Relatório Técnico de Implementação: Projeto Ludo

MC322 - Programação Orientada a Objetos

24 de novembro de 2025

Introdução

Aqui detalhamos as mudançasna arquitetura, adições funcionais e decisões técnicas tomadas durante a implementação do projeto Ludo. Cada alteração é acompanhada de sua justificativa técnica.

1. Novas classes e adições de arquitetura

Pacote `game`

- **Classe `ConfigJogo` (Estática):** Persiste as configurações escolhidas na Cena de Menu (JavaFX) até que a Cena de Jogo seja carregada e o `MotorJogo` instanciado.

Pacote `controller`

- **Classe `ControladorMenu`:** Separação de responsabilidade. O `ControladorJogo` não deve gerenciar a lógica de configuração pré-jogo. O ajuste de escala garante responsividade da GUI em diferentes resoluções.
- **Classe `ControladorVitoria`:** Desacoplamento do fluxo de "Fim de Jogo". Permite que a tela de vitória seja uma cena independente.

2. Refinamentos do Core

Classe `MotorJogo`

- **Uso de Reflection em `iniciarNovoJogo`:** Elimina a necessidade de estruturas condicionais (`switch/case`) para criar tipos de jogadores. Aumenta a extensibilidade (ex: adicionar um `JogadorRede` futuro sem alterar o motor).
- **Deep Copy em `carregarJogo`:** Preserva a referência de memória do objeto `MotorJogo` que o `ControladorJogo` possui. Se a instância fosse substituída, a GUI perderia o vínculo com o jogo lógico.
- **Getters Adicionais (`getJogadores`, `getValorDadoAtual`):** Necessidade da camada de Visão (GUI) para renderizar o estado atual dos peões e do dado após cada atualização.

Classe `Tabuleiro`

- **Mapas Auxiliares (`indicesEntradaReta`, `casasBase`):** Otimiza o desempenho do cálculo de movimentação e evita iterações desnecessárias para encontrar casas especiais durante a renderização.
- **Métodos de Indexação (`getIndiceCircuito`, `isCasaDaRetaFinal`):** Permite que o Controlador mapeie objetos lógicos `Casa` para coordenadas visuais (X, Y) na tela sem que o Modelo conheça a GUI.

Classe `JogadorIA`

- **Método `melhorPeao()`:** Encapsula a inteligência da IA. Prioriza: 1. Sair da base; 2. Capturar; 3. Fugir; 4. Avançar.

3. GUI

Classe `ControladorJogo` (View-Controller)

- **Classe Interna `CellPos`:** Facilita o cálculo matemático de posicionamento no `GridPane` ou `Pane`, desacoplando a lógica de pixels.
- **Renderização Procedural (`desenharTabuleiroBase`, `desenharDado`):** Garante que os gráficos sejam vetorizados e não percam qualidade ao redimensionar a janela, além de facilitar a mudança dinâmica de cores.
- **Métodos de Animação (`gerenciarTurnoIA` com `PauseTransition`):** Melhoria de UX (Experiência do Usuário). Permite que o jogador humano visualize o dado rolado e o movimento da peça da máquina, que de outra forma seriam instantâneos.

```
classDiagram
    %% PACOTE GAME
    namespace game {
        class MotorJogo {
            - Tabuleiro tabuleiro
            - List~Jogador~ jogadores
            - Jogador jogadorAtual
            - EstadoJogo estado
            + iniciarNovoJogo(List~Class~, List~Cor~)
            + rolarDado()
            + tentarMoverPeao(Peao)
            + carregarJogo(int slot)
        }
        class Tabuleiro {
            - List~Casa~ casasCircuito
            - Map~Cor, Integer~ indicesEntradaReta
            + moverPeao(Peao, Casa)
            + getCasaDestino(Peao, int)
            + getIndiceCircuito(Casa) int
        }
        class Jogador {
            <<Abstract>>
            # List~Peao~ peoes
            + getPeoesValidos(int, Tabuleiro)
        }
        class JogadorIA {
            + fazerJogada(int)
            - melhorPeao(List~Peao~, int) Peao
        }
        class ConfigJogo {
            <<Static>>
            - List~Class~ tiposJogadores
            - List~Cor~ coresJogadores
            + configurar()
        }
        class Peao {
            - Cor cor
            - Casa casaAtual
            + voltarParaBase()
        }
        class Dado { + rolar() int }
    }

    %% PACOTE CONTROLLER
    namespace controller {
        class ControladorMenu {
            + initialize()
            - iniciarFluxoJxJ(int)
        }
        class ControladorJogo {
            - MotorJogo motorJogo
            - Map~Peao, Circle~ mapaPeoes
            + initialize()
            + handleRolarDado()
            - gerenciarTurnoIA()
            - desenharTabuleiroBase()
        }
        class ControladorVitoria {
            + configurarVencedor(String, String)
        }
```

```
61      }
62
63      %% RELACOES
64      MotorJogo *-- Tabuleiro
65      MotorJogo *-- Dado
66      MotorJogo o-- Jogador
67      Jogador <|-- JogadorIA
68      Jogador *-- Peao
69      ControladorMenu ..> ConfigJogo : Configura
70      ControladorJogo --> MotorJogo : Controla
```