Engenharia de Software II
Entrega: 08/05/2024
Valor: 5 pontos

EXERCÍCIOS DE IMPLEMENTAÇÃO DE PADRÕES (em dupla)

1. CRIACIONAIS (Abstract Factory)

Considere os seguintes conceitos do mundo real: pizzaria, pizzaiolo, pizza, consumidor. Considere ainda que em uma determinada pizzaria, dois pizzaiolos se alternam. Um deles trabalha segundas, quartas e sextas e só sabe fazer pizza de calabresa (queijo + calabresa + tomate), o outro trabalha terças, quintas e sábados e só sabe fazer pizza de presunto (queijo + presunto + tomate). A pizzaria fecha aos domingos. Tente mapear os conceitos acima para o padrão Abstract Factory (hierarquia de fábricas, hierarquia de produtos, cliente) e implemente um programa que receba uma data como parâmetro (formato dd/mm/yyyy) e imprima os ingredientes da pizza que é feita no dia ou, se a pizzaria estiver fechada, informe isso na tela. Agora imagine que a pizzaria agora faz também calzones (novamente, de calabresa ou presunto). Complemente a solução com mais este componente.

(Builder)

Na cadeia de restaurantes *fast-food PatternBurgers* há um padrão para montagem de lanches de crianças. O sanduíche (hambúrguer ou *cheeseburger*), a batata (pequena, média ou grande) e o brinquedo (carrinho ou bonequinha) são colocados dentro de uma caixa e o refrigerante (coca ou guaraná) é entregue fora da caixa. A classe abaixo é dada para representar o pedido de um consumidor:

```
import java.util.*;

public class Pedido {
	private Set<String> dentroDaCaixa = new HashSet<String>();
	private Set<String> foraDaCaixa = new HashSet<String>();

	public void adicionarDentroDaCaixa(String item) {
		dentroDaCaixa.add(item);
	}

	public void adicionarForaDaCaixa(String item) {
		foraDaCaixa.add(item);
	}

	public String toString() {
		StringBuffer buffer = new StringBuffer();
		buffer.append("Seu pedido:\n");
		buffer.append("Dentro da caixa:\n");
		for (String item : dentroDaCaixa) buffer.append("\t" + item + "\n");
		buffer.append("Fora da caixa:\n");
		for (String item : foraDaCaixa) buffer.append("\t" + item + "\n");
		buffer.append("\nTenha um bom dia!\n\n");
		return buffer.toString();
	}
}
```

Neste caso, o padrão *Builder* pode ser usado para separar as tarefas do atendente e do funcionário que monta o pedido. Somente este último sabe como montar os pedidos segundo os padrões da empresa, mas é o atendente quem lhe informa quais itens o consumidor pediu.

Implemente a simulação do restaurante *fast-food* descrita acima utilizando o padrão *Builder* e escreva uma classe cliente que pede um lanche ao atendente, recebe-o do outro funcionário e imprime o pedido.

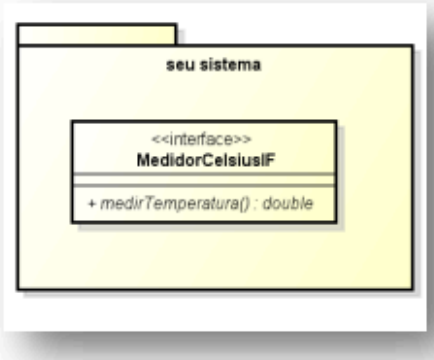
2. Estruturais (Adapter)

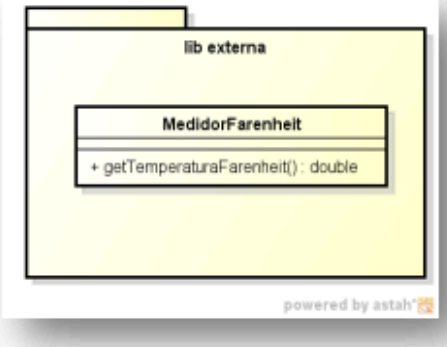
Suponha que você está implementando o sistema de uma estação meteorológica brasileira cuja medida utilizada para temperatura é *Celsius*. Uma das interfaces do seu sistema é a interface *MedidorCelsiusIF*, cujo método *medirTemperatura()* retorna sempre um valor double referente à temperatura na **medida Celsius**. O gerente do projeto decidiu comprar o termômetro no exterior, e, portanto este equipamento só mede no **formato Farenheit**. Abaixo veja a classe *MedidorFarenheit* da lib adquirida e o diagrama de classes do projeto.

P.S. 1: a classe *MedidorFarenheit* da lib adquirida já está compilada – não podemos alterá-la.

P.S. 2: **°C = (°F − 32) ÷ 1,8** ➔ fórmula para converter a temperatura de Farenheit para Celsius

```java
import java.util.Random;
public class MedidorFarenheit {
	public double getTemperaturaFarenheit(){
		return new Random().nextDouble(); //simulando o termometro
	}
}
```





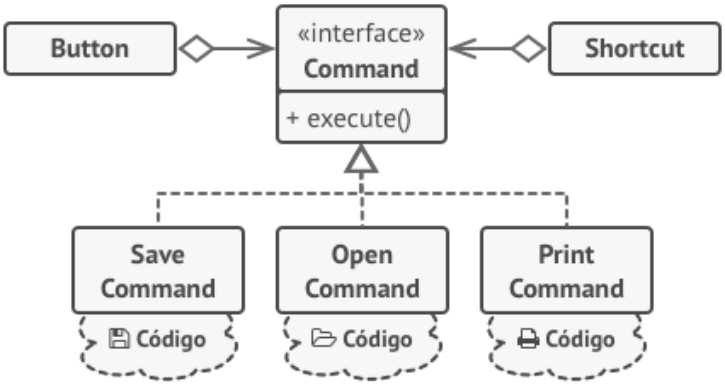
a) Crie um Adaptador (***AdaptadorFarenheitParaCelsiusObjectAdapter***) utilizando a abordagem ***Object Adapter*** para utilizar a biblioteca comprada.

b) Crie um Adaptador (***AdaptadorFarenheitParaCelsiusClassAdapter***) utilizando a abordagem ***Class Adapter*** para utilizar a biblioteca comprada.

c) Explique a razão de ambos os adaptadores implementados respeitar o princípio "aberto/fechado".

3. **Comportamentais (Command)**

O *Command* nos obriga a separar o que é código de execução de um comando de outros códigos como interface, controle de sessão, etc...



Faça uma implementação para o exemplo acima.