

1º Trabalho de Programação Orientada a Objetos  
Curso de Engenharia de Computação - UEMG Ituiutaba  
1º Semestre de 2019

Aluno(a): \_\_\_\_\_

Aluno(a): \_\_\_\_\_

1. Na programação orientada a objetos, são características dos objetos:  

(a) As classes, os métodos e as mensagens.	(d) A instanciación, a generalização e a especialização.
(b) A identidade, os atributos e as operações.	
(c) O encapsulamento, a herança e o polimorfismo.	(e) A classificação, a composição e a decomposição.
2. Em programação orientada a objetos, as propriedades que definem a estrutura e o comportamento de um objeto são especificadas para a classe da qual o objeto é instância e são válidas para alguns objetos dessa classe. Assinalar **Certo** ou **Errado**. ( ) Certo ( ) Errado
3. Na programação orientada a objetos, objetos que possuam operações (métodos) que se comportem da mesma maneira, mesmo que possuam atributos diferentes, podem ser agrupados em uma mesma classe. Assinalar **Certo** ou **Errado**. ( ) Certo ( ) Errado
4. O encapsulamento em Java somente pode ser realizado por meio do modificador de acesso *public*. Assinalar **Certo** ou **Errado**. ( ) Certo ( ) Errado
5. Os três elementos básicos quando contidos num arquivo fonte Java devem obrigatoriamente se apresentar na seguinte ordem:  

(a) import, package e class.	(c) class, import e package.	(e) package, import e class.
(b) class, package e import.	(d) package, class e import.	
6. A **JVM** mais o núcleo de classes da plataforma **Java** e os arquivos de suporte formam  

(a) o J2EE.	(b) o JDK.	(c) o JRE.	(d) uma JSP.	(e) uma API.
-------------	------------	------------	--------------	--------------
7. Sobre a criação de objetos na linguagem de programação **Java**, analise as assertivas e assinale a alternativa que aponta a(s) **correta(s)**.
  - I. Nesta linguagem de programação, a criação de objetos é gerenciada por um algoritmo de escalonamento, onde característica como prioridade, acesso a recurso são considerados.
  - II. Objetos são criados por expressões contendo a palavra-chave *new*. Criar um objeto a partir de uma definição de classe é também conhecido como instanciación; assim, objetos são muitas vezes chamados de instâncias.
  - III. Referências a objetos são *null* quando elas não referenciam algum objeto.
  - IV. Objetos recentemente criados são colocados em uma área de memória do sistema conhecido como *heap*. Todos os objetos são acessados via referências a objetos – qualquer variável que possa aparentar um objeto, na realidade, contém uma referência àquele objeto.

- (a) Apenas I.
- (b) Apenas I, II e III.
- (c) Apenas I, III e IV.
- (d) Apenas II, III e IV.
- (e) I, II, III e IV.

8. Acerca de **construtores** em **Java**, marque a alternativa correta.

- (a) Quando não é declarado nenhum construtor na classe, a **JVM** o cria. Esse construtor é o construtor opcional, ele não recebe nenhum argumento e o corpo dele é vazio. A partir do momento que o construtor é declarado, o construtor *default* não é mais fornecido.
- (b) Um construtor é um método especial, já que possui retorno.
- (c) O construtor dar a possibilidade ou obriga o usuário de uma classe a passar argumentos para o objeto durante o processo de criação do mesmo.
- (d) Os construtores permitem modificar suas saídas.
- (e) O construtor de uma classe pode ser invocado a qualquer momento.

9. Acerca do uso do *this* em **Java**, analise as seguintes afirmativas:

- I. O *this* é utilizado pelo objeto para acessar uma referência a si próprio.
- II. O *this* é utilizado em métodos não-estáticos.
- III. O *this* é aceito em métodos estáticos.

Podemos afirmar que:

- (a) Todas as afirmativas estão corretas.
- (b) Todas as afirmativas estão incorretas.
- (c) Apenas a afirmativas I e II estão corretas.
- (d) Apenas as afirmativas II e III estão corretas.

10. Sobre a linguagem de programação **Java**, analise as assertivas e assinale a alternativa que aponta a(s) correta(s).

- I. Nesta linguagem de programação, programas são construídos a partir de classes. A partir de uma definição de classe, podemos criar qualquer quantidade de objetos, que são conhecidos como instâncias daquela classe.
- II. Uma classe, nesta linguagem de programação, contém membros, sendo campos e métodos as principais espécies. Campos (atributos) são variáveis de dados que pertencem ou à própria classe ou a objetos da classe; eles constituem o estado do objeto ou classe.
- III. Encontramos também, em uma classe **Java**, métodos. Métodos são coleções de comandos que operam sobre os campos para manipular o estado. Comandos definem o comportamento de classes; eles podem atribuir valores a campos e outras variáveis, avaliar expressões aritméticas, invocar métodos e controlar o fluxo de execução.
- IV. Uma classe, nesta linguagem, pode ser compilada para *bytecodes*.

- (a) Apenas I.
- (b) Apenas I, II e III.
- (c) Apenas I, III e IV.
- (d) Apenas II, III e IV.
- (e) I, II, III e IV.

11. Observe o código abaixo, em **Java**:

```
public class Conta {
    private double saldo;
    private double limite;
    private Cliente titular;

    public double getSaldo() {
        return this.saldo + this.limite;
    }

    public getTitular() {
        return titular;
    }

    public void setTitular( Cliente titular ) {
        this.titular = titular;
    }
}
```

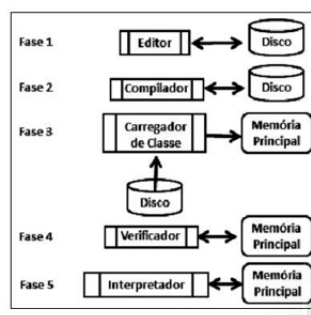
Acerca do código acima, analise as seguintes afirmativas:

- I. O código não possibilita a invocação do método **getLimite()**.
- II. O método **setTitular()** permite a exibição do titular da classe **Conta**.
- III. O método **getTitular()** permite alterar o titular da classe **Conta**.

Podemos afirmar que:

- (a) Todas as afirmativas estão corretas.
- (b) Todas as afirmativas estão incorretas.
- (c) Apenas a afirmativa I está correta.
- (d) Apenas a afirmativa II está correta.
- (e) Apenas a afirmativa III está correta.

12. Observe a figura a seguir que representa o ambiente **Java** típico.



Analise as afirmativas correspondentes a duas atividades desenvolvidas em duas fases.

- I. A partir do código fonte, os *bytecodes* são criados.
- II. Os *bytecodes* são lidos e traduzidos para uma linguagem que o computador pode entender, possivelmente armazenando valores de dados enquanto executa o programa.

As atividades descritas são realizadas, respectivamente, nas seguintes fases

- (a) 2 e 4.
- (b) 2 e 5.
- (c) 3 e 4.
- (d) 3 e 5.
- (e) 1 e 2.

13. **Seja o compilador.** Cada um dos arquivos **Java** desta questão representa um programa fonte completo. Sua tarefa é personificar o compilador e determinar se cada um deles pode ser compilado. Se não puderem ser compilados, como Você os corrigiria, e se eles forem compilados, qual seria sua saída.

```
1: // Arquivo: Semaforo.java
2: package questao14.classe;
3: public class Semaforo {
4:     private int vermelhoVerdeAmarelo;
5:
6:     public Semaforo() {
7:         vermelhoVerdeAmarelo = 0;    // 0=vermelho,1=verde,2=amarelo
8:     }
9:
10:    public void alternarSinal() {
11:        vermelhoVerdeAmarelo++;
12:        vermelhoVerdeAmarelo = vermelhoVerdeAmarelo % 3;
13:    }
14:
15:    public void sinalAtual() {
16:        System.out.println("Sinal: " + vermelhoVerdeAmarelo )
17:    }
18:}
```

```
1: // Arquivo: UsaSemaforo.java
2: package questao14.aplicacao;
3: public class UsaSemaforo {
4:     public static void main( String[] args ) {
5:         semaforo.alternarSinal();
6:         semaforo.alternarSinal();
7:         semaforo.sinalAtual();
8:     }
9: }
```

**OBS:** as linhas numeradas nas classes são apenas para facilitar a(s) referência(s) na resolução da questão.