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
2021
Aula 07 – Herança e Polimorfismo II

Atenção

- Código inicial a ser usado na resolução dos exercícios encontra-se **disponível no Discord**.
- 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.
- 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 1

Considere as classes **Musica**, **Artista** e **Playlist**, implementadas nas aulas anteriores.

Na classe **Playlist**:

- Além de armazenar músicas, a **Playlist** deve armazenar **Artistas**. Com isso, implemente o método `bool adicionar(Artista *a)`, o qual possibilita a adição de um **Artista** à **Playlist**. Crie no construtor um vetor alocado dinamicamente para armazenar os artistas, o qual deve ter tamanho máximo `maximoValor`. No método `adicionar`, se já houver `maximoValor` de **Artistas** ou caso o **Artista** tenha sido previamente adicionado (o mesmo objeto), o método deve retornar `false`. Ao adicionar um **Artista** a uma **Playlist** deve-se adicionar todas as suas músicas à **Playlist**. Caso não seja possível adicionar uma das músicas de um **Artista** devido a espaço insuficiente, o método deve retornar `false` (sem adicionar músicas e o artista). Caso uma das músicas não seja adicionada por ser repetida, o método ainda deve retornar `true`. Por exemplo, considere uma playlist com as músicas: *Redemption Song*, *Stairway to Heaven* e *Alien*. Caso o artista "Bob Marley" tenha as músicas *Redemption Song* e *Buffalo Soldier*, ele deve ser adicionado à playlist desde que exista espaço o suficiente para a adição da música *Buffalo Soldier* (Note que *Redemption Song* não deve ser adicionada pois já está contida na playlist). Caso seja possível adicionar o **Artista** à **Playlist**, o método deve retornar `true`.

  Dica: A classe **Playlist** possui um método `temMusica` que retorna `true` caso uma música já tenha sido adicionada.

- Implemente também os métodos `int getQuantidadeDeArtistas()` e `Artista** getArtistas()` que devem retornar a quantidade de artistas já adicionados ao vetor e o vetor alocado dinamicamente que contém os artistas da **Playlist**, respectivamente.
- No destrutor, destrua apenas o vetor alocado. Não destrua os **Artistas** e tampouco as **Musicas**.

Note que o método `adicionar` foi ***sobrecarregado***. Os métodos públicos dessa classe são apresentados a seguir:

```
class Playlist {
public:
    Playlist(string nome, int maximoValor);
    ~Playlist();

    string getNome();
    int getDuracao();

    int getQuantidadeDeMusicas() const;
    int getQuantidadeDeArtistas() const;

    Musica** getMusicas();
    Artista** getArtistas();

    bool adicionar(Musica *m);
    bool adicionar(Artista *a);

    void imprimir();
};
```

Implemente apropriadamente os métodos conforme especificado e adicione atributos conforme necessário. **NÃO** altere as classes **Musica** e **Artista** fornecidas.

Implemente a função teste considerando os seguintes passos:

1. Crie a `Playlist` *Favoritos* e `maximoValor` 2;
2. Crie o `Artista` *Tom Jobim*, de `quantidadeMaxima` 2;
3. Adicione a música *Aguas de marco*, de duração 214, a *Tom Jobim* e à *Playlist Favoritos*;
3. Adicione a música *Samba de uma nota so*, de duração 180, a *Tom Jobim*;
6. Adicione *Tom Jobim* a *Favoritos*;
7. Imprima *a Playlist Favoritos*.

Exercício 2
Implemente a classe **Banda**, cujos métodos públicos são apresentados abaixo. **Inclua os arquivos ".h" e ".cpp" referentes à classe Banda ao projeto do Code::Blocks** (clique com o botão direito no projeto e selecione "Add files..").

```
class Banda: public Artista {
public:
    Banda(string nome, int maximoValor);
    ~Banda();

    bool adicionar(Artista* e);
    double getNota();
};
```

Redefina o método `getNota` em **Banda** de forma que a nota retornada seja a nota média de todos os artistas que fazem parte da banda, com um bônus de um ponto. Em outras palavras, a nota da banda deve ser a média da nota de seus artistas (obtida pelo método getNota de **Artista**) somada a um. A nota máxima da banda deve ser cinco, ou seja, caso a nota ultrapasse cinco quando somado o bônus, retorne o valor 5. Por exemplo, considere a Banda *Elis &* Tom composta pelos artistas *Tom Jobim, com nota 3, e Elis Regina, com nota 4.* A Banda deve ter nota ((3 + 4) / 2) + 1 = 4,5.
Não é necessário modificar a implementação de `adicionar` entregue.

Altere o arquivo `Artista.h` para que o método mais especializado seja sempre utilizado, independentemente do tipo da variável. Mas não modifique a **implementação** dos métodos em `Artista.cpp`.

Exercício 3
Na classe **Playlist**, mostrada no primeiro exercício, implemente o método `getBandas`. A assinatura da classe deve ficar da seguinte forma:

```
class Playlist {
public:
    Playlist(string nome, int maximoValor);
    ~Playlist();

    string getNome();
    int getDuracao();

    int getQuantidadeDeMusicas() const;
    int getQuantidadeDeArtistas() const;

    Musica** getMusicas();
    Artista** getArtistas();

    Banda** getBandas(int &quantidade);

    bool adicionar(Musica* e);
    bool adicionar(Artista* e);

    void imprimir();
};
```

Esse método deve retornar um vetor alocado dinamicamente contendo todas as **Bandas** do vetor que armazena os **Artistas** de uma **Playlist**. Além disso, deve-se retornar a quantidade de **Bandas** por meio do parâmetro `quantidade`, passado por referência. Caso a **Playlist** não possua nenhuma **Banda**, deve-se retornar NULL e deve ser atribuído 0 à `quantidade`.

Dicas: Para facilitar, aloque um vetor com tamanho `maximoValor`.

Além disso, modifique a implementação do método imprimir de `Banda` para que seja um refinamento. Antes de invocar o método da superclasse, esse método deve imprimir a quantidade de membros da `Banda` da seguinte forma (sem pular linha no final):

```
Banda de <quantidadeDeArtistas> membros
```

Testes do Judge

Exercício 1

- Playlist: adiciona Artista com músicas em Playlist vazia;
- Playlist: adiciona Artista com músicas em Playlist cheia sem músicas do artista;
- Playlist: adiciona Artista com músicas em Playlist cheia que já contém músicas do artista;
- Playlist: adiciona Artista com músicas em Playlist com todas menos uma música do artista (com espaço para adição);

- Playlist: adiciona Artista com músicas em Playlist com todas menos uma música do artista (sem espaço para adição);
- Playlist: adiciona Artista em Playlist sem espaço para o Artista.
- Playlist: adiciona Artista repetido em Playlist.
- Teste da função teste.

Exercício 2

- Banda: getNota de Banda com nota inferior a cinco;
- Banda: getNota de Banda com nota superior a cinco;
- Banda: getNota de Banda através de uma variável de tipo Artista.

Exercício 3

- Playlist: getBandas em playlist sem bandas;
- Playlist: getBandas em playlist com uma banda no começo do vetor;
- Playlist: getBandas em playlist com uma banda no meio do vetor;
- Playlist: getBandas em playlist com uma banda no fim do vetor;
- Playlist: getBandas em playlist com mais de uma banda;
- Banda: imprimir;