Computação Orientada a Objetos - 2025

Lista de exercícios

1. Considere o cenário em que uma classe qualquer define um atributo de instância, e que seja esperado que códigos que estejam no papel de “clientes/usuários” desta classe sejam capazes tanto de obter o valor atual do atributo, quanto de alterá-lo. Por que, mesmo neste caso, é recomendável proteger o atributo, tornando-o privado, e definir *getters* e *setters* para o atributo em questão?

2. O código **Expressoes.java** declara algumas classes que podem ser usadas para representar expressões matemáticas diversas (envolvendo os operadores de soma e subtração), assim como calcular os valores de tais expressões. A partir do código apresentado, reescreva-o de modo a melhorá-lo, gerando um código mais enxuto, com menos cadeias de condicionais, e mais fácil de estender. Após a reescrita, implemente os operadores aritméticos de multiplicação e divisão.

3. Considere as seguintes classes:

```
class Lista {

  // Implementa uma lista linear de propósito geral.

  public Lista(){ ... }
  public void insere(Object obj, int indice){ ... }
  public void set(Object obj, int indice) { ... }
  public Object get(int indice) { ... }
  public Object remove(int indice) { ... }
  public int tamanho() { ... }
}

class Fila extends Lista {

  // Implementa uma fila, aproveitando a
  // implementação já existente da lista,
  // através do mecanismo de herança

  public Fila() { /*TODO*/ }
  public void insere(Object obj) { /*TODO*/ }
  public Object remove() { /*TODO*/ }
}
```

A partir das classes apresentadas:

a) Escreva as implementações do construtor e métodos da classe **Fila**. Pense na melhor implementação possível, partindo da proposta de se implementar a nova classe como uma classe derivada da classe **Lista**.
b) Por melhor que seja sua implementação do item (a), haverá uma falha de encapsulamento muito importante na classe **Fila**. Aponte onde está tal falha de encapsulamento.
c) Mantendo o uso de herança para implementar a classe **Fila**, proponha uma solução para a falha de encapsulamento.
d) Proponha uma outra solução para a falha de encapsulamento que não use herança, mas que continue reaproveitando a implementação já existente da classe **Lista**.
e) Em relação à classe **Lista**, perceba que os métodos *insere*/*set*/*get*/*remove* podem falhar caso os índices especificados não sejam índices válidos da lista. Como falhas nestas operações (devido à passagem de índices inválidos como parâmetro) poderiam ser sinalizadas para quem usa tais métodos, sem realizar mudanças no tipo de retorno dos métodos em questão?

4. Explique como cada um dos recursos abaixo disponíveis na linguagem Java possibilitam a criação de um código com melhor qualidade:

a) Possibilidade de criação de pacotes.
b) Mecanismo para tratamento de exceções.
c) Tipos genéricos.

5. Explique como **composição** pode ser usada como uma alternativa à **herança**.

6. Suponha que você esteja trabalhando na implementação de uma certa classe, e você precise disponibilizar a informação de quantas instâncias desta classe foram criadas desde o momento em que o programa iniciou sua execução, até o instante atual. Qual seria uma boa forma de implementar tal funcionalidade?

7. Você está trabalhando com um projeto Java que possui os seguintes pacotes: **projeto** (do qual faz parte o programa principal codificado no arquivo fonte **Ex7.java**), **projeto.mat** (ao qual pertencem as classes **Circulo**, **Triangulo** e **Retangulo**) e **projeto.graficos** (ao qual pertencem as classes **Poligono**, **Reta**, **Circulo**). Assumindo que as demais classes estão devidamente implementadas nos seus respectivos pacotes, explique por que o código **Ex7.java** não compila, e o que fazer para corrigi-lo?

8. Descreva as diversas maneiras de se utilizar, dentro do seu código, uma classe que pertence a um pacote diferente do pacote atual (isto é, diferente do pacote ao qual pertence a classe "cliente/usuária"). Escreva um trecho de código para ilustrar cada maneira.

9. Quais as saídas geradas pelo programa **Ex9.java**, para os seguintes pares de números digitados como entrada pelo usuário: "20 16", "8 8", "23 9.5" "10 20"? Explique o que acontece com o fluxo de execução do programa em cada um dos cenários.

10. Qual a utilidade de um bloco ***finally***?

11. Para que servem as instruções ***throw*** e ***throws*** relacionadas ao mecanismo para tratamento de exceções presente na linguagem Java?

12. Que característica uma classe deve ter para que suas instâncias possam ser lançadas?

13. Quais as diferenças que existem entre as exceções *verificadas* e *não verificadas*?

14. Imagine que você está trabalhando no desenvolvimento de uma biblioteca que encapsula um determinado conjunto de funcionalidades, a serem usadas por um outro grupo de desenvolvedores. Suponha que um dos métodos, de uma das classes desta biblioteca, só deve executar determinada ação se uma condição for satisfeita, e caso a condição não seja satisfeita uma exceção é lançada como forma de sinalizar ao usuário da classe/método que algo de errado aconteceu. Para cada possibilidade, listada nos itens (a), (b) e (c), referente ao tipo de exceção a ser lançada, discuta como ela afeta o lado dos desenvolvedores-usuários da biblioteca, e qual delas você escolheria usar:

    a) lançar uma instância do tipo **Exception**.
    b) lançar uma instância de uma classe desenvolvida por você que estenda **Exception**.
    c) lançar uma instância de uma classe desenvolvida por você que estenda **RuntimeException**.

15. Implemente uma **Fila** genérica que capaz de armazenar objetos de qualquer tipo. A classe **Fila** deve implementar o método **adiciona**, para a inserção de elementos, o método **remove**, para a remoção, e o método **tamanho** que devolve a quantidade de elementos armazenados. Internamente, sua classe deve usar um vetor (*array*) para armazenar os objetos contidos na fila.

16. Implemente um programa de teste para a classe **Fila** do exercício anterior, com os seguintes métodos genéricos: **criaFila** que recebe um vetor (*array*) de objetos genéricos e devolve uma **Fila** do tipo genérico correspondente que contenha os elementos do vetor; e **processaFila** que recebe uma **Fila** genérica e consome seus elementos, imprimindo os objetos nela guardados. Usando estes métodos genéricos, implemente um programa de teste que crie filas de diversos tipos e processe seus elementos.

17. Ainda considerando a classe de teste do exercício 16, modifique o método **processaFila** de modo que seja devolvido o elemento de menor valor presente na fila. Modifique seu programa de teste para que o elemento devolvido pelo novo método **processaFila** também seja impresso na saída do seu programa. Que tipo de modificação este novo funcionamento do método **processaFila** irá demandar?