**IPOO 2024 - 2025**

Introdução à Programação Orientada por Objetos

RELATÓRIO DE PROJETO

MineSweeper

|  |  |  |
| --- | --- | --- |
| **Turma** | **Número** | **Nome** |
| PL5 | 202100642 | Gonçalo Martim dos Santos Dernedde Sequeira |
| PL5 | 201902549 | Thiers Pinto de Mesquita Neto |
|  | | **Docente Orientador:** |
|  | | Profª. Paula Miranda |

**Data:** Dezembro / 2024

## ÍNDICE

[1. INTRODUÇÃO 5](#_Toc14820)

[2. DESCRIÇÃO DA IMPLEMENTAÇÃO 6](#_Toc14821)

[3. ESTRUTURA GERAL DO PROGRAMA 7](#_Toc14822)

[4. FUNCIONALIDADES IMPLEMENTADAS 10](#_Toc14823)

[5. DECISÕES DE DESIGN 11](#_Toc14824)

[6. EXEMPLO DE EXECUÇÃO 12](#_Toc14825)

[7. CONCLUSÃO 14](#_Toc14826)

[8. REFERÊNCIAS 15](#_Toc14827)

## Índice de Figuras

Figura 1 - Representação do Menu inicial do Jogo. ............................................................... 12

Figura 2 - Representação do começo do jogo/início de nova partida. ............................. 13

Figura 3 - Vitória ............................................................................................................................ 13

## Índice de Tabelas

Tabela 1 - Descrição da estrutura principal do programa. Fonte: o autor. ........................... 7

Tabela 2 - Principais métodos e funções da Classe Main ........................................................ 8

Tabela 3 - Principais Métodos e Funções da Classe Board ...................................................... 8 Tabela 4 - Descrição das Funcionalidades Implementadas. Fonte: o autor. ...................... 10 Tabela 5 - Descrição gráfica de caracteres representativos de estado das células. Fonte: o autor. ................................................................................................................................................. 11

# 1. INTRODUÇÃO

O projeto apresentado consiste no desenvolvimento de uma aplicação em Java para o jogo **Campo Minado (Minesweeper)**, implementada com conceitos de **Introdução** **à** **Programação Orientada a Objetos (IPOO)**. Este projeto tem como principal objetivo aplicar os fundamentos teóricos aprendidos durante a disciplina de Introdução à Programação Orientada a Objetos (IPOO) na prática, promovendo o desenvolvimento de habilidades técnicas e lógicas essenciais para a construção de aplicações robustas.

O **Campo Minado (Minesweeper)** é um jogo de lógica popular, no qual o jogador deve identificar todas as células que não contêm minas em um tabuleiro. Neste jogo, cada célula revela informações sobre o número de minas nas proximidades, e o desafio reside em usar essas informações para evitar acionar uma mina.

# 2. DESCRIÇÃO DA IMPLEMENTAÇÃO

O projeto **MineSweeper** foi desenvolvido em Java, utilizando os princípios da programação orientada a objetos, com o objetivo de criar uma versão funcional e interativa do jogo.

Para tal, foram utilizadas Classes, que embora tenha sido colocada a restrição que não poderia haver dependências dentro das Classes, foram usadas algumas Classes com funções meramente de auxílio, como por exemplo uma Classe Coordinate (coordenada) que torna mais fácil e prático o tratamento e retorno deste tipo de dados. Esta Classe armazena meramente 2 valores do tipo inteiro servindo como uma referência útil à gestão das células do tabuleiro. Estas práticas foram devidamente aprovadas pelo Responsável pela Unidade Curricular durante uma das aulas Teórico-Práticas.

Quanto aos métodos e fluxo do programa, estes estão presentes nos capítulos que se seguem com descrições detalhadas do seu funcionamento. É, no entanto, aconselhado que se faça uma leitura dos comentários e javadoc presente no código anexado.

Seguidamente, serão descritos a estrutura geral, as funcionalidades implementadas e as decisões de design adotadas durante o desenvolvimento.

# 3. ESTRUTURA GERAL DO PROGRAMA

O programa foi desenvolvido seguindo uma estrutura, organizada em diferentes classes para facilitar a manutenção e a compreensão do código. Cada componente do jogo **Campo Minado (Minesweeper)** foi encapsulado em classes que representam entidades e responsabilidades específicas. Encontra-se descrita na tabela 1 a estrutura principal do programa.

|  |  |
| --- | --- |
| ***Classes*** | ***Explicação*** |
| Board | É a classe principal que representa o tabuleiro do jogo |
| Coordinate | Uma classe auxiliar para gerir as coordenadas das células (**linha** e **coluna**). |
| Game | Classe auxiliar para armazenar o estado de um tabuleiro, a alcunha do jogador e o estado do jogo (utilizado posteriormente para apresentar os jogos ganhos) |
| Main | Classe que gere o fluxo do programa, onde estão presentes todas as funções relativamente à interação entre o utilizador e o programa (Menus, comandos, etc…) |

*Tabela 1 - Descrição da estrutura principal do programa. Fonte: o autor.*

|  |  |
| --- | --- |
| **Classe Main (Principais métodos e funções)** | |
| showStartMenu() | É o primeiro método chamado no arranque do main. Mostra o menu inicial na consola ao utilizador, indicando 3 possíveis ações:   1. Começar Jogo 2. Ver os últimos 10 jogos ganhos 3. Sair   Também responsável pela gestão da  escolha efetuada pelo utilizador |
| interpretCommands() | Método responsável pela interpretação dos comandos inseridos durante a execução do jogo. |
| startGame() | Método responsável por instanciar um novo tabuleiro e por toda a gestão necessária ao iniciar um novo jogo. |
| openCell() | Abre uma célula no tabuleiro atualmente instanciado. |
| saveGame() | Método responsável por tratar todos os dados a ser guardados em caso de vitória para posteriormente poder exibir os últimos 10 jogos salvos. Os dados são armazenados em um array de objetos Game (Classe  auxiliar já anteriormente justificada) |
| showLastGames() | Método responsável por imprimir todos os últimos 10 jogos guardados. |
| asciiConvert | Função útil para converter valores de caracteres para numérico (útil por exemplo nos comandos inseridos em que A = 65, por isso se utilizador inserir o valor A, a função irá subtrair 65 ficando assim 0, que é um valor já indexável em array. Todas estas justificações encontram-se devidamente comentadas no código). |

Tabela 2 - Principais métodos e funções da Classe Main

|  |  |
| --- | --- |
| **Classe Board (Principais métodos e funções)** | |
| placeMines() | Método que gera minas em posições  aleatórias do tabuleiro |
| setNeighboringMineCounts() | Método que calcula e define o número de minas na vizinhança das células |
| countNeighboringMines() | Função que calcula no número de minas vizinhas à célula |
| revealCell() | Revela uma mina, mas apenas caso esta ainda não tenha sido revelada e seja uma posição válida. Posteriormente, se as células vizinhas não tiverem minas ou minas vizinhas, abre-as também. |
| updateVisual() | Atualiza visualmente o aspeto de uma célula em uma determinada posição, passando por várias verificações de forma a saber, por exemplo, se está em modo batota ou não, pois se estiver será atualizada com o respetivo caracter. |
| toggleFlagStatus() | Um simples toggle da bandeira |
| setLosingBoard() | Atualiza visualmente o tabuleiro para o estado pretendido ao perder o jogo. |
| getRandomCleanCoordinate() | Usado para o comando /hint, retorna uma coordenada que seja válida e que não possua minas. |

Tabela 3 - Principais Métodos e Funções da Classe Board

# 4.FUNCIONALIDADES IMPLEMENTADAS

|  |  |
| --- | --- |
| **Iniciar um novo jogo** | Permite ao utilizador inserir um *nickname* (ou caso não adicione é atribuído “Anonymous”) e começar um novo jogo, inicializando um tabuleiro e gerando minas configurados aleatoriamente. |
| **Marcar e desmarcar bandeiras** | O utilizador pode marcar bandeiras para sinalizar possíveis minas, com limite de bandeiras equivalente ao número de minas. |
| **Revelar Células** | Abre uma célula do tabuleiro, mostrando o número de minas adjacentes ou, no caso de mina, acabando o jogo. |
| **Verificar vitória** | Determina se o jogador ganhou e regista a vitória no histórico. |
| **Exibir as 10 últimas vitórias** | Mostra o *nickname* e o tempo gasto em cada vitória armazenada no histórico. |
| ***Cheat Mode*** | Permite visualizar as minas no tabuleiro |

*Tabela 4 - Descrição das Funcionalidades Implementadas. Fonte: o autor.*

# 5.DECISÕES DE DESIGN

A implementação do **MineSweeper** foi projetada por forma a proporcionar uma experiência interativa e intuitiva ao jogador, mantendo a simplicidade característica do jogo clássico. Para isso, foi criada uma **interface textual baseada em comandos**, onde o jogador interage diretamente com o tabuleiro digitando instruções na consola. Essa abordagem foi escolhida por ser versátil, leve e fácil de implementar, adicionando um sistema de Comandos.

O sistema de comandos foi desenvolvido para ser eficiente e autoexplicativo, permitindo que o jogador execute ações essenciais para a dinâmica do jogo. Por conseguinte, destacamos as principais decisões tomadas:

**Diversidade de Comandos**: A lista de comandos cobre todas as ações necessárias, desde abrir células (/open) e marcar minas com bandeiras (/flag), até aos comandos utilitários como /help para consultar instruções e /quit para encerrar o jogo. Esta diversidade contribui para assegurar uma experiência completa, mesmo em um ambiente simplificado.

**Modos Extras**: O comando /cheat foi implementado para fins de depuração e como um recurso para jogadores que desejam explorar o jogo sem restrições, ativando um modo em que as minas são visíveis. O comando /hint, por sua vez, fornece dicas, sugerindo células seguras para abrir, oferecendo suporte a novos jogadores ou auxiliando em momentos críticos.

**Exibição Dinâmica do Tabuleiro:** Utilizar caracteres específicos para representar o estado das células notabuleiro:

|  |  |
| --- | --- |
| ■ | Para células fechadas. |
| ▶ | Para bandeiras. |
| ◆ | Para minas no modo de batota. |

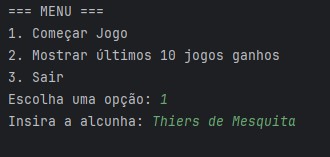
*Tabela 5 - Descrição gráfica de caracteres representativos de estado das células. Fonte: o autor.*

# 6.EXEMPLO DE EXECUÇÃO

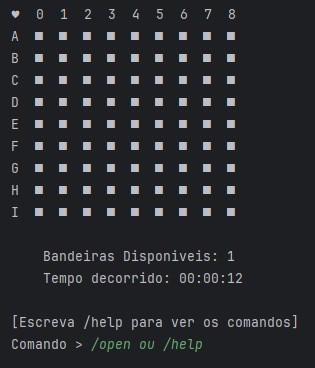
Ao iniciar o jogo **MineSweeper**, é apresentado ao jogador um menu com três opções:

1. **Começar Jogo**: Inicia uma nova partida. Após escolher esta opção, será solicitado ao jogador que este insira um nome, ou apelido, para a sua identificação.
2. **Mostrar Últimos 10 Jogos Ganhos**: Exibe uma lista com as últimas 10 vitórias registadas.
3. **Sair**: Fecha o jogo.

Para iniciar o jogo, o utilizador/jogador deve escolher a opção “**1. Começar Jogo”** no menu e inserir um nome de utilizador quando solicitado. Seguidamente, o jogo será carregado, iniciando e, o jogador poderá interagir com o tabuleiro.



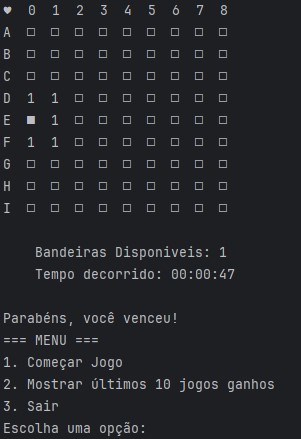
*Figura 1 - Representação do Menu inicial do Jogo.*



*Figura 2 - Representação do começo do jogo/início de nova partida.*

Nesta fase o utilizador encontra-se já em jogo, podendo inserir os comandos /help, que devolve uma listagem dos comandos disponíveis e quais as suas funções. Existem ainda os comandos:

* /open c n , onde c é um caracter e n é um número. Este comando abre a célula indicada.
* /flag c n , onde c e n funcionam da mesma forma que previamente mencionado. Coloca uma bandeira na célula caso não exista já lá uma e retira caso já exista.
* /hint, que indica uma célula ainda não aberta livre de minas.
* /cheat, que ativa o modo batota permitindo a visualização dos locais das minas.
* /quit, que desiste do jogo

 *Figura 3 - Vitória*

# 7. CONCLUSÃO

O projeto **MineSweeper** permitiu aplicar e reforçar os conceitos de programação orientada a objetos por meio do desenvolvimento de um jogo interativo. Com a implementação de funcionalidades essenciais e uma interface de comandos simples, foi possível criar uma experiência funcional e dinâmica para o jogador.

Foram utilizadas no seu desenvolvimento algumas técnicas interessantes como a conversão entre ascii e numérico (necessário para funcionar como índices de array). Esta técnica aplica conhecimentos de Arquitetura Computacional e de Microprocessadores, mas que também foram abordados nas aulas Teórico-Práticas de IPOO.

Todo o código encontra-se devidamente comentado e documentado em javadoc. Vale notar, no entanto, que devido ao uso dos caracteres utilizados no código, o javadoc é impossível de gerar. Isto acontece porque ele acusa alguns dos caracteres utilizados de serem ilegais, lançando múltiplos erros para a consola no processo de geração. No entanto, todas as funções de teor importante do código têm uma secção de javadoc dedicada, apenas não é possível gerar o documento final.

Em suma, a realização deste projeto permitiu aplicar e consolidar os conteúdos teóricos aprendidos, tendo como objetivo a realização de um jogo, o que torna a aprendizagem não só mais dinâmica como também mais divertida.

# 8. REFERÊNCIAS

* [1] – Página com suportes de estudo sobre os temas lecionados, disponibilizado via

Moodle IPS

* + [https://moodle.ips.pt/2425/course/view.php?id=1263#section-4](https://moodle.ips.pt/2425/course/view.php?id=1263%23section-4)

* [2] – Ferramenta de apoio Python Tutor, que permite visualizar detalhes como o fluxo da memória e o tipo de gestão de variáveis (array e objetos sendo tipos referenciados) o [https://pythontutor.com/render.html#mode=edit](https://pythontutor.com/render.html%23mode=edit)

* [3] – Código disponibilizado pelos docentes via github, foi daqui retirada inspiração para o funcionamento do interpretCommands() com recurso a um do while. Esta implementação foi necessária devido a erros prévios.
  + [https://github.com/estsetubalipoo/TP\_07\_Exercicios/blob/master/src/main/java/Ex4.java](https://github.com/estsetubal-ipoo/TP_07_Exercicios/blob/master/src/main/java/Ex4.java)

* [4] – Docentes Bruno Silva e Paula Miranda. Os docentes foram consultados relativamente a algumas dúvidas e serviram como boas referências à resolução de alguns dos erros existentes.