer recebe 1 carta virada para cima e uma virada para baixo. 2) Cada carta vale o seu numero, as figuras valem 10 e o As vale 1 ou 11. 3) O jogador pode pedir cartas ou pode manter-se. 4) O dealer pede sempre cartas até que o seu valor seja 17 ou superior. 5) O objetivo do jogo é ter um total de valor maior que o dealer sem deixar passar de 21. 6) O jogador e o Dealer devem ter o mesmo valor de mão então é um empate.]

Este frame mostra exatamente o que foi concebido no esboço da parte B.

Frame Resultados



Este frame mostra exatamente o que foi concebido no esboço da parte B

Conclusão

Este relatório detalhou a implementação de um jogo de BlackJack singleplayer em JAVA, seguindo o plano elaborado na Parte A e mostrando que aspetos foram alterados em relação ao plano inicial. A aplicação utiliza a biblioteca Swing para a interface gráfica e tem a funcionalidade de leitura e escrita de dados em XML. Este projeto consolidou os conhecimentos teóricos adquiridos ao longo do semestre na disciplina de mop, fortalecendo as capacidades de desenvolvimento de aplicações na linguagem JAVA.