Projeto Padoca

Idealizadores:

Lisiane Hoffmeister
Lucca Mota Sindeaux

O tema do projeto será um sistema de delivery de tortas. A ideia é a execução de serviço que permite que o usuário confeccione tortas doces e salgadas personalizadas. Tendo isto em mente, visualizamos a criação de uma superclasse abstrata chamada “Torta”, cujas herdeiras concretas seriam as macro-classificações “Doce” e “Salgada”, que o conjunto se torna nosso objeto.

Como padrão estrutural, optamos pelo Decorator. Já que ele permite adicionar uma função extra dinamicamente à funcionalidade principal sem aumentar a complexidade do código. No nosso projeto a ideia de Decorator é a adição de embalagens especiais ou cartas para as tortas entregues. No caso específico de adicionar a mensagem com o decorator, a mensagem desejada será escrita dentro da função de criar o cartão.

Para que o Decorator funcione é necessário uma herança entre a superclasse do objeto concreto e a classe Decorator que implementa o método construtor. Ao chamarmos as funções do modelo para dentro de nossa classe de decorar tortas, criamos este link.

Ou seja, para a implementação correta: as funcionalidades serão escritas no Decorator com os valores já adicionados durante o processo de criação de torta. Aqui está a leveza do código, pois todo este processamento é feito pelo objeto concreto, seus resultados são copiados e adiciona-se a eles as funcionalidades únicas do Decorator, neste caso, decorar a torta com cartas ou embalagens únicas.

O decorator é um componente extensionista de nossas classes, sendo inclusive totalmente opcional como um adicional ao objeto concreto, o arquivo sequer será chamado se durante o uso da aplicação o cliente optar por não incluir as decorações extras.

```
    @Override
    public void preparar() {
        tortaDecorada.preparar();
    }

    @Override
    public void exibirDetalhes() {
        tortaDecorada.exibirDetalhes();
    }

    @Override
    public String getTamanho() {
        return tortaDecorada.getTamanho();
    }

    @Override
    public String getCamada() {
        return tortaDecorada.getCamada();
    }

    @Override
    public String getRecheio() {
        return tortaDecorada.getRecheio();
    }

    @Override
    public String getCobertura() {
        return tortaDecorada.getCobertura();
    }
}
```

```
package com.doceria.padoca.decorator;

import com.doceria.padoca.model.Torta;

public abstract class TortaDecorator extends Torta {
    protected Torta tortaDecorada;

    public TortaDecorator(Torta tortaDecorada) {
        this.tortaDecorada = tortaDecorada;
    }
```

```java
public class CartaoPresente extends TortaDecorator {

	public CartaoPresente( Torta torta ) {
		super(torta);
	}

	@Override
	public void preparar() {
		super.preparar();
		System.out.println( "Adicionando cartão de presente." );

		Scanner scanner = new Scanner(System.in);
		System.out.println("Digite a mensagem que deseja no cartão:");
		String mensagem = scanner.nextLine();

		System.out.println("Mensagem no cartão: \"" + mensagem + "\"");
	}
```