Para encapsular um atributo, deixando-o visível apenas para a classe que o contém, utilizamos qual palavra-chave?

Private.

Para encapsularmos um atributo, utilizamos a palavra-chave private; com isso, o mesmo ficará visível apenas para classe da qual ele é membro

Método especial destinado ao preparo de novos objetos durante sua instanciação. Pode ser acionado por meio do operador new, recebendo parâmetros como métodos comuns, o que permite caracterizar os objetos já na instanciação. Trata-se de

A resposta correta é:

construtor.

O modelo de sistema que representa a realidade em termos de objetos é chamado de Programação Orientada a Objetos (POO). Nessa modelagem, temos os objetos que têm propriedades e comportamentos. Dentre as alternativas a seguir, assinale aquela que melhor define o conceito de encapsulamento.

A resposta correta é:

Encapsulamento é o mecanismo utilizado com a finalidade de esconder os detalhes de implementação para garantir maior segurança e domínio da complexidade.

Sobre objetos, podemos afirmar que:

A resposta correta é: São as representações de uma classe com características e ações.

Criar uma classe, em Java, chamada Idade que possui o método calcular Idade que recebe como parâmetro o ano atual e o ano de nascimento de uma pessoa e retorna a idade dessa pessoa. OBS: Criar apenas essa classe

public class Main {

      public static void main(String[] args) {

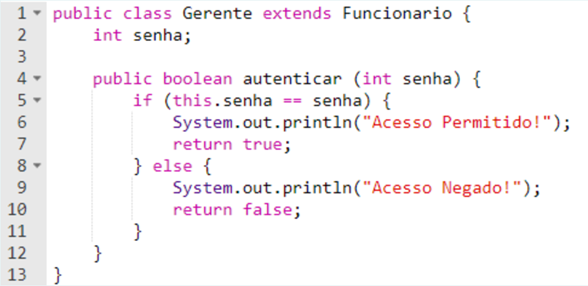
          Idade idadeObjeto = new Idade();

          int idade = idadeObjeto.calcularIdade(2023, 1993);                                  System.out.println("A idade é: " + idade);

      }

}

Considere o trecho de código em Java:



Observando-se os conceitos de orientação a objetos, expostos no trecho em Java,

A resposta correta é:

a classe Gerente herda todos os atributos e métodos da classe Funcionario. Isso é expresso pelo uso da palavra chave extends.

Uma das técnicas de programação muito utilizada dentro do paradigma da Orientação a Objetos é o encapsulamento, onde você como programador deverá escolher quando utilizá-la. O real motivo de trabalharmos com esta técnica é:

A resposta correta é:

Usar como Auxílio para impedir que classes externas alterem diretamente o estado e o comportamento do objeto.

O paradigma de programação orientado a objetos tem sido largamente utilizado no desenvolvimento de sistemas. Considerando o conceito de herança, avalie as afirmações abaixo.

I. Herança é uma propriedade que facilita a implementação de reuso.

II. Quando uma subclasse é criada, essa herda todas as características da superclasse, não podendo possuir propriedades e métodos próprios.

III. Herança múltipla é uma propriedade na qual uma superclasse possui diversas subclasses.

IV . Extensão é uma das formas de se implementar herança.

É correto apenas o que se afirma em:

A resposta correta é:

I e IV.

Na programaçãoorientada a objetos,  polimorfismo significa "muitas formas", é o termo definido em linguagens orientadas a objeto, como por exemplo Java, C# e C++, que permite ao desenvolvedor usar o mesmo elemento de formas diferentes. Um tipo de polimorfismo é

A resposta correta é:

polimorfismo de sobrecarga.

O modificador FINAL, quando aplicado no atributo da classe o torna uma constante e, só pode ser inicializado uma única vez, ou diretamente na declaração da classe ou no método construtor. No entanto, esse operador pode ser utilizado em métodos e classes. Qual alteração no método o operador final pode gerar?

**Texto de resposta**

Quando o modificador final é aplicado a um método, ele indica que o método não pode ser sobrescrito por subclasses. Isso significa que a implementação do método na classe em que ele é declarado é a versão final e não pode ser alterada nas subclasses.

Quando um método é declarado como final, ele impede que as subclasses substituam ou modifiquem seu comportamento. Isso pode ser útil em situações em que se deseja garantir que o comportamento de um método não seja alterado nas subclasses, mantendo a consistência e a integridade do código.

É importante notar que o modificador final pode ser aplicado tanto a métodos quanto a classes e atributos, mas cada um tem um efeito diferente:

* final em um método: Impede que o método seja sobrescrito nas subclasses.
* final em uma classe: Impede que a classe seja estendida, ou seja, não pode ter subclasses.
* final em um atributo: Torna o atributo uma constante, ou seja, seu valor não pode ser alterado após a inicialização.

Portanto, a alteração que o operador final pode gerar em um método é impedir que ele seja sobrescrito nas subclasses. Isso garante a manutenção do comportamento definido na classe original e evita que o método seja modificado ou substituído de forma indesejada.

Um conceito utilizado em POO no qual a mesma assinatura de um método seja utilizado para implementar, por exemplo, três versões diferentes desse método, cada uma com um número distinto de argumentos entre si. Essa definição corresponde ao conceito de

A resposta correta é:

Sobrecarga

Sobre [interfaces](https://educacaoonline.unifametro.edu.br/mod/lti/view.php?id=328209) em Java, responda VERDADEIRO ou FALSO.

[Interfaces](https://educacaoonline.unifametro.edu.br/mod/lti/view.php?id=379580) são classes 100% abstratas. → VERDADEIRO,

Nas [Interfaces](https://educacaoonline.unifametro.edu.br/mod/lti/view.php?id=379580), os atributos são implicitamente private, static, final. → FALSO,

Nas [Interfaces](https://educacaoonline.unifametro.edu.br/mod/lti/view.php?id=328209), os métodos são implicitamente public abstract. → VERDADEIRO.

Escreva **V** ou **F** conforme seja verdadeiro ou falso o que se afirma a seguir sobre tratamento de exceção:  
(   ) Para lidar com uma exceção, deve-se colocar qualquer código que pode lançar uma exceção em uma instrução *catch*..

(   ) É possível ter vários blocos *catch* para a mesma cláusula *try* para tratar diferentes exceções.

(   ) É possível declarar mais de uma exceção na cláusula *throws*.

(   ) Se o desenvolvedor usa o bloco *try-catch* para tratar uma ou mais exceções em um método, ele não pode mais usar a cláusula *throws* na assinatura do mesmo método para lançar exceções.  
(   ) Caso queira lançar uma exceção em seu programa, o programador deverá utilizar a palavra reservada throw seguida da exceção a ser lançada.

A resposta correta é:

V F V F V.

Na linguagem Java, a sintaxe correta de um bloco de controle de exceção é:

A resposta correta é:

try { // código a ser executado } catch (TipoExcecao nomeExcecao) { // tratamento da exceção }

Na linguagem de programação Java, a exceção que ocorre quando tentamos indexar um array além de seus limites. *Qual e a exceção dada:*

A resposta correta é:

ArrayIndexOutOfBoundsException.

Sobre os tipos genéricos *(generics)* em Java, analise as afirmações abaixo.

1) Tipos genéricos permitem abstrair sobre tipos. Os exemplos mais comuns são os tipos coleções.

2) Métodos genéricos são métodos que introduzem os seus próprios parâmetros de tipo. Isso é semelhante ao declarar um tipo genérico, mas o escopo do parâmetro de tipo é limitado ao método em que é declarado.

3) Métodos genéricos estáticos são permitidos, bem como construtores de classe genéricos. Porém, métodos genéricos não estáticos não são permitidos.

4) Entre as vantagens dos tipos genéricos, temos a simplicidade e a robustez. Por um lado, não é necessário *type casting* para extrair os objetos das coleções. Por outro, o compilador não permite colocar na coleção elementos incompatíveis com os tipos declarados.

5) Os parâmetros de tipo de um tipo genérico podem ser ou não limitados. O limite de um parâmetro de tipo restringe os tipos que podem ser usados como argumento. O parâmetro de tipo limitado dá acesso aos métodos do “tipo limite”.

6) os tipos genéricos (Generics) têm o propósito de criar conjuntos com consistência entre os tipos.

Qual das afirmativas acima não está correta? Justifique a sua resposta.

3) Métodos genéricos estáticos são permitidos, bem como construtores de classe genéricos. Porém, métodos genéricos não estáticos não são permitidos.

Em Java, métodos genéricos não estão restritos a serem apenas estáticos. É perfeitamente possível ter métodos genéricos não estáticos em classes genéricas. A restrição se aplica apenas a variáveis de instância não estáticas.