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 07 – Herança e Polimorfismo II

Atenção

1. As definições das classes usadas nos exercícios encontram-se **disponíveis no e-Disciplinas**. Use o código fornecido.
2. Os nomes, os atributos, os métodos, e as respectivas assinaturas das classes dadas **devem seguir o especificado** em cada exercício para fins de correção automática.
3. A **ordem de declaração** de atributos e métodos fornecidos **não deve ser alterada**. Caso contrário, poderá haver redução automática da nota.
4. A função `main` **não deve ser submetida**. Caso contrário, a correção automática retornará um *Compilation Error.*

Considere as classes **Filme** e **Catalogo**, implementadas nas aulas anteriores. Elas são fornecidas para esta aula.

Exercício 01

Altere a **Catalogo.** Seus métodos públicos são apresentados a seguir:

```
Catalogo(int tamanhoMaximo);
~Catalogo();

bool adicionar(Filme *f);
bool adicionar(string categoria);
int getDuracaoTotal();
int getTotalDeVisualizacoes();
Filme** getFilmes();
string* getCategorias();

void imprimir();
```

- O **Catalogo**, além de armazenar filmes, agora deve armazenar as categorias de seus filmes (por exemplo, categorias como filme premiado, clássico, remake, etc…). Portanto, na classe **Catalogo** fornecida, crie e implemente o método `bool adicionar(string categoria)`, que adiciona uma determinada categoria ao catálogo. Nesse método, a função deve retornar `false` caso o vetor de categorias já esteja completamente

preenchido ou se a categoria que se deseja adicionar já existe no vetor. Caso contrário, a categoria deve ser adicionada e, por fim, retorna-se `true`. Altere a classe no que for necessário para que ela armazene como atributo interno a quantidade de categorias adicionadas.

- No construtor, crie um vetor alocado dinamicamente para armazenar as categorias, de tamanho máximo `tamanhoMaximo` informado no construtor (ou seja, o mesmo tamanho máximo dos filmes). Lembre-se de deletar esse vetor no destrutor.

  Perceba que o método adicionar foi **sobrecarregado.**

- Implemente o método `string* getCategorias()`, que retorna o vetor de categorias do **Catalogo**.
- Implemente o método `void imprimir()`. Para um Catalogo com n filmes e k categorias, deve-se imprimir o seguinte:
  - Na primeira linha, imprime-se a quantidade de filmes e categorias como abaixo:

  `Catalogo: <n> filmes e <k> categorias`
  - Em seguida, chame o método `imprimir` para cada filme no catálogo.

  Assim, por exemplo, seja um catálogo com os seguintes filmes: *Casablanca* com 100 minutos de duração, 5 visualizações; e *Titanic* com 120 minutos de duração, 3 visualizações. Considere também que o catálogo possua as categorias Clássico e Oscar. O método `imprimir()` para esse catálogo deve gerar a seguinte saída:

```
Catalogo: 2 filmes e 2 categorias
Filme: Casablanca - 100 minutos - 5 visualizacoes
Filme: Titanic - 120 minutos - 3 visualizacoes
```

- Implemente a função `teste1()` conforme os passos a seguir:

1. Crie um catálogo de tamanho 2;
2. Crie o filme *Madagascar* de duração 100 minutos;
3. Chame o método `assistir` para *Madagascar* com os valores 20, 30 e 50;

4. Adicione *Madagascar* ao catálogo criado;
5. Adicione a categoria *Animacao* ao catálogo;
6. Imprima o catálogo;
7. Delete o filme e o catálogo criados.

Exercício 02
Implemente a classe **Lancamento**, que é uma classe filha de **Filme**. Seus métodos públicos são apresentados a seguir:

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

void assistir(int tempo);
```

- Redefina o método `assistir` da classe mãe para que a visualização de um lançamento seja contada para qualquer valor de tempo maior que zero e menor ou igual ao valor de duração. Para valores incoerentes, isto é, negativos ou maiores que o valor de duração, não incremente as visualizações do lançamento. Modifique o que for necessário na classe **Filme** para que haja ligação dinâmica. Lembre-se também de alterar o destrutor apropriadamente, pelo mesmo motivo. Caso contrário, o destrutor de `Lancamento` não seria chamado.
- Implemente a função `teste2()` conforme os passos a seguir:

1. Repita os passos de 1 a 5 do `teste1`;
2. Crie o lançamento *Tempos Modernos* de duração 100 minutos e faça uma variável Filme apontar para ele;
3. Chame o método `assistir` para *Tempos Modernos* com os mesmos valores do passo 3 do teste1;
4. Adicione *Tempos Modernos* ao catálogo criado;
5. Imprima o catálogo.
6. Delete o filme, o lançamento e o catálogo criados.

Compare os valores de visualização dos filmes *Madagascar* com *Tempos Modernos*, que receberam o mesmo tratamento do método `assistir`. Perceba o comportamento diferente do método `assistir` dependendo de se o objeto é Filme ou Lancamento. Isso é a consequência da ligação dinâmica!

Exercício 03

Implemente o método `Lancamento** getLancamentos(int &quantidadeLancamentos)` em **Catalogo**, que deve retornar um vetor composto pelos lançamentos que existem no catálogo, caso existam. Se não existir lançamentos no catálogo, retorne NULL (e `quantidadeLancamentos` deve ser retornado como `0`). Crie o vetor de lançamentos dentro do método (ou seja, cada vez que o método for chamado, crie um novo vetor dinamicamente). Para simplificar, você pode alocar um vetor de `tamanhoMaximo`). Além disso, note que o método também retorna a quantidade de lançamentos que existem no vetor, uma vez que `quantidadeLancamentos` é passado por referência. Não se esqueça de inicializá-lo em `0` dentro do método.

Os métodos públicos de **Catalogo** devem ficar (não altere os demais métodos):

```
Catalogo(int tamanhoMaximo);
~Catalogo();


bool adicionar(Filme *f);
bool adicionar(string categoria);
int getDuracaoTotal();
int getTotalDeVisualizacoes();
Filme** getFilmes();
string* getCategorias();
Lancamento** getLancamentos(int &quantidadeLancamentos);

void imprimir();
```

- Implemente a função `teste3()` conforme os passos a seguir:

1. Crie um catálogo de tamanho 2;
2. Chame o método `getLancamentos` para obter a quantidade de lançamentos do catálogo;
3. Imprima a quantidade de lançamentos do catálogo (pule uma linha no final);
4. Crie o **Filme** *Madagascar* de duração 100 minutos (não o assista) e adicione-o ao catálogo;

5. Repita os passos 2 e 3;
6. Crie o **Lancamento** *Tempos Modernos* de duração 100 minutos (não o assista) e adicione-o ao catálogo;
7. Repita os passos 2 e 3;
8. Delete o filme, o lançamento e o catálogo criados.

Testes do Judge

Exercício 1

- `Catalogo` Teste `adicionar` categoria com catálogo vazio;
- `Catalogo` Teste `adicionar` categoria já adicionada;
- `Catalogo` Teste `adicionar` categoria com catálogo cheio;
- `Catalogo` Teste `getCategorias`;
- `Catalogo` Teste `imprimir`;
- Teste da função `teste1`.

Exercício 2

- `Lancamento` Teste `assistir` com tempo maior que 0 e menor que a duração do filme;
- `Lancamento` Teste `assistir` com tempo menor ou igual a 0;
- `Lancamento` Teste `assistir` com tempo maior que a duração do filme;
- Teste da função `teste2`.

Exercício 3

- `Catalogo` Teste `getLancamentos` com 5 filmes e nenhum lançamento;
- `Catalogo` Teste `getLancamentos` com 5 filmes e 1 lançamento;
- `Catalogo` Teste `getLancamentos` com 5 filmes e 3 lançamentos;
- Teste da função `teste3`.