Linguagem Java para aplicativo Web

Curso prático de linguagem Java

Rogério Laureano Martins

Minha gratidão a Deus por me permitir compartilhar o que tenho aprendido. Minha gratidão à minha esposa Beatriz e ao meu filho João por terem paciência. Minha gratidão aos professores e youtubers que de alguma forma procura passar o conhecimento pessoal às pessoas.

Rogério Laureano Martins

Linguagem Java

A linguagem Java é uma poderosa ferramenta de programação. Neste livro temos temos diversos conceitos desta linguagem.

Com o Java, o desenvolvimento de 4 tipos de programas são bastante populares.

Standalone → Programas populares para rodar em desktop. Semelhantes a Media Player, antivírus etc.

Web → Páginas HTML dinâmicas

Aplicativos corporativos → Aplicações para empresas como bancos, varejo, indústria etc.

Mobile → Aplicativos para equipamentos mobile como telefone celular, tablets etc.

Para tanto, forma criadas 4 plataformas de desenvolvimento para o java:

Java SE → Plataforma de desenvolvimento java que inclui todos os recursos de IO, Expressões regulares etc.

Java EE → Plataforma para desenvolvimentos corporativos e Web. É um pedaço importante do Java SE.

Java ME → Plataforma "micro" do Java. Utilizada principalmente para Mobile.

Java FX → Plataforma especial utilizada principalmente para aplicativos Web mais ricos.

História do Java

Em 1991 o projeto Java se iniciava com uma equipe formada por James Gosling, Mike Sheridan e Patrick Naughton. O time de desenvolvimento se denominou Green Team. Por isso o primeiro nome foi Greentalk. Posteriormente a linguagem mudou de nome para Oak, que era uma árvore típica dos Estados Unidos, França, Romênia etc. Como já existia uma empresa chamada Oak, mudou-se o nome para Java. Este nome foi dado em homenagem a ilha de Java, que tem uma história com a produção de café. Por isso também que o símbolo do java é uma chícara de café.

Features Java

O projeto Java, foi feito com o intuito de criar uma linguagem portável, segura e simples. Abaixo a lista dos principais features:

- Programa Simples → Os desenvolvedores pretenderam tirar a dificuldade da linguagem. É muito semelhante a linguagem C. Porém, sem as complicações de ponteiros. Este, por exemplo, podem ser usados. Porém, são internos no Java.

- Orientação por objeto → O Java é uma linguagem orientada por objetos. Possibilita a criação de Objetos, Classes, Heranças, Polimorfismo e todos os conceitos de orientação por objetos.

- Portável → O bytecode gerado é o mesmo em qualquer interpretador Java

- Multiplataforma → Por rodar em uma máquina virtual, "JVM" é portátil e roda em diversos sistemas operacionais.

- Segura → Por estar na JVM, tem mais uma camada de segurança independente do sistema operacional

- Robusta → Ferramente robusta no desenvolvimento

- Archetecture neutral → Por exemplo, os tamanhos dos tipos primitivos são reservados sempre o mesmo tamanho de bits.

- Linguagem interpretada → Ao rodar o aplicativo criado em Java, é necessário que se cria um bytecode. Ou seja, é feito uma "compilação" no Java para criar este bytecode. Após isso o Java interpreta o bytecode e executa o programa.

- Multithreaded → O Java permite programas com diversas threads em paralelo

- Distribuída → Facilmente distribuido

- Dinâmica → Linguagem bastante dinâmica e versátil

Preparação para programar em Java

Para executar qualquer programa java, você precisa

- Instale o JDK se você não o tiver instalado, faça o download do JDK e instale-o.
- Defina o caminho do diretório jdk
- Crie o programa java
- Compile e execute o programa java

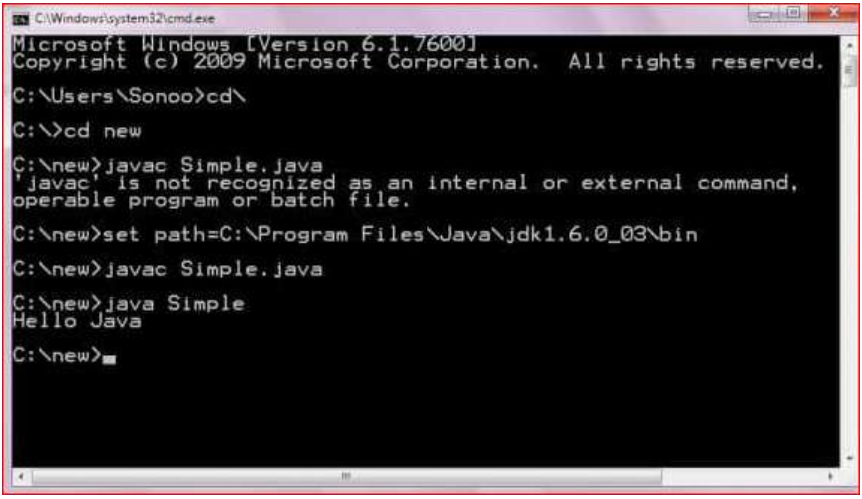
É necessário definir o caminho para o uso de ferramentas como javac, java, etc.

Se você estiver salvando o arquivo de origem Java dentro do diretório JDK / bin, não será necessário definir o caminho, pois todas as ferramentas estarão disponíveis no diretório atual.

No entanto, se você tiver seu arquivo Java fora da pasta JDK / bin, será necessário definir o caminho do JDK.

Existem duas maneiras de definir o caminho em Java:
1 – Temporário
2 – Permanente

1) Como definir o caminho temporário do JDK no Windows
Para definir o caminho temporário do JDK, você precisa seguir as seguintes etapas:
Abra o prompt de comando
Copie o caminho do diretório JDK / bin
Escreva no prompt de comando: defina path = copied_path
Por exemplo:
definir caminho = C: \ Arquivos de Programas \ Java \ jdk1.6.0_23 \ bin
Vamos ver na figura abaixo:



2) Como definir o caminho permanente do JDK no Windows

Para definir o caminho permanente do JDK, siga estas etapas:

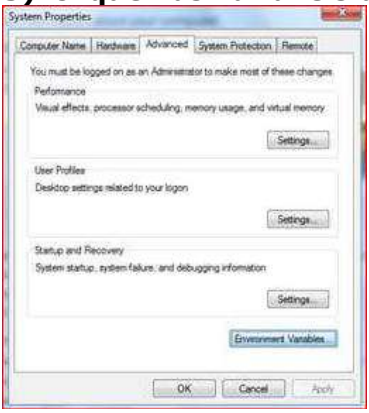
Vá para Propriedades do MyComputer -> guia avançado -> variáveis de ambiente -> nova guia da variável do usuário -> caminho de gravação no nome da variável -> caminho de gravação da pasta bin no valor da variável -> ok -> ok -> ok

Por exemplo:

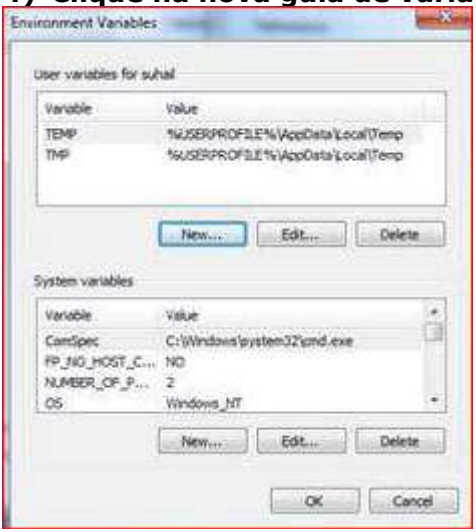
1) Vá para propriedades do MyComputer

2) Clique na aba avançada

3) Clique nas variáveis de ambiente

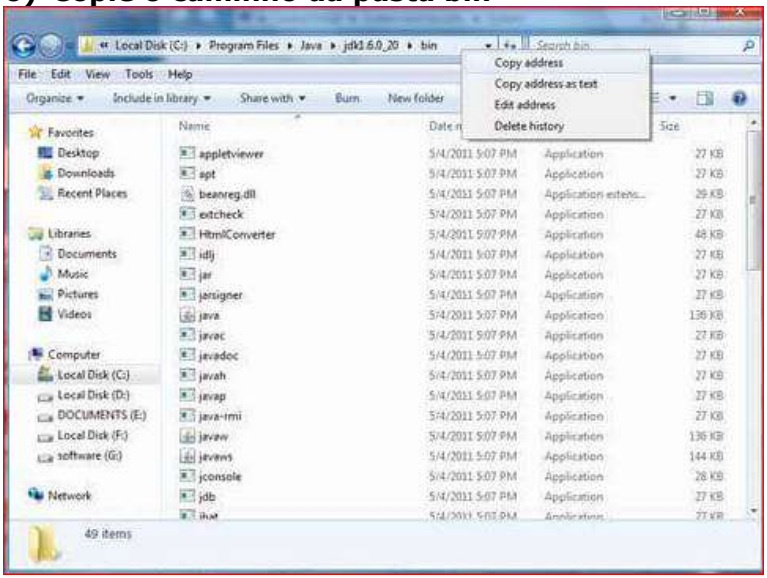


4) Clique na nova guia de variáveis do usuário



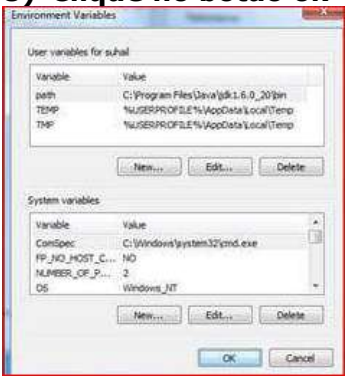
5) Escreva o caminho no nome da variável

6) Copie o caminho da pasta bin

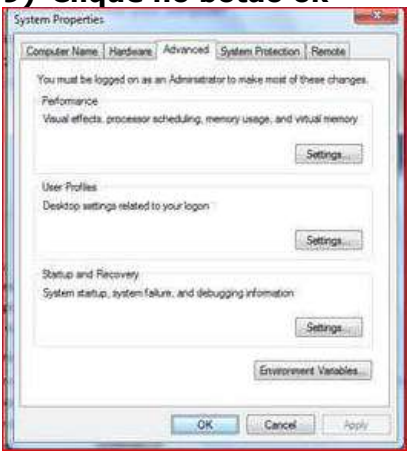


7) Cole o caminho da pasta bin no valor da variável

8) Clique no botão ok



9) Clique no botão ok



Agora seu caminho permanente está definido. Agora você pode executar qualquer programa de java a partir de qualquer unidade.

Configurando o caminho Java no Linux OS

Definir caminho no sistema operacional Linux é o mesmo que definir o caminho no sistema operacional Windows. Mas, aqui usamos a ferramenta de exportação em vez de definir. Vamos ver como definir o caminho no sistema operacional Linux:

```
exportar PATH = $ PATH: /home/jdk1.6.01/bin/
```

Aqui, instalamos o JDK no diretório inicial em Raiz (/ home).

JSP programação Web

Somado com o TomCat, o Java permite a criação de programas Web (intranet ou internet) através do JSP.

O pré-requisito, é que tenha o TomCat instalado.
Em meu exemplo, foi instalado em C:\apache-tomcat-9.0.30

Assim, todas os arquivos com extensão .jar, ficarão no diretório:
C:\apache-tomcat-9.0.30\lib

E todos os arquivos com extensão .jsp ficarão no diretório:
C:\apache-tomcat-9.0.30\webapps\ROOT

Exibição de mensagem e textos

O JSP é organizado por tags, diretivas e elementos de ação. Também possui alguns objetos implícitos. Segue abaixo a explicação de cada um destes elementos.

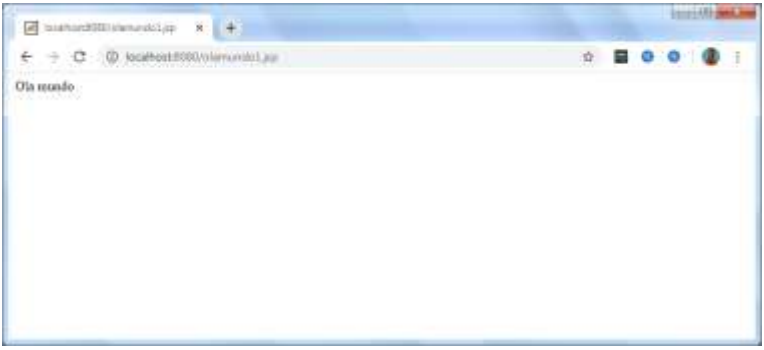
Tipos de tags

Existem diversos tipos de tags no JSP. As tags de elementos de script são divididas três subdivisões: tag scriptlet, tag expression e tag declaration. Além destas, também temos as tags de diretivas e as tags de elementos de ação

Tag de scriptlet

A tag scriptlet, premite acoplar os comandos em Java dentro dela. A tag é "<% [comandos] %>". A interação com o HTML é feita através de objetos implícitos do JSP. No exemplo abaixo o objeto implício é o "out":

```
<html>
<body>
<%
  out.print("Ola mundo");
%>
</body>
</html>
```

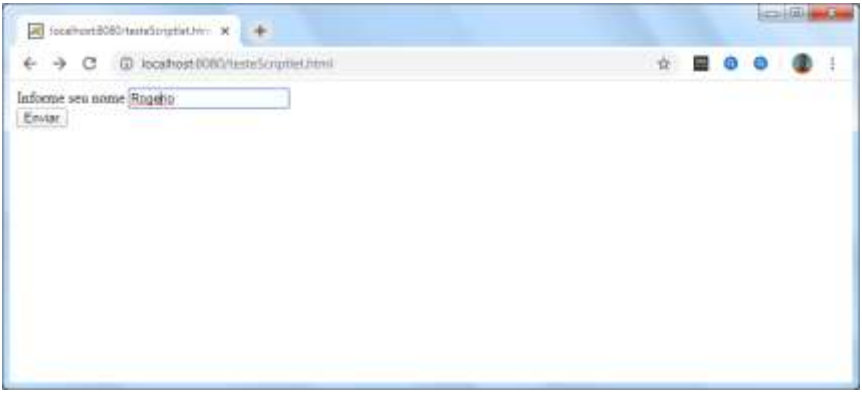


Exemplo 2 com interação do objeto request:
Arquivo (testeScriptlet.html):

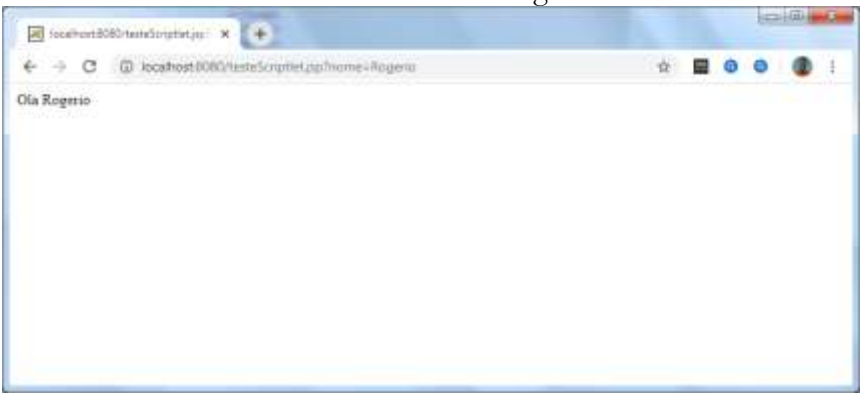
```
<html>
<body>
<form action="testeScriptlet.jsp">
Informe seu nome <input type="text" name="nome"><br/>
<input type="submit" value="Enviar">
</form>
</body>
</html>
```

Arquivo (testeScriptlet.jsp)

```
<html>
<body>
<%
String nome=request.getParameter("nome");
out.print("Ola " + nome);
%>
</form>
</body>
</html>
```



Ao clicar em "Enviar" o sistema exibe o seguinte:



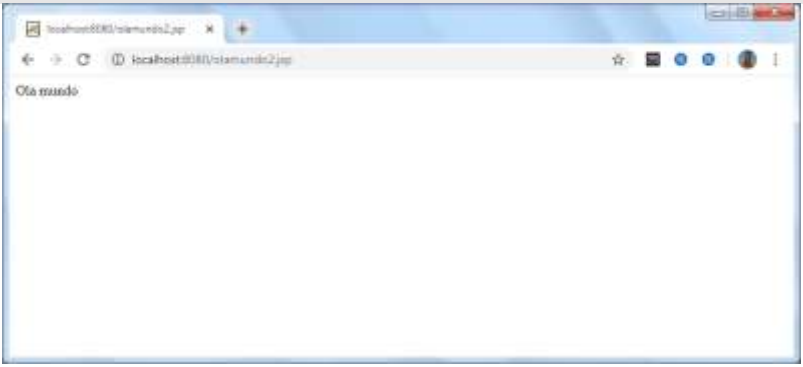
Tag expression

A tag de expressão, permite interagir diretamenente com o HTML com comando em Java, não utilizando os objetos implícitos do JSP que são de saída para o HTML. A tag é a seguinte "<%= [comandos] %>".

Exemplo de utilização:

```
<html>
```

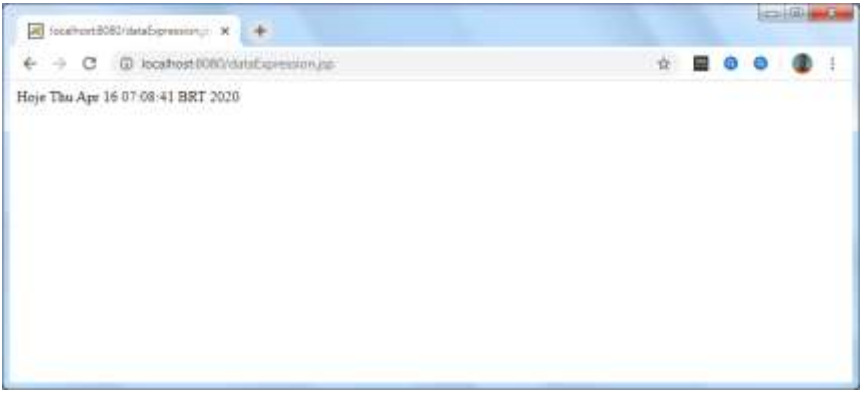
```
<body>
<%= "Ola mundo" %>
</body>
</html>
```



Note que não foi utilizado o objeto "out" ou o objeto "response". Apenas foi colocada a string de saída.

Outro exemplo de tag de expressão utilizando um comando do Java:

```
<html>
<body>
Hoje <%= java.util.Calendar.getInstance().getTime() %>
</body>
</html>
```



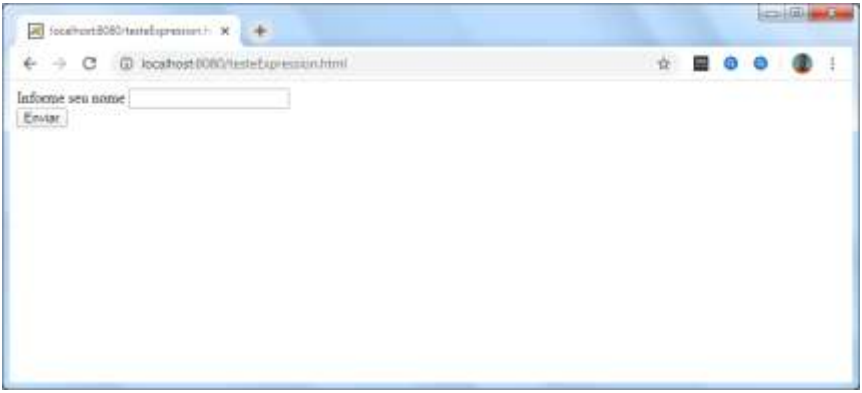
Exemplo de tag expression com interação de objeto implícito do JSP:

Arquivo (testeExpression.html):

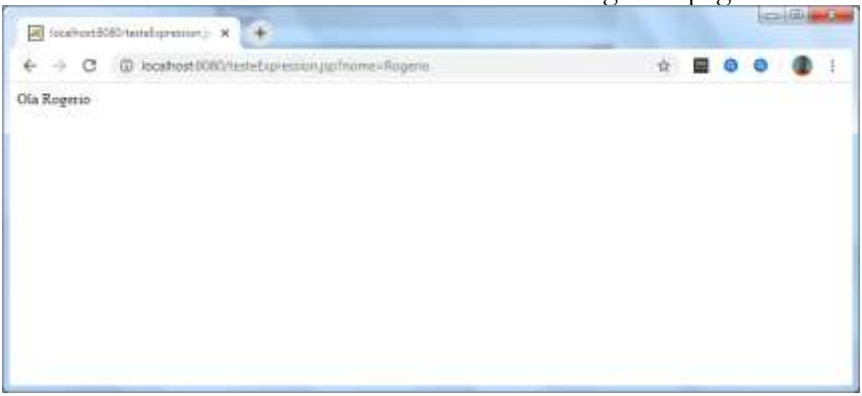
```
<html>
<body>
<form action="testeExpression.jsp">
Informe seu nome <input type="text" name="nome"><br/>
<input type="submit" value="Enviar">
</form>
</body>
</html>
```

Arquivo (testeExpression.jsp)

```
<html>
<body>
<%=
"Ola " + request.getParameter("nome")
%>
</form>
</body>
</html>
```



Ao clicar no botão "Enviar" o sistema exibe a seguinte página:



Tag declaration

A tag de declaração, permite declarar variáveis ou funções que retornam valores dentro de um arquivo jsp. A tag é a seguinte:
<%! [declaração] %>

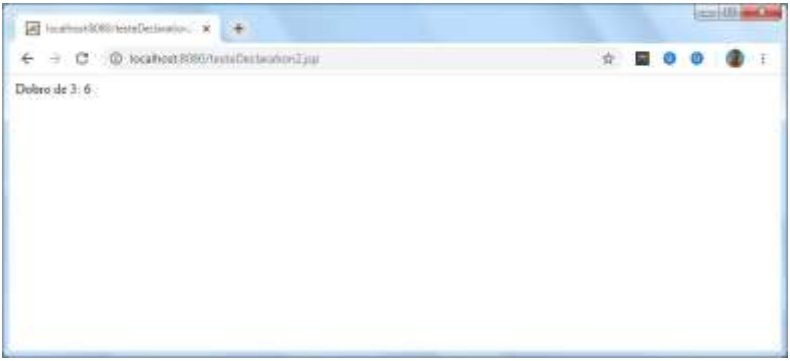
Exemplo:

```
<html>
```

```
<body>
<%! int valor=1; %>
<%= "Valor da variavel: " + valor %>
</body>
</html>
```

Exemplo 2:

```
<html>
<body>
<%!
int dobro (int n){
  return n * 2;
}
%>
<%= "Dobro de 3: " + dobro(3) %>
</body>
</html>
```



Variáveis

Quanto ao tipo de informação primitivos que as variáveis suportam, destaco:

byte → Número inteiro de -128 à 127
short → Número inteiro de -32.768 à 32.767
int → Número inteiro de -2.147.483.648 até 2.147.483.647
long → Número inteiro de -9.223.372.036.854.770.000 à -9.223.372.036.854.770.000
float → Número real (com valores fracionados) de $3,4^{-38}$ à $3,4^{38}$
double → Número real de $1,7^{-308}$ à $1,7^{308}$
char → Caracter que vai de 0 à 65535
boolean → Valor verdadeiro e falso.

O java é case sensitive. Ou seja, faz diferença entre letras maiúsculas e minúsculas no momento da codificação. Abaixo temos a lista de classes básicas do java e a referência com as variáveis primitivas:

Byte → Classe que contem regras, transformações dos valores primitivos do tipo byte.
Short → Classe que contem regras, transformações dos valores primitivos do tipo short.
Integer → Classe que contem regras, transformações dos valores primitivos do tipo int.
Long → Classe que contem regras, transformações dos valores primitivos do tipo long.
Float → Classe que contem regras, transformações dos valores primitivos do tipo float.
Double → Classe que contem regras, transformações dos valores primitivos do tipo double.
String → Classe que contem regras, transformações dos valores primitivos do tipo char.

Quanto ao tipo de declaração das variáveis, o Java pode ser de três tipos:

```
class A{
  int vIntancia = 50; //Variável de instância
  static int m = 100; //Variável estática
  void method(){
    int n=90; //Variável local
  }
}
```

Variável de instância é o tipo de variável que é declarada com não estática a nível da classe. Seu valor não é compartilhado para outras instâncias da mesma classe

Variável estática é o tipo de variável com a cláusula "static". Seu valor é compartilhado para outras instâncias da mesma classe. Não pode ser declarada como local.

Variável local é o tipo de variável declarada dentro de um método da classe.

Quanto aos acessos, temos 4 tipos:

Access modifiers	**Dentro da classe**	**Dentro da package**	**Fora da package em uma subclasse**	**Fora da package**
private	Permitido	Não perm	Não perm	Não perm
default	Permitido	Permitido	Não perm	Não perm
protected	Permitido	Permitido	Permitido	Não perm
public	Permitido	Permitido	Permitido	Permitido

private → Apenas a classe.
default → Na classe e na package em que esta classe pertence.
protected →Além de permitir na classe e na package também permite em uma sub-classe fora da package
public → Acesso irrestrito à variável ou método.

OBS: O conceito de package será melhor entendido mais à frente no livro.

Além do static, temos outras cláusulas que não são modificadores de acesso.

static → Valor que pode ser utilizado em outras instâncias
abstract → Declara variáveis abstratas

synchronized → Utilizado em threads
volatile → Utilizado em threads
final → Utilizado para finalizar implementações de classes, métodos e variáveis

No caso de inteiros, podemos fazer na entrada das variáveis a notação decimal, binária e hexadecimal. Na saída decimal, octal e hexadecimal. Esta funcionalidade é bastante útil na linguagem java principalmente para programação de microcontroladores. Segue os exemplos:

Exemplo com variáveis do tipo int com escrita em decimal saída em decimal

```
<html>
<body>
<%!
  int A = 15;
%>
<%= "".format("%d",A) %>
</body>
</html>
```

Exemplo com variável do tipo int com escrita em binário saída em decimal

```
<html>
<body>
<%!
  int A = 0b1111;
%>
<%= "".format("%d",A) %>
</body>
</html>
```

Exemplo com variável do tipo int com escrita em hexadecimal saída em decimal

```
<html>
<body>
<%!
  int A = 0xf;
%>
<%= "".format("%d",A) %>
</body>
</html>
```

Exemplo com variáveis do tipo int com escrita em decimal saída em hexadecimal

```
<html>
<body>
<%!
  int A = 15;
%>
<%= "".format("%x",A) %>
</body>
</html>
```

Exemplo com variável do tipo int com escrita em binário saída em hexadecimal

```
<html>
<body>
<%!
  int A = 0b1111;
%>
<%= "".format("%x",A) %>
</body>
</html>
```

Exemplo com variável do tipo int com escrita em hexadecimal saída em hexadecimal

```
<html>
<body>
<%!
  int A = 0xf;
%>
<%= "".format("%x",A) %>
</body>
</html>
```

Exemplo com variáveis do tipo int com escrita em decimal saída em octal

```
<html>
<body>
<%!
  int A = 15;
%>
<%= "".format("%o",A) %>
</body>
</html>
```

Exemplo com variável do tipo int com escrita em binário saída em octal

```
<html>
<body>
<%!
  int A = 0b1111;
%>
<%= "".format("%o",A) %>
</body>
</html>
```

Exemplo com variável do tipo int com escrita em hexadecimal saída em octal

```
<html>
<body>
<%!
  int A = 0xf;
%>
<%= "".format("%o",A) %>
</body>
</html>
```

Exemplo com variável

```
<html>
<body>
<%!
  byte A = 15;
%>
<%= "".format("%d",A) %>
</body>
</html>
```

Exemplo com variável short

```
<html>
<body>
<%!
  short A = 15;
%>
<%= "".format("%o",A) %>
</body>
</html>
```

Exemplo com variável long

```
<html>
<body>
<%!
  long A = 15;
```

```
%>
<%= "".format("%d",A) %>
</body>
</html>
```

Operações Numéricas

Segue tabela com os sinais e funções das expressões numéricas.

Sinal	Funcionalidade
+	Possibilita a soma entre dois números
-	Possibilita a subtração entre dois números
*	Possibilita a multiplicação entre dois números
/	Possibilita a divisão entre dois números
%	Possibilita a recuperar o resto de uma divisão.
++[var]	Pré-soma. Onde [var] é a variável
[var]++	Pós-soma. Onde [var] é a variável
--[var]	Pré-subtração. Onde [var] é a variável
[var]--	Pós-subtração. Onde [var] é a variável
[var]+=[val]	Soma implícita. Onde [var] é a variável e [val] o valor
[var]-=[val]	Subtração implícita. Onde [var] é a variável e [val] o valor
[var]*=[val]	Multiplicação implícita. Onde [var] é a variável e [val] o valor
[var]/=[val]	Divisão implícita. Onde [var] é a variável e [val] o valor
~	Subtrai 1 e inverte sinal. Exemplos: (10 vira -11) e (-10 vira 9)
>>	Regressão geométrica
<<	Progressão geométrica
pow	Possibilita a potenciação entre dois números
sqrt	Possibilita a raíz entre dois números
cbrt	Possibilita a raíz cúbica
()	Agrupador de expressão numérica
abs	Permite arredondar em valor absoluto
round	Permite arredondar para o valor fracionado mais

Sinal	Funcionalidade
	próximo
ceil	Permite arredondar para cima os números fracionados
floor	Permite arredondar para baixo os números fracionados
abs	Retorna o valor absoluto positivo de um número
acos	Função retorna o arco cosseno do parâmetro x; isto é, o valor cujo cosseno é x. O parâmetro de entrada x deve estar entre -1 e 1. O valor de retorno é em radianos, entre 0 e pi
asin	Função retorna o arco seno do parâmetro x; isto é, o valor cujo cosseno é x. O parâmetro de entrada x deve estar entre -1 e 1. O valor de retorno é em radianos, entre 0 e pi
atan	Função retorna o arco tangente do parâmetro x; isto é, o valor cujo cosseno é x. O parâmetro de entrada x deve estar entre -1 e 1. O valor de retorno é em radianos, entre 0 e pi
atan2	Função que retorna tangente entre dois números
cos	Função retorna o cosseno de f em radianos. O valor de retorno é de -1 a 1.
cosh	Função retorna o cosseno hiperbólico de x, definido matematicamente como (ex + e-x) / 2. Se o valor de x for muito grande (se ocorrer um estouro), a função falhará.
exp	A função retorna o valor de e - a base dos logaritmos naturais - aumentada para a potência x (ou seja, ex).
sin	Função retorna o seno de f em radianos. O valor de retorno é de -1 a 1.
sinh	Função retorna o cosseno hiperbólico de x, definido matematicamente como (ex + e-x) / 2. Se o valor de x for muito grande (se ocorrer um estouro), a função falhará.
tan	Função retorna a tangente de x em radianos.
tanh	Função retorna a tangente hiperbólica de x, definida matematicamente como seno (x) / cosseno (x).

Exemplo de adição entre dois números
Arquivo (opNumerica1.html):

```
<html>
<body>
<form action="opNumerica1.jsp">
Informe o inteiro <input type="text" name="A"><br/>
Informe o inteiro <input type="text" name="B"><br/>
<input type="submit" value="Enviar">
</form>
</body>
</html>
```

Arquivo (opNumerica1.jsp)

```
<html>
<body>
<%
    int A, B, C;
    A = Integer.parseInt(request.getParameter("A"));
    B = Integer.parseInt(request.getParameter("B"));
    C = A + B;
    out.print("".format("Resultado: %d", C);
%>
</form>
</body>
</html>
```

O sistema exibe a tela