



Primeira Avaliação

1. Crie as classes a seguir, em Java, utilizando os conceitos e padrões da orientação a objetos (3.0):

1,0

Atleta
- nome : String
- habilidade : int

2,0

Equipe
- nome : String
- atletas : ArrayList<Atleta>
+ adicionarAtleta(Atleta a) : void <i>bool</i>
+ removerAtleta(String nome) : bool
+ mostrarEquipe() : void
+ habilidadeTotalEquipe() : int

- adicionarAtleta – Adiciona um novo atleta na equipe, retorna *true* se a operação foi realizar com sucesso e *false* caso contrário;
- removerAtleta – Remove o atleta que tenha o nome informado, retorna *true* se a operação foi realizar com sucesso e *false* caso contrário;
- mostrarEquipe – Mostra todos os dados da equipe: nome, quantidade de atletas e as informações completas dos atletas;
- habilidadeTotalEquipe – Retorna o somatório da habilidade de todos os atletas.

2. Assinale a alternativa que melhor define o conceito de encapsulamento:

- a) A habilidade de um objeto chamar outro objeto.
- b) A possibilidade de uma classe ter várias implementações do mesmo método.
- ☒ c) O princípio de ocultar os detalhes internos de uma classe e fornecer acesso controlado.
- d) A forma como Java implementa múltiplas heranças.

3. O modificador private em Java indica que:

- a) O atributo é visível em qualquer pacote.
- ☒ b) O atributo ou método pode ser acessado apenas dentro da mesma classe.
- c) O atributo pode ser herdado, mas não sobrescrito.
- d) O método não pode ser reescrito em subclasses.

4. O que é sobrecarga de métodos (overloading)?

- a) Usar herança para modificar métodos herdados.
- ☒ b) Criar métodos com o mesmo nome, mas com parâmetros diferentes na mesma classe.
- c) Implementar métodos abstratos em classes concretas.
- d) Alterar a visibilidade de um método herdado.

5. Quais as características fundamentais para que uma linguagem de programação seja considerada orientada a objetos?

- a) Compilação, tipagem dinâmica, e segurança.
- b) Modularização, programação funcional, tipagem fraca e desempenho.
- ☒ c) Encapsulamento, herança, polimorfismo e abstração.
- d) Tipagem estática, uso de ponteiros, recursividade e portabilidade.

6. Observe as classes a seguir e responda a questões abaixo.

```
class Curso {  
    private String nome;  
    private List<Professor> professores;  
  
    public Curso(String nome) {  
        this.nome = nome;  
        this.professores = new ArrayList<>();  
    }  
    public void adicionarProfessor(Professor p) {  
        professores.add(p);  
    }  
}  
  
class Professor {  
    private String nome;  
  
    public Professor(String nome) {  
        this.nome = nome;  
    }  
}
```

- a) Qual o tipo de relacionamento entre as classes Curso e Professor (associação, agregação ou composição)? Justifique sua resposta.
- b) O que mudaria na relação se os objetos Professor fossem criados dentro da classe Curso e não pudessem existir fora dela?