FGV EMAp
ESCOLA DE MATEMÁTICA APLICADA

Prova 5 de Engenharia de Software
11 de Novembro de 2025

Nome: _____

Instruções Gerais

- Todas as questões da prova devem ser organizadas em **dois arquivos apenas**:
  1. Um módulo denominado `funcoes.py`, no qual deverão ser implementadas todas as funções desenvolvidas ao longo da prova;
  2. Um arquivo `main.py`, que atuará como **driver code**, responsável por importar o módulo `funcoes.py` e demonstrar o funcionamento das funções nele implementadas.
- O código deverá apresentar-se **claro, devidamente organizado e adequadamente documentado**. Apenas serão aceitas soluções que evidenciem legibilidade, coerência estrutural e rigor metodológico.

Questão Única

Você está projetando um módulo de **processamento de tarefas** em uma plataforma de análise de dados empresariais. Cada tarefa (por exemplo, *importar dados*, *gerar relatório*, *enviar alertas*) pode envolver diferentes comportamentos de execução, validação e registro de operações. O objetivo é projetar um sistema **flexível, extensível e com baixo acoplamento**, capaz de:

- incluir novos tipos de tarefa sem alterar o código existente;
- manter um **histórico** de execuções ou cálculos;
- permitir, se possível, **configuração dinâmica** do comportamento de cada operação.

(a) Escolha e Justificativa – 4 pontos

Você deve **justificar qual combinação de padrões** usará para resolver o problema:

1. **Strategy + Template Method**, ou
2. **Chain of Responsibility + Command**.

A justificativa deve abordar:

- O **motivo da escolha** dos padrões (quando e por que usá-los neste contexto);

- Como a combinação escolhida **reduz o acoplamento** e **aumenta a extensibilidade** do sistema;
- Descreva, de forma comparativa, o que mudaria em sua solução caso tivesse optado pela outra combinação de padrões. Analise o que seria potencialmente melhor ou pior em termos de clareza, flexibilidade e complexidade do código.

(b) Implementação mínima – 6 pontos

Não há solução única — a correção considerará **coerência entre escolha e código**.

Se escolher *Strategy + Template Method*:

No arquivo `funcoes.py`:

1. Crie uma classe abstrata `TaskTemplate` com o método `execute()` (template) e três passos:
   - `load_data()`
   - `process()` (método abstrato)
   - `summarize()`
2. Crie duas subclasses concretas (`ImportTask`, `ReportTask`) que redefinam o método `process()` de modo distinto.
3. Crie uma classe `TaskStrategyContext` que receba dinamicamente uma *estratégia de processamento* (objeto com método `run(data)`), permitindo trocar o comportamento em tempo de execução.

No arquivo `main.py`:

- Demonstre a execução de duas tarefas com o Template Method;
- Troque a estratégia de execução de uma das tarefas dinamicamente;
- Mostre a saída de console com os resultados de cada abordagem.

Se escolher *Chain of Responsibility + Command*:

No arquivo `funcoes.py`:

1. Defina uma classe abstrata `Handler` com o método `handle(request)` e uma referência ao próximo handler;
2. Implemente duas subclasses (`ValidationHandler`, `ExecutionHandler`) que processam ou repassam a requisição;

3. Crie uma classe `TaskCommand` que encapsula uma operação, contendo os atributos `handler_name`, `timestamp` e `result`;
4. Mantenha uma lista `history` de comandos executados.

No arquivo `main.py`:

- Monte uma cadeia de handlers que simule a execução de uma tarefa de análise;
- Execute três requisições diferentes;
- Exiba o histórico de comandos processados com o nome do handler e o resultado.