Módulo: Programação Orientada à Objetos I – Polimorfismo

1. O que é polimorfismo? Descreva com suas palavras em uma frase.

Permite que um mesmo método, definido numa superclasse ou interface, apresente vários comportamentos diferentes dependendo da subclasse aplicada.

2. Explique porque o polimorfismo pode ser uma estratégia útil.

O polimorfismo faz com que o código seja melhor interpretado visto que o nome identificador do método partilha um "core" do que faz o método especificando-o para cada necessidade, assim, melhorando a compreensão da regra de negócio aplicada no método.

3. Em Java, declaramos uma variável usando a sintaxe `[Tipo] [variável] = [valor]`. Considerando a representação de herança onde `Animal` é a superclasse e `Cachorro` a subclasse, quais são TODAS as formas possíveis de declarar uma variável para a nova instância `new Cachorro()`?

Animal nomeDoCachorro = new Cachorro();
Cachorro nomeDoCachorro = new Cachorro();

4. É possível aplicar o princípio do polimorfismo tanto para interfaces quanto para classes.

Verdadeiro.

5. Quando definimos uma relação de herança entre duas classes, a classe-filha possui disponível os métodos herdados de sua classe-pai mais os métodos definidos nela mesmo.

Verdadeiro.

6. Quando usamos polimorfismo, os métodos específicos da classe-filha ficam inacessíveis, pois estamos representando aquele objeto através da classe-pai.

Falso.

7. Considere a interface `Animal` com os métodos `alimentar` e `respirar`. Considere também a classe `Cachorro` que implementa a interface `Animal`, e define seus próprios métodos `brincar` e `comunicar`. Quais métodos estarão disponíveis ao declarar a variável `Cachorro cachorro = new Cachorro`?

Os métodos disponíveis na variável cachorro será 'brincar', 'comunicar', 'alimentar' e 'respirar'.

8. Criamos uma classe Cachorro com os seus métodos próprios `brincar` e `comunicar`. Além disso, `Cachorro` implementa a interface `Animal` que possui definição para o método `alimentar`. Se aplicarmos o princípio do polimorfismo declarando uma variável do tipo `Animal cachorro = new Cachorro()`, quais métodos estarão disponíveis na variável `cachorro`?

Os métodos disponíveis na variável cachorro será 'brincar', 'comunicar' e 'alimentar'.

9. Ao criar uma classe chamada `Aluno` e invocar que uma variável `Aluno aluno = new Aluno();` para exibir em console através de `System.out.println(aluno);`, o resultado é semelhante a `Aluno@6a6824be`. Por quê?

Quando acionamos um objeto dessa forma, a visualização não é possível, pois o objeto nada mais é que um array de informações, com chaves e dados, assim, para visualizar um dado específico é necessário identificar o que quer acessar dentro do array de informações do objeto, ex: Objeto.metodox().

10. Ao declarar uma variável do tipo `Aluno aluno = new Aluno();`, e exibi-la em console através de `System.out.println(aluno);` o resultado foi `Aluno@6a6824be`. Considerando que a classe `Aluno` possui os atributos nome e matrícula, como poderíamos definir um método que exiba corretamente essas informações em console?

Para exibir essas informações de forma prática é possível gerar um método especial chamado toString(), ele imprimirá todas as informações dos atributos presentes na classe quando acionado.