

CLASSES ABSTRATAS / POLIMORFISMO PARAMÉTRICO

DISCIPLINA: PROGRAMAÇÃO ORIENTADA A OBJETOS

Data de entrega: até 24 de abril de 2022.

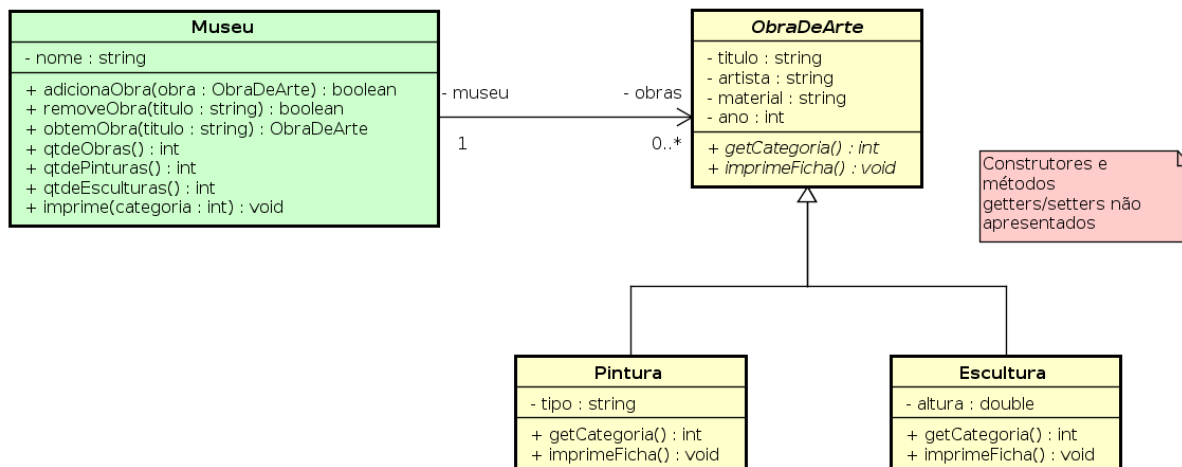
Professores: Delano Beder & Renato Bueno

1 Introdução

Vamos supor que vocês fazem parte de uma equipe de desenvolvimento que foi contratada para desenvolver um sistema de gerenciamento (catálogo) de obras de arte de um museu. Após a fase de levantamento de requisitos do sistema, a equipe identificou que as obras de arte são apenas de duas categorias — pinturas e esculturas — e que compartilham algumas características em comum: título, artista, material e ano de criação. Dessa forma, a modelagem orientada a objetos do sistema é composta das 4 classes discutidas nas próximas seções.

1.1 Classes do Catálogo

O catálogo de obras de arte será implementado através das seguintes classes: `ObraDeArte`, `Pintura`, `Escultura` e `Museu`. Organize suas classes no *namespace* **catalogo**.



Classe ObraDeArte

A classe abstrata `ObraDeArte` deverá conter atributos **privados** para armazenar os seguintes dados sobre as mídias: **título**, **artista**, **material** sobre o qual esta foi feita (papel, tela, madeira, etc) e **ano de criação**. Esta classe deverá também conter um ou mais métodos (métodos *getters/setters*) que permitam a atribuição e recuperação de valores para cada um dos atributos.

A classe `ObraDeArte` deve conter pelo menos os seguintes métodos/construtores:

- `ObraDeArte(string titulo, string artista, string material, int anoCriacao)`. Construtor único da classe que recebe os valores iniciais dos atributos **título**, **artista**, **material** e **ano de criação**.
- `int getCategoria()`. Método *abstrato* que deve ser implementado pelas subclasses de `ObraDeArte`.
- `void imprimeFicha()`. Método *abstrato* que deve ser implementado pelas subclasses de `ObraDeArte`. Esse método imprime uma ficha contendo os dados da obra de arte semelhante às apresentadas a seguir.

(a) Ficha de uma Pintura

Categoria: Pintura Título: Mona Lisa Artista: Leonardo da Vinci Material: Madeira Ano: 1503 Tipo: Óleo

(b) Ficha de uma Escultura

Categoria: Escultura Título: David Artista: Michelangelo Material: Mármore Ano: 1501 Altura: 1.99
--

Classe Pintura

A classe `Pintura` representa uma pintura e deve conter o seguinte atributo: **tipo** (óleo, aquarela, etc). Esta classe deverá também conter um ou mais métodos (métodos *getters/setters*) que permitam a atribuição e recuperação de valores para cada um dos atributos.

Essa classe deve conter pelo menos os seguintes métodos:

- `Pintura(string titulo, string artista, string material, int anoCriacao, string tipo)`. Construtor único da classe que recebe os valores iniciais dos atributos.
- `int getCategoria()`. Implementação do método que retorna um inteiro que representa a categoria.
[Pintura = 1 e Escultura = 2]
- `void imprimeFicha()`. Implementação do método que imprime os dados da pintura.
[Exemplo: ver ficha (a) apresentada anteriormente]

Classe Escultura

A classe **Escultura** representa uma escultura e deve conter o seguinte atributo: **altura**. Esta classe deverá também conter um ou mais métodos (métodos *getters/setters*) que permitam a atribuição e recuperação de valores para cada um dos atributos.

Essa classe deve conter pelo menos os seguintes métodos:

- **Escultura(string titulo, string artista, string material, int anoCriacao, double altura)**. Construtor único da classe que recebe os valores iniciais dos atributos.
- **int getCategoria()**. Implementação do método que retorna um inteiro que representa a categoria. [Pintura = 1 e Escultura = 2]
- **void imprimeFicha()**. Implementação do método que imprime os dados da escultura. [Exemplo: ver ficha (b) apresentada anteriormente]

Classe Museu

A classe **Museu** possui o atributo **nome** que representa o nome do museu. Esta classe deverá também conter um ou mais métodos (métodos *getters/setters*) que permitam a atribuição/recuperação de valores do atributo **nome**. Por fim essa classe possui o atributo **obras** que representa um relacionamento de **1 para N** com a classe abstrata **ObraDeArte**.

A classe **Museu** deve conter pelo menos os seguintes métodos:

- **boolean adicionaObra(ObraDeArte* obra)**. Adiciona uma obra de arte ao catálogo. O método deve restringir a inserção de uma obra duplicada.
- **boolean removeObra(string titulo)**. Remove uma obra de arte (cujo título é igual ao parâmetro passado) do catálogo.
- **ObraDeArte* obtemObra(string titulo)**. Método que retorna a obra de arte cujo título é igual ao parâmetro passado e **null**, caso a obra de arte não seja encontrada no catálogo.
- **int qtdeObras()**. Fornece a quantidade de obras no catálogo.
- **int qtdePinturas()**. Fornece a quantidade de pinturas no catálogo.
- **int qtdeEsculturas()**. Fornece a quantidade de esculturas no catálogo.
- **void imprime(int categoria)**. Imprime as fichas das obras de arte da coleção, ordenadas pelo ano + título. Isto é, as obras de arte são ordenadas levando em consideração o **ano** da obra de arte. Caso duas obras de arte tenham o mesmo **ano**, então são ordenadas pelo **título**.
 - categoria = 0. Imprime todas as obras de arte.
 - categoria = 1. Imprime apenas as pinturas.
 - categoria = 2. Imprime apenas as esculturas.

2 Observações importantes

2.1 Sobre a elaboração:

- Este exercício-programa deve ser elaborado em grupo (de até 5 estudantes).
- Vocês devem utilizar **apenas** os conceitos apresentados em aula.
 - Vocês devem implementar as classes em C++.
 - Os atributos das classes devem ser **privados**.
 - Organizem suas classes no *namespace* **catalogo**.
 - Utilizem uma estrutura de dados mais apropriada da API STL C++.
- Compactem o código-fonte em um arquivo <NroGrupo>.zip
Exemplo: Grupo01.zip (Cuidado para não enviar o arquivo errado!)
- O prazo de entrega é o dia 24 de abril de 2022 às 23h59.
- A entrega será feita unicamente pelo ambiente moodle (<https://ava2.ead.ufscar.br>). Não serão aceitos trabalhos enviados por email.
- Guardem uma cópia do seu programa entregue.

2.2 Sobre a avaliação:

- Não serão toleradas cópias! Exercícios copiados (com ou sem eventuais disfarces) receberão nota ZERO. O exercício do aluno alvo da cópia também receberá nota ZERO.
- Exercícios com erros de sintaxe (ou seja, erros de compilação) receberão nota ZERO.
- Os exercícios serão avaliados segundo os seguintes critérios:
 - Soma simples dos valores obtidos nos itens de 1 a 2
 1. Atendimento às normas de boas práticas de programação (comentários, endentação, nomes de variáveis, estruturação do código, modularização, etc) [0..10]
 2. Corretude na implementação da atividade [0..90]