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

2022

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".
- Comente a função *main* ao submeter.

Exercício 01

Considere a classe **Filme**, já fornecida. Implemente agora a classe **Lancamento**, filha de **Filme**. Essa classe recebe no construtor o preço de aluguel do lançamento em `preco` e a data de lançamento através do parâmetro `dataDeLancamento`. A data de lançamento será um inteiro com quatro dígitos no formato DDMM em que DD corresponde ao dia e MM ao mês de lançamento. Além disso, a classe também recebe os parâmetros herdados da classe **Filme**. A seguir são apresentados os métodos específicos a essa classe.

```
Lancamento(int preco, int dataDeLancamento, string nome, int duracao);
~Lancamento();

int getPreco();
int getDataDeLancamento();

void imprimir();
```

- Altere a visibilidade dos atributos necessários da classe **Filme** (private ou protected).
- Implemente os métodos `getPreco()` e `getDataDeLancamento()`, que retornam o preço e a data de lançamento, respectivamente.
- Implemente o método `imprimir()` para realizar a impressão (pulando uma linha ao final):

  `<nome> - sai em <dataDeLancamento> por <preco> reais`

  Por exemplo, para o **Lancamento** ABC planejado para 1312 (treze de dezembro) custando 20 reais de aluguel, seria impresso:

  `ABC - sai em 1312 por 20 reais`

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

1. Declare uma variável da classe `Filme` e atribua-lhe um objeto da classe `Lancamento` de nome *Sonic 2* com 110 minutos de duração, 12 reais de preço de aluguel e previsto para 20 de Março (2003);
2. Use `static_cast` para fazer a conversão do filme *Sonic 2* para um lançamento e imprima.
3. Delete *Sonic 2*.

Observação: na aula que vem veremos o que significa a palavra reservada *virtual* no destrutor de Filme. Por enquanto não se preocupe com isso.

Exercício 02
Implemente a classe **ListaDeFilmes**, correspondente a uma lista de filmes que um usuário pode criar para ver depois. Cada lista contém um vetor de ponteiros do tipo **Filme**, alocado dinamicamente. Adicione os atributos necessários para o funcionamento da classe e implemente seus métodos.

```
class ListaDeFilmes{
public:
    ListaDeFilmes(int quantidadeMaxima);
    virtual ~ListaDeFilmes();

    bool adicionar(Filme* f);
    bool remover(Filme* f);
    Filme** getFilmes();
    int getQuantidadeDeFilmes();

    void imprimir();
};
```

- Declare um vetor de **Filme** como atributo (para armazenar os filmes a assistir), mas apenas o crie no construtor, com tamanho `quantidadeMaxima`.
- No destrutor, destrua apenas o vetor de filmes alocado dinamicamente. **Não destrua os Filmes**.
- O método **getFilmes** retorna o vetor com os filmes a assistir adicionados.
- O método **getQuantidadeDeFilmes** retorna a quantidade de filmes a assistir adicionados (use um atributo adicional para guardar essa informação e não se esqueça de inicializá-lo).
- O método **adicionar** não deve adicionar o mesmo filme mais de uma vez, nem adicionar uma quantidade acima de **quantidadeMaxima**. Se for possível adicionar o filme, retorne **true**; caso contrário, retorne **false**.
    - Veja se o filme 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 filme especificado não esteja no vetor. A ação de remover um filme deve ser implementada de forma a prevenir que haja posições livres no interior da lista, sem trocar sua ordem original. Para tanto, é

necessário mover todos os filmes do vetor que estão depois do filme apagado uma posição para trás, como exemplificado abaixo. Se for possível remover o filme, retorne **true**; caso contrário, retorne **false**.
    - Por exemplo: a lista de filmes [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** já consta implementado.
- Implemente a função `teste2,` que segue os seguintes passos:

  1. Crie um `Filme` de nome *Pequena Sereia* com duração de 83 minutos;

  2. Crie um `Lancamento` de nome *Sonic 2* com 110 minutos de duração, 12 reais de preço de aluguel e previsto para 20 de Março;

  3. Crie uma lista com dois filmes a assistir (use esse valor como máximo), nesta ordem: *Pequena Sereia* e *Sonic 2*;

  4. Imprima a lista criada;

  5. Remova o Filme *Pequena Sereia* da lista.

  6. Imprima a lista criada;

  7. Delete, nesta ordem, a *lista*, *Pequena Sereia* e *Sonic 2*.

Para pensar: Verifique a forma como é feita a impressão do **`Lancamento`** ***Sonic 2*** na impressão da lista. Como seria o esperado?

Testes do Judge
Exercício 1
- Lancamento é classe filha de Filme;
- Métodos: getNome e getDuracao;
- Método: getPreco e getDataDeLancamento;
- Teste do método imprimir;
- Teste da função teste.

Exercício 2
- ListaDeFilmes com objetos Filme: getters
- ListaDeFilmes com objetos Lacamento: getters;
- ListaDeFilmes com objetos Filme e Lancamento: getters;
- Adicionar filmes iguais;
- Adicionar com vetor cheio;
- Remover com filme inválido;
- Remover com lista vazia;
- Remover com filme válido;
- Teste da função teste.