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 4 – Encapsulamento

Atenção

- Código inicial a ser usado na resolução dos exercícios encontra-se **disponível no Discord**.
- Submeta um arquivo comprimido (faça um ".zip" – **não pode ser ".rar"**) colocando apenas os arquivos ".cpp" e ".h". Não crie pastas no "zip".
- Ao enviar para o Judge mantenha o `#define NUMERO_MAXIMO_VALORES 10` com o valor 10, por motivos de correção.

Utilize o código fornecido no e-Disciplinas para implementar as classes **`Musica`** e **`Playlist`**, aplicando os conceitos de encapsulamento vistos em aula.

Exercício 1
No arquivo teste.cpp é fornecida a definição e a implementação da classe `Musica`. Separe-a em dois arquivos, "**Musica.h**" e "**Musica.cpp**". O arquivo .h deve conter apenas a definição. O arquivo .cpp deve conter apenas a implementação. Use adequadamente as diretivas de compilação.

Implemente os métodos `getNome`, `setNome`, `getDuracao` e `setDuracao` apropriadamente (retorne o valor do respectivo atributo no get e defina o valor do respectivo atributo no set). Defina a visibilidade dos atributos e dos métodos de modo que os atributos sejam acessíveis apenas pelos métodos da classe e os métodos sejam acessíveis externamente.

Complete o código fornecido, para tanto considere os seguintes passos na função `teste`:

1. Crie uma `Musica` de nome *Roses* e duração 180 segundos;
2. Avalie *Roses* com notas 3, 3, e 1;
3. Imprima *Roses*;

Exercício 2
Implemente a classe `Playlist`, definida a seguir. **Inclua os arquivos ".h" e ".cpp" referentes a essa classe ao projeto do Code::Blocks** (clique com o botão direito no projeto e selecione "Add files.."). Note que essa classe utiliza a classe `Musica`, do Exercício 1. Novamente, é necessário separar a classe em dois arquivos, "**Playlist.h**" e "**Playlist.cpp**", usando adequadamente as diretivas de compilação. Defina corretamente a **visibilidade** de seus métodos e atributos.

```
#define NUMERO_MAXIMO_VALORES 10

class Playlist {
    int getDuracaoTotal();
    bool adicionar(Musica* m);

    void setNome(string nome);
    string getNome();
    int getQuantidade();

    void imprimir();
};
```

A implementação da classe Playlist deve atender aos seguintes requisitos:

- O método `setNome` deve definir o nome da `Playlist`. O nome é obtido pelo método `getNome`.
- O método `getDuracaoTotal` deve retornar o somatório da duração de cada música na `Playlist`**.** Caso não existam músicas adicionadas à playlist, esse método deve retornar -1.
    - **Dica:** Use o método da classe `Musica`.
- O método `adicionar` deve adicionar, se possível, o objeto do tipo `Musica` passado como parâmetro ao vetor **musicas** (crie este atributo). Caso o vetor já esteja completamente preenchido ou o objeto já tenha sido adicionado, o método não modifica o vetor e retorna **false**. Caso seja bem sucedido, deve retornar **true**. Utilize a constante `NUMERO_MAXIMO_VALORES` como o número máximo de músicas que a Playlist comporta. Controle a quantidade de objetos adicionados com um atributo `quantidade`, ou seja, caso exista apenas uma música na Playlist, a `quantidade` será igual a 1; caso existam duas músicas, a `quantidade` será igual a 2.
- O método `getQuantidade` retorna o número de músicas que foram adicionadas ao vetor `musicas`, retornando o valor do atributo `quantidade`.
- O método `imprimir` deve mostrar na tela as informações da Playlist, seguindo o formato:

  `<nome> - <duração total> segundos no total`

  Além de imprimir as informações de todas as músicas que foram adicionadas à Playlist. Por exemplo:

  ```
  Estrangeiras – 390 segundos no total
  Roses – 180 segundos - <número> avaliacao
  Overdue – 210 segundos - <número> avaliacao
  ```

- Altere a função `teste,` para tanto considere os seguintes passos:

    1. Crie uma Musica de nome Roses e duração 180 segundos;
    2. Avalie Roses com notas 3, 3, e 1;
    3. Crie uma outra `Musica` de nome *Overdue* e duração 210 segundos;
    4. Avalie *Overdue* com notas 1, 5, e 4;

5. Crie uma `Playlist` de nome *Estrangeiras*;
6. Adicione *Roses* e *Overdue* em *Estrangeiras*;
7. Imprima *Estrangeiras.*

Dica: a classe `Musica` já possui um método `imprimir` – utilize-o para facilitar a implementação do método imprimir da `Playlist`!

Lembre-se de utilizar a diretiva *#ifndef* nos arquivos de cabeçalhos ("**.h**") para evitar problemas de conflitos de definição de classes causados por múltiplas inclusões de um mesmo cabeçalho.

Dicas importantes
1. Os nomes, os atributos, os métodos, e as respectivas assinaturas das classes dadas **devem seguir exatamente o especificado** para fins de correção automática.

2. A função `main` não deve ser submetida. Caso contrário, a correção automática retornará um *Compilation Error.*

Testes do Judge
Exercício 1
- Teste dos *setters* e dos *getters;*
- Teste `imprimir;`
- Teste da função `teste`.

Exercício 2
- Teste dos *setters* e dos *getters* da `Playlist`;
- Teste `adicionar` com vetor vazio;
- Teste `adicionar` com vetor parcialmente preenchido;
- Teste `adicionar` com vetor cheio;
- Teste `getDuracaoTotal` com vetor vazio;
- Teste `getDuracaoTotal` com vetor parcialmente preenchido;
- Teste `getDuracaoTotal` com vetor cheio;
- Teste da função `teste`.