Programação Orientada por Objetos

2019/2020

Trabalho de Laboratório – Curso EI

2

Objetivos:
Introdução à herança de classes.

Programas:
Pretende-se desenvolver um programa que suporte a gestão e classificação de obras de arte de um museu.

Regras de implementação:

- Criar a aplicação utilizando o IDE BlueJ.
- Implementar o código necessário e testar no fim de cada nível.
- Use as convenções de codificação adotadas para a linguagem Java (ver **Notas**).
- No método `main` separe o código de cada nível imprimindo uma linha do tipo:
  - `"//NIVEL N  ************************************"`.

Implementação:

Nível 1:

- Implemente uma classe `Room` com os seguintes atributos:
  - `number` - um número inteiro atribuído automaticamente com base no número de ordem de criação do objeto;
  - `name` – uma String com o nome da sala;
  - `area` - um número decimal com a área da sala.
- Crie um construtor sem argumentos.
- Crie um construtor com parâmetros para todos os atributos à exceção de `number` (que deverá ser incrementado automaticamente para cada objeto sala criado).
- Crie o método seletor para o atributo `number`.
- Crie os métodos seletores e modificadores para os atributos `name`, e `area`. Valide os parâmetros recebidos.
- Redefina o método `toString()`, para devolver apenas o nome da sala.
- Para testar mais facilmente, crie uma classe `Program` com um único método estático `void main()` onde irá colocar o código de teste.
- Neste método `main` crie as dez salas do museu, adicionando-as a uma coleção de salas.
- Apresente na consola o resultado da aplicação do método `toString()` às instâncias criadas.

Nível 2:

- Implemente uma classe `WorkOfArt` com os seguintes atributos:
  - `code` – um número inteiro atribuído automaticamente com base no número de ordem de criação do objeto;
  - `onDisplay` – um booleano que indica se a obra de arte se encontra em exibição (defina as constantes `OFF_DISPLAY` (false) e `ON_DISPLAY` (true));
  - `artist` – uma String com o nome do artista;
  - `title` – uma String com o nome da obra de arte.
- Crie um construtor com todos os parâmetros necessários (na criação do objeto uma obra de arte esta encontra-se sempre fora de exibição (`OFF_DISPLAY`)).
- Crie um construtor sem argumentos que reutilize o construtor anterior.
- Crie os métodos seletores e modificadores para os atributos `onDisplay`, `artist` e `title` e apenas o método seletor para os restantes atributos. Valide os parâmetros recebidos.
- No método `main` crie duas `WorkOfArt` e imprima os nomes dos artistas e das obras de arte na consola.

Nível 3:

- Implemente a classe **`Painting`** que herda de **`WorkOfArt`** e tem como atributos:
  - **`artMovement`** – uma String com o nome do movimento artístico;
  - **`inkType`** – Enumerado com os seguintes valores possíveis: **`OIL`**, **`WATERCOLOR`**, **`PASTEL`**, **`ACRYLIC`** e **`OTHER`**;
  - **`Support`** – Enumerado com os seguintes valores possíveis: **`ON_CANVAS`**, **`ON_PAPER`**, **`ON_WOOD`**, **`ON_GLASS`** e **`OTHER`**.
  - Crie um construtor sem argumentos e um construtor com todos os parâmetros.
  - Crie os métodos seletores e modificadores.
- Implemente a classe **`Sculpture`** que herda de **`WorkOfArt`** e defina nessa classe o atributo:
  - **`material`** - uma String com o material da escultura;
  - Crie o método seletor e modificador do atributo **`material`**;
  - Crie um construtor sem argumentos e um construtor com os parâmetros **`artist`**, **`title`** e **`material`**.
- Redefina os métodos **`toString`** das classes **`Sculpture`** e **`Painting`** para retornar a informação dos atributos da superclasse e da respetiva classe. Para melhor codificação sugere-se a reutilização do método **`toString()`** da classe **`WorkOfArt`**.
- No método **`main`**, crie uma pintura e uma escultura, aplicando o **princípio da substituição** coloque os objetos criados numa única coleção. Apresente na consola o resultado da aplicação do método **`toString()`** ao conjunto de obras de arte nesta coleção.

Nível 4:

- Crie uma classe **`Museum`** com uma coleção que relaciona uma obra de arte com a respetiva sala de exibição.
- Crie um construtor sem argumentos.
- Implemente o método **`addToMuseumCollection`** para adicionar e associar uma obra de arte (ainda não associada) a uma sala.
- Implemente também o método **`removeFromMuseumCollection`** para remover uma obra de arte da coleção do museu.
- Crie um método **`toString()`** que devolve a lista das obras de arte na coleção do museu.
- No método **`main`**, crie uma instância de **`Museum`**, adicione e associe a pintura e a escultura criadas no nível anterior a salas diferentes e imprima na consola o resultado da aplicação do método **`toString()`** à coleção de obras de arte do museu.

Nível 5:

- Na classe **`Museum`**, implemente o método **`listArtInRoom()`** que devolve a lista das obras de arte associadas à sala fornecida como parâmetro.
- Implemente também o método **`printAllByRoom()`** que apresenta a sala e para cada sala as obras de arte associadas, para todas as salas existentes na coleção e sem repetir nenhuma utilizando o **processamento funcional**.
- No método **`main`**, crie mais duas obras de arte e adicione-as a uma sala ainda sem obras de arte e apresente na consola o resultado do método **`listArtInRoom()`** fornecendo como parâmetro esta sala. Apresente também na consola o resultado da aplicação do método **`printAllByRoom()`**.

Notas:

Para os identificadores siga as convenções adotadas normalmente, em particular:

1) A notação **camelCase** para o nome das variáveis locais e identificadores de atributos e métodos.
2) A notação **PascalCase** para os nomes das classes.

3) Não utilize o símbolo '_', nem abreviaturas nos identificadores.