**LISTA DE EXERCÍCIOS - ORIENTAÇÃO A OBJETOS COM JAVA**

**08.** Sobre as características da Orientação a Objetos em Java, assinale verdadeiro

( V) ou falso (F ):

|  |  |
| --- | --- |
| V | Java possui menos de 9 tipos primitivos. |
| F | Não é possível comparar tipos primitivos utilizando o operador “==”. |
| V | Atributos de interface são sempre *final*. |
| V | Palavras reservadas do Java são aquelas que não podem ser usadas para nomear membros de classes. |
| V | Em Java toda variável, sem exceção, deve ser declarada. |
| V | As subclasses podem adicionar membros próprios. |
| V | Variáveis polimórficas podem referenciar objetos de uma classe ( subclasse) da superclasse declarada. |
| F | Em métodos, o modificador *abstract* obriga que suas subclasses não abstratas implementem o método. |
| F | Um método definido como *final* pode ser sobreposto apenas por uma classe descendente. |
| F | Classe com modificador *final* só pode ser especializada por herança. |

|  |  |
| --- | --- |
| V | O modificador *final* estabelece que um atributo não pode ter seu valor modificado. |
| V | A visibilidade definida por *public* permite que um determinado atributo seja acessível a partir de quaisquer métodos, objetos e classes. |
| F | O modificador *protected* não restringe acesso oriundo de outro pacote. |
| V | Os atributos e métodos privados de uma classe são acessíveis apenas nos métodos da própria classe. |
| V | A estrutura *switch* aceita qualquer tipo de dado primitivo do Java. |
| V | Encapsulamento consiste em proteger os atributos de acessos e modificações não controladas, centralizando o gerenciamento e a validação dos dados antes de serem armazenados pelos objetos. |
| V | O encapsulamento permite definir o grau de visibilidade dos atributos de uma classe, estabelecendo restrições e permissões por métodos ao sistema. |
| V | Não se implementa o encapsulamento em interface. |
| F | A implementação de uma interface obriga a classe a implementar todos os métodos definidos, a não ser que a classe seja definida como abstrata, podendo assim deixar a implementação para as suas subclasses não abstratas. |
| V | O polimorfismo de sobrecarga pode ser utilizado para distinguir, em uma determinada classe, dois métodos com o mesmo nome, mas com parâmetros diferentes. |
| V | Polimorfismo é o conceito que define que mais vários métodos, com o mesmo nome, podem Implementar diferentes formas de executar, dependendo de como ele é acionado. |
| F | *Override* é um tipo de Polimorfismo que só ocorre em caso de herança. |
| V | Na herança, todos os atributos são herdados, inclusive os privados. |
| V | Em Java as subclasses herdam atributos e métodos da classe *Object*. |
| F | Uma classe Java pode herdar de uma única classe na herança simples, e de várias na herança múltipla. |
| V | Além de herdar entidades de sua classe-pal, uma classe derivada pode modificar métodos herdados, inclusive podendo até acrescentar novas entidades, sem afetar a estrutura da classe que a originou. |

**09.** Marque a afirmativa **incorreta**:

1. Encapsulamento representa a proteção das propriedades referentes a um determinado objeto.
2. O encapsulamento permite concentrar as regras de negócio da classe dentro dela mesma.
3. Encapsulamento representa a proteção das propriedades referentes a um determinado objeto.
4. **O encapsulamento é obrigatório para os atributos de uma classe.**
5. Um atributo encapsulado não pode ser acessado diretamente por outro objeto ou classe.
6. Os métodos de acesso do encapsulamento (get/set) não devem ser privados.

**10.** A habilidade de duas ou mais classes, derivadas da mesma superclasse, responderem à mesma solicitação cada qual à sua maneira, é o conceito de:

1. Simetria
2. Abstração
3. **Polimorfismo**
4. Encapsulamento

(E)Reutilização

( F) Herança

1. A respeito das palavras reservadas do Java, associe a coluna da esquerda à coluna da direita.

|  |  |
| --- | --- |
| (A) super | (F) Método sem retorno |
| (B) default | (S) Define constantes |
| (C) new | (G) Tipo de dado |
| (D) this | (R) Implementa herança |
| (E) throw | (T) Declara importação |
| (F) void | (H) Implementa uma interface |
| (G) int | (J) Classe |
| (H) implements | (I) Incremento |
| (I) ++ | (Q) Laço de repetição |
| (J) class | (E) Levanta uma exceção |
| (K) get | (K) Sem significador |
| (L) static | (C) Instancia a class |
| (M) public | (N) Encerra a execução do método |
| (N) return | (O) Define membro de classe |
| (O) String | (P) Definição de uma classe |
| (P) abstract | (D) Auto referência |
| (Q) do | (A) Referencia classe pai |
| (R) extends | (L) Define classe sem objeto |
| (S) final | (B) *Else* do switch |
| (T) import | (M) Modificador de visibilidade |

1. Define com suas palavras o que se pede:

(A)Classe: Conjunto de métodos e atributos.

(B)Objeto: É o fruto derivado de uma classe após ela ser instanciada.

1. O que compõe a assinatura de um método? Exemplifique.

Modificador de acesso – Tipo de permissão de acesso daquele método (public, private, etc)

Tipo de retorno – Qual o tipo de dado que aquele método vai retornar (void, int, String, etc)

Nome do método – Qual o nome que usaremos para chamar aquele método em nosso programa.

Parâmetros – As variáveis que vão está passando os dados que serão usados no método, isso somente no caso de métodos com essa característica.

Exceções – Informa um possível erro que pode acontecer durante a execução do metodo.