


Bacharelado em Ciência da Computação		 Unifal <small>Universidade Federal de Alfenas</small>
Disciplina: Orientação a Objetos	Período: 3º	
Valor: 10 pontos (peso 0.45)	Data: 16/05/2024	
Professor: Rodrigo Martins Pagliares		
Discente:		

1. Uma classe abstrata pode definir simultaneamente métodos abstratos e não abstratos?

- a. Não, ela deve ter somente métodos de um tipo ou outro.
- b. Não, ela deve ter somente métodos abstratos.
- c. Sim, uma classe só se torna concreta quando todos os métodos abstratos de uma hierarquia de classes abstratas são implementados.
- d. Sim, uma classe abstrata com métodos concretos e abstratos é conhecida como classe híbrida.
- e. Nenhuma das alternativas.

2. Quanto às classes concretas e abstratas, selecione a alternativa correta:

- a. Tanto a classe concreta, como a abstrata, não podem ser instanciadas.
- b. Somente a classe abstrata pode ser instanciada diretamente.
- c. Tanto a classe concreta, como a abstrata, podem ser instanciadas.
- d. Somente a classe concreta pode ser instanciada diretamente.
- e. Nenhuma das demais alternativas.

3. Se uma LojaOnline possui mais de um Cliente, uma variável chamada contador é necessária para sabermos o ponto de inserção de um objeto do tipo Cliente dentro de uma array criado para armazenar os clientes da loja.

obs: ... significa código omitido, não importante na questão.

```
public class LojaOnline {
    private String nome;
    ...
    _____;

    ...
}
```

- a. private integer contador
- b. private int contador
- c. contador
- d. private contador integer
- e. Nenhuma das demais alternativas.

4. Uma transportadora possui uma frota de veículos. Supondo a correta implementação da classe `Veiculo`, qual alternativa é resultado do correto preenchimento das lacunas no código?

```
public class Transportadora {
    private _____ frota;
    public _____ contador = 0;

    public Transportadora() {
        this.frota = _____ Veiculo[100];
    }

    public boolean adicionarVeiculo( _____ veiculo) {
        _____ = veiculo;
        _____ ;
        return true;
    }

    public _____ getVeiculo(int indice) {
        return _____;
    }
}
```

- a. `Veiculo[], final int, new, Veiculo, frota[contador], contador++, Veiculo, frota[contador]`
- b. `Veiculo[], static int, new, Veiculo, this.frota[contador], contador++, Transportadora, frota[contador]`
- c. `Veiculo[], final int, new, Veiculo, frota[contador], contador++, Veiculo, frota[contador]`
- d. `Veiculo[], static int, new, Veiculo, frota[contador], contador++, Veiculo, frota[indice]`
- e. `Veiculo[], final int, new, Veiculo, frota[contador], contador, Veiculo, frota[indice]`

5. Qual o resultado ao se executar este programa? (as reticências indicam código omitido. O discente deverá assumir que este código omitido foi codificado corretamente).

```
class AtribuicaoReferencia {
    ... String nome;

    ... AtribuicaoReferencia(... nome) {
        this.nome = nome;
    }

    public void setNome(String nome) {
        this.nome = ...;
    }
    public ... getNome() {
        return nome;
    }

    public ... void manipularReferencia(AtribuicaoReferencia r1, AtribuicaoReferencia r2) {
        r1.setNome("Macaco");
        r2 = r1;
        r2.setNome("Morcego");
    }
}
```

```

public static void main(String[] args) {
    ... pet1 = new AtribuicaoReferencia("Cachorro");
    ... pet2 = new AtribuicaoReferencia("Gato");
    manipularReferencia (pet1, pet2);
    System.out.println(pet1.getNome() + "," + pet2.getNome());
}
}

```

- Exibe: Morcego, Morcego
- Exibe: Cachorro, Morcego
- Exibe: Morcego, Gato
- Exibe: Morcego, Macaco
- Nenhuma das demais alternativas.

6. Complete a(s) 2 lacuna(s) para que seja invocada de forma corretamente o método calcular

```

public class BootStrap {
    public static void main(String[] args) {
        _____;
        _____;
    }

    public void calcular() {}
}

```

- BootStrap boot = new BootStrap(); boot.calcular();
- calcular(); final;
- BootStrap.calcular(); static;
- calcular(); static;
- Nenhuma das demais alternativas.

7. Qual a alternativa melhor retrata o conceito associado ao operador instanceof em Java?

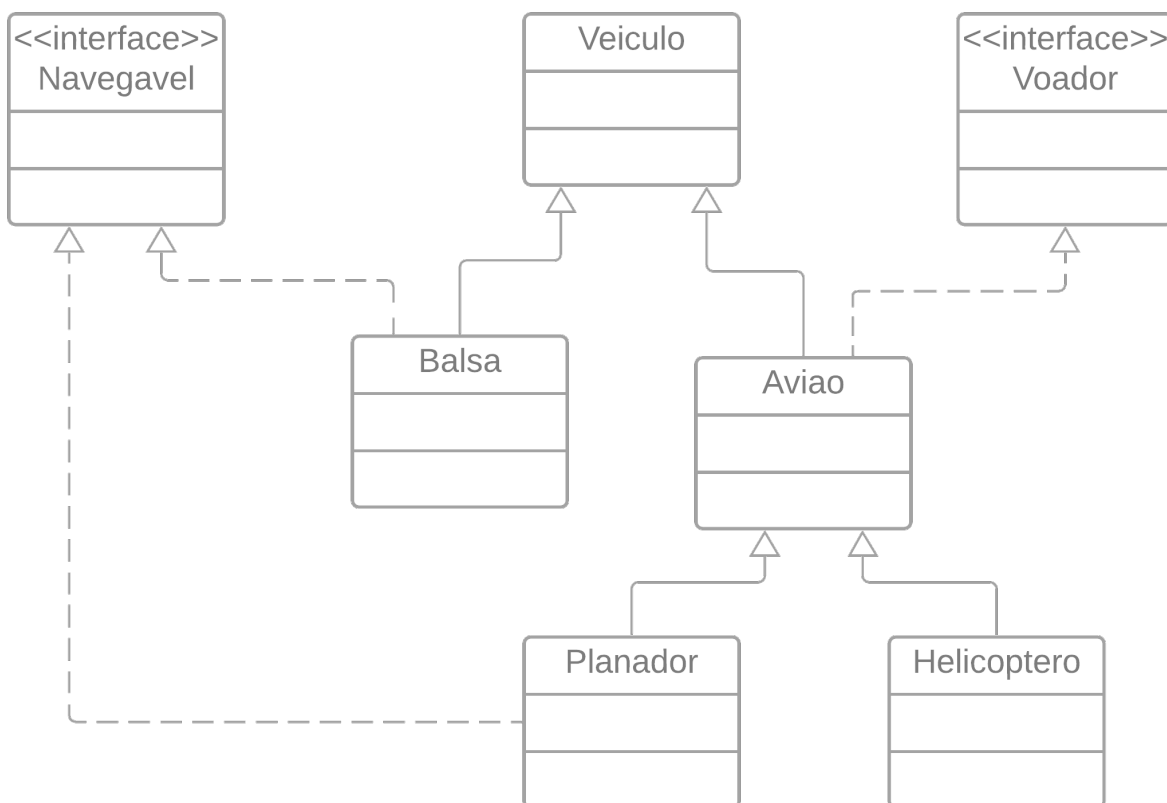
- Usamos o operador `instanceof` quando possuímos uma variável do tipo superclasse referenciando um objeto do tipo subclasse na *heap* e necessitamos invocar métodos específicos do objeto do tipo subclasse.
- Usamos o operador `instanceof` quando possuímos uma variável do tipo superclasse referenciando um objeto do tipo subclasse na *heap* e precisamos derivar o objeto da superclasse para invocação de seus métodos.
- Devemos evitar o uso do operador `instanceof` já que pelo fato de termos sobrescrita de métodos, não é necessário fazer conversão entre objetos já que o método sobrescrito na subclasse é invocado.
- Usamos o operador `instanceof` quando possuímos uma variável do tipo subclasse referenciando um objeto do tipo superclasse na *heap* e necessitamos invocar métodos específicos do objeto do tipo subclasse.
- Nenhuma das demais alternativas.

8. Para responder a questão, considere as classes x e y , tal que x estende (herda de) y . A classe x possui um método denominado `calcula` e a classe y possui um método denominado `calcula`.

Considere que o método `calcula` em x retorna um valor inteiro que é o dobro do único valor inteiro passado como parâmetro e o método `calcula` em y retorna um valor inteiro que é a soma dos dois valores inteiros passados como parâmetro. Se x e y são objetos das classes x e y , respectivamente, então a execução da expressão `x.calcula(y.calcula(10,11))` resulta no valor:

- a. 21
- b. 22
- c. 42
- d. 43
- e. Nenhuma das demais alternativas.

9. Selecione a alternativa que indica todos os tipos de objetos que podem ser referenciados por uma variável do tipo `Voador` no diagrama de classes.



- a. Objetos do tipo `Aviao`, `Planador` e `Helicoptero`.
- b. Objetos do tipo `Balsa`, `Aviao`, `Planador` e `Helicoptero`.
- c. `Voador` é uma interface. Não pode referenciar objetos concretos.
- d. Objetos do tipo `Veiculo`, `Balsa`, `Aviao`, `Planador` e `Helicoptero`.
- e. Nenhuma das demais alternativas.

10. As variáveis de instância devem ser iniciadas nestas lacunas.

```
public class Cliente {
    private String nome;
    private String sobrenome;

    public Cliente()
        ...
    }

    public Cliente(String nome) {
        ...
    }

    public Cliente(String nome, String sobrenome) {
        _____; // Esta questão
        _____; // Esta questão
    }
}
```

- a. nome;
sobrenome;
- b. this.nome = nome;
this.sobrenome = sobrenome;
- c. this(nome) = nome;
this(sobrenome) = sobrenome;
- d. nome = this.nome;
sobrenome = this.sobrenome;
- e. static nome = nome;
static sobrenome = sobrenome;

11. Transportadora é um MetodoEnvio. Qual código deverá ser inserido na lacuna para iniciação da variável de instância endereco?

```
class MetodoEnvio {

    .... double valorEnvio;

    MetodoEnvio(double valorEnvio) {

        ...

    }

}
```

```
public class Transportadora ...{

    private String endereco;

    public Transportadora(double valorEnvio, String endereco){

        ...
        _____; // esta questão
    }
}
```

- a. this.endereco = new Endereco()
- b. this(endereco)
- c. this.endereco = endereco
- d. super(valorEnvio, endereco)
- e. super(endereco)

12. Qual dessas palavras reservadas pode ser usada para prevenir que um método seja sobrescrito?

- a. static
- b. constant
- c. protected
- d. final
- e. Nenhuma das demais alternativas.

13. Sobre os pilares da Programação Orientada a Objetos (POO), analise as afirmativas abaixo e assinale a alternativa correta.

I. Polimorfismo protege o acesso direto (referência) aos atributos de uma instância, fora da classe onde estes foram declarados.

II. Herança é usada na intenção de reaproveitar código ou comportamento generalizado ou especializar operações ou atributos.

III. Encapsulamento permite que referências de tipos de classes mais abstratas representem o comportamento das classes concretas que referenciam.

- a. Apenas a afirmativa II está correta
- b. Apenas a afirmativa I está correta
- c. Apenas as afirmativas I e III estão corretas
- d. Apenas as afirmativas I e II estão corretas
- e. Nenhuma das demais alternativas.

14. Assinale a alternativa correta com relação à programação orientada a objetos:

- a. O polimorfismo é um mecanismo da Orientação a Objetos que permite criar novas classes a partir da composição de classes já existentes, usando delegação para aproveitar características existentes na classe a ser composta.
- b. A herança 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.
- c. O polimorfismo serve para controlar o acesso aos atributos e métodos de uma classe. É uma forma eficiente de proteger os dados manipulados dentro da classe, além de determinar onde esta classe poderá ser manipulada.
- d. A herança é um mecanismo da Orientação a Objetos que permite criar novas classes a partir de classes já existentes, aproveitando-se das características existentes na classe a ser estendida.
- e. O encapsulamento 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.

15. Ambas classes, `Veiculo` e sua subclasse `Carro`, têm um método `acelerar()`. Se `v` é uma variável do tipo `Veiculo` que aponta para um objeto do tipo `Carro`, qual será o resultado da execução do código abaixo:

```
v.acelerar ();
```

- a. O método `acelerar()` definido em `Veiculo` será executado.
- b. O método `acelerar()` definido em `Carro` será executado.
- c. O compilador irá reclamar que o método `acelerar()` foi definido duas vezes.
- d. O compilador irá usar as regras de *overloading* (*sobrecarga*) para decidir qual método invocar.
- e. Nenhuma das demais alternativas.

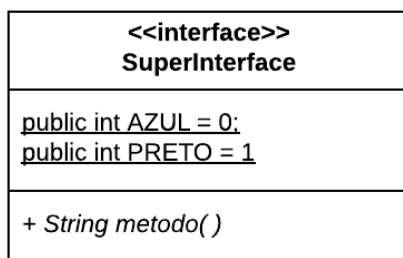
16. Suponha que a classe `Roedor` tem uma classe filha `Rato` e outra classe filha `Camundongo`. A classe `Camundongo` tem uma classe filha `CamundongoDeBolso`. Observe o seguinte código:

```
Roedor roedor;  
Rato rato = new Rato();  
Camundongo camundongo = new Camundongo();  
CamundongoDeBolso cdb = new CamundongoDeBolso();
```

Qual das seguintes atribuições causará um erro de compilação?

- a. `roedor = rato;`
- b. `roedor = camundongo;`
- c. `cdb = null;`
- d. `cdb = rato;`
- e. Nenhuma das demais alternativas.

17. Crie constantes conforme diagrama de classes



```
public interface SuperInterface {
```

```
_____ // Esta questão
```

```
_____ // Esta questão
```

```
    public String metodo() ;
```

```
}
```

- a. public static final int AZUL = 0;
 public static final int PRETO = 1;
- b. public static int AZUL = 0;
 public static int PRETO = 1;
- c. public abstract final int AZUL = 0;
 public abstract final int PRETO = 1;
- d. public protected final int AZUL = 0;
 public protected final int PRETO = 1;
- e. public package final int AZUL = 0;
 public package final int PRETO = 1;

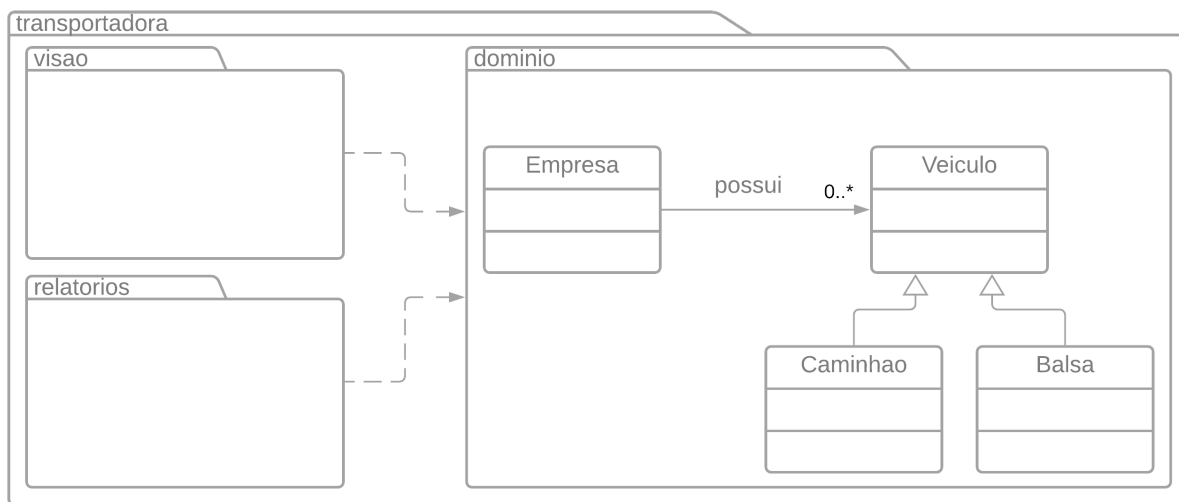
18. Qual é a restrição existente ao uso da referência *super* em um construtor?

- a. Ela somente pode ser usada no construtor da superclasse.
- b. Somente uma subclasse pode usá-la.
- c. Ela deve ser a última instrução do construtor.
- d. Ela deve ser a primeira instrução do construtor.
- e. Nenhuma das demais alternativas.

19. A Programação Orientada a Objetos tem alguns mecanismos para garantir maior segurança no código. Mais especificamente, a recomendação é utilizar a visibilidade `private` para os atributos de uma classe. Assim, considerando que uma classe tenha sido criada com atributos de visibilidade `private` indique a alternativa correta:

- a. Atributos e métodos são acessíveis somente dentro das classes que pertençam ao mesmo pacote.
- b. Aqueles que tenham acesso à classe terão acesso também a qualquer membro.
- c. Atributos e métodos são acessíveis dentro da própria classe, das subclasses e das classes que façam parte do mesmo pacote.
- d. A classe pode ser instanciada por qualquer outra classe.
- e. o acesso aos atributos dessa classe deve ser feito somente por métodos definidos na mesma classe que contêm os atributos desejados.

20. Qual a instrução em Java representa a correta definição de pacote para a classe `Balsa` no diagrama de classes abaixo?



- a. package transportadora
- b. package dominio.transportadora
- c. package transportadora.dominio
- d. package transportadora.visao.relatorios.dominio
- e. Nenhuma das demais alternativas.

Boa Prova !
Rodrigo Martins Pagliares