



(✓) Curso Técnico

Quem Investe no futuro faz QI

Informática para Internet

Desenvolvimento de Sistemas

Web III

Unidade V

1. POO – Programação Orientada a Objetos

A linguagem Java é orientada a objetos, ou seja, possibilita a criação de códigos denominados classes, sendo que as classes são compostas por atributos e métodos e a partir das classes é possível criar objetos.

As classes são organizadas em pacotes, pois os sistemas de software empresariais geralmente são grandes e necessitam ser criados em partes relativamente independentes para serem viáveis. Na linguagem Java as classes, possibilitam esta operação.

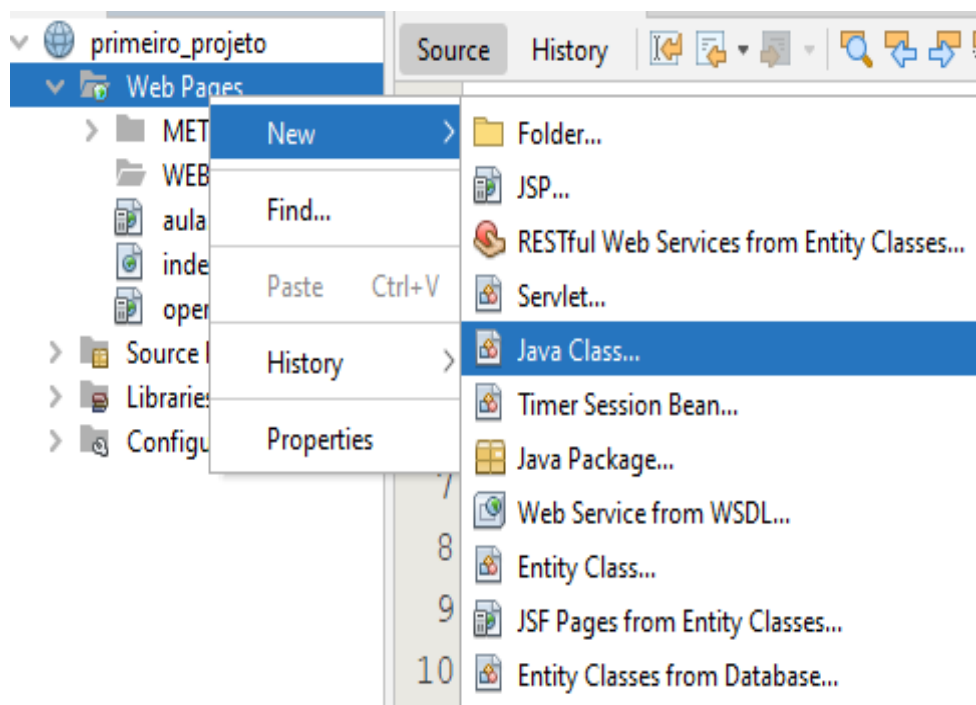
2. Classes

Uma classe é uma forma ou gabarito para a definição de objetos. Através da definição de uma classe, descreve-se que propriedades/atributos os seus objetos terão. Ainda se descreve os comportamentos/funcionalidades dos seus objetos, chamando-os de métodos.

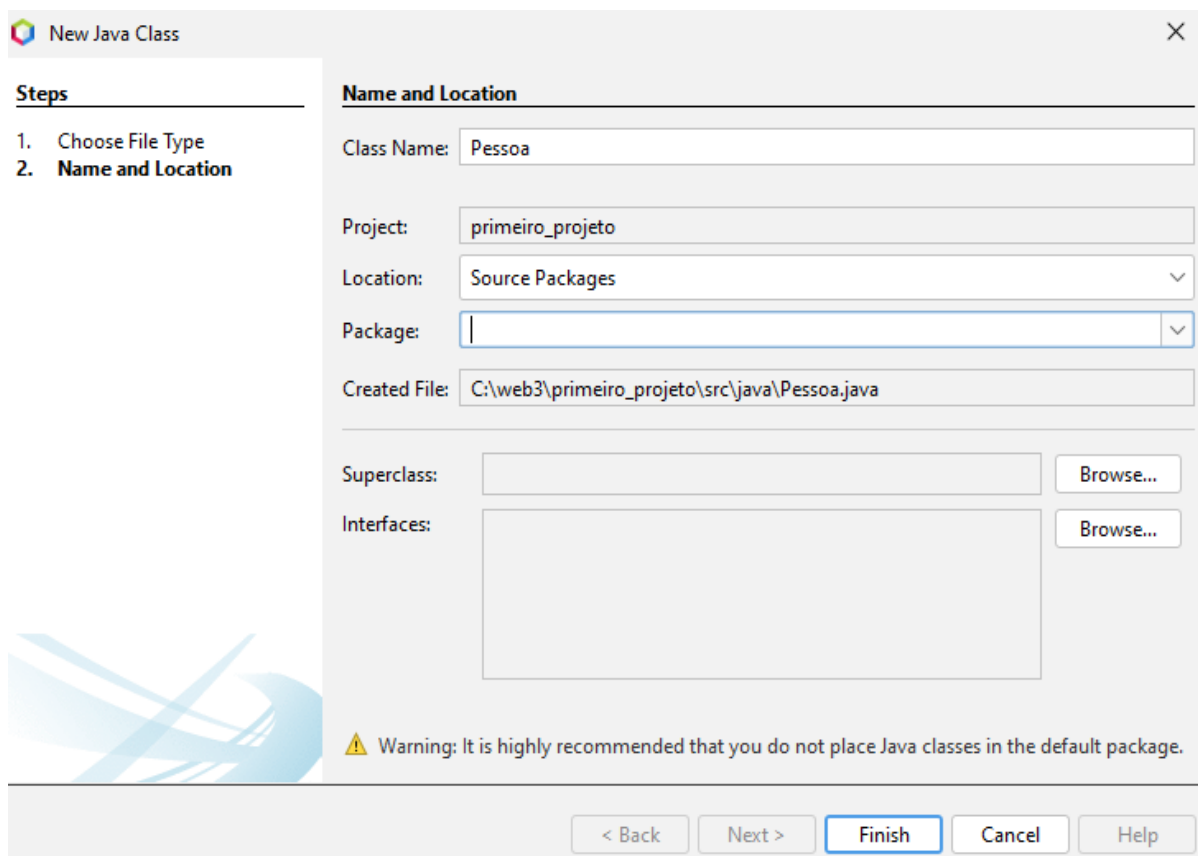


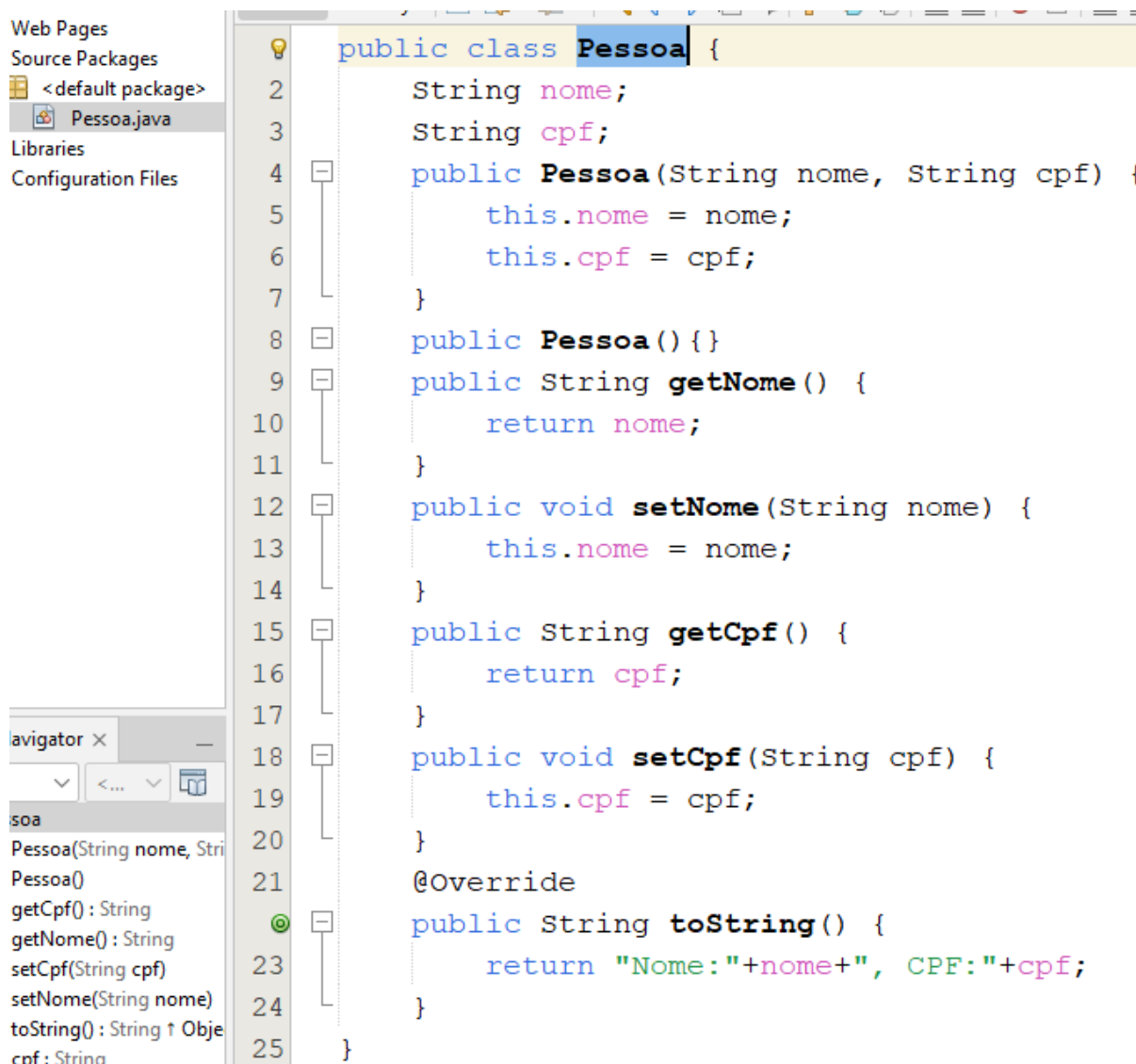
Sintaxe genérica de declaração de uma classe:

```
[Opções] class NomeClasse {  
  
    [atributos]  
  
    [construtores]  
  
    [métodos]  
  
}
```



Vamos criar uma classe chamada Pessoa:





```

1 public class Pessoa {
2     String nome;
3     String cpf;
4     public Pessoa(String nome, String cpf) {
5         this.nome = nome;
6         this.cpf = cpf;
7     }
8     public Pessoa() {}
9     public String getNome() {
10         return nome;
11     }
12     public void setNome(String nome) {
13         this.nome = nome;
14     }
15     public String getCpf() {
16         return cpf;
17     }
18     public void setCpf(String cpf) {
19         this.cpf = cpf;
20     }
21     @Override
22     public String toString() {
23         return "Nome: "+nome+", CPF: "+cpf;
24     }
25 }

```

Web Pages
Source Packages
<default package>
Pessoa.java
Libraries
Configuration Files

soa
Pessoa(String nome, Stri
Pessoa()
getCpf() : String
getNome() : String
setCpf(String cpf)
setNome(String nome)
toString() : String ↑ Obj
cpf : String

Ao analisar a classe Pessoa percebe-se que a classe detém dois atributos, dois construtores e cinco métodos, conforme detalha-se:

Nas linhas 2 e 3 do código acima foram definidos dois atributos do tipo String com os nomes **nome** e **cpf**.

Nas linhas 4, 5, 6 e 7 do código acima foi definido o construtor cheio, isto é, quando um objeto for instanciado e o construtor cheio for acionado, todos os atributos deverão ser inicializados(receber algum dado).

Na linha 8 do código acima foi definido o construtor vazio, isto é, quando um objeto for instanciado e o construtor vazio for acionado, não haverá necessidade de inicializar os atributos(atribuir dados).

CURSO TÉCNICO EM INFORMÁTICA PARA INTERNET

Nas linhas 9, 10, 11 do código acima foi definido o método `get` chamado `getNome()`, cuja função é retornar o dado armazenado no atributo `nome`.

Nas linhas 12, 13, 14 do código acima foi definido o método `set` chamado `setNome()`, cuja função é armazenar dados no atributo `nome`.

Nas linhas 15, 16, 17 do código acima foi definido o método `get` chamado `getCpf()`, cuja função é retornar o dado armazenado no atributo `cpf`.

Nas linhas 18, 19, 20 do código acima foi definido o método `set` chamado `setCpf()`, cuja função é armazenar dados no atributo `cpf`.

Nas linhas 21, 22, 23 e 24 do código acima foi sobrescrito o método `toString()`, importa dizer que é um método que já existe na estrutura de árvore da linguagem Java, mas permite o `override` ou a sobrescrita, isto é, definição nos moldes que interessam. A função é retornar na forma de `String` (texto) os dados armazenados no objeto.

Mesmo diante dos exemplos e explicações, vale pontuar que o nome da classe é uma referência ou identificador para a classe e deve sempre começar com letra maiúscula e sem caracteres especiais ou acentos e se for um nome composto não poderá ter espaço entre as palavras, pois deve seguir um padrão internacional. É relevante saber que o nome da classe permite referenciá-la posteriormente, na criação de um objeto ou na utilização de um recurso estático.

Ainda vale pontuar que um atributo descreve uma propriedade da classe, sendo cada atributo identificado por nome/referência e um **tipo de dado** associado.

Os tipos de dados na linguagem Java são:

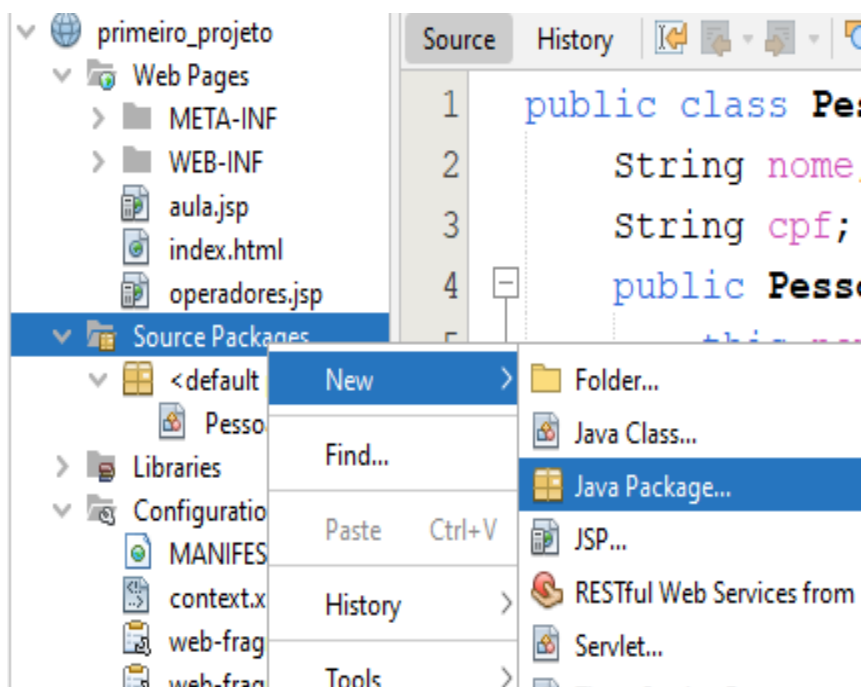
Nome	Especificações	Valor Padrão	Grupo
byte	Possui 1 byte de informação ou 8 bits, aceita valores entre -128 a 127;	0	Primitivo -> Inteiro
short	Possui 2 bytes de informação ou 16 bits, aceita valores entre -32.768 a 32.767;	0	Primitivo -> Inteiro
int	Possui 4 bytes de informação ou 32	0	Primitivo -> Inteiro

	bits, aceita valores entre - 2.147.483.648 a 2.147.483.647		
long	Possui 8 bytes de informação ou 64 bits, aceita valores entre - 9,22337E+18 a 9,22337E+18	0L	Primitivo -> Inteiro
float	Possui 4 bytes de informação ou 32 bits, aceita valores entre IEEE 754 $\pm 1,40129846432481707e-45$ a $3,40282346638528860e+38$	0.0f	Primitivo -> Reais
double	Possui 8 bytes de informação ou 64 bits, aceita valores entre IEEE 754 $\pm 4,94065645841246544e-324$ a $1,79769313486231570e+308$	0.0d	Primitivo -> Reais
boolean	Possui 1 bit, permitindo os valores false e true	false	Primitivo -> Lógico
char	Possui 16 bits ou 2 bytes, pois permite armazenar um caractere Unicode, sendo seu valor mínimo '\u0000' (ou 0) e seu valor máximo '\uffff' (ou 65535)	'\u0000'	Primitivo -> Caracteres

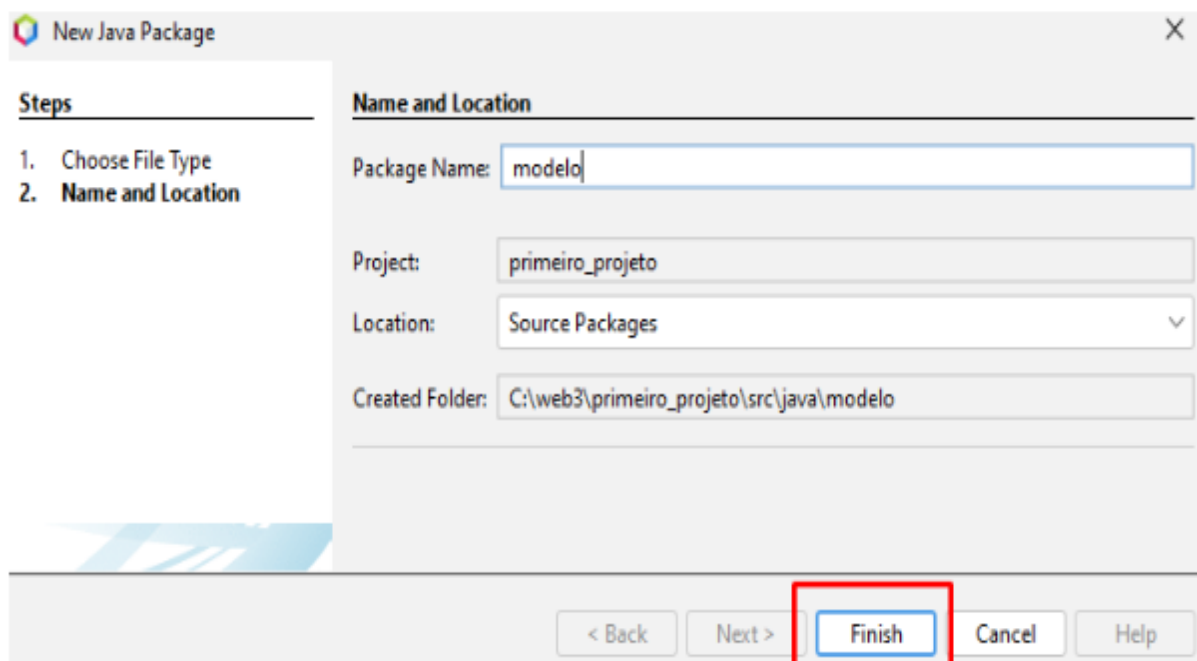
Vale salientar que é comum na linguagem Java a declaração de variáveis do tipo String, pois a classe String permite a criação de objetos/variáveis que possibilitam o armazenamento de um conjunto de caracteres de todos os tipos(literal/texto).

String	Permite o armazenamento de um conjunto de caracteres do mesmo tipo ou de tipos diferentes, desde que estejam entre aspas.	null	Classe
--------	---	------	--------

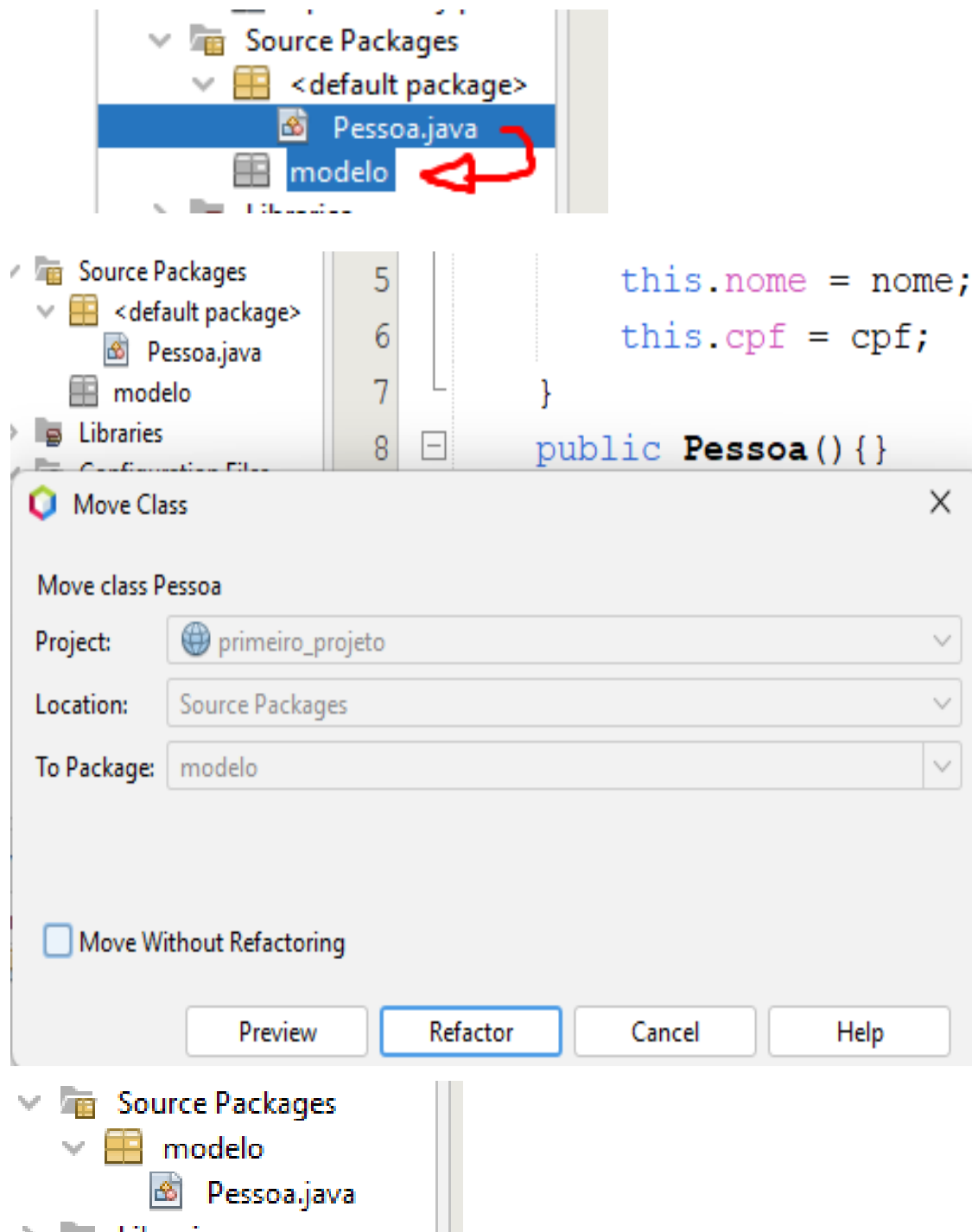
Para utilizar a classe acima mencionada é importante criar um package (pacote) para agrupamento das classes:



É comum chamar o pacote que agrupa as classes de modelo ou model, conforme o conceito MVC:



Na sequência “arrasta-se” a classe Pessoa para dentro do pacote modelo:



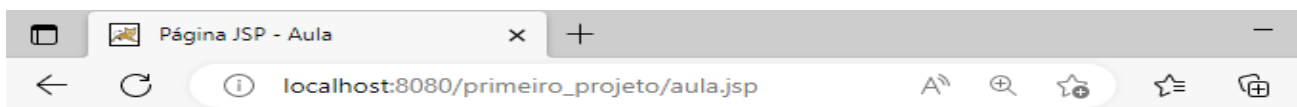
Após a inserção da classe Pessoa no Pacote modelo é possível importar a classe Pessoa para outros documentos do projeto, utilizando a instrução `<%@page import="modelo.Pessoa"%>`, no topo da página que utilizará o código da classe Pessoa, conforme exemplifica-se:


```

aula.jsp x
Source History
<%@page contentType="text/html" pageEncoding="UTF-8"%>
1 <%@page import="modelo.Pessoa"%>
2 <!-- Importando a classe pessoa do pacote modelo-->
3 <!DOCTYPE html>
4 <html>
5 <head>
6 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
7 <title>Página JSP - Aula</title>
8 </head>
9 <body>
10 <% // Criando a referência ou objeto chamado p1
11 Pessoa p1;
12 // instanciando ou inicializando o objeto chamado p1, com o construtor cheio
13 p1 = new Pessoa("Maria", "999.999.999-99");
14 //outra maneira de criar um objeto, já instanciando
15 Pessoa p2= new Pessoa("João", "888.888.888-88");
16 //Criando outra instância com o construtor vazio
17 Pessoa p3= new Pessoa();
18 %>
19 <p>Conteúdo do objeto p1 : <%=p1.toString() %></p>
20 <p>Conteúdo do objeto p2 : <%=p2.toString() %></p>
21 <p>Conteúdo do objeto p3 : <%=p3.toString() %></p>
22 <!--outra maneira de exibir-->
23 <p>Valor do atributo nome do objeto p1 : <%=p1.getNome() %></p>
24 <p>Valor do atributo cpf do objeto p2 : <%=p2.getCpf() %></p>
25 </body>
26 </html>

```

Como resultado será exibido no navegador:



Conteúdo do objeto p1 : Nome:Maria, CPF:999.999.999-99

Conteúdo do objeto p2 : Nome:João, CPF:888.888.888-88

Conteúdo do objeto p3 : Nome:null, CPF:null

Valor do atributo nome do objeto p1 : Maria

Valor do atributo cpf do objeto p2 : 888.888.888-88

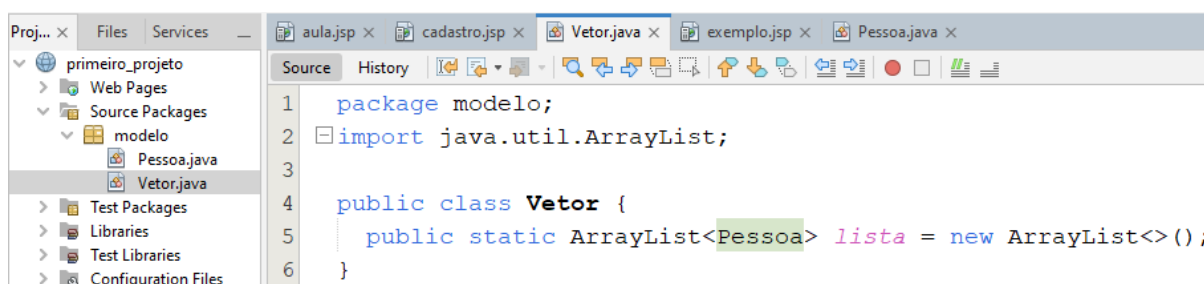
2. ArrayList

É uma classe da linguagem Java que utiliza matrizes dinâmicas para armazenar objetos. A classe ArrayList estende da classe AbstractList e implementa a interface List, fornecendo a facilidade de acesso aleatório, pois baseado em índices. Observa-se que conforme a quantidade de objetos ou elementos, bem como características da estrutura computacional, a performance pode ser lenta no processo de remoção de um elemento, pois a lista é reorganizada.

Sintaxe genérica de declaração de um ArrayList:

```
ArrayList<Classe> nomeDoArrayList=new ArrayList<>();
```

Exemplo utilizando a classe Pessoa criada neste conteúdo:



```
1 package modelo;
2 import java.util.ArrayList;
3
4 public class Vetor {
5     public static ArrayList<Pessoa> lista = new ArrayList<>();
6 }
```

Uma variável/referência estática da classe ArrayList é criado com o nome **lista**, e instanciado como novo ArrayList, dentro de uma classe chamada Vetor no arquivo Vetor.java. Neste contexto, explica-se cada linha abaixo:

```
1 package modelo;
2 import java.util.ArrayList;
3 public class Vetor {
4     public static ArrayList<Pessoa> lista = new ArrayList<>();
5 }
```

- Na linha 1 é feita a definição do pacote da classe;
- Na linha 2 é feita a importação da classe ArrayList para que seja possível utilizá-la na classe Vetor;
- Na linha 3 é iniciada a declaração da classe Vetor;
- Na linha 4 é criada a variável/referência estática(que pode ser acionada a partir do nome da classe e não após a criação/declaração de um

objeto da classe, ou seja, está vinculada à classe e não aos objetos da classe) **lista**.

- Ainda na linha 4, após o sinal de igualdade(=) a variável **lista** é instanciada como um novo ArrayList;

Para ampliar o entendimento é importante a exemplificação do uso do ArrayList criado no exemplo acima num documento web Java com a extensão .jsp:

```
1 <%@page import="modelo.*" %>
2 <%@page contentType="text/html" pageEncoding="UTF-8"%>
3 <!DOCTYPE html>
4 <html>
5   <head>
6     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
7     <title>Exemplo ArrayList</title>
8   </head>
9   <body>
10    <%
11      Vetor.lista.add(new Pessoa("Maria", "88888"));
12      out.print(Vetor.lista.toString());
13    %>
14  </body>
15 </html>
```

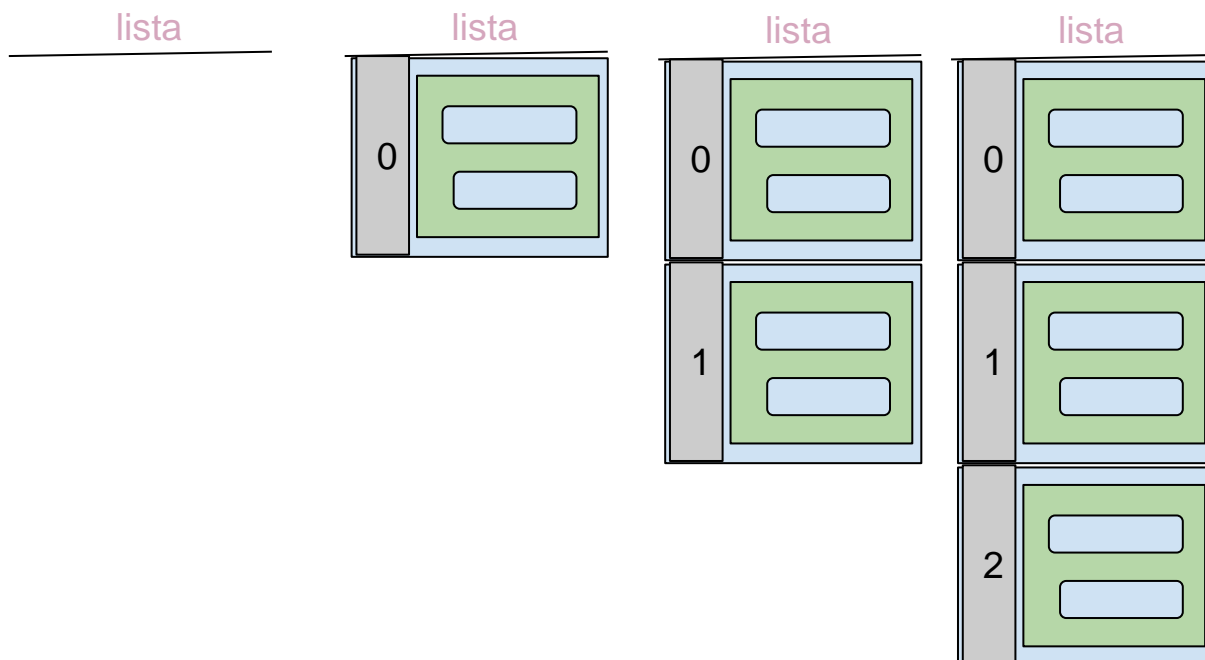
Na linha 1 do exemplo acima é feita a importação de todas as classes do pacote modelo, quando o signo * é utilizado após o ponto final.

A importação do pacote modelo é imprescindível para o funcionamento ou execução das linhas 11 e 12, pois importa as classes Pessoa e Vetor.

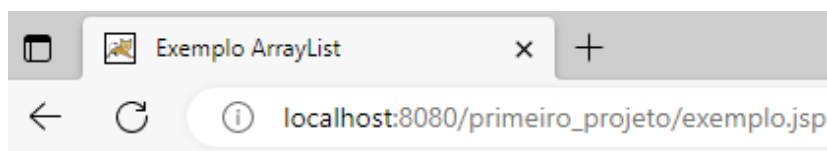
Na linha 11 do exemplo acima é feita a adição de um novo objeto da classe Pessoa à referência/variável **lista**, que é um ArrayList.

Na linha 12 do exemplo acima é feita a exibição de todos os objetos contidos na referência/variável **lista**, que é um ArrayList.

Vale ressaltar que se o navegador for atualizado mais duas vezes ocorrerá o que está nos diagramas abaixo, i.e., serão adicionados novos objetos no ArrayList **lista** com os mesmos valores, mas em índices subsequentes, pois o ArrayList é dinâmico e o método add permite a adição de um elemento/objeto na última posição do ArrayList:

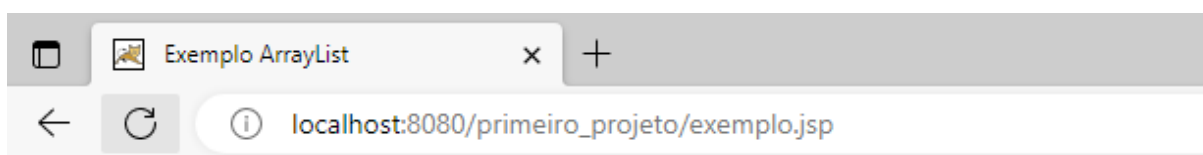


No navegador será visualizado da seguinte maneira:



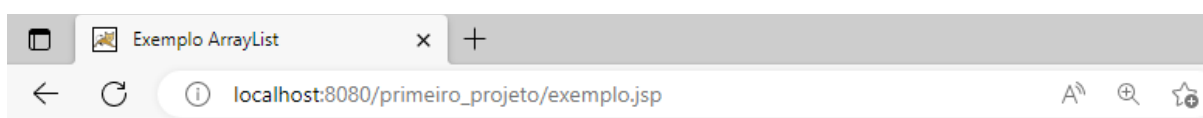
[Nome:Maria, CPF:88888]

Após atualizar:



[Nome:Maria, CPF:88888, Nome:Maria, CPF:88888]

Após atualizar:



[Nome:Maria, CPF:88888, Nome:Maria, CPF:88888, Nome:Maria, CPF:88888]

O método `get` permite trabalhar com um objeto/elemento específico de um `ArrayList`, ou seja, num `ArrayList` com cinco objetos/elementos, é possível indicar com qual elemento específico será executada uma determinada instrução. Para ampliar o entendimento imagina-se que haja a necessidade de realização de uma operação com a elemento/objeto cujo índice seja 2. Assim o comando a ser executado para trabalhar com o elemento será o `nomeDoArrayList.get(2)`.

O método `size` permite saber a quantidade de objetos/elementos de um `ArrayList`, ou seja, permite saber o tamanho de um `ArrayList`. Para ampliar o entendimento imagina-se que haja a necessidade de percorrer um `ArrayList` exibindo todos os nomes, mas não se sabe quantos elementos/objetos há neste `ArrayList`, neste contexto o método `size` será muito útil para o código. Para utilizar o método `size` basta digitar `nomeDoArrayList.size()`.

O método `remove` permite remover um objeto/elemento específico de um `ArrayList`, ou seja, num `ArrayList` com cinco objetos/elementos, é possível indicar com qual elemento específico será excluído. Para ampliar o entendimento imagina-se que haja a necessidade de exclusão de um elemento/objeto cujo índice seja 1. Assim o comando a ser executado para remover o elemento será o `nomeDoArrayList.remove(1)`.

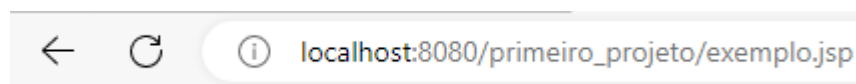
O método `clear` permite remover todos os objetos/elementos de um `ArrayList`, ou seja. Assim o comando a ser executado para remover todos os elemento será o `nomeDoArrayList.clear()`.

Importante um exemplo comentado com o uso dos métodos supracitados:

```
<%@page import="modelo.*" %>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Exemplo ArrayList</title>
</head>
<body>
```

```
<%
    Vetor.lista.add(new Pessoa("Maria","88888"));
    Vetor.lista.add(new Pessoa("João","77777"));
    Vetor.lista.add(new Pessoa("José","66666"));
    Vetor.lista.add(new Pessoa("Laura","55555"));
    Vetor.lista.add(new Pessoa("Carla","44444"));
    out.print("Quantidade de elementos do ArrayList:"+Vetor.lista.size());
    out.print("<h1>Todos os elementos</h2>");
    for(int i=0;i<Vetor.lista.size();i++)
        out.print(Vetor.lista.get(i).toString()+"<br>");
    //abaixo é removido o elemento de índice 2
    Vetor.lista.remove(2);
    out.print("Quantidade de elementos do ArrayList:"+Vetor.lista.size());
    out.print("<h1>Todos os elementos</h2>");
    for(int i=0;i<Vetor.lista.size();i++)
        out.print(Vetor.lista.get(i).toString()+"<br>");
    //abaixo todos os elemento são removidos
    Vetor.lista.clear();
    out.print("Quantidade de elementos do ArrayList:"+Vetor.lista.size());
    out.print("<h1>Todos os elementos</h2>");
    for(int i=0;i<Vetor.lista.size();i++)
        out.print(Vetor.lista.get(i).toString()+"");
%>
</body>
</html>
```

Quando este código for executado o navegador exibirá:



Quantidade de elementos do ArrayList:5

Todos os elementos

Nome:Maria, CPF:88888

Nome:João, CPF:77777

Nome:José, CPF:66666

Nome:Laura, CPF:55555

Nome:Carla, CPF:44444

Quantidade de elementos do ArrayList:4

Todos os elementos

Nome:Maria, CPF:88888

Nome:João, CPF:77777

Nome:Laura, CPF:55555

Nome:Carla, CPF:44444

Quantidade de elementos do ArrayList:0

Todos os elementos

Referências

- ★ <https://www.dca.fee.unicamp.br/cursos/PooJava/Aulas/poojava.pdf>
- ★ <https://blog.betrybe.com/java/java-string/>
- ★ <https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html>
- ★ <https://www.feg.unesp.br/Home/PaginasPessoais/CristovaoCunha/TreinamentoSGCD/matelli/como-usar-arraylist-em-java.pdf>