Lista das palavras reservadas

**Modificadores de acesso:**

. private, protected e public

**Modificadores de classes, variaveis ou métodos:**

. abstract, class, extends, final, implements, interface, native, new, static, strictfp, synchronized, transient, volatile

**Controle de fluxo:**

. break, case, continue, default, do, else, for, if, instanceof, return, switch, while

**Tratamento de erros:**

. assert, catch, finally, throw, throws, try

**Controles de pacote:**

. import, package

**Tipos primitivos:**

. boolean, byte, char, double, float, int, long, short

**Variaveis de referencia:**

. super, this

**Retorno de método:**

**.** void

**Palavras que não são usadas atualmente:**

. const, goto

O que fazem

**Modificadores de acesso:**

·         **private:** Faz com que um método ou variável possa ser acessado somente de dentro da própria classe;

·         **protected:** Faz com que um método ou variável possa ser acessado somente por classes do mesmo pacote ou por subclasses da classe;

·         **public:** Faz com que uma classe, método ou variável possa ser acessado a partir de qualquer outra classe.

**Modificadores de classe, método e variável:**

·         **abstract**: Usada para declarar uma classe que não pode ser instanciada ou método que deve ser implementado por subclasses concretas;

·         **class:** Palavra chave para identificar uma classe;

·         **extends:** Usada para estender uma classe a outra, isto é, indicar para uma superclasse que uma subclasse está estendendo, (herança);

·         **final:**  Vimos no artigo anterior, torna impossível estender uma classe, sobrepor um método ou reiniciar uma variável;

·         **implements:** Usada para indicar a uma interface que uma classe a implementará;

·         **interface:** Palavra chave para identificar uma interface;

·         **native**: Indica que um método foi escrito em uma linguagem que só pode ser usada em uma plataforma, exemplo o C++;

·         **new**: Usada para instanciar um objeto;

·         **static**: Indica que o método ou a variável pertence a classe e não a instancia;

·         **strictfp**: Usada na frente de um método para indicar que os números de ponto flutuantes seguirão regras restritas de ponto flutuante em todas as expressões;

·         **syncronized:** Indica que um método só pode ser acessado por um seguimento de cada vez;

·         **transient:** Impede que os campos sejam serializados;

·         **volatile:** Indica que a variável pode não ficar sincronizada por ser usada em fragmentos.

**Controle de fluxo:**

·         **break:** Faz o fluxo sair do bloco do código onde a palavra foi inserida.

·         **case:** Executa um bloco de código dependendo da instrução switch.

·         **continue:** Faz com que o código existente após essa instrução não seja executado em um loop, indicando a próxima interação do loop.

·         **default:** Executará esse bloco de código se nenhuma das instruções switch/case for atendida.

·         **do:** Executa um bloco de código um vez e, em seguida junto com a instrução while, processa um teste para determinar se o bloco deve ser executado novamente.

·         **else:** Executa um bloco de código alternativo se uma condição if for falsa.

·         **for:** Usada para executar um loop condicional em um bloco de código.

·         **if:** Usada para executar um teste lógico verdadeiro ou falso.

·         **instanceof:** Determina se um objeto é a instancia de uma classe, superclasse ou interface.

·         **return:** Faz o fluxo sair de um método sem executar nenhum código que seja após a instrução (pode retornar uma variável opcionalmente);

·         **switch:** Indica a variável a ser confrontada com as instruções case.

·         **while:** Executa um bloco de código repetidamente, enquanto um certa condição for verdadeira.

**Tratamento de Erros:**

·         **catch:** Declara o bloco de código usado para tratar uma execução.

·         **finally:** Bloco de código, geralmente após uma instrução try-catch, que vai ser executado independente de que fluxo do programa for usado no tratamento de uma exceção.

·         **throw:** Usada para passar uma execução para o método que chamou o método atual.

·         **throws:** Indica o método que passará uma execução para o método que o chamou.

·         **try:** Bloco de código que será executado, podendo, porém, causar uma exceção.

·         **assert:** Avalia uma expressão condicional para verificar as alternativas do programador.

**Controle de Pacotes:**

·         **import**: Fala-se que é para importar pacotes ou classes para o código, o que não é verdade, veremos mais a frente.

·         **package:** Especifica a que pacote todas as classes de um arquivo fonte pertence.

**Primitivas:**

·         **boolean:** Verdadeiro ou Falso;

·         **byte:** Inteiro de 8 bits (com sinal);

·         **char:** Unidade Unicode (de 16 bits, sem o sinal);

·         **double:** Número de ponto flutuante com 64 bits (com sinal);

·         **float:** Número de ponto flutuante com 32 bits (com sinal);

·         **int:** Número inteiro de 32 bits (com sinal);

·         **long:** Número inteiro de 64 bits (com sinal);

·         **short:** Número inteiro de 16 bits (com sinal);

**Palavras chaves de variáveis:**

·         **super:** Variável de referência que diz respeito a superclasse imediata;

·         **this:** Variável de referência que diz respeito a instancia atual de um objeto;

**Palavra void para tipos retornados:**

·         **void:** Indica que não há um tipo de retorno para o método.

**Palavras reservadas não utilizadas:**

·         **const:** Não use esta palavra para declarar constantes use public static final;

·         **goto:** Não foi implementada na linguagem Java por ser considerada perigosa.

**Objeto:** Uma entidade que possui estado e comportamento é conhecida como um **objeto**, por exemplo, cadeira, bicicleta, marcador, caneta, mesa, carro, etc. Pode ser física ou lógica (tangível e intangível). O exemplo de um **objeto** intangível é o sistema bancário.

**Pacote:** ou package na tecnologia **Java** nada mais é **do que** um conjunto de classes localizadas na mesma estrutura hierárquica de diretórios.

**Classe:** é um elemento do código **Java** que utilizamos para representar objetos do mundo real. Dentro dela é comum declararmos atributos e métodos, que representam, respectivamente, as características e comportamentos desse objeto. Neste documento será apresentado como declarar e utilizar uma **classe em Java**.

**Atributos e métodos: Atributos** são as propriedades de um objeto. **Métodos** são as ações que um objeto pode realizar. Os objetos são características definidas pelas classes. Neles é permitido instanciar objetos da classe para inicializar os **atributos** e invocar os **métodos**.

**Biblioteca:** é uma coleção de subprogramas utilizados no desenvolvimento de software. **Bibliotecas** contém código **e** dados auxiliares, que provém serviços a programas independentes, o que permite o compartilhamento **e** a alteração de código **e** dados de forma modular.

**Angular:** é uma plataforma e framework para construção da interface de aplicações usando HTML, CSS e, principalmente, JavaScript, criada pelos desenvolvedores da Google. ... **Angular** nos ajuda a criar Single-Page Applications com uma qualidade e produtividade surpeendente!

**Controller**: (Controlador), que é responsável por ligar as duas camadas anteriores, ou seja, o **Controller** liga a View ao Modelo e direciona o fluxo da aplicação. Isto permite escolher qual View será exibida ao usuário em resposta à entrada e a lógica de negócio que será processada em cima dela.

**View:** é a apresentação de dados, é a saída, é como o usuário irá ver o que foi produzido por uma ação da aplicação, e é a forma como uma entrada de dados ocorrerá e iniciará uma ação por parte do usuário