Relatório: Implementação do Jogo Zumbicídio

# Introdução

Neste relatório, apresentamos o jogo "Zumbicídio", uma adaptação do jogo de tabuleiro "Zombicide" implementada em Java. O jogo se passa em uma cidade infestada de zumbis, onde o jogador assume o papel de um sobrevivente com a missão de eliminar todos os mortos-vivos enquanto explora o ambiente em busca de recursos para sobreviver.

O desenvolvimento seguiu princípios de Programação Orientada a Objetos, permitindo uma representação natural dos elementos do jogo como classes e objetos. A implementação foi estruturada nos seguintes **packages:** A lógica do jogo(modelo), a interface gráfica(interfaceGrafica), os controladores de interação(controle) e utilidades(util).

A estrutura geral do jogo consiste em um tabuleiro 10x10 que representa a cidade, onde o jogador navega entre células que podem conter paredes, baús com itens ou diferentes tipos de zumbis. O jogador pode mover-se, atacar inimigos com diferentes armas e usar itens para se curar. A condição de vitória é eliminar todos os zumbis do mapa, enquanto a derrota ocorre quando o jogador perde toda sua saúde.

# Estrutura do Projeto

O projeto foi organizado em quatro pacotes principais, cada um com responsabilidades específicas:

1. **modelo**: Classes que representam os dados e a lógica de negócio do jogo
2. interfaceGrafica: Classes da interface gráfica
3. **controle**: Classes que controlam o fluxo do jogo
4. **util**: Classes utilitárias de suporte

## Diagrama de Classes SimplificadoA screenshot of a computer AI-generated content may be incorrect.

# Aplicação de Conceitos de Orientação a Objetos

# Herança

A herança foi amplamente utilizada no projeto, permitindo a especialização de comportamentos específicos a partir de classes base. Exemplos notáveis incluem:

 A classe abstrata Personagem sendo estendida por Jogador e Zumbi, fornecendo atributos comuns como posição e saúde.

 A classe abstrata Zumbi sendo especializada em ZumbiComum, ZumbiRastejante, ZumbiCorredor e

ZumbiGigante, cada um com comportamentos específicos.

 A classe abstrata Item sendo estendida por Atadura, TacoBeisebol e Revolver, permitindo diferentes funcionalidades ao serem utilizados pelo jogador.

Esta abordagem permitiu reutilização de código e facilitou a implementação de novos tipos de personagens e itens.

## Polimorfismo

O polimorfismo foi essencial para tratar diferentes tipos de entidades de maneira uniforme:

Os diferentes tipos de zumbis são tratados como Zumbi em muitas partes do código, mas cada um responde diferentemente a ações como movimento e ataque.

O método moverEmDirecaoAoJogador() é sobrescrito em cada subclasse de Zumbi para implementar comportamentos específicos.

A interface gráfica renderiza diferentes ocupantes de uma célula sem precisar conhecer seus tipos específicos. O método aplicar() da classe Item comporta-se de maneira específica para cada tipo de item quando utilizado pelo jogador.

## Classes Abstratas

O uso de classes abstratas permitiu definir comportamentos comuns enquanto delegava implementações específicas às subclasses:

Personagem: Define atributos e métodos comuns a todos os personagens, mas deixa comportamentos específicos para Jogador e Zumbi.

Zumbi: Implementa comportamentos gerais de zumbis, mas delega comportamentos específicos para as subclasses.

Item: Define a interface para todos os itens, mas cada subclasse implementa seu próprio comportamento quando aplicado ao jogador.

## Encapsulamento

Implementamos encapsulamento através de modificadores de acesso apropriados:

Atributos como saude, posicaoX, e posicaoY são protegidos (protected) na classe Personagem, permitindo acesso apenas às subclasses.

Métodos públicos como getters e setters fornecem acesso controlado aos atributos privados.

Classes como Celula encapsulam seu estado interno, expondo apenas métodos necessários como

temParede() e estaOcupada().

## Exceções

Utilizamos tratamento de exceções para lidar com situações anormais, como:

Falha ao carregar arquivos de mapa (FileNotFoundException). Erros ao carregar imagens para a interface gráfica.

Validação de movimentos inválidos e ações proibidas.

## Padrões de Projeto

Implementamos vários padrões de projeto para melhorar a estrutura e a manutenção do código:

**Singleton**: Para o ImageLoader, garantindo que exista apenas uma instância que gerencia o carregamento de imagens.

**Factory Method**: Em MapaController para criar diferentes tipos de zumbis e itens.

## Composição

Implementamos composição de célula para item

**Controlador Principal (Jogo) e seus Componentes:** A classe Jogo atua como coordenadora central, instanciando e gerenciando componentes essenciais como Tabuleiro, Jogador e diferentes controladores. Estes componentes são criados dentro da própria classe e não possuem significado fora de seu contexto.

**Estrutura do Tabuleiro**: A classe Tabuleiro mantém uma matriz de objetos Celula que são instanciados durante sua construção. Cada célula representa uma unidade do espaço de jogo e só existe enquanto o tabuleiro existir.

**Elementos de Interface Gráfica:** Nos painéis da interface (como StatusPanel), componentes visuais são instanciados e gerenciados pelo painel principal, seguindo o padrão Composite do framework Swing.

**Elementos do Jogo:** Classes como Bau exemplificam composição ao gerenciar listas de itens e potencialmente um ZumbiRastejante escondido, sendo exclusivamente responsáveis por esses objetos**.** Passos para Executar o Programa.

## Requisitos

 Java Development Kit (JDK) 8 ou superior

 Um ambiente de desenvolvimento Java como Eclipse, IntelliJ IDEA ou NetBeans

## Instalação

1. Clone ou faça download do projeto
2. Importe-o como projeto Java em seu ambiente de desenvolvimento
3. Certifique-se de que a estrutura de pastas inclui resources/images/ e resources/maps/ com os respectivos arquivos
4. Compile e execute a classe TelaInicial.java que contém o método main()

## Como Jogar

### Tela Inicial

Ao iniciar o jogo, você verá a tela inicial com o título "ZUMBICÍDIO" Selecione a dificuldade (Fácil, Médio ou Difícil)

Clique em "Jogar" para iniciar o jogo normal ou "DEBUG" para ver todo o mapaA screenshot of a computer

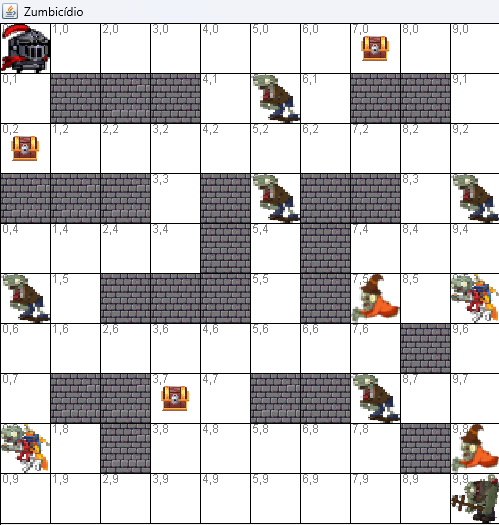
AI-generated content may be incorrect.

### Navegação

O jogador é representado por uma imagem no tabuleiro

Clique em uma célula adjacente (horizontal ou vertical) para mover-se

Você pode ver apenas células em linha reta até encontrar um obstáculo (parede, baú ou zumbi) As células não visíveis aparecem em cinza claro



### Combate

Ao mover-se para uma célula ocupada por um zumbi, inicia-se um combate Escolha como atacar (mãos, taco ou revólver) dependendo das armas disponíveis Observe o resultado do ataque e o contra-ataque do zumbi

Você pode tentar fugir movendo-se para uma célula vazia adjacente

A screenshot of a computer

AI-generated content may be incorrect.

### Itens

Ao mover-se para uma célula com um baú, você o abre automaticamente Você pode encontrar:

 Atadura: permite recuperar 1 ponto de saúde

 Taco de beisebol: melhora seus ataques corpo-a-corpo  Revólver: permite ataques à distância mais poderosos

 Zumbi rastejante: pode causar dano se você falhar no teste de percepção

A screenshot of a computer error message

AI-generated content may be incorrect.

### Ações

Use o botão "Curar" para utilizar uma atadura e recuperar saúde O botão "Sair" permite encerrar o jogo atual

A close up of a button

AI-generated content may be incorrect.

### Fim de Jogo

Vitória: Elimine todos os zumbis do mapa Derrota: Sua saúde chega a zero

Após o fim do jogo, você pode escolher "Reiniciar Jogo" ou "Novo Jogo"

A screenshot of a computer error message

AI-generated content may be incorrect.

# Dificuldades Encontradas

Durante o desenvolvimento do jogo, enfrentamos diversos desafios:

1. **Implementação da visibilidade**: Um dos maiores desafios foi implementar o sistema de visibilidade do jogador, que deveria permitir ver apenas células em linha reta até encontrar um obstáculo. Encontramos dificuldades em coordenar esta mecânica com o modo DEBUG.
2. **Mecânica de combate**: A implementação do sistema de combate, com as diferentes interações entre tipos de armas e zumbis, exigiu uma cuidadosa integração entre várias classes.
3. **Remoção de zumbis**: Enfrentamos um problema em que os zumbis não eram corretamente removidos do jogo após serem derrotados, o que exigiu uma revisão das classes CombateController e Jogo.
4. **Carregamento de imagens**: A integração das imagens com a interface gráfica apresentou desafios, especialmente para garantir que fossem carregadas corretamente e exibidas nas dimensões apropriadas.
5. **Leitura de mapas**: O processamento dos arquivos de mapa e a conversão das letras para os elementos correspondentes no jogo demandou ajustes para funcionar corretamente.
6. **Organização das classes**: Enfretamos um problema na orientação das classes para o desenvolvimento do programa.

# Conclusão

O desenvolvimento do jogo Zumbicídio foi uma experiência enriquecedora que nos permitiu aplicar conceitos de Programação Orientada a Objetos em um projeto real. A divisão do trabalho em dupla facilitou a implementação do projeto, com cada membro contribuindo em áreas específicas como modelagem de classes, interface gráfica e lógica de jogo.

A utilização de conceitos como herança, polimorfismo e encapsulamento tornou o código mais organizado e extensível. Além dos conceitos técnicos, o projeto nos proporcionou uma experiência valiosa em colaboração, resolução de problemas e tomada de decisões em grupo. Os desafios enfrentados serviram como oportunidades de aprendizado e aprimoramento de nossas habilidades de programação orientada a objetos.

# Propostas de Trabalhos Futuros

Para futuras melhorias e expansões do jogo Zumbicídio, propomos as seguintes possibilidades:

1. **Trilha sonora e efeitos sonoros**: Adicionar música de fundo e efeitos sonoros para aumentar a imersão.
2. **Mais tipos de zumbis**: Expandir a variedade de inimigos com novos tipos, habilidades e comportamentos.
3. **Sistema de pontuação**: Implementar um sistema de pontos e um ranking de melhores jogadores.
4. **Editor de mapas**: Desenvolver uma ferramenta que permita aos usuários criar e compartilhar seus próprios mapas.
5. **Narrativa e missões**: Adicionar uma história e objetivos específicos além de simplesmente eliminar todos os zumbis.
6. **Personalização de personagens**: Permitir que os jogadores escolham e personalizem seus personagens com diferentes habilidades e aparências.

O potencial para expansão é vasto, e estas propostas representam apenas algumas das possibilidades para levar o jogo a um novo patamar de jogabilidade e engajamento.

Feito por: Santiago Affonso e Caio Peter.