Universidade Federal do Maranhão - UFMA
Disciplina: Linguagem de Programação II – 2025.2
Professor: Tiago Bonini Borchartt
Nome:

Data: 19/01/26

7,5

Reposição P1

1. Sobre os pilares da Orientação a Objetos, assinale a alternativa que define corretamente o conceito de Encapsulamento:
a) É a capacidade de um objeto assumir diferentes formas dependendo do contexto.
b) É o mecanismo que permite criar novas classes a partir de classes existentes.
c) É a técnica de ocultar os detalhes internos de implementação de um objeto e expor apenas uma interface pública controlada.
d) É a definição de um modelo abstrato que não pode ser instanciado diretamente.
e) É a capacidade de converter tipos primitivos em objetos Wrapper automaticamente.

2. Em Java, qual é a principal diferença conceitual entre uma Classe e um Objeto?
a) A Classe é uma instância em memória, enquanto o Objeto é o código fonte.
b) A Classe é o modelo (planta baixa), enquanto o Objeto é a instância concreta criada a partir desse modelo.
c) Objetos são armazenados na memória Stack, enquanto Classes são armazenadas na memória Heap.
d) Uma Classe não pode ter métodos, apenas atributos.
e) Não há diferença, são sinônimos absolutos na linguagem Java.

3. Analise o trecho de código abaixo:
Pessoa p1 = new Pessoa("Ana");
O que o operador *new* realiza nesta linha?
a) Declara a variável p1 do tipo Pessoa.
b) Copia os dados de um objeto existente para p1.
c) Aloca memória para um novo objeto do tipo Pessoa e invoca seu construtor.
d) Verifica se a classe Pessoa existe no pacote atual.
e) Transforma a String "Ana" em um objeto genérico.

4. Sobre o método Construtor em Java, é INCORRETO afirmar:
a) Deve ter exatamente o mesmo nome da classe.
b) É executado automaticamente no momento da instanciação do objeto.
c) Pode ser sobrecarregado (ter múltiplas versões com assinaturas diferentes).
d) Deve especificar um tipo de retorno, geralmente void.
e) Se nenhum construtor for declarado explicitamente, o compilador fornece um padrão.

5. Qual modificador de acesso deve ser utilizado em um atributo de classe para garantir o encapsulamento, permitindo acesso direto apenas dentro da própria classe?
a) public
b) static
c) protected
d) private
e) final

6. O que significa declarar um atributo como *static* em uma classe Java?
a) O atributo não pode ter seu valor alterado após a inicialização.
b) O atributo pertence à classe e é compartilhado por todas as instâncias dessa classe.
c) O atributo é visível apenas dentro do pacote onde a classe foi criada.
d) O atributo deve ser obrigatoriamente um tipo primitivo.
e) O atributo só existe enquanto o método onde ele foi criado estiver sendo executado.

7. Observe a definição de relacionamento abaixo: "É um relacionamento 'Todo-Parte' forte, onde a parte não tem existência independente do todo. Se o objeto 'Todo' for destruído, as 'Partes' também são." Este conceito refere-se à:
a) Associação Simples.
b) Agregação.
c) Composição.
d) Herança.
e) Implementação.

8. Qual das alternativas melhor descreve a diferença entre Agregação e Composição?
a) Na Agregação, as partes morrem com o todo; na Composição, elas sobrevivem.
b) Na Agregação, as partes podem existir independentemente do todo; na Composição, o ciclo de vida das partes está atrelado ao todo.
c) A Agregação usa a palavra-chave extends.
d) A Composição permite herança múltipla.
e) Não há diferença prática.

9. Em um diagrama de classes, o relacionamento de Associação é geralmente representado por:
a) Uma linha contínua conectando duas classes.
b) Um losango preenchido (preto).
c) Um losango vazio (branco).
d) Uma seta com linha tracejada.
e) Um triângulo vazio.

10. A palavra-chave *this* dentro de um método de instância refere-se a:
a) A classe atual (o tipo estático).
b) O objeto atual que está executando o método.
c) A superclasse da classe atual.
d) Uma variável local do método.
e) O método main da aplicação.

11. Qual interface do *Java Collections Framework* deve ser utilizada quando precisamos de uma coleção que NÃO permita elementos duplicados?
a) List
b) Map
c) Queue
d) Set
e) Vector

12. Sobre a classe *ArrayList*, assinale a alternativa correta:
a) Garante elementos únicos.
b) É uma lista encadeada.
c) É baseada em um array redimensionável e permite acesso rápido aos elementos via índice.
d) Tem tamanho fixo, que deve ser definido no momento da criação.
e) Não permite a inserção de valores null.

13. Considere o uso de *HashMap<K, V>*. Qual afirmação é verdadeira?
a) Ele armazena os elementos na ordem em que foram inseridos.
b) Ele permite chaves duplicadas.
c) Ele utiliza um par Chave/Valor para armazenar dados e não permite chaves duplicadas.
d) É uma coleção sincronizada por padrão.
e) O acesso é feito por índice numérico sequencial.

14. Para percorrer todos os elementos de uma *List<String>* nomes, qual das sintaxes abaixo corresponde ao "*Enhanced for*" (for-each) do Java?
a) for (nomes.iterator() : String s)
b) for (String s : nomes)
c) foreach (String s in nomes)
d) for (int i = 0; i < nomes.length; i++)
e) loop (nomes as s)

15. O que são "*Generics*" (ex: <String> em List<String>) e qual sua principal vantagem?
a) São anotações para documentação.
b) Permitem que uma lista armazene qualquer tipo de objeto misturado.
c) Permitem definir o tipo dos elementos em tempo de compilação, garantindo segurança de tipos (Type Safety).
d) Servem para converter automaticamente tipos primitivos em objetos.
e) Garantem que a coleção será sincronizada.

16. Qual a diferença de performance principal entre *ArrayList* e *LinkedList*?
a) ArrayList é mais rápida para remover elementos no meio da lista.
b) ArrayList é mais rápida para busca por índice (get), enquanto LinkedList é mais eficiente para inserções/remoções frequentes.
c) LinkedList consome menos memória que ArrayList.
d) ArrayList não suporta Generics.
e) Não há diferença perceptível de performance.

17. O que acontece se tentarmos adicionar um tipo incompatível em uma lista tipada (ex: adicionar um Integer em uma *List<String>*)?
a) O código compila, mas lança exceção na execução.
b) O compilador gera um erro em tempo de compilação.
c) O Integer é convertido automaticamente para String.
d) O valor é adicionado como null.
e) O Java cria uma nova lista genérica.

18. Analise o código:
class Carro { Motor m = new Motor(); }
Este código é um exemplo clássico de implementação de qual relacionamento?
a) Associação fraca.
b) Composição.
c) Agregação.
d) Herança.
e) Polimorfismo.

19. Dado o seguinte trecho:
lista.add("A");
lista.add("A");
Qual será o tamanho da lista (assumindo que lista é um *ArrayList*)?
a) 1
b) 2
c) 3
d) Erro de compilação.
e) Lança uma exceção.

20. Para ordenar uma lista de objetos customizados (ex: *List<Aluno>*), a classe Aluno deve implementar qual interface para definir sua "ordem natural"?
a) Serializable
b) Cloneable
c) Comparator
d) Comparable
e) Iterable