Lista 3 - POO

Prof. Dr. Alexandre Garcia de Oliveira

Exercício 1

Para montar expressões numéricas computacionais, pode-se usar os conceitos de Programação Orientada a Objetos (POO). A classe `Número` possui um campo inteiro e um construtor. A classe `Termo` possui apenas o método `mostrar()`, que retorna a expressão montada em uma `String`. A classe `Operação` possui dois atributos de tipo `Termo`. Sabe-se que um `Número` também é um `Termo`. O método `mostrar()` deve ser sobrescrito em `Número` e mostrar apenas o atributo inteiro na tela. Sabe-se que `Soma` é uma `Operação` e `Multiplicação` também. O método `mostrar()` em `Soma` e `Multiplicação` mostra a concatenação da chamada "recursiva" de `mostrar()` (use os dois atributos) concatenados com os símbolos "+" e "*", respectivamente. Implemente esta situação e desenhe seu diagrama de classes.

Exercício 2

Usando o exercício 1, crie a classe `Eval` que tenha um método `calcular(Operacao o)` para calcular o resultado de uma operação através de um objeto `Termo` passado como parâmetro.

Exercício 3

Imagine que você está desenvolvendo um sistema para uma escola que precisa gerenciar informações sobre disciplinas e estudantes. Cada disciplina pode ter diferentes tipos de avaliação (prova, trabalho, projeto) e cada estudante pode estar matriculado em várias disciplinas. Você deve usar conceitos de polimorfismo para implementar o seguinte:

- **Classe** `Disciplina`:
  - Possui um nome e uma lista de estudantes matriculados.
  - Possui métodos para adicionar estudantes e calcular a média das notas.
- **Classe** `Estudante`:
  - Possui um nome e um número de matrícula.
  - Possui um método para adicionar uma nota em uma disciplina e outro para calcular a média geral do estudante em todas as disciplinas.
- **Classe** `Avaliacao` **(abstrata)**:

  - Possui um método abstrato `calcularNota()`, que retorna a nota do estudante.
  - Possui um método `mostrarDetalhes()` que retorna uma string com os detalhes da avaliação.

- **Subclasses de** `Avaliacao`:
  - **Classe** `Prova`: Representa uma prova, com atributos específicos como nota e peso.
  - **Classe** `Trabalho`: Representa um trabalho, com atributos específicos como nota e data de entrega.
  - **Classe** `Projeto`: Representa um projeto, com atributos específicos como nota e número de etapas.

Implemente esta situação em Java e desenhe o diagrama de classes correspondente.

Exercício 4

Decida se as assertivas abaixo são verdadeiras ou falsas:

1. É possível ter construtores em classes abstratas;
2. Uma classe abstrata deve possuir apenas métodos abstratos;
3. O modificador `protected` não permite que uma subclasse acesse métodos e atributos;
4. Uma interface pode ser implementada apenas uma vez;
5. Uma interface pode ser definida como uma coleção de assinaturas de métodos e constantes;
6. Uma interface pode possuir atributos (não constantes);
7. Herança múltipla é permitida em Java;
8. Uma classe abstrata pode possuir um construtor privado;
9. Polimorfismo é o mesmo que sobrecarga e sobrescrita;
10. `Object` é subclasse de toda classe;
11. A linha: `"Cachorro o = new Object();"` está correta;