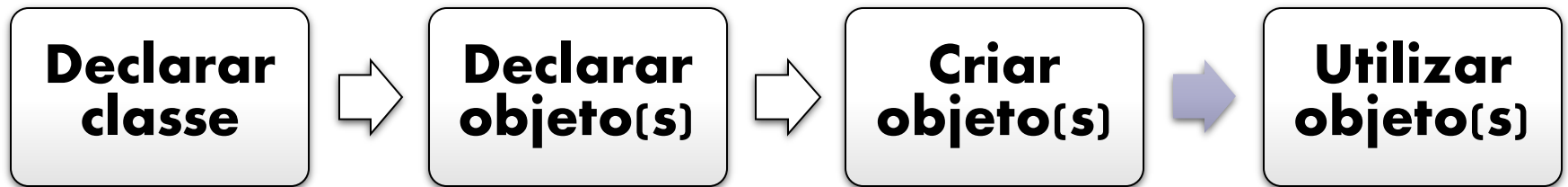


# **Implementação Classes e objetos**

# Utilização de POO



# **Declaração de classe**

# Elementos da classe

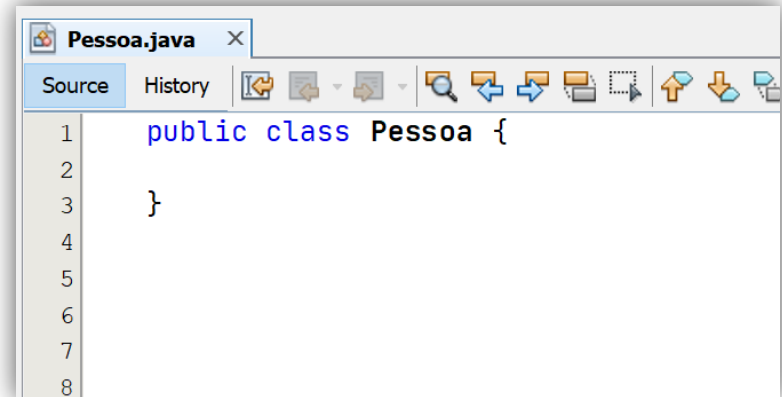
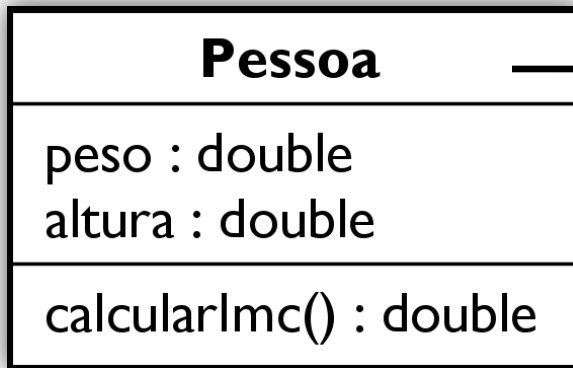
<b>Classe</b>
atributos
operacoes()

➡ **Uma classe Java (arquivo .java)**

➡ **Variáveis**

➡ **Métodos**

# Classe Java

A screenshot of a Java code editor window titled 'Pessoa.java'. The editor shows the following code:

```
1 public class Pessoa {  
2  
3 }  
4  
5  
6  
7  
8
```

# Tradução dos atributos em variáveis de instância

- Em geral, os atributos da classe são traduzidos em Java para **variáveis de instância**.
- Sintaxe: `tipoDado nomeAtributo`

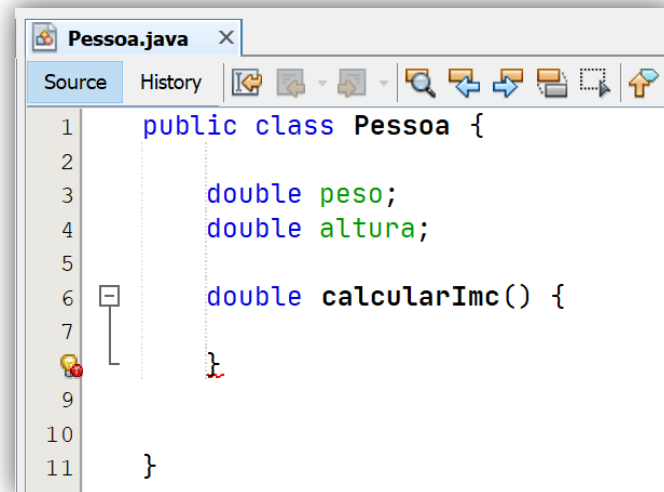
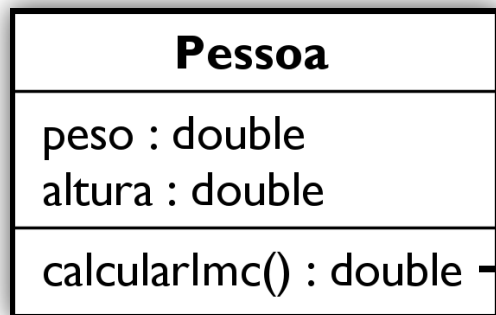
Pessoa
peso : double altura : double
calcularImc() : double



```
Pessoa.java x
Source History
1 public class Pessoa {
2
3     double peso;
4     double altura;
5
6 }
7
8
```

# Tradução das operações em métodos

- As operações da classe são traduzidas em Java em **métodos**
- Sintaxe: `tipoDeDado nomeMétodo(parâmetros)`



```
Pessoa.java x
Source History
1 public class Pessoa {
2
3     double peso;
4     double altura;
5
6     double calcularImc() {
7
8     }
9
10
11 }
```

- O corpo do método realiza alguma computação e pode:
  - Alterar o estado do objeto
  - Não alterar o estado do objeto, mas devolver um valor

# Métodos - devolvendo um valor

Método que não altera o estado do objeto, mas devolve um valor

```
12 public class Pessoa {  
13  
14     double altura;  
15     double peso;  
16  
17     double calcularImc() {  
18         return peso / (altura * altura);  
19     }  
20  
21 }
```

Métodos que retornam um valor devem declarar o tipo do dado devolvido

O corpo do método pode utilizar o valor dos atributos para realizar alguma computação



# Métodos – alterando estado do objeto

Talvez o método realiza alguma computação que causa a mudança de estado do objeto. Neste caso, geralmente não devolve um valor.

Pessoa
peso : double altura : double
calcularImc() : double comer(quantidade : double) : void

Como o método apenas alterará o estado do objeto, deve indicar que não devolverá valor algum (void)

Método utilizado para indicar que a pessoa se alimentou de uma determinada quantidade que está expressa em gramas.

```
Pessoa.java x
Source History
1 public class Pessoa {
2
3     double peso;
4     double altura;
5
6     void comer(double quantidade) {
7         peso = peso + (quantidade/1000);
8     }
9
10    double calcularImc() {
11        return peso / (altura * altura);
12    }
13
14 }
```

# **Declaração e criação de objetos**

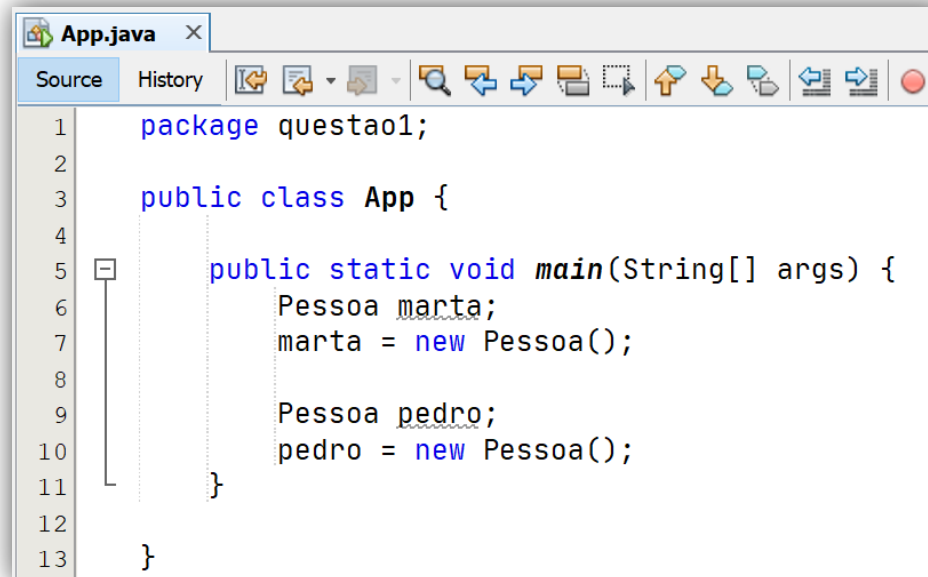
# Declaração e criação de objetos

- Declaração de variável de referência, para acessar um objeto:

Sintaxe: `Classe nomeVariavel;`

- Criação de objetos:

Sintaxe: `nomeVariavel = new Classe();`



```
1 package questao1;
2
3 public class App {
4
5     public static void main(String[] args) {
6         Pessoa marta;
7         marta = new Pessoa();
8
9         Pessoa pedro;
10        pedro = new Pessoa();
11    }
12
13 }
```

# Manipulação de objetos

# Manipulação de objetos

- Para alterar valor de atributos de um objeto ou solicitar a execução de métodos, deve-se utilizar o **operador de membro**
- O operador de membro é o ponto (.)
- A sintaxe de uso é:

```
nomeVariavel.membro
```

- Exemplo:

```
marta.altura = 1.70;  
marta.peso = 79;  
double imc = marta.calcularImc();
```

# Variável do tipo referência

```
Pessoa p1;  
p1 = new Pessoa();
```



Não é variável de tipo de dado primitivo.  
É uma variável do **tipo referência**

Variável do tipo referência armazena o endereço de um objeto.

# Áreas de Memória do Java

- **Stack**
  - Área de memória utilizada para armazenar:
    - variáveis locais
    - Variáveis paramétricas (parâmetros de métodos)
    - chamadas de funções
- **Heap**
  - Armazena os objetos

# O operador new

- O operador **new** cria objetos (instanciação de classe).
- O operador **new** faz três operações:
  - 1) Cria o objeto na memória, alocando espaço para armazenar valores para suas variáveis de instância;
  - 2) Inicializa as variáveis de instância
  - 3) Retorna o endereço de memória criado pelo objeto.

