PROGRAMAÇÃO ORIENTADA A OBJETOS
PROJETO FINAL – 2025-2
REFATORAÇÃO DO STARTUP GAME

1. Descrição do Game

O Startup Game é uma simulação em turnos (rodadas) onde cada usuário gerencia uma startup. Em cada rodada, os jogadores podem tomar até N decisões (configurável) dentre opções como Marketing, Equipe, Produto, Investidores e Cortar Custos. As decisões impactam Caixa, Receita Base, Reputação e Moral da startup. Após o número total de rodadas, o jogo calcula um score final e apresenta o ranking.

2. Descrição do Projeto/Objetivo

Refatorar o jogo Startup Game aplicando boas práticas de Programação Orientada a Objetos e padrões de projeto. O código fornecido inicialmente é monolítico (arquivo único) e deve ser transformado em um projeto modular e testável, com camadas definidas, persistência de dados, padrões Strategy (obrigatório) e uso de Value Objects fornecidos. O projeto deve ser versionado em um repositório Git. O professor deverá ser incluído como colaborador no repositório. O link do projeto deve constar nos entregáveis. O professor fará o clone do repositório para testes das features e arguição do grupo.

3. Descrição do arquivo Main.java recebido

O arquivo Main.java contém duas classes no mesmo arquivo: public class Main e class Startup. Este arquivo já implementa o jogo no terminal, com decisões e pontuação. A classe Startup mantém o estado da startup e a classe Main contém o loop do jogo, lógica de aplicação das decisões e exibição dos resultados.

4. Detalhamento das Tarefas

As tarefas obrigatórias incluem:

- Separação de responsabilidades (cada classe em seu arquivo).
- Leitura de configurações do arquivo game.properties (total de rodadas e decisões por rodada).
- Persistência de dados utilizando H2 (salvar startups, rodadas e decisões aplicadas).
- Implementação do padrão Strategy para decisões.
- Uso obrigatório das classes VO fornecidas (Dinheiro, Percentual, Humor).
- Tratamento obrigatório de exceções.

As tarefas opcionais incluem (escolher 2 a 5):

- Implementar Observer (eventos do jogo com listeners).
- Implementar Command (undo/replay).
- Implementar State (máquina de estados).
- Criar relatórios avançados ou exportação CSV.
- Criar Bots (IA simples) para decisões automáticas.
- Testes JUnit e seed determinística para resultados reproduzíveis.

5. Detalhes da Estrutura Recebida no ZIP

O ZIP fornecido já contém a seguinte estrutura mínima com stubs (esboços) e classes VO prontas:

```
src/
  config/Config.java      (leitura de game.properties)
  model/Startup.java      (stub - entidade principal do jogo (startup))
  model/Deltas.java       (stub - encapsular as variações (custos, bonus, etc.))
  model/vo/Dinheiro.java (VO - Value Object para representar valores monetários)
  model/vo/Percentual.java            (VO pronto)
  model/vo/Humor.java                 (VO pronto)
  actions/DecisaoStrategy.java        (interface Strategy)
  actions/DecisaoFactory.java         (fábrica)
  actions/[estratégias].java          (stubs)
  persistence/DataSourceProvider.java (H2)
  persistence/[repositories].java     (stubs)
  engine/GameEngine.java              (stub)
  engine/ScoreService.java            (stub)
  ui/ConsoleApp.java                  (stub)
  Main.java                           (inicia ConsoleApp)
resources/
  game.properties    (já configurado com total.rodadas=8 e max.decisoes.por.rodada=3)
  schema.sql         (arquivo para criação das tabelas no H2)
Enunciado_Projeto.md (detalhes do projeto)
README.md            (como compilar e rodar)
```

6. Entregáveis

- Código-fonte completo no repositório Git.
  Versionamento no Git: O projeto deve ser versionado em um repositório Git. O professor será incluído como colaborador para verificar commits. A frequência, autoria e qualidade dos commits (mensagens claras, commits incrementais) serão avaliadas como parte da nota do projeto. Commits únicos ou ausência de histórico podem impactar negativamente a avaliação.
- Link do repositório Git fornecido ao professor.
- "schema.sql" completo com tabelas do H2.
- "README.md" com instruções de build, execução e configuração.
- "RELATORIO.md" indicando quais itens obrigatórios e opcionais foram implementados.
- Prints de tela/execução demonstrando funcionamento.
- (Opcional) Relatório ".csv" ou ".txt" com métricas e ranking final.

7. Rubrica de Avaliação

Critério	Peso
Arquitetura & POO (SRP/OCP/encapsulamento/exceções)	30%
Padrões (Strategy obrigatório + opcionais escolhidos)	25%
Persistência & Configuração (H2 + game.properties)	20%
Testes & Qualidade	15%
UX Console & Relatórios	10%
Bônus (Observer, Command, State, VO, Bots, Relatórios avançados)	até +10%
Controle de versão (Git) — frequência e autoria dos commits	Incluído na avaliação geral (impacta nota final)