Projeto PI - Padrão Observer

Nome do padrão: Observer

Resumo da implementação

Escolhemos o padrão Observer para o projeto porque ele é ideal para cenários em que vários objetos

Motivo da escolha

- Baixo acoplamento: o sujeito não precisa saber quem são os observadores — só notifica uma lista d
- Flexibilidade: novos observadores (ex.: SMS, e-mail, push) podem ser adicionados sem modificar a
- Coesão de responsabilidade: o sujeito mantém o estado; os observadores lidam com as ações reação

Contrato / comportamento esperado

- Inputs: eventos de mudança no estado do menu/produto (ex.: disponibilidade, promoção).
- Outputs: notificações disparadas aos observadores cadastrados.
- Erros: registro nulo/duplicado de observadores deve ser tratado de forma segura.

Classes envolvidas (sugestão de mapeamento para o projeto atual)

- Subject / Observable (interface)
  - Responsabilidade: permitir registrar, remover e notificar observadores.
  - Exemplo de papel no projeto: `MenuService` pode delegar um `MenuSubject` para gerenciar assinat

- Observer (interface)
  - Responsabilidade: definir método `update(...)` que será chamado quando ocorrer evento.
  - Exemplo de classes concretas: `ClientObserver`, `PromotionObserver`, `InventoryObserver`.

- MenuSubject (implementação concreta do Subject)
  - Responsabilidade: manter lista de observadores; disparar eventos quando há alteração de menu/pr

- ClientObserver (implementação concreta do Observer)
  - Responsabilidade: receber a notificação e acionar `ClientService` (ou `NotificationService`) pa

- NotificationService (serviço existente ou novo)
  - Responsabilidade: encapsular lógica de envio de e-mail / push / SMS.

Mapeamento com classes já presentes no projeto

- `MenuService` (src/main/java/com/basilios/basilios/service/MenuService.java) — papel de coordenad
- `Client` / `Usuario` (src/main/java/com/basilios/basilios/model/Client.java, Usuario.java) — repr
- `ClientService` / `UsuarioService` (src/main/java/com/basilios/basilios/service/) — podem ser cor
- `Product` (src/main/java/com/basilios/basilios/model/Product.java) — mudanças de disponibilidade
- `NotificationService` (sugerir criação em `service/`) — envia mensagens reais para os observadore

Sugestão rápida de implementação (códigos de exemplo)

1) Interface Subject (Observable)

```java
public interface Subject {
    void registerObserver(Observer o);
    void removeObserver(Observer o);
    void notifyObservers(String event, Object payload);
}
```

2) Interface Observer

```java
public interface Observer {
    void update(String event, Object payload);
}
```

```

3) Implementação concreta (MenuSubject)

```java
import java.util.concurrent.CopyOnWriteArrayList;

public class MenuSubject implements Subject {
    private final CopyOnWriteArrayList<Observer> observers = new CopyOnWriteArrayList<>();

    @Override
    public void registerObserver(Observer o) {
        if (o != null && !observers.contains(o)) observers.add(o);
    }

    @Override
    public void removeObserver(Observer o) {
        observers.remove(o);
    }

    @Override
    public void notifyObservers(String event, Object payload) {
        for (Observer o : observers) {
            try {
                o.update(event, payload);
            } catch (Exception ex) {
                // log e continue
            }
        }
    }

    // método chamado quando o menu muda
    public void onMenuChanged(String event, Object payload) {
        notifyObservers(event, payload);
    }
}
```

4) Exemplo de Observer (ClientObserver)

```java
public class ClientObserver implements Observer {
    private final Long clientId;
    private final NotificationService notificationService;

    public ClientObserver(Long clientId, NotificationService notificationService) {
        this.clientId = clientId;
        this.notificationService = notificationService;
    }

    @Override
    public void update(String event, Object payload) {
        // forma simples de notificação — delega ao serviço
        notificationService.notifyClient(clientId, "Evento: " + event + " - " + String.valueOf(payl
    }
}
```

Trechos de código "printados" (pelo menos 2) — exemplos acima

- Print 1: Interface `Subject` (veja trecho acima)
- Print 2: Implementação `MenuSubject` (veja trecho acima)

Observações finais
- Para integrar, sugiro criar os arquivos em `src/main/java/com/basilios/basilios/infra/observer/`
```

- Se desejar, posso gerar os arquivos Java completos já colocados no projeto e executar uma compila

Assinatura da equipe

Todos os membros devem enviar a mesma cópia deste PDF; este arquivo foi gerado automaticamente para