

PUC Minas

Curso: Ciência da Computação

Disciplina: Engenharia de Software II

Exercício Resolvido: Sistema de Gerenciamento de Tarefas com Design Patterns

Neste exercício resolvido, apresento um **Sistema de Gerenciamento de Tarefas (Task Manager)** que demonstra a aplicação prática de três padrões de design fundamentais:

- **Padrão Criacional:** Factory Method
- **Padrão Estrutural:** Decorator
- **Padrão Comportamental:** Observer

O sistema permitirá que os usuários criem, visualizem e gerenciem diferentes tipos de tarefas, adicionem recursos extras às tarefas e implementem um sistema de notificações quando o status das tarefas for alterado.

Design Patterns Implementados

1. Factory Method (Padrão Criacional)

Será usado para criar diferentes tipos de tarefas (Pessoal, Trabalho, Estudo) sem expor a lógica de criação ao cliente.

2. Decorator (Padrão Estrutural)

Permitirá adicionar funcionalidades extras às tarefas de forma dinâmica, como prioridade alta, etiquetas coloridas e datas de vencimento.

3. Observer (Padrão Comportamental)

Implementará um sistema de notificações onde vários observadores (como notificação na tela, email simulado, log) serão notificados quando o status de uma tarefa mudar.

Documentação de Instalação

Requisitos

- Um navegador web moderno (Chrome, Firefox, Edge, Safari)
- Servidor web local (opcional) - pode ser Apache, Nginx, ou simplesmente o Live Server do VS Code

Instalação e Execução

Método 1: Usando Live Server no Visual Studio Code

- **Instalar o Visual Studio Code:**
 - Baixe e instale o Visual Studio Code.
- **Instalar a extensão Live Server:**
 - Abra o VS Code
 - Vá para a aba de extensões (ou pressione `Ctrl+Shift+X`)
 - Pesquise "Live Server" (autor: Ritwick Dey)
 - Clique em "Install"
- **Preparar os arquivos do projeto:**
 - Crie uma pasta para o projeto (ex: designpatterns)
 - Crie a estrutura de pastas conforme descrito:

Estrutura de arquivos

```
/designpatterns/  
├── index.html  
├── css/  
│   └── styles.css  
└── js/  
    ├── app.js  
    ├── models/  
    │   ├── task.js  
    │   └── notification.js  
    ├── patterns/  
    │   ├── factory.js  
    │   ├── decorator.js  
    │   └── observer.js  
    └── ui/  
        └── ui-controller.js
```

- **Executar a aplicação:**
 - Abra a pasta do projeto no VS Code
 - Clique com o botão direito no arquivo `index.html`
 - Selecione "Open with Live Server"
 - O navegador será aberto automaticamente com a aplicação rodando

Método 2: Usando um servidor web existente

- **Preparar os arquivos do projeto:**
 - Crie a mesma estrutura de pastas e arquivos descrita acima
- **Copiar para o diretório do servidor web:**
 - Copie toda a pasta `task-manager` para o diretório do seu servidor web
 - Por exemplo, para Apache: `/var/www/html/designpatterns/`
- **Acessar a aplicação:**
 - Abra um navegador e acesse a URL correspondente
 - Por exemplo: `http://localhost/designpatterns/`

Método 3: Abrindo diretamente no navegador (para demonstração rápida)

- **Preparar os arquivos do projeto:**
 - Crie a mesma estrutura de pastas e arquivos descrita acima
- **Abrir o arquivo HTML:**
 - Navegue até a pasta do projeto no seu explorador de arquivos
 - Dê um duplo clique em `index.html` para abri-lo no seu navegador padrão
 - Nota: Alguns recursos não funcionarão desta forma, mas o básico funcionará para demonstração dos padrões

Como usar a aplicação

- **Criar uma tarefa (Factory Method):**
 - Preencha o título e descrição no formulário "Criar Tarefa"
 - Escolha o tipo de tarefa (Pessoal, Trabalho ou Estudo)
 - Clique em "Criar Tarefa"
- **Adicionar recursos extras às tarefas (Decorator):**
 - Selecione uma tarefa existente no menu suspenso
 - Marque as opções desejadas (Alta Prioridade, Etiqueta Colorida, Data de Vencimento)
 - Configure as opções adicionais se necessário
 - Clique em "Aplicar Recursos"
- **Gerenciar notificações (Observer):**
 - Marque ou desmarque os tipos de observadores que deseja ativar
 - Altere o status de uma tarefa clicando nos botões "Em Andamento" ou "Concluída"
 - Observe as notificações que aparecem nos diferentes canais

Conclusão

Este exercício demonstra a implementação prática de três padrões de design fundamentais em uma aplicação web:

- **Factory Method (Criacional):** Para criar diferentes tipos de tarefas
- **Decorator (Estrutural):** Para adicionar funcionalidades às tarefas de forma flexível
- **Observer (Comportamental):** Para implementar um sistema de notificações

A aplicação é completamente funcional e pode ser usada como base para o aprendizado de padrões de design ou expandida para incluir funcionalidades adicionais.

Exercício em dupla (5 pontos): Entrega 13/05

1a parte: Personalizando a aplicação existente

- **Adicionar novos tipos de tarefa:**
 - Crie uma nova classe de tarefa em `js/models/task.js` que estenda a classe `Task`
 - Adicione um novo case no método `createTask()` do `TaskFactory`
 - Adicione a nova opção no select HTML
 - **Criar novos decoradores:**
 - Crie uma nova classe em `js/patterns/decorator.js` que estenda `TaskDecorator`
 - Sobrescreva os métodos necessários para adicionar o novo comportamento
 - Adicione a interface do usuário correspondente em `index.html`
 - Atualize `applyDecorators()` em `ui-controller.js`
 - **Adicionar novos observadores:**
 - Crie uma nova classe em `js/patterns/observer.js` que estenda `Observer`
 - Implemente o método `update()`
 - Adicione a interface do usuário em `index.html`
 - Atualize `initObservers()` e `updateObservers()` em `ui-controller.js`
-

2a parte: Criar mais funcionalidades que abordem mais 3 Design Patterns, sendo 1 criacional, 1 estrutural e 1 comportamental.

Observação: caso os alunos preferam, podem criar outra aplicação web mas que contenham os 3 design patterns.