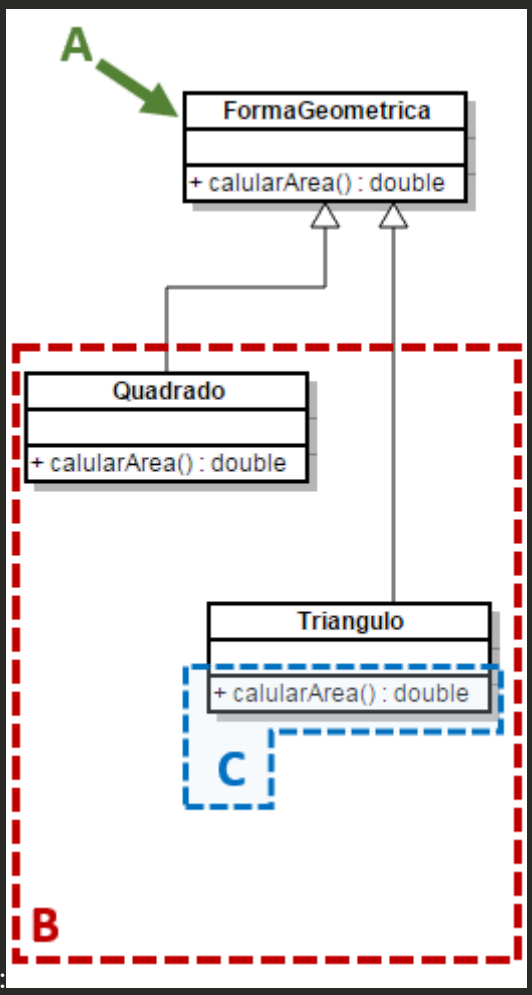
A+ Alterar modo de visualização

Peso da Avaliação 1,50
Prova 40579650
Qtd. de Questões 10
Acertos/Erros 6/4
Nota 6,00

1 Java é uma linguagem de programação interpretada orientada a objetos desenvolvida na década de 90 por uma equipe de programadores. Observe a figura anexa que ilustra uma situação típica em que podemos aplicar o conceito de herança de classes em Java. Considerando os conceitos de programação orientada a objetos, analise as sentenças a seguir:

I- A classe FormaGeometrica, indicada pela letra A, é considerada a subclasse das classes contidas na área indicada pela letra B.
II- Quadrado é uma classe filha da classe FormaGeometrica.
III- A classe Triangulo tem como superclasse a classe FormaGeometrica.



Agora, assinale a alternativa CORRETA:

A As sentenças I e II estão corretas.

B As sentenças I e III estão corretas.

C As sentenças II e III estão corretas.

D Somente a sentença I está correta.

2 A utilização do recurso de Pacotes em Java permite uma melhor organização do código-fonte, além de permitir que duas ou mais classes tenham o mesmo nome em pacotes diferentes. Juntamente com eles, o uso dos modificadores de visibilidade compõe uma fórmula poderosa no tocante à definição de segurança e encapsulamento de um programa ou aplicação. Sobre a utilização de modificadores de visibilidade na linguagem de programação Java, classifique V para as sentenças verdadeiras e F para as falsas:

( ) O modificador "default", aplicado para uma classe, permite o acesso a essa classe por uma classe de outro pacote através do comando import.
( ) O modificador "default", definido para um método contido em uma classe pública, impede que o método seja acessado por outra classe criada em outro pacote.
( ) O modificador "private", definido para um método contido em uma classe com o modificador "default", não impede que o método seja acessado por outra classe criada no mesmo pacote.
( ) O modificador "public", definido para um método contido em uma classe pública, elimina qualquer tipo de restrição de acesso ao método, mesmo que ele seja acessado em uma classe criada em outro pacote.

Assinale a alternativa que apresenta a sequência CORRETA:

A F - F - F - F.

B V - V - V - F.

C F - V - F - V.

D V - F - F - V.

3

Quando uma classe herda de outra, ela herda implementação, atributos e comportamento. Isso significa que todos os métodos e atributos disponíveis na interface externa da classe mãe estarão também na interface externa da filha (SINTES, 2002). Uma classe construída através de herança pode ter tipos importantes de métodos e atributos.

FONTE: SINTES, Anthony. Aprenda programação orientada a objetos em 21 dias. Tradução João Eduardo Nóbrega Tortello. São Paulo: Pearson Education do Brasil, 2002.

Sobre os tipos de métodos e atributos, assinale a alternativa INCORRETA:

A Sobreposto.

B Recursivo.

C Novo.

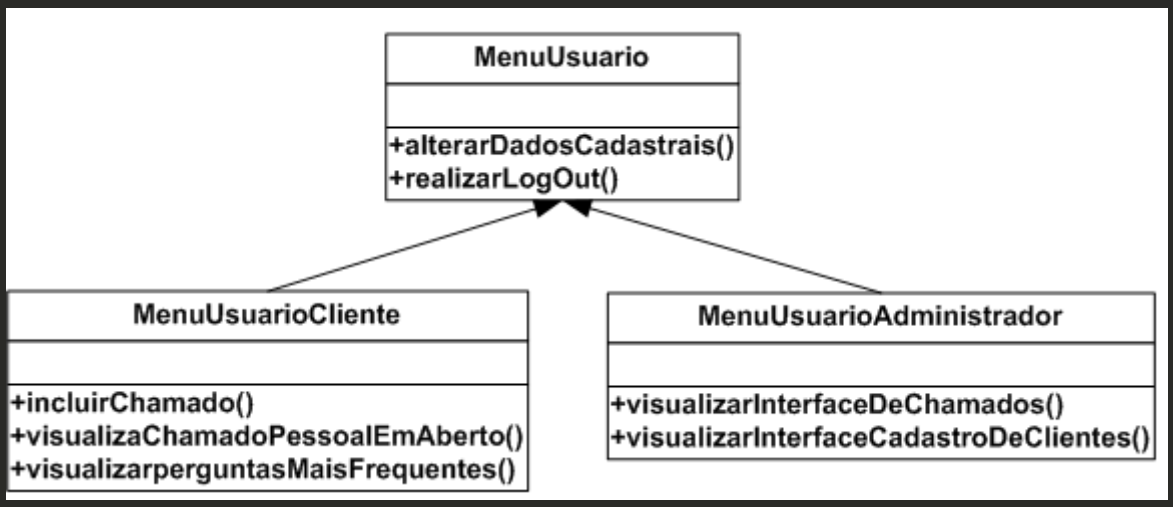
D Replicado.

4

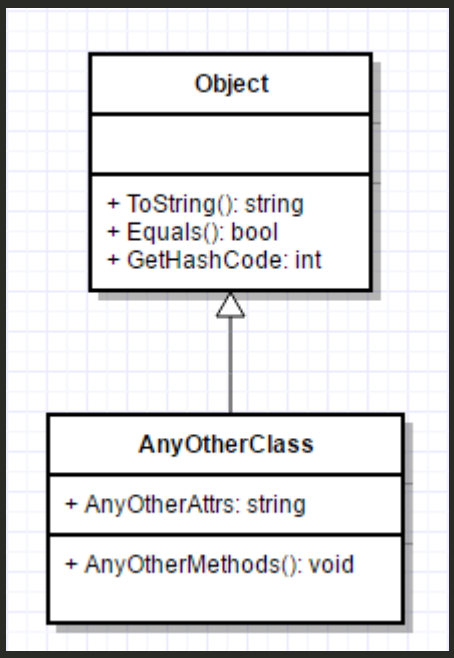
Polimorfismo é o princípio pelo qual duas ou mais classes derivadas de uma mesma superclasse podem invocar métodos que têm a mesma identificação (assinatura), mas comportamentos distintos, especializados para cada classe derivada, usando para tanto uma referência a um objeto do tipo da superclasse.

Sobre um exemplo de polimorfismo, assinale a alternativa CORRETA:

A

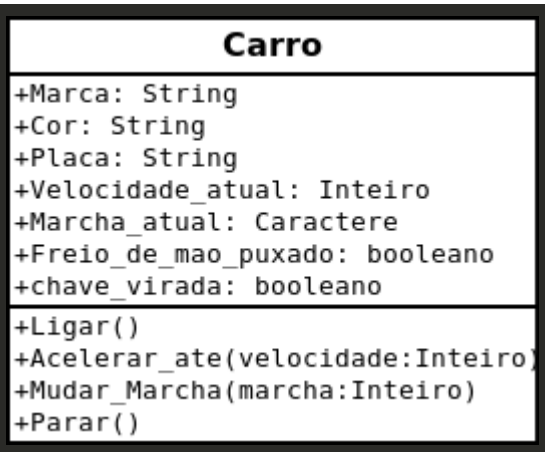


B



C Todas estão corretas!

D



5 A coesão é a característica que faz com que uma classe tenha somente uma responsabilidade dentro do sistema, o que faz com que a sua correção, em caso de erros ou a manutenção em caso de evolução, seja simplificada. No que se refere à coesão de uma classe, assinale a alternativa CORRETA:

A A coesão, embora seja uma característica importante, não auxilia o encapsulamento de uma classe.

B Tanto a coesão quanto o encapsulamento são obtidos na linguagem de programação Java através do modificador de acesso private.

C Uma classe coesa é, necessariamente, uma classe com alto acoplamento.

D Se uma classe tiver uma e somente uma responsabilidade, a probabilidade de esta sofrer manutenção diminui.

6

Existem três características mais importantes na Programação Orientada a Objetos (POO). Neste sentido, temos um conceito de programação orientada a objetos que promove a reutilização de software.

Qual é esse conceito?

A Abstração de dados.

B Herança.

C Polimorfismo.

D Sobrecarga de métodos.

7 O conceito de classes abstrata, através de suas características especiais, fornece ao conceito de herança uma maior confiabilidade uma vez que as classes modeladas com herança precisam ser melhor idealizadas e projetadas. Dessa forma, avalie o código-fonte da classe "Competidor" e classifique V para as sentenças verdadeiras e F para as falsas:

```
(1) package fontes;

(2) public class Competidor{
(3)    private int pais;
(5)    private String modalidade;
(6)    private boolean medalistaOlimpico;
(7)
(8)    public void competir(String modalidade){
(9)        System.out.println("É especialista em " + modalidade);
(10)   }
(11)
(12)   public abstract void competirPor(int pais){
(13)       System.out.println("O atleta compete por " + pais);
(14)   }
(15)
(16)}
```

( ) Existe um erro de sintaxe na linha (2) que é resolvido adicionando-se a palavra-chave "abstract", antes da palavra chave "Class".
( ) O método "competir(String modalidade)" está gerando um erro de sintaxe, e adicionando a palavra chave "abstract", antes da palavra-chave "void", resolverá o erro gerado.
( ) O método "competirPor(int pais)" está gerando um erro de sintaxe, resolvido eliminando-se a sua implementação, deixando-se apenas a sua assinatura.
( ) O método "competirPor(int pais)" está gerando um erro de sintaxe, e adicionando a palavra-chave abstract, antes da palavra- chave "Class", além de eliminar a sua implementação, deixando-se apenas a sua assinatura, resolverá o erro gerado.

Assinale a alternativa que apresenta a sequência CORRETA:

A V - V - F - V.

B V - F - F - V.

C V - V - V - F.

D F - F - V - V.

8 Classes abstratas são um recurso poderoso da linguagem Java para criação de abstrações. Utilizando classes abstratas pode-se alcançar bons níveis de reutilização de código-fonte, além de fortalecer e favorecer o conceito de polimorfismo, permitindo a reutilização e a extensão de estado e comportamento entre classes. Sobre as características das classes abstratas, analise as sentenças a seguir:

I- Classes abstratas, por via de regra, não podem gerar objetos, mas construtores abstratos, construtores especiais, podem ser utilizados por essas classes para permitir que objetos sejam criados.
II- Os métodos concretos de uma classe abstrata podem ser compartilhados, sem alterações, por todas as suas classes concretas geradas.
III- Em uma hierarquia de classes abstratas, a primeira ocorrência de uma classe concreta deve

implementar todos os métodos abstratos presentes na hierarquia.
IV- Em uma hierarquia de classes abstratas, a primeira ocorrência de uma classe concreta deve alterar todos os métodos concretos presentes na hierarquia.

Assinale a alternativa CORRETA:

A As sentenças II e III estão corretas.

B As sentenças II e IV estão corretas.

C As sentenças I e II estão corretas.

D As sentenças III e IV estão corretas.

9 A utilização de classes abstratas permite uma maior confiabilidade na definição da hierarquia de herança, pois possui características especiais que impedem sua utilização de forma errônea por outros desenvolvedores. Considerando que as palavras sublinhadas na figura a seguir representam erros de compilação, classifique V para as sentenças e F para as falsas:

( ) Bastaria colocar a palavra reservada abstract na frente da palavra class para resolver todos os problemas da classe.
( ) Uma classe abstrata pode possuir métodos concretos.
( ) Somente classes abstratas podem possuir métodos abstratos.
( ) O primeiro método imprime() deve receber um parâmetro de qualquer tipo para que seu erro de compilação seja corrigido.

Agora, assinale a alternativa que apresenta a sequência CORRETA:

```
1  package basico;
2
3  public class Abstrata {
4
5  private int codigo;
6
7  public abstract
8          String imprime(){
9      return "Olá";
10 }
11
12 public void
13     imprime(){
14     System.out.
15     println("olá");
16 }
17 }
```

A V - F - F - V.

B V - V - F - V.

C F - V - V - F.

D V - V - F - F.

10

Na Implementação em Java, encapsular, basicamente, significa ocultar. No caso específico do Java, marcamos com modificadores de visibilidade os atributos, métodos ou classes que desejamos encapsular. São quatro os modificadores de visibilidade da linguagem de programação Java: Private; Public; Default; Protect.

Sobre a definição para “Protect”, assinale a alternativa CORRETA:

A Fornece acesso dentro da aplicação onde estiver declarado, no caso de classes, atributos ou métodos. Será visível a todas as demais classes da aplicação, independentemente do pacote onde estiver. Logicamente, existe a necessidade da referenciação da classe que se deseja utilizar, caso esta esteja em outro pacote.

B O modificador default é implementado simplesmente sem colocar nenhum modificador na frente da classe, atributo ou método. Ele indica que existe visibilidade dentro do pacote onde você estiver.

C Este modificador indica que o atributo ou método será visível somente na subclasse de um relacionamento de herança.

D Fornece acesso somente dentro da classe onde estiver declarado, no caso de atributos ou métodos. Classes privadas somente fazem sentido se forem internas a outra classe.

Imprimir