ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO
Departamento de Engenharia de Computação e Sistemas Digitais

PCS 3111 - Laboratório de Programação Orientada a Objetos para Engenharia Elétrica

2023

Aula 06 – Herança e Polimorfismo I

Atenção

- Código inicial a ser usado na resolução dos exercícios encontra-se **disponível no e-Disciplinas**.
- Submeta um arquivo comprimido (faça um ".zip" – **não pode ser ".rar"**) colocando apenas os arquivos ".cpp" e ".h". Não crie pastas no "zip".

Exercício 01

Considere a classe **Produto**, já fornecida[1]. Implemente agora a classe **Combo**, filha de **Produto**, que representa um produto formado pela combinação de 2 ou mais produtos. Nela, o preço é calculado através da somatória dos preços de cada **Produto**, seguido pela aplicação do desconto. Por exemplo:

Um combo com
- um **Produto A** de **50 reais**,
- um **Produto B** de **30 reais** e
- um **desconto de 30%** (desconto = 0.3)

terá preço de (50+30)*(1-0.3) = 80*0.7 = **56 reais**

Essa classe recebe no construtor:
- o nome do Combo;
- o vetor dos produtos desse combo;
- a quantidade de produtos no vetor;
- o desconto aplicado à soma dos preços dos produtos do combo.
- Note que para criar um combo é necessário chamar o construtor do produto indicando valor 0 e então atualizar o preço de acordo com o vetor de produtos recebido. **Observe também que para alterar um atributo de uma classe mãe é necessário que ele tenha uma visibilidade específica.**

O desconto é um double que deve pertencer ao intervalo de [0,1], onde 0 significa ausência de desconto e 1 significa 100% de desconto.

A seguir são apresentados os métodos específicos a essa classe.

[1] Na aula que vem será explicado o significado da palavra `virtual` no destrutor de **Produto**. Mantenha o `virtual` mas não se preocupe com isso.

```
Combo(string nome, Produto **p, int quantidadeDeProdutos, double desconto);
~Combo();
Produto **getProdutos();
int getQuantidadeDeProdutos();
double getDesconto();
```

- Altere a visibilidade dos atributos (private ou protected) necessários da classe **Produto**, fornecida.
- Implemente os métodos a seguir::
  - Os métodos abaixo devem retornar os valores fornecidos no construtor
    - **getProdutos()
    - getQuantidadeDeProdutos()
    - getDesconto()
  - getPreco() deve retornar o preço final do Combo conforme descrito no início do enunciado. Não defina esse método na classe Combo.
  - Não destrua os produtos no destrutor.

Por exemplo, para o **Combo** com nome “Combo 2” com **20% de desconto** contendo:

- uma “Pipoca Grande” custando 20 reais e
- uma “Coca-Cola 1L” custando 15 reais

A saída de imprimir() esperada é:

Produto: `Combo 2 - 28 reais`

- Implemente a função `teste1` (em teste.cpp) da seguinte maneira:

1. Crie os produtos:
   - `Pipoca Grande - 20 reais`
   - `Coca-Cola 1L - 15 reais`
2. Crie um vetor com tamanho 2 e guarde os produtos na ordem acima, isto é, {Pipoca Grande, Coca-Cola 1L}.
3. Crie o Combo “Combo 2”.
   - com os 2 produtos acima e
   - com 20% de desconto (desconto = 0.2)
4. Imprima o Combo criado acima.
5. Delete o Combo e os produtos criados

Exercício 02
Implemente a classe **Pedido**, correspondente a uma lista de produtos que um usuário deseja comprar. Cada **Pedido** contém um vetor de ponteiros do tipo **Produto**, alocado dinamicamente. Adicione os atributos necessários para o funcionamento da classe e implemente seus métodos.

```
Pedido(int quantidadeMaxima);
virtual ~Pedido();

bool adicionar(Produto *p);
bool remover(Produto *p);
Produto **getProdutos();
int getQuantidadeDeProdutos();
double getValor();
void imprimir();
```

- Declare um vetor de **Produto** como atributo (para armazenar os produtos), mas apenas o crie no construtor, com tamanho `quantidadeMaxima`.
- No destrutor, destrua apenas o vetor de produtos alocado dinamicamente. **Não destrua os Produtos**.
- O método **getProdutos** retorna o vetor com os produtos do pedido adicionados.
- O método **getQuantidadeDeProdutos** retorna a quantidade de produtos adicionados (use um atributo adicional para guardar essa informação e não se esqueça de inicializá-lo).
- O método **getValor** deve calcular a soma dos valores dos produtos no **Pedido**.
- O método **adicionar** não deve adicionar o mesmo **Produto** mais de uma vez, nem adicionar uma quantidade acima de **quantidadeMaxima**. Se for possível adicionar o produto, retorne **true**; caso contrário, retorne **false**.
    - Veja se o produto já foi adicionado pelo uso de "==" para comparar se as variáveis apontam para o mesmo objeto na memória.
- O método **remover** não deve modificar o vetor caso o produto especificado não esteja no vetor. A ação de remover um produto deve ser implementada de forma a prevenir que haja posições livres no interior da lista, sem trocar sua ordem. Para tanto, é necessário mover todos os produtos do vetor que estão depois do produto apagado uma posição para trás, como exemplificado abaixo. Se for possível remover o produto, retorne **true**; caso contrário, retorne **false**.
    - Por exemplo: a lista de produtos [a, b, c, d, X] - em que "X" representa uma posição livre - teve seu método *remover* chamado para "b". A lista resultante deve ser [a, c, d, X, X]

- O método **imprimir** de Pedido já consta implementado nos arquivos fornecidos.

- Implemente a função `teste2`, que segue os seguintes passos:

1. Faça os passos 1, 2 e 3 do teste1.
2. Crie os produtos:
    - `Pipoca Pequena - 15 reais`
    - `Pepsi 500ml - 10 reais`
3. `Crie um Pedido de tamanho 3.`
4. `Adicione “Combo 2” ao pedido.`
5. `Adicione ao pedido “Pipoca Pequena” e “Pepsi 500ml” (nessa ordem).`
6. `Imprima o pedido.`
7. `Remova “Pipoca Pequena” do pedido.`
8. `Imprima o pedido novamente.`
9. `Delete o pedido, o combo e os produtos criados.`

Note o funcionamento do princípio da substituição na classe **Pedido**: é possível guardar no vetor de **Produtos** objetos do tipo **Combo**, pois este é filho de Produto.

Testes do Judge
Exercício 1
- Combo eh classe filha de Produto
- Métodos getNome de Combo e Produto
- Método getPreco de Produto
- Método getPreco de Combo
- Teste do método imprimir produto
- Teste da função teste1

Exercício 2
- Test case: Pedido com objetos Produto getters
- Test case: Pedido com objetos Combo getters
- Test case: Pedido com objetos Produto e Combo getters
- Test case: Adicionar produtos iguais
- Test case: Adicionar com vetor cheio
- Test case: Remover primeira posição
- Test case: Remover primeira posição duas vezes
- Test case: Remover ultima posição
- Test case: Remover meio do vetor
- Test case: Remover meio do vetor duas vezes
- Test case: Remover ultima posição duas vezes
- Test case: Remover com produto que não está no vetor
- Test case: Remover com lista vazia
- Test case: Teste da função teste2