Mini EP 4 - Padrões e Contextos

Neste EP, trabalharemos com padrões de O.O. e alguns contextos de aplicação. Você deve descrever contextos de aplicação para padrões. Sigam o exemplo dado e escrevam os contextos que vocês pensarem **para dois padrões**. O exemplo dá um contexto e um padrão que pode ser utilizado. Vocês podem fazer:

- dois contextos, cada um com um padrão
- um contexto que combine com dois padrões diferentes, desde que fique claro qual o uso de cada padrão

O que você deve fazer?

Submeter um PDF contendo a descrição do(s) contexto(s) com os padrões utilizados, conforme exemplificado a seguir. A escolha dos padrões é livre, contendo apenas uma restrição: o padrão Singleton não será aceito, pois seu uso é desencorajado na comunidade de engenharia de software.
Por ser um mini EP, **pseudo-código não é obrigatório,** mas será muito bem-vindo**.**

Exemplo

Contexto

Você está desenvolvendo uma plataforma de apoio ao ensino que oferece uma gama de recursos, como submissão de tarefas e registro de presença. Em particular, ao final do semestre, o professor calcula a média de cada aluno usando esta plataforma. Para facilitar a vida, alguns critérios comuns já são oferecidos, como médias simples e ponderada entre as notas das tarefas e um limite inferior de presença. Além disso, a plataforma oferece a possibilidade do professor definir algum critério além dos prontos, que ficará disponível a todos os demais professores da plataforma.

Problema

Como escrever um código que seja facilmente extensível para aceitar novos critérios de aprovação?

Solução

Faça cada critério de aprovação em uma classe diferente que implemente a mesma interface `CriterioDeAprovacao` e faça com que o código cliente `AprovadorDeAlunos` possa definir qual ***estratégia*** de `CriterioDeAprovacao` será utilizado. [padrão Strategy]

Código (em Kotlin)

```
interface CriterioDeAprovacao {
    fun avalia(aluno: Aluno): Status
}

class AprovadorDeAlunos {
    private var criterio: CriterioDeAprovacao = MediaSimples()
    private var alunos: Set<Aluno>

    fun redefineCriterio(novoCriterio: CriterioDeAprovacao) {
        criterio = novoCriterio
    }

    fun fechaNotas() = alunos.forEach { aluno ->
        val status = criterio.avalia(aluno)
        // ...
    }
}

class MediaSimples : CriterioDeAprovacao {
    override fun avalia(aluno: Aluno): Status { /* … */ }
}

class MediaPonderada : CriterioDeAprovacao {
    override fun avalia(aluno: Aluno): Status { /* … */ }
}
```