ANÁLISE COMPARATIVA DE VERSÕES JOGO DE TABULEIRO

Aluno: Davi da Silva Fonteles
Curso: Ciência da Computação
Professor: Paulo Henrique Maia

Introdução

O objetivo deste relatório é realizar uma análise comparativa entre duas versões do projeto Jogo de Tabuleiro, desenvolvidas em Java, com foco em qualidade de código, estruturação, aplicação de princípios de orientação a objetos e uso de padrões de projeto. A primeira versão do código apresentava uma estrutura funcional, mas com diversas limitações quanto à modularidade, reutilização e manutenção. Já a segunda versão foi reestruturada com base em boas práticas de design de software, aplicando os padrões Strategy, Factory, Singleton e Facade, promovendo uma arquitetura mais limpa e escalável.

Resumo da Arquitetura Anterior

A versão inicial do projeto foi desenvolvida com base em herança e polimorfismo entre as classes JogadorGeral, JogadorSortudo, JogadorAzarado e JogadorNormal. No entanto, todo o controle de regras do jogo foi concentrado na classe Tabuleiro, especificamente dentro do método main, gerando:

- Alta complexidade ciclomatica (mais de 80, segundo análise anterior no SonarQube).
- Violação do princípio de responsabilidade única (SRP), com o main cuidando de lógica de fluxo, regras do jogo, entrada de dados e controle de estado.
- Baixo encapsulamento: diversos atributos públicos.
- Falta de reutilização e dificuldade de estender o sistema.

Essa estrutura tornava o código difícil de manter e pouco flexível para futuras modificações.

Resumo da Nova Arquitetura

Na nova implementação, o projeto foi completamente reestruturado em pacotes temáticos:

- estrategia/: encapsula o comportamento das casas especiais usando o padrão Strategy.
- fabrica/: utiliza o padrão Factory para instanciar jogadores e casas.
- Jogo: atua como Facade, concentrando o fluxo principal do jogo.
- Tabuleiro: agora segue o padrão Singleton.
- CasaX.java: cada tipo de casa é implementada como uma subclasse.
- JogadorX.java: mantém a herança e polimorfismo para os tipos de jogador.

Essa arquitetura favorece o baixo acoplamento, alta coesão e aderência ao princípio aberto/fechado (OCP).

Comparativo de Métricas e Qualidade

Métrica/Critério	Código Antigo	Novo Código
Total de classes	6 Modularizadas por função	20+ Modularizadas por função (por tipo de jogador, casa, estratégia, etc.)
Encapsulamento	Ruim(Atributos públicos)	Bom (Getters/setters usados)
Responsabilidade única	Violado (Tudo no main)	Cada classe tem função clara
Complexidade do método principal	Alta (Tinha **mais de 120 linhas**, **várias estruturas de decisão** (if, switch, for, while)	Baixa/moderada (A lógica foi **dividida entre várias classes** (ex: EstrategiaSorte, FabricaJogador, etc.)
Aplicação de padrões de projeto	Nenhum	Strategy, Factory, Facade, Singleton
Reutilização de código	Baixa (Regras duplicadas)	Alta (Estratégias reutilizáveis)
Boas práticas	Fraca (Nomeação inconsistente, código acoplado, baixa legibilidade)	Forte (Nomeação clara, modularização, organização e princípios de design)
Facilidade para testes	Baixa (Difícil testar isoladamente)	Alta (Componentes isolados e desacoplados)

Principais Code Smells Eliminados na Nova Versão

- God Class: Tabuleiro centralizava toda a lógica → resolvido com múltiplas classes e uso de Jogo como Facade.
- Long Method: o main gigante foi decomposto.
- Switch Smell: trocas de tipo de jogador foram substituídas por fábricas.
- Feature Envy: acesso direto a atributos dos jogadores foi substituído por métodos.
- Naming issues: nomes como JogarDados foram corrigidos conforme convenção.

Conclusão

A nova implementação do jogo de tabuleiro representa uma evolução significativa em termos de qualidade de código. A aplicação dos princípios da orientação a objetos e padrões de projeto contribuiu para um código mais modular, reutilizável, testável e escalável. Além de resolver diversos problemas encontrados na análise anterior, a nova estrutura também facilita futuras expansões do projeto.

A estrutura do projeto pode ser visualizada no link: https://www.figma.com/board/XjrrazXW2GLOpBhMlyNgCN/Sem-t%C3%ADtulo?node-id=0-1&t=zaEX9YWJ8clNCrUc-1 ou no diagrama de classes em anexo, que ilustra as relações entre as entidades principais e os padrões de design utilizados.