**Exercícios**de Avaliação 4

**Disciplina de Projeto e Arquitetura de Software**

**Alunos**

Alberto Rocha Pinalli

Luiz Guerra

Vitor Demenighi

Neste exercício de avaliação os alunos deverão explorar os padrões de projeto trabalhados nos últimos dias:

**- Singleton**

**- Observer**

**- Fachada**

**- Factory**

**- Strategy.**

O problema a ser resolvido continua o mesmo do exercício proposto, o e-commerce.

Sugestões de aplicação destes padrões para este problema:

1. **singleton**, para garantir um único carrinho de compras para um cliente;
2. **observer**, atualização do carrinho a cada produto selecionado pelo cliente;
3. **strategy e factory** para implementar as diferentes formas de pagamento e o acesso ao cliente a estas informações (cartão de débito, cartão de crédito, boleto bancário e paypal).
4. **fachada**; classe a ser definida na camada de negócios como um único ponto de acesso da camada de apresentação a ela.

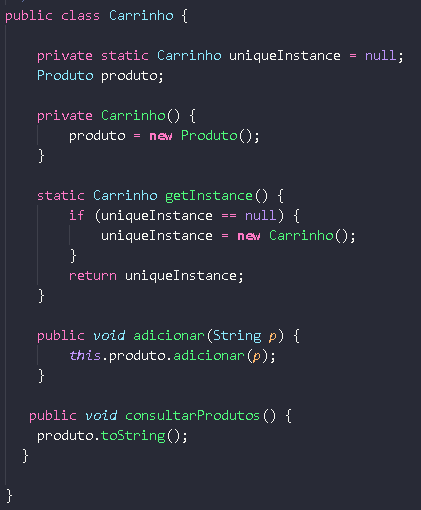
**Cada grupo deve entregar um arquivo texto com uma descrição e screen shots das classes/métodos implementados para cada padrão de projeto**. **Este documento deve conter o link do git para acesso ao projeto.**

**Padrão Singleton**

**Classe Carrinho**

Tem por objetivo garantir a criação de uma única instancia doobjeto carrinho através do método getInstance()

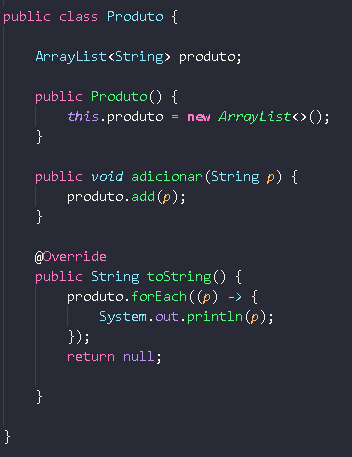
Possui os métodos adicionar e consultar produtos no carrinho e o método construtor



**Classe Produto**

Tem por finalidade criar um ArrayList de produtos para inserir no Carrinho

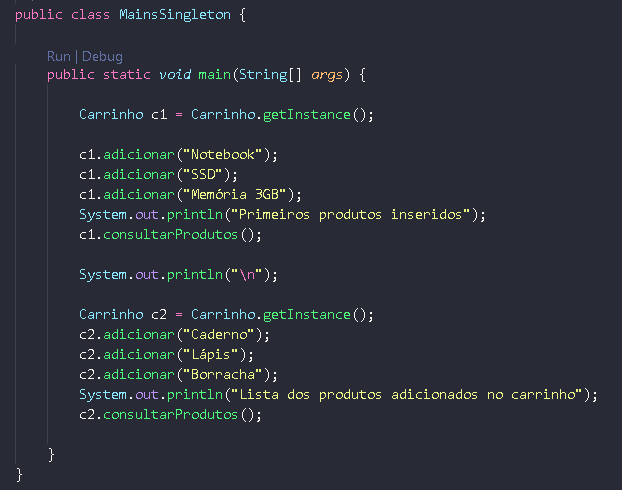
Possui o método construtor, adicionar produtos e o ToString



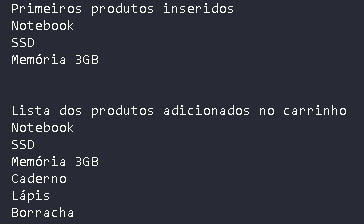
**Classe MainSingleton**

Tem como objetivo instanciar o Carrinho, adicionar produtos no carrinho chamando o método adicionar

E a apresentação do resultado ao usuário



**Console**

****

**Padrão Oberver**

**Classe Subject**

O objetivo da classe é permitir ao usuário escolher um produto para adicionar ao carrinho

Possui a declaração de um ArrayList para inserção de produtos através do método adicionar

Possui o método construtor e o método consultar para a visualização de produtos no carrinho

Possui o método inseriu implementado da interface Observer, método que atualiza as informações quando há alteração no carrinho o método inseriu é chamado pelo método notificarObserver da classe ConcreteObserver



**Classe ConcreteObserver**

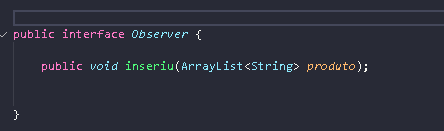
É a classe observada, possui um ArrayList de produtos e um ArrayList de observers

Além do método construtor, possui os métodos adicionar, responsável por adicionar os produtos no carrinho, o método addObserver, responsável por adicionar um observer, o método removeObserver responsável pela remoção de um observer e o método notificarObserver que é chamado pelo método inseriu da classe Subject e o método toString



**Interface Observer**

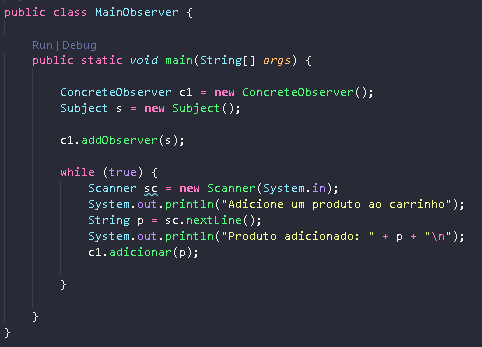
Responsável por garantir que a classe Subject implemente o método inseriu, responsável por notificar a classe ConcreteObserver sobre alterações no carrinho



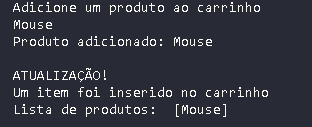
**Classe MainObserver**

Responsável por instanciar as classes ConcreteObserver e Subject para poderem criar o objeto c1 e inserir produtos no carrinho

Possui o método addObserver para um observer poder informar das alterações realizadas no carrinho



**Console**

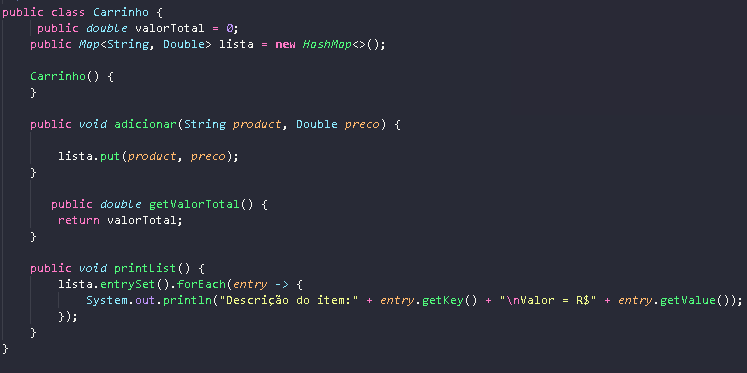
****

**Padrão Strategy**

**Classe Carrinho**

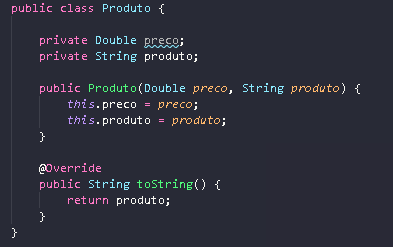
Possui um Map de produtos e valor

Além do método construtor, possui os métodos adicionar, responsável por adicionar os produtos no carrinho e valor, o método getValorTotal, responsável por retornar o valor total do carrinho, o método printList responsável pela descrição dos itens e seus valores

****

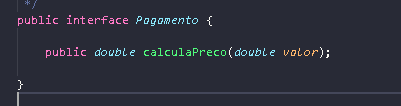
**Classe Produto**

Além do método construtor, possui o toString e as variáveis preco e produto



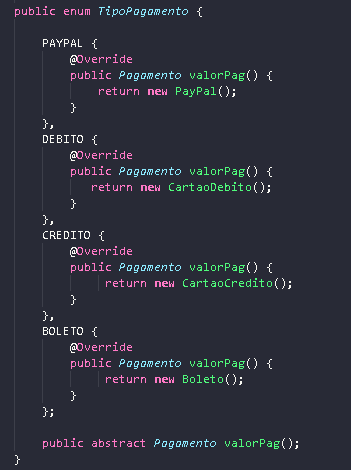
**Interface Pagamento**

Responsável por assegura que as classes CartaoDebito, cartaoCrediro, Paypal e Boleto implementem o método calculaPreco



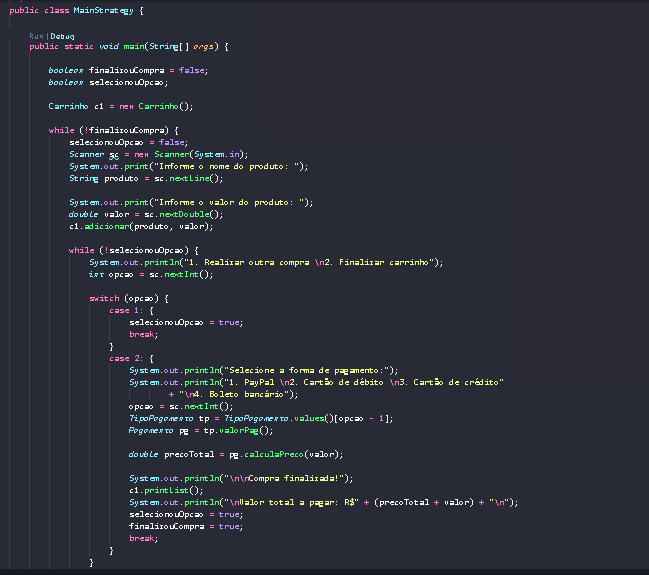
**Enum tipoPagamento**

Responsável por implementar os métodos próprios implementados pelo método abstrato Pagamento valorPag()

****

**Classe MainStrategy**

Responsável por instanciar as classes Carrinho para podere criar o objeto c1 e inserir produtos no carrinho chamando os métodos responsáveis para adicionar e apresentar os itens do carrinho ao usuario

****