Aluno(a): _____

Observação: Resolver 8 das 9 questões propostas.

1. Na **orientação a objetos**, o conceito de **encapsulamento** corresponde à propriedade de

   (a) receber, por uma classe, uma mensagem sem parâmetros.
   (b) utilizar estruturas de matrizes quadradas nos programas desenvolvidos.
   (c) usar variáveis e constantes do tipo inteiro nos métodos das classes implementadas.
   (d) esconder ou ocultar detalhes da implementação de uma dada classe de outras classes.
   (e) ter um conjunto de objetos com a mesma classe.

2. O município de Ituiutaba possui uma população aproximada de 104.067 (estimativa IBGE 2018) de habitantes, e teve um aumento médio de mais de 3.000 novos habitantes na última década. Na associação da frase acima aos conceitos da modelagem orientada a objetos, é correto afirmar que **Ituiutaba**, **população** e **aumento médio**, referem-se, respectivamente, a

   (a) classe, objeto, instância de classe.
   (b) objeto, atributo, implementação por um método da classe do respectivo objeto.
   (c) classe, objeto, atributo.
   (d) objeto, instância, operação.
   (e) classe, objeto, associação pelo método de agregação.

3. Uma classe **Java** pode ser instanciada por um comando, cuja sintaxe é

   (a) nomeObjeto nomeClasse = new nomeObjeto();
   (b) nomeClasse nomeObjeto = new nomeClasse();
   (c) nomeClasse nomeinstancia = new nomeObjeto();
   (d) nomeInstancia nomeObjeto = new nomeInstancia();
   (e) nomeInstancia nomeClasse = new nomeInstancia();

4. O método **construtor** é um tipo especial de rotina que toda classe **Java** possui. É uma característica de todo método **construtor** na linguagem **Java**:

   (a) obrigatoriedade de sua declaração.
   (b) desnecessária alocação de memória para sua execução.
   (c) atribuição de nome diferente da classe a que pertence.
   (d) ausência de especificação de tipo de retorno.

5. A plataforma **Java** disponibiliza um interpretador que traduz, em tempo de execução, o *bytecode* para instruções nativas do processador, permitindo, dessa forma, que uma mesma aplicação seja executada em qualquer plataforma computacional que possua essa implementação. Trata-se de

   (a) JavaBeans.
   (b) Java API.
   (c) JVM
   (d) JDK
   (e) JAVA VIRTUAL MAQUINE

6. Considere a classe Java abaixo.

```
public class Processo {
        private String numeroProcesso;
}
```

   Um desenvolvedor **Java** afirma, corretamente, que:

   (a) para incluir um valor no atributo **numeroProcesso** através de um objeto dessa classe será necessário criar um método privado **getNumeroProcesso**.
   (b) a instrução **Processo p = new Processo("10453");** instancia um objeto dessa classe utilizando o construtor padrão (*default*).

(c) para permitir encapsulamento, os novos atributos e métodos a serem incluídos nessa classe terão que ser privados.

(d) poderão ser incluídos nessa classe um construtor que não recebe parâmetros e um construtor que recebe como parâmetro o número do processo.

(e) não será possível instanciar um objeto dessa classe, pois ela não tem construtor.

7. Para executar um programa **Java** deve ocorrer um processo que envolve compilação e interpretação. Quando se compila uma classe com extensão **.java** é gerado um arquivo com extensão

(a) **.class**, conhecido como *bytecode*, que pode ser compilado pela **JVM**.

(b) **.jar**, conhecido como *bytecode*, que pode ser lido pela **JVM**.

(c) **.class**, que instala a classe na memória virtual para ser executada.

(d) **.jar**, que quando executado, cria um arquivo **.class**, que é interpretado pela **JVM**.

(e) **.class**, conhecido como *bytecode*, que pode ser interpretado pela **JVM**.

8. Métodos **sobrecarregados** de uma classe são distinguidos por um compilador **Java** por meio

(a) da observação de seus tipos de retorno.

(b) da análise de suas assinaturas.

(c) da combinação de seus respectivos nomes e números de parâmetros.

(d) da comparação de seus respectivos tipos e ordem de parâmetros.

9. Analise o seguinte trecho de código abaixo em **Java**:

```
1  package provaprof2011;
2
3  public class Carro {
4      private String placa;
5      private int renavam;
6      private String cor;
7      // Primeiro construtor
8      public Carro(String p, int r){
9          placa = p;
10         renavam = r;
11     }
12     // Segundo construtor
13     public Carro(String c, int r){
14         cor = c;
15         renavam = r;
16     }
17     // Terceiro construtor
18     public Carro(int r, String c){
19         cor = c;
20         renavam = r;
21     }
22 }
```

Em relação ao trecho de código, é **CORRETO** afirmar que.

(a) A classe **Carro** contém erro(s). O segundo construtor e o terceiro construtor inicializam os mesmos atributos da classe.

(b) A classe **Carro** contém erro(s). Nenhum construtor inicializa todos os atributos da classe.

(c) A classe **Carro** não contém erros e será compilada com sucesso.

(d) A classe **Carro** contém erro(s). O primeiro construtor e o segundo construtor tem a mesma assinatura.

(e) A classe **Carro** é uma classe aplicação.