

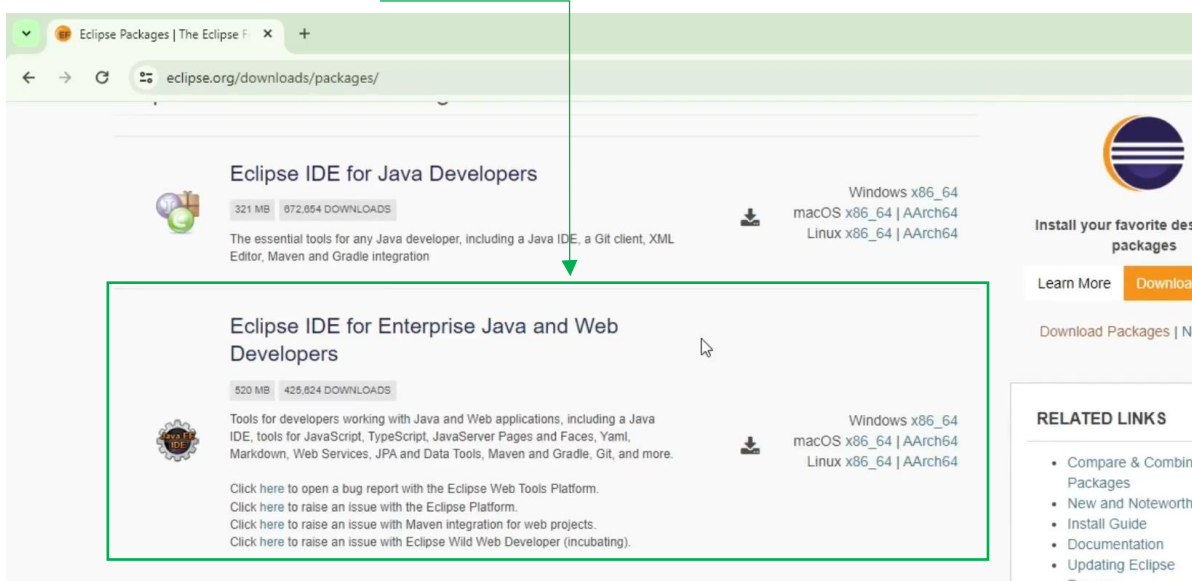
## Roteiro para aula prática 1 - Classe POJO e sobrescrita

**Objetivo da aula:** relembrar classe POJO e sobrescrita de métodos.

Nessa aula prática será utilizado o Eclipse.

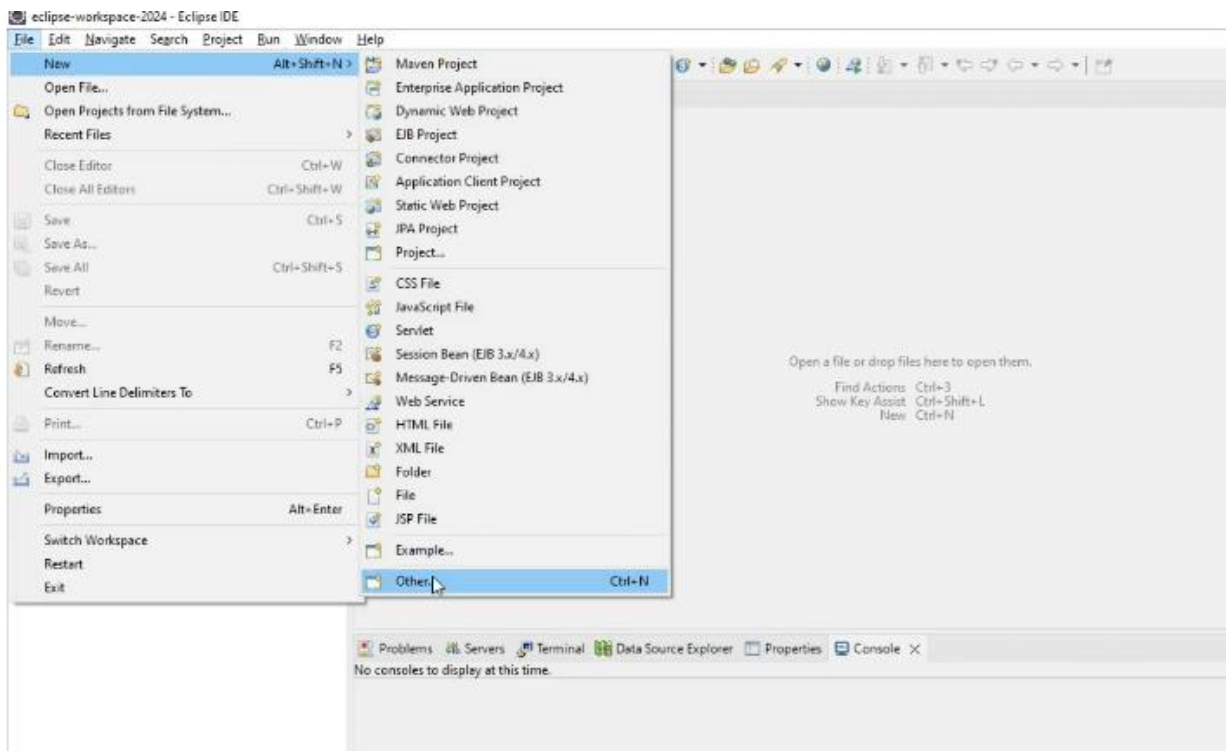
**Para baixar o eclipse:** <https://www.eclipse.org/downloads/packages/>

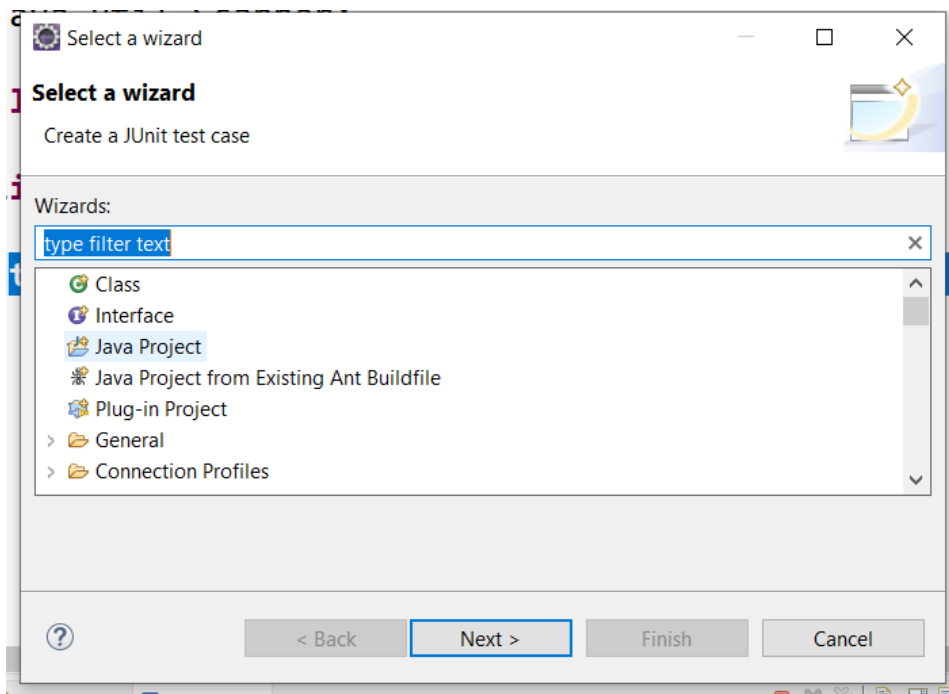
Pode baixar a versão Java para Web



## Criando Projeto Java

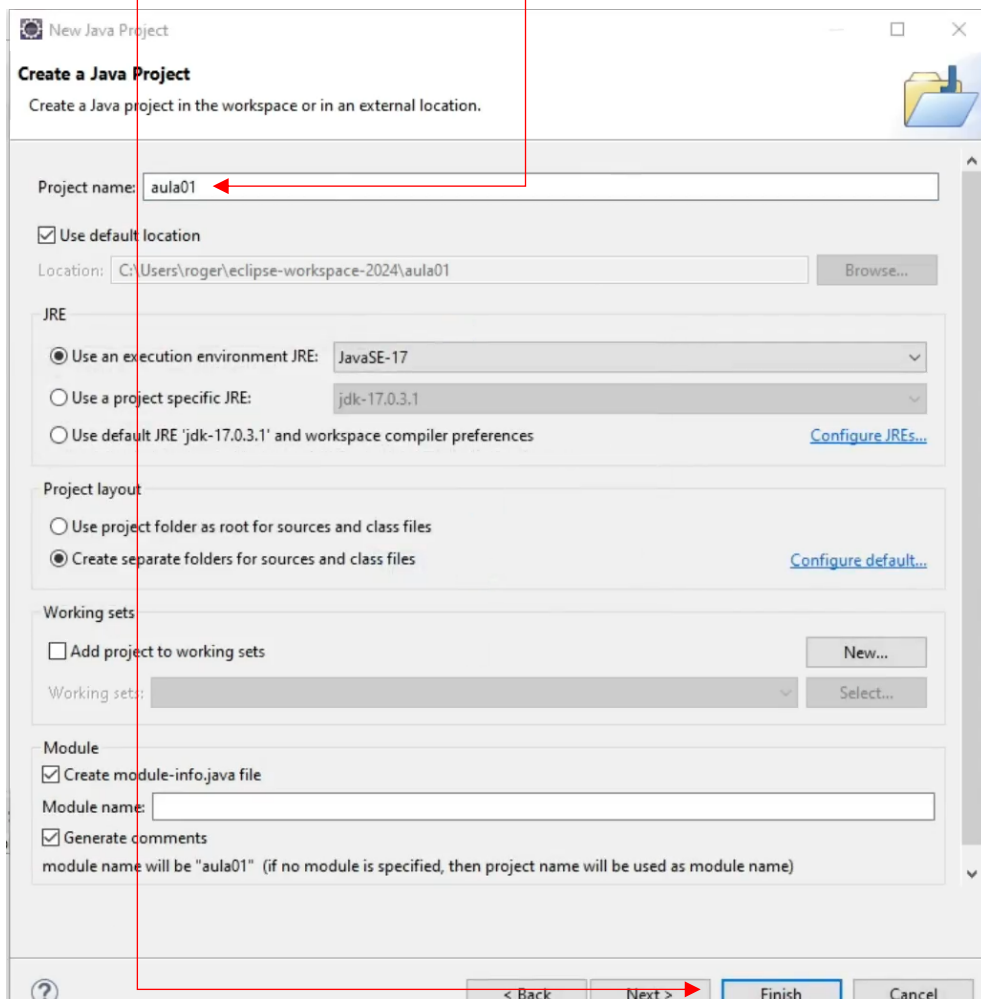
Vamos criar um projeto Java em File > New > Other>Java Project





Vamos dar um nome ao projeto: aula01

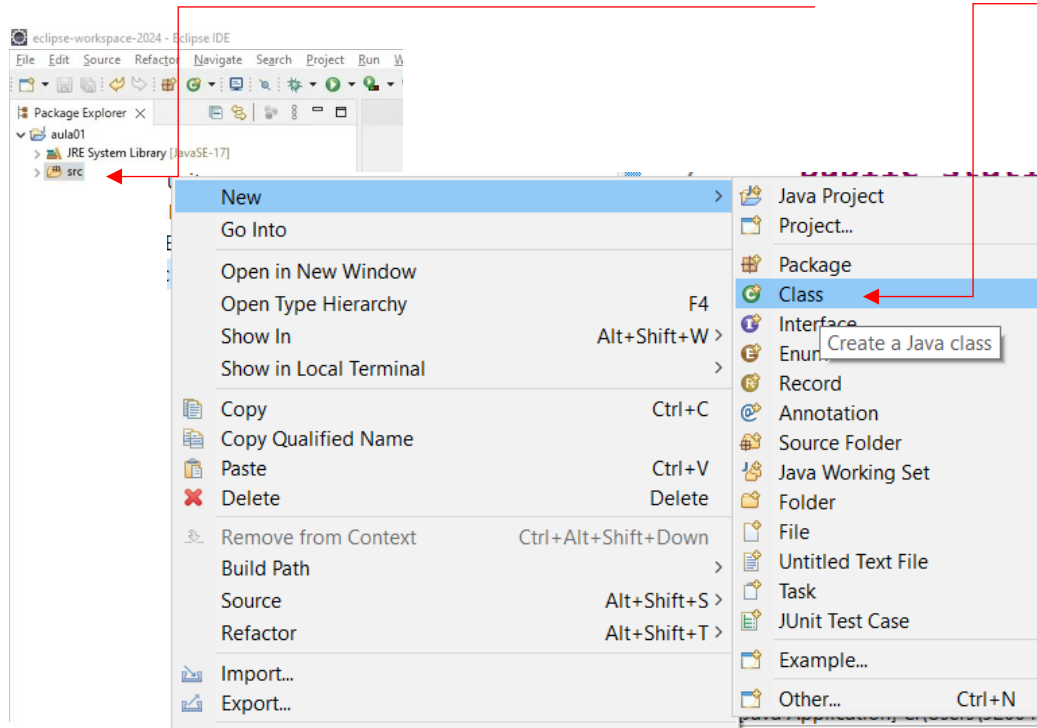
Clicar em Finish



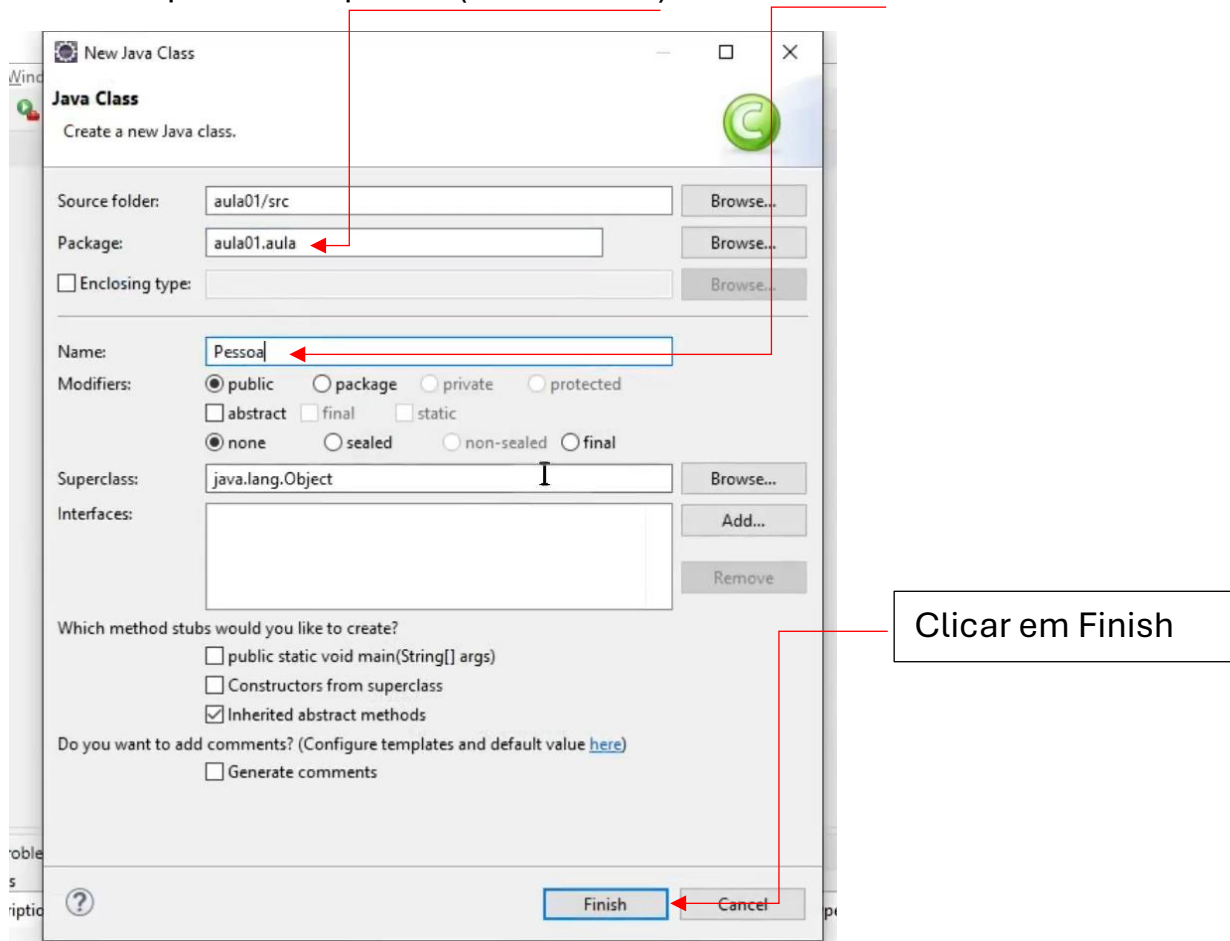
## Criando classe POJO "Plain Old Java Object" (Objeto Java Simples)

Vamos criar classe POJO Pessoa :

Clique com o botão direito do mouse na pasta src, escolha New > Class



Vamos especificar o pacote(aula01.aula) e um nome à nossa classe: Pessoa



## Criando atributos

Nossa classe Pessoa terá apenas 2 atributos: nome e idade.

Para o encapsulamento inicial marcaremos os atributos como privados e posteriormente ofereceremos métodos públicos de captura e configuração (getters e setters).

Abaixo podemos ver o código da nossa classe Pessoa.

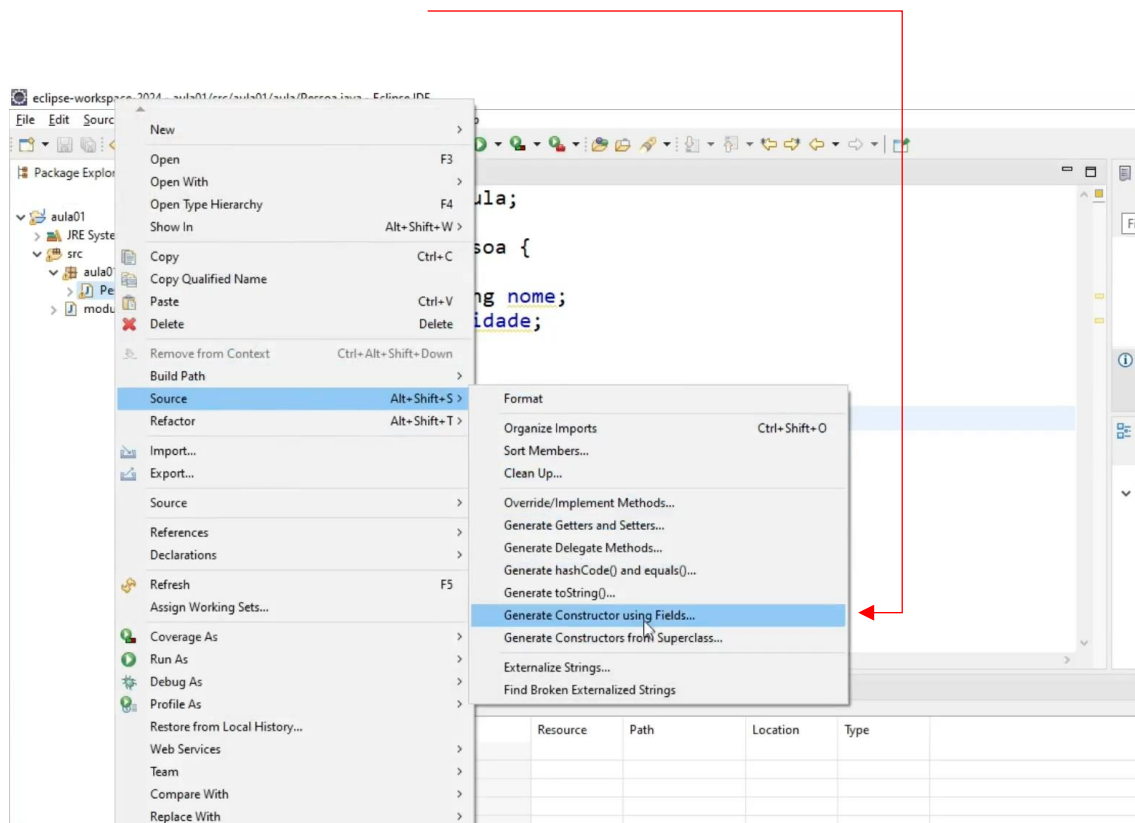
## Pessoa.java

```
public class Pessoa {  
  
    private String nome;  
    private int idade;  
  
}
```

## Criando Construtor

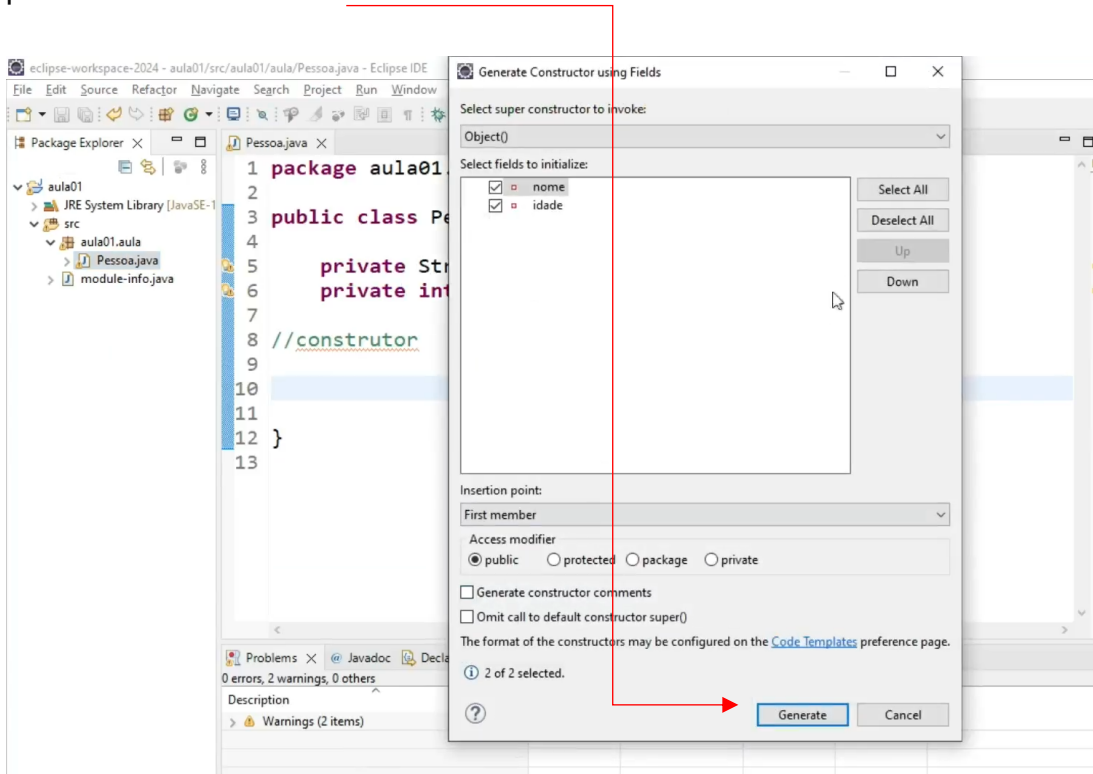
Vamos implementar o construtor padrão e alternativo utilizando a IDE.

Para isso, selecione a classe Pessoa e clique com botão direito do mouse. No menu suspenso escolha Source > Generate Constructor using Fields

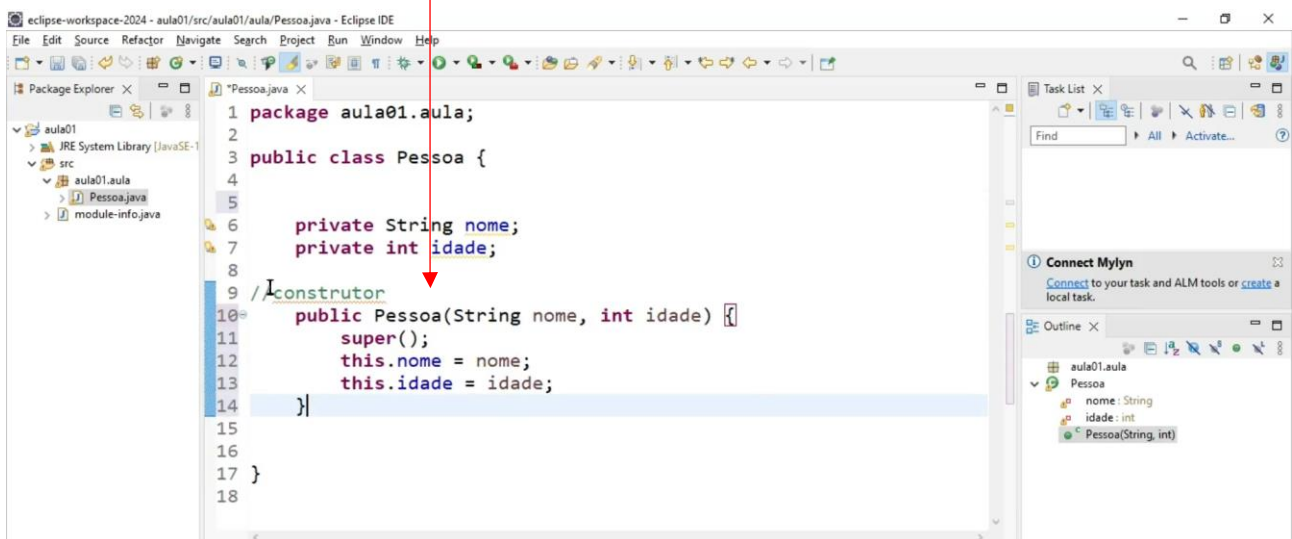


A seguinte janela se abrirá (Generate Construtor using Fields).

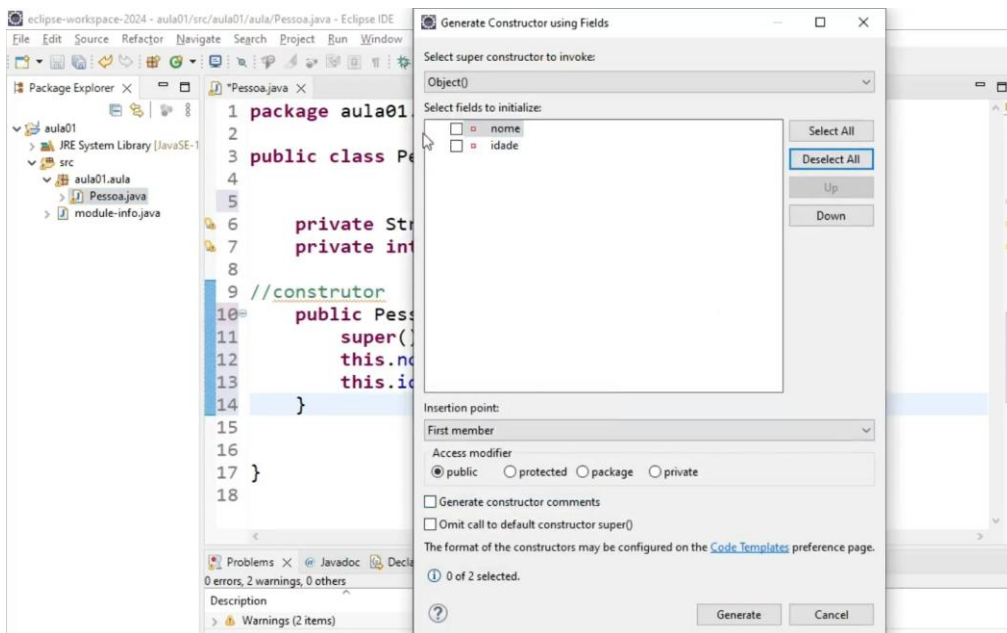
Vamos selecionar os campos nome e idade (pode também clicar no botão Select All) e posteriormente clicar no botão Generate.



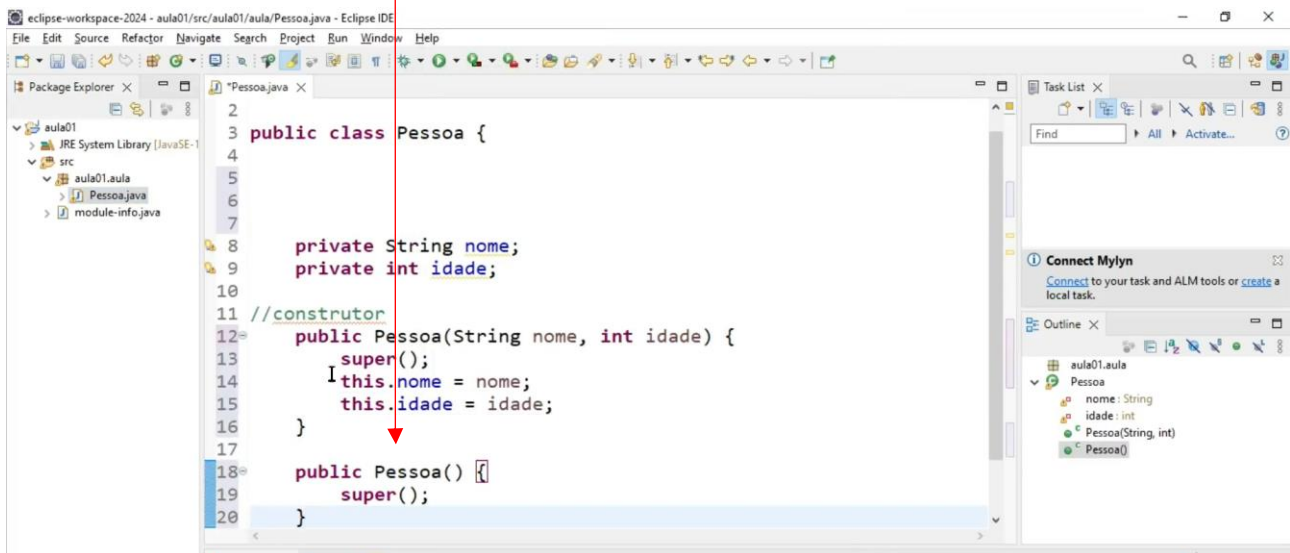
A IDE irá criar o construtor como mostrado na imagem abaixo



Vamos criar também um construtor vazio, que é o construtor padrão. Sendo assim não será necessário selecionar os atributos nome e idade.



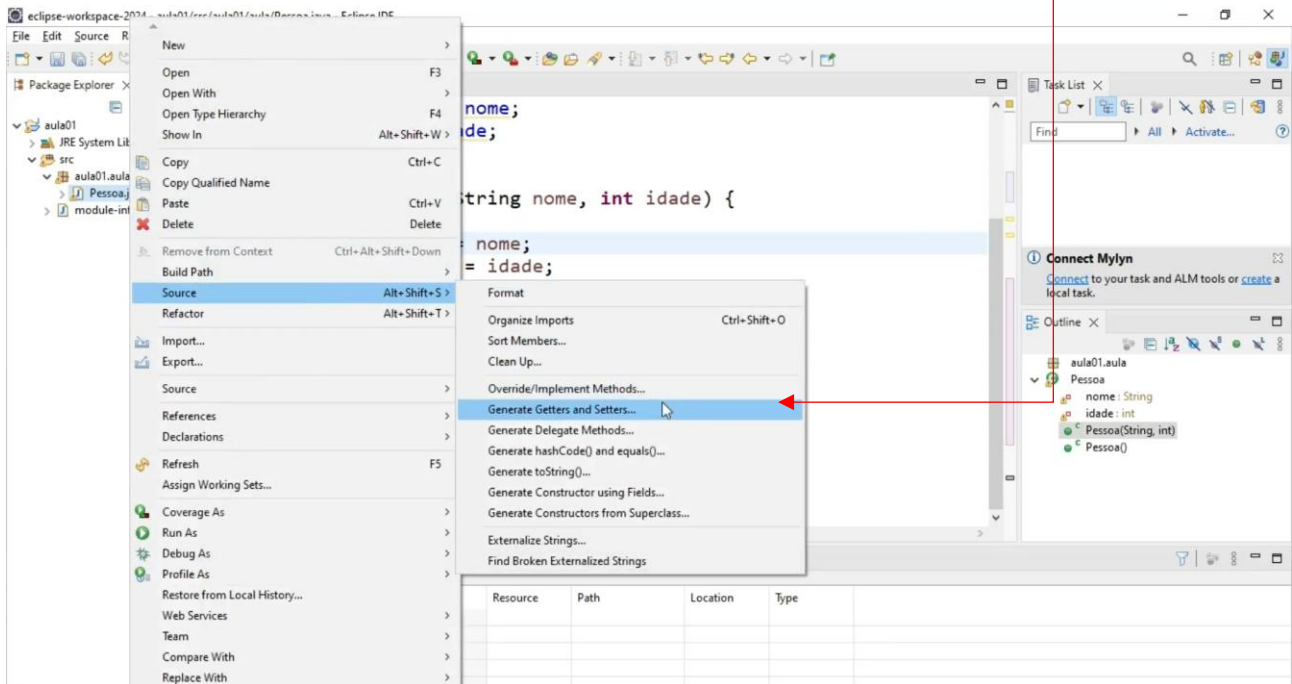
Temos abaixo o código do construtor padrão criado pela IDE.



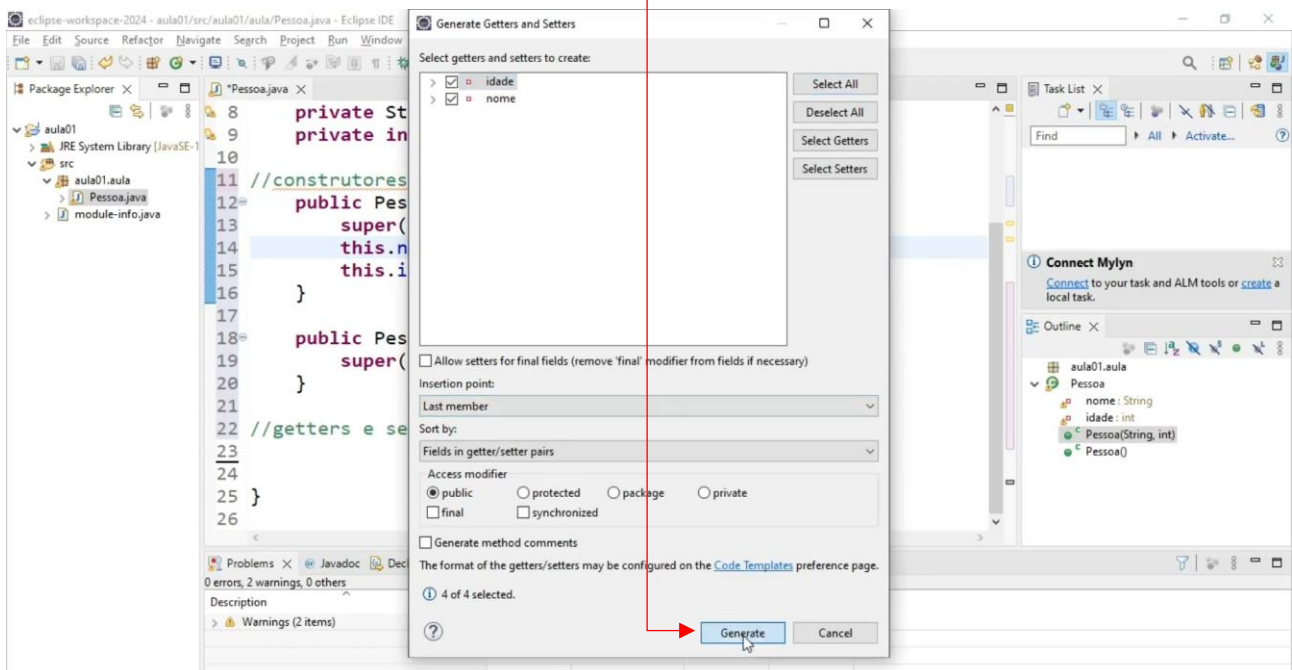


## Criando métodos getters e setters

Para a criação dos métodos getters e setter, vamos utilizar o mesmo processo que utilizamos para o construtor. Selecione a classe Pessoa e clique com botão direito do mouse. No menu suspenso escolha Source > Generate Getter and Setters

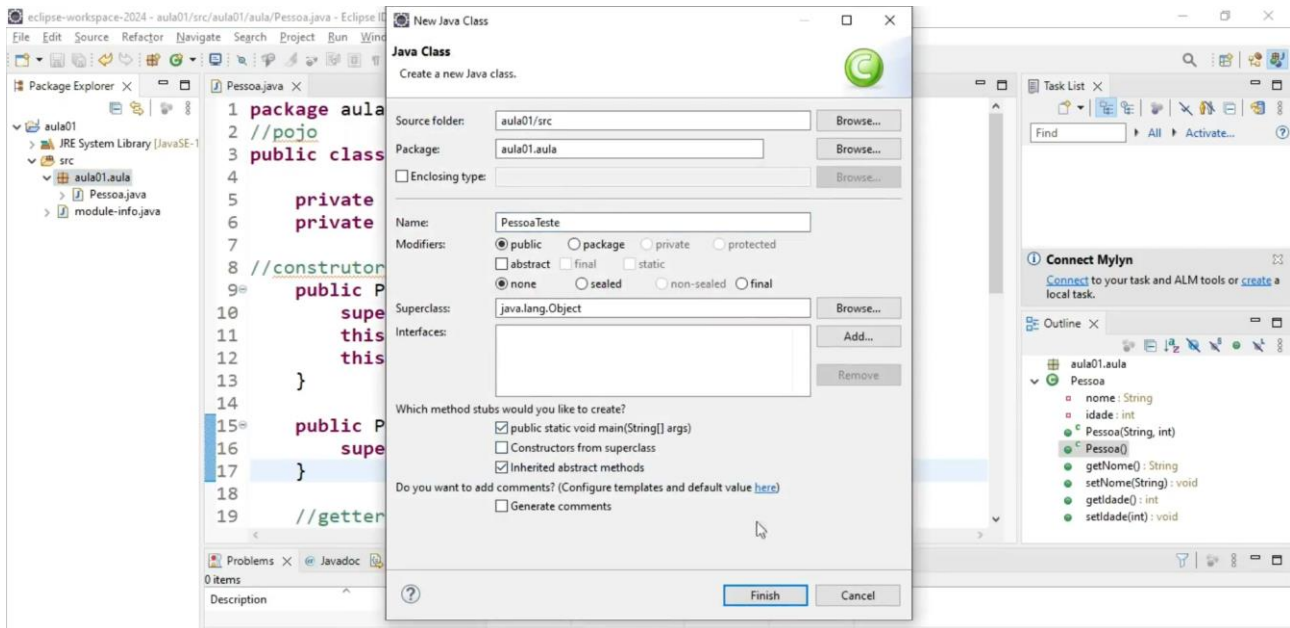


Selecionar idade e nome e clicar em Generate.

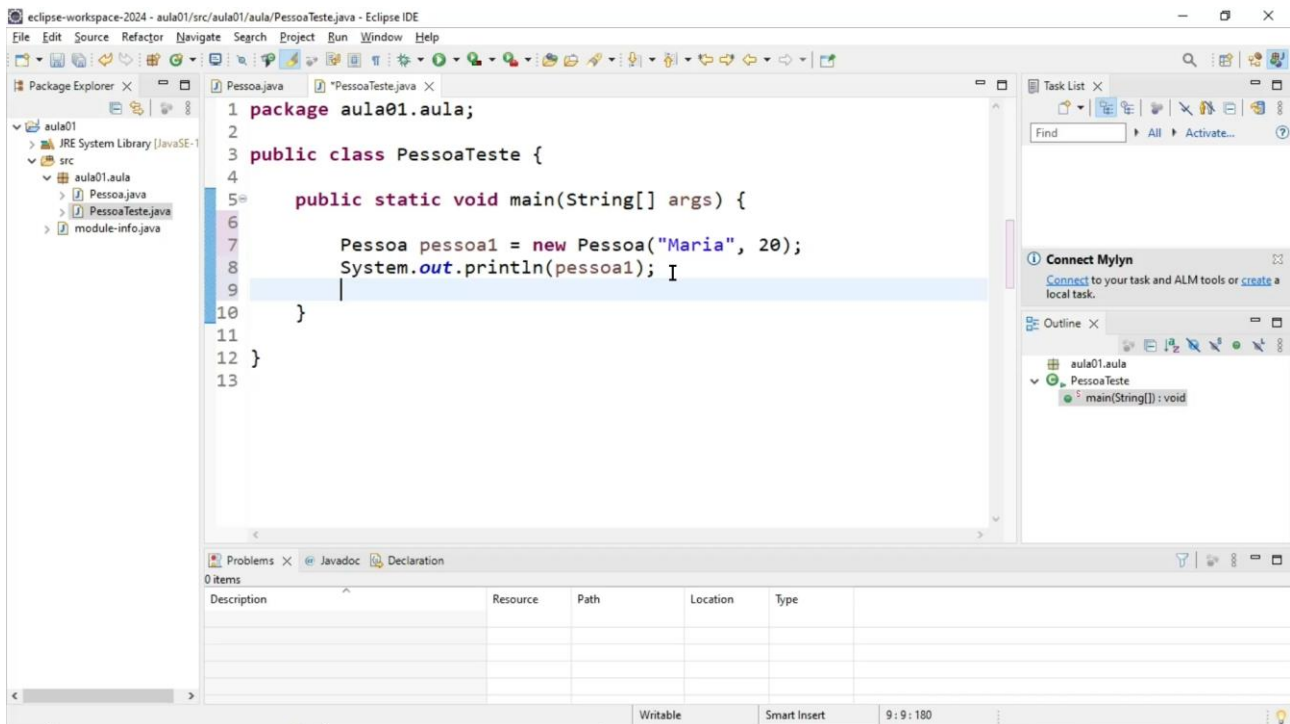


## Criando a classe de Teste

Agora que terminamos a classe POJO, vamos sobrescrever os métodos toString, equals e hashCode. Para testar e compreender porque sobrescrevemos essas classes, vamos primeiro criar uma classe de teste. Nossa classe de teste irá se chamar PessoaTeste.java.



Nossa classe de teste tem a seguinte aparência





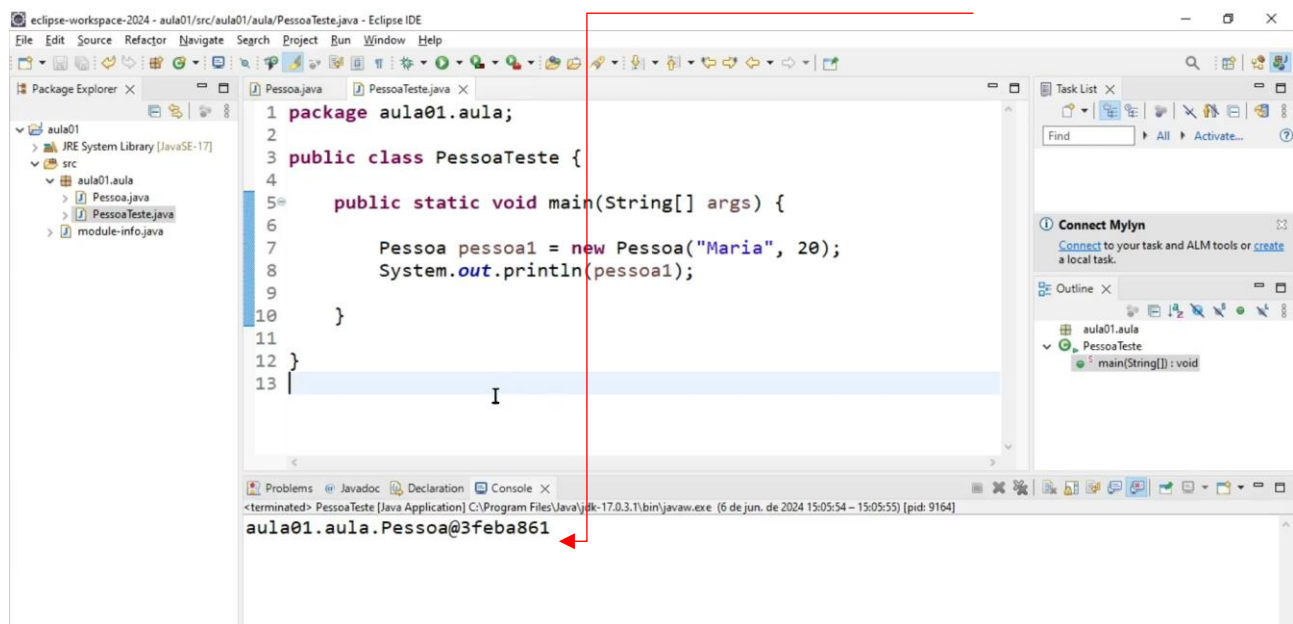
## Método toString

Abaixo temos o código da classe de teste para testarmos o método toString.

```
public class PessoaTeste {  
    public static void main(String[] args) {  
        Pessoa pessoa1 = nova Pessoa("Maria", 20);  
        System.out.print(pessoa1);  
    }  
}
```

O que será impresso?

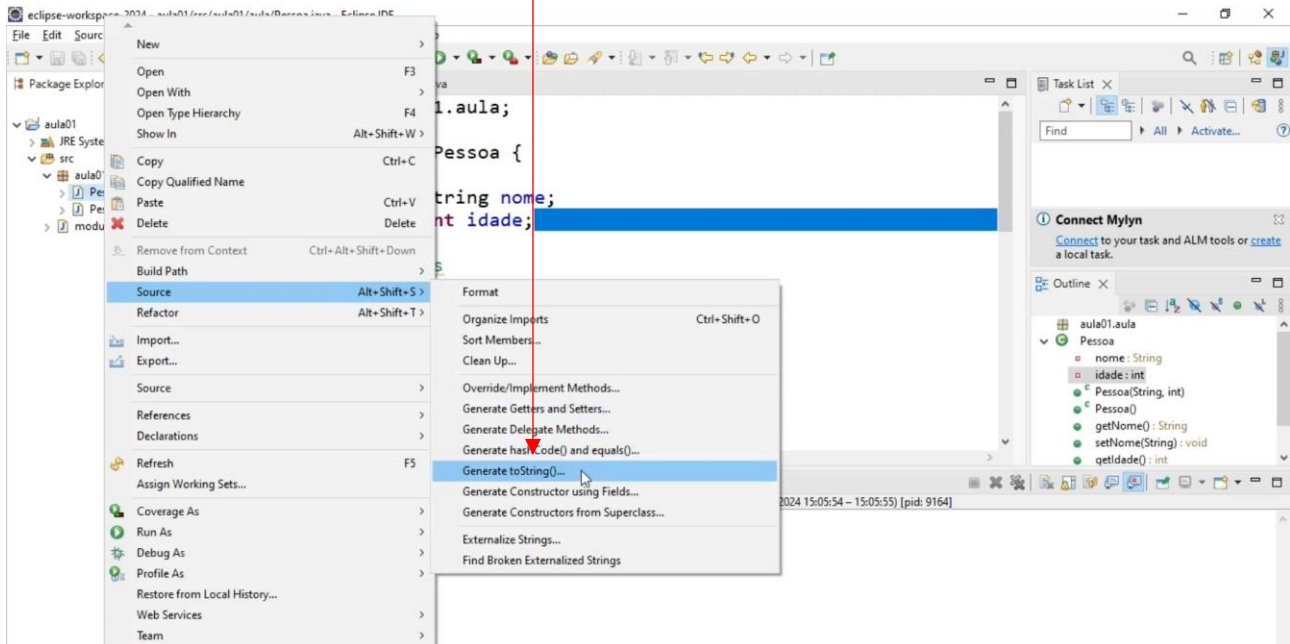
Se o método toString() não for sobrescrito em uma classe em Java, ele imprimirá uma representação padrão da instância do objeto.



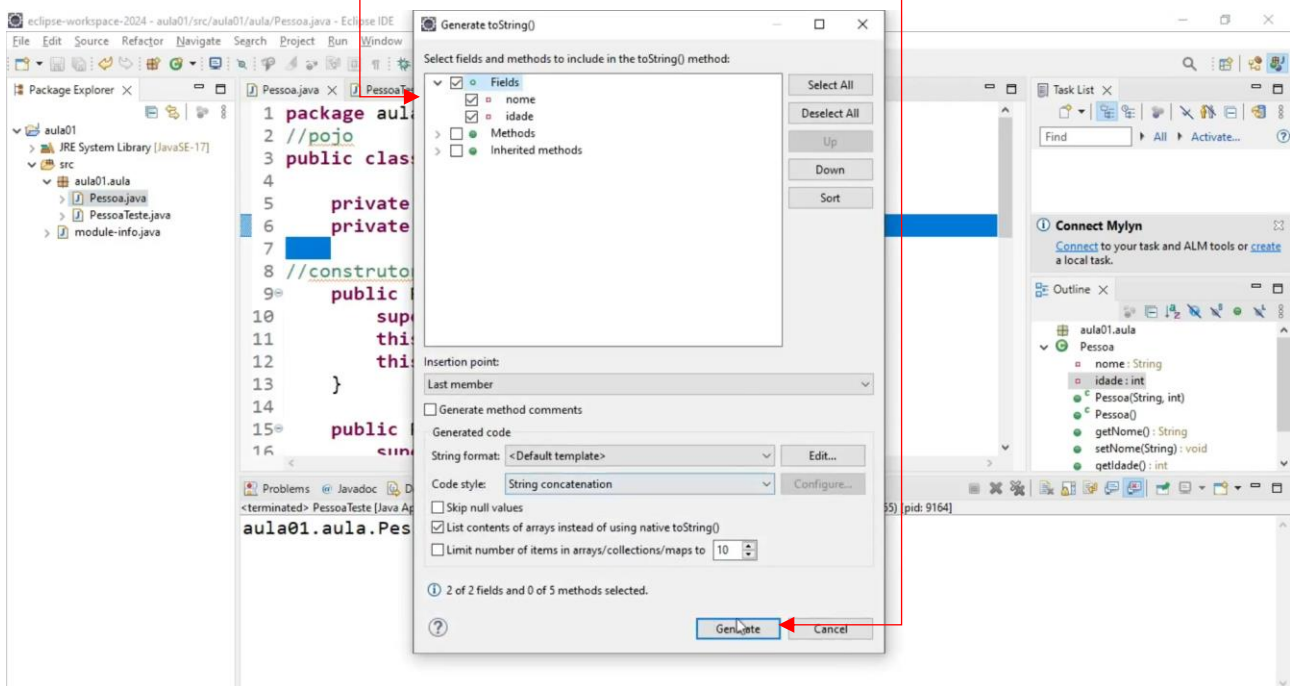
Essa informação impressa não é muito informativa e normalmente não é adequada para representar o estado de um objeto de maneira útil. É uma boa prática sobrescrever toString() para fornecer uma representação mais legível e significativa do objeto. A seguir vamos sobrescrever o método toString utilizando a IDE.

## Sobrescrevendo o método toString

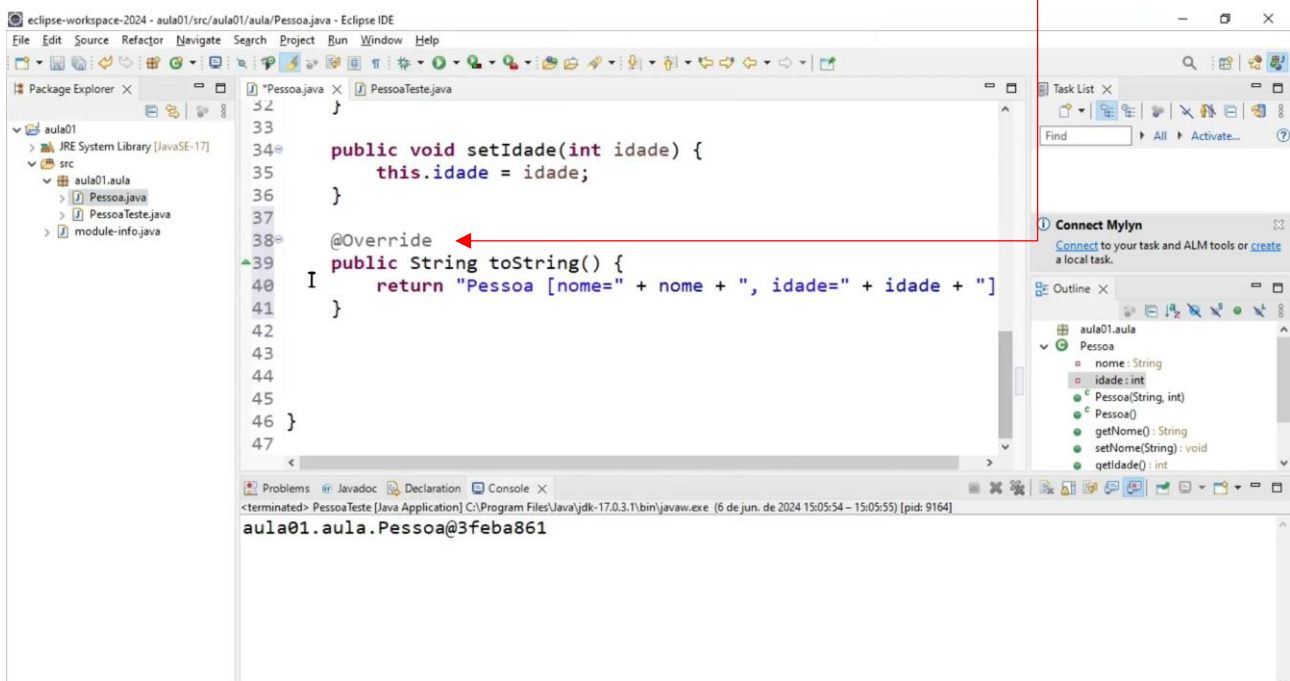
Selecione a classe Pessoa e clique com botão direito do mouse. No menu suspenso escolha Source > Generate toString().



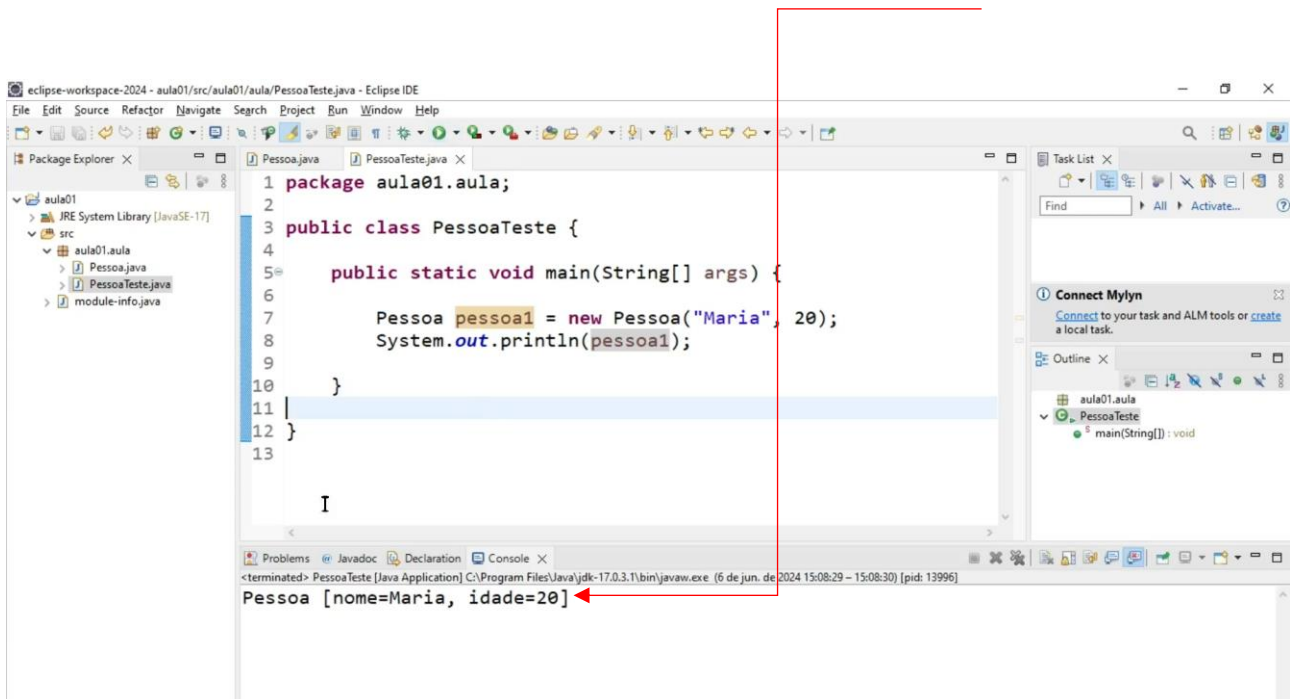
Selecione os campos(Fields) nome e idade e clique em Generate.



Observe no código gerado que ele já será gerado com a notação `@Override`.



Agora vamos executar a classe de teste novamente. Observe a saída.



Verifique no Console que foi impresso uma informação útil, com o nome e idade da `pessoa1`.

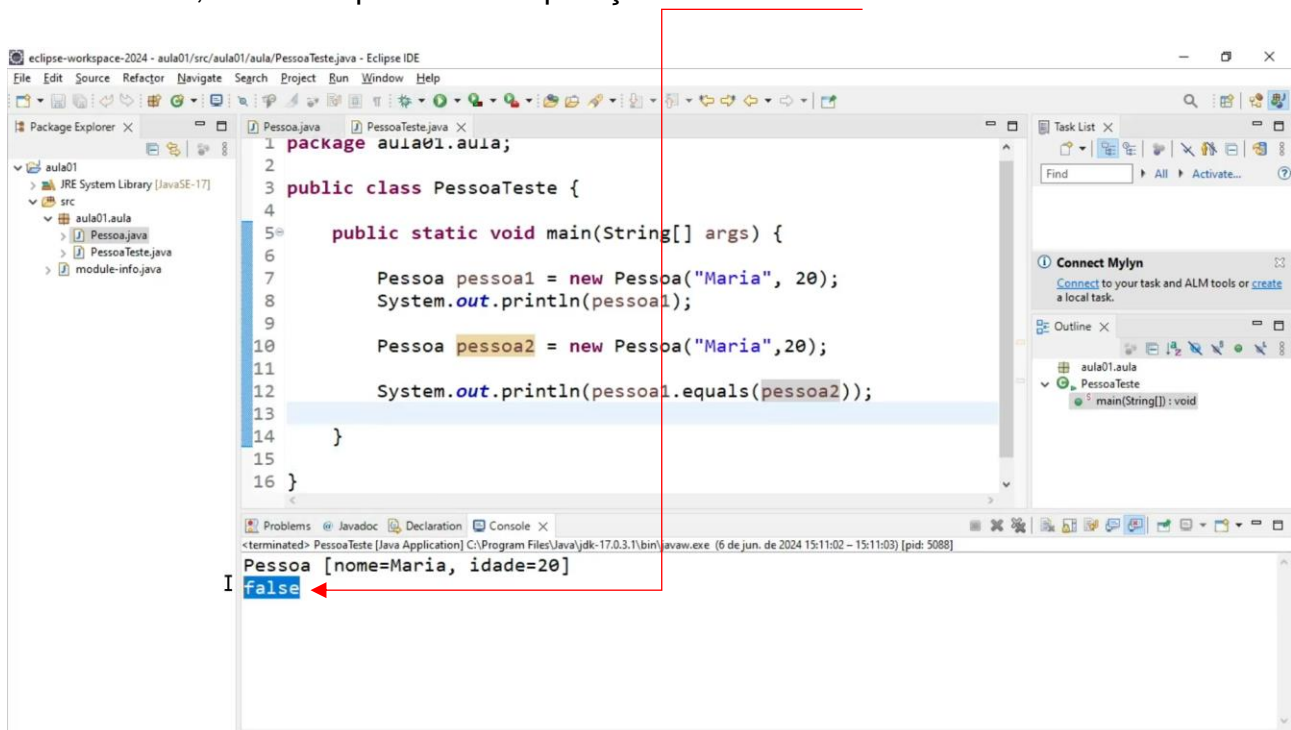
## Método equals

Agora veremos sobre o método equals(). Sua implementação original leva em consideração que dois objetos são iguais se apontam para o mesmo endereço na memória.

```
public class TestePessoa {  
    public static void main(String[] args) {  
  
        Pessoa pessoa1 = new Pessoa("Maria", 20);  
        Pessoa pessoa2 = new Pessoa("Maria", 20);  
  
        System.out.println(pessoa1.equals(pessoa2));  
  
    }  
}
```

Qual a saída?

Ao testarmos, veremos que essa comparação retornou false.



Para funcionar corretamente, devemos sobrescrever o método equals. Mas antes de sobrescrever esse método iremos ver o comportamento do método hashCode.

## Método hashCode

O método hashCode monta a tabela hash de modo correto usado para encontrar objetos em uma coleção. O método hashCode tem como objetivo fornecer um valor hash para a instância de um objeto, que é uma representação numérica do objeto. Este valor é utilizado por estruturas de dados baseadas em hash para armazenar e recuperar objetos de forma

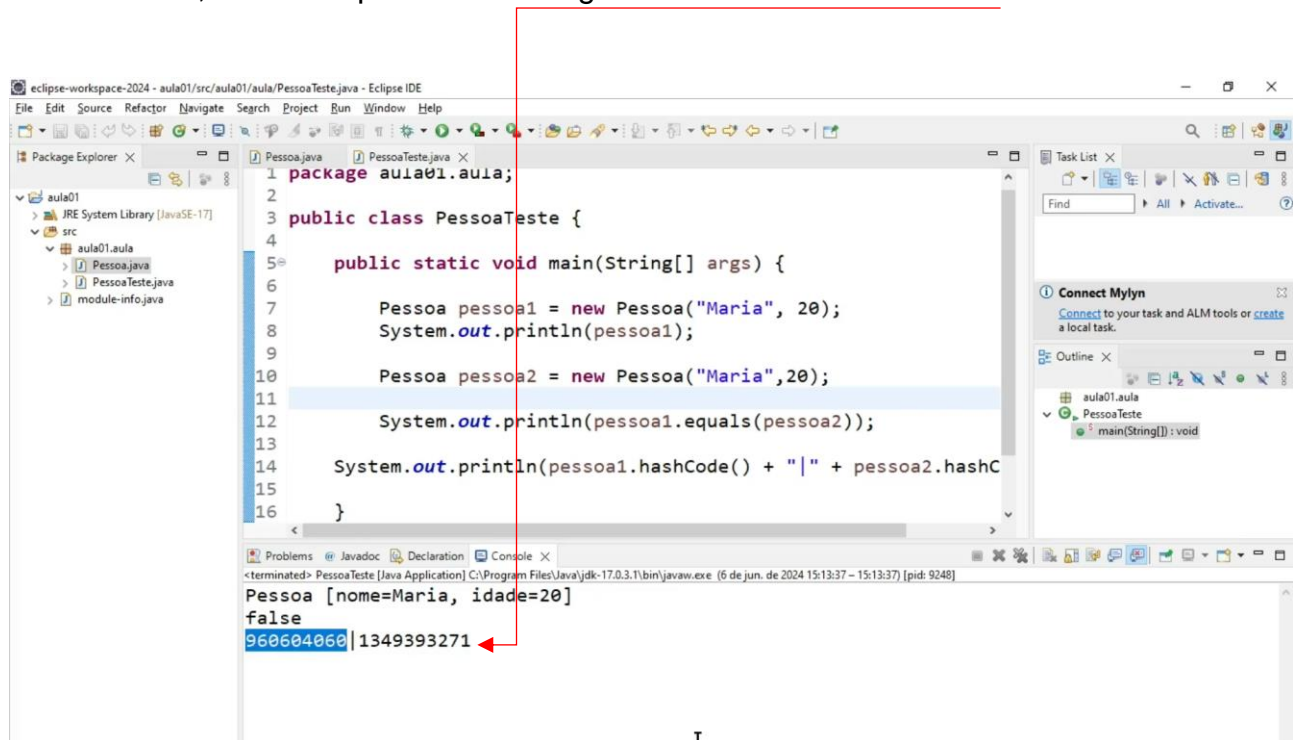
eficiente. É importante que objetos iguais (conforme determinado pelo método equals) tenham o mesmo valor hash.

Testaremos a aplicação a seguir adicionando a linha com o método hashCode.

```
public class TestePessoa {  
    public static void main(String[] args) {  
  
        Pessoa pessoa1 = new Pessoa("Maria", 20);  
        Pessoa pessoa2 = new Pessoa("Maria", 20);  
  
        System.out.print(pessoa1); }  
        System.out.println(pessoa1.equals(pessoa2));  
  
        System.out.println(pessoa1.hashCode() + " | " +  
        pessoa2.hashCode());  
    }  
}
```

Qual a saída?

Ao testarmos, veremos que os dois códigos retornados são diferentes.

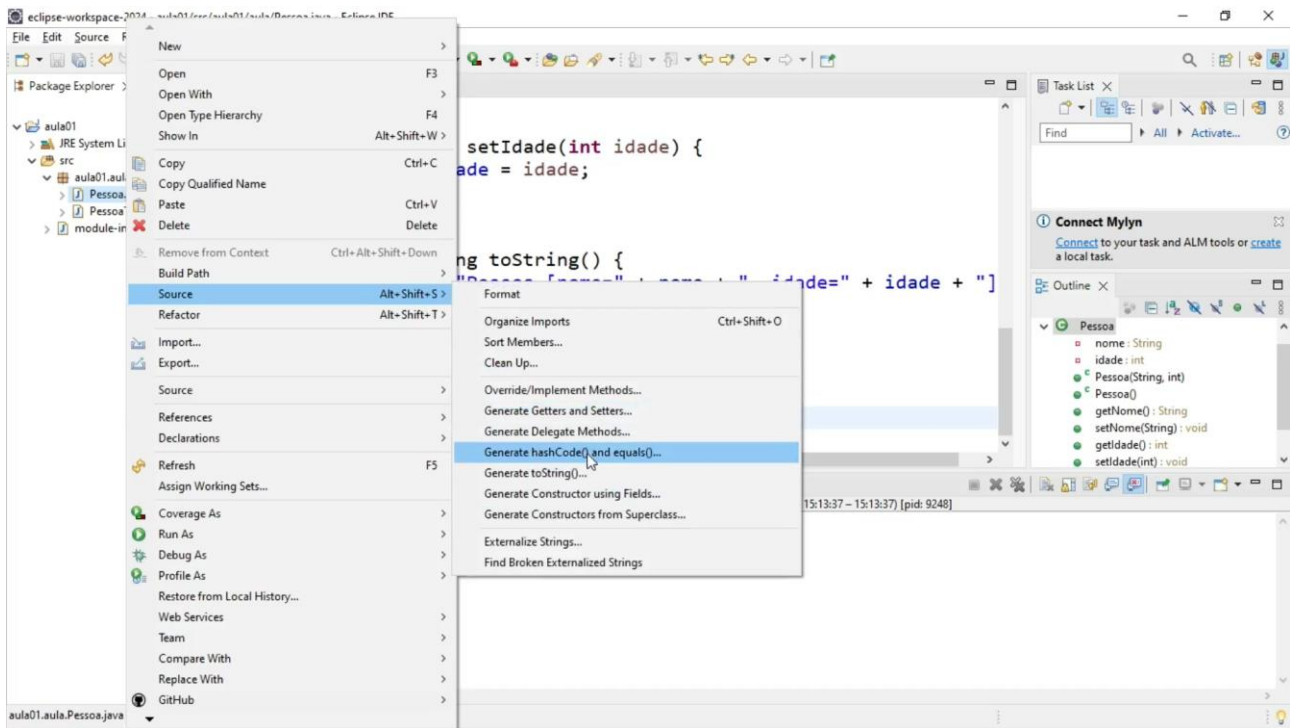


Para que funcione corretamente, devemos sobrescrever o método hashCode.

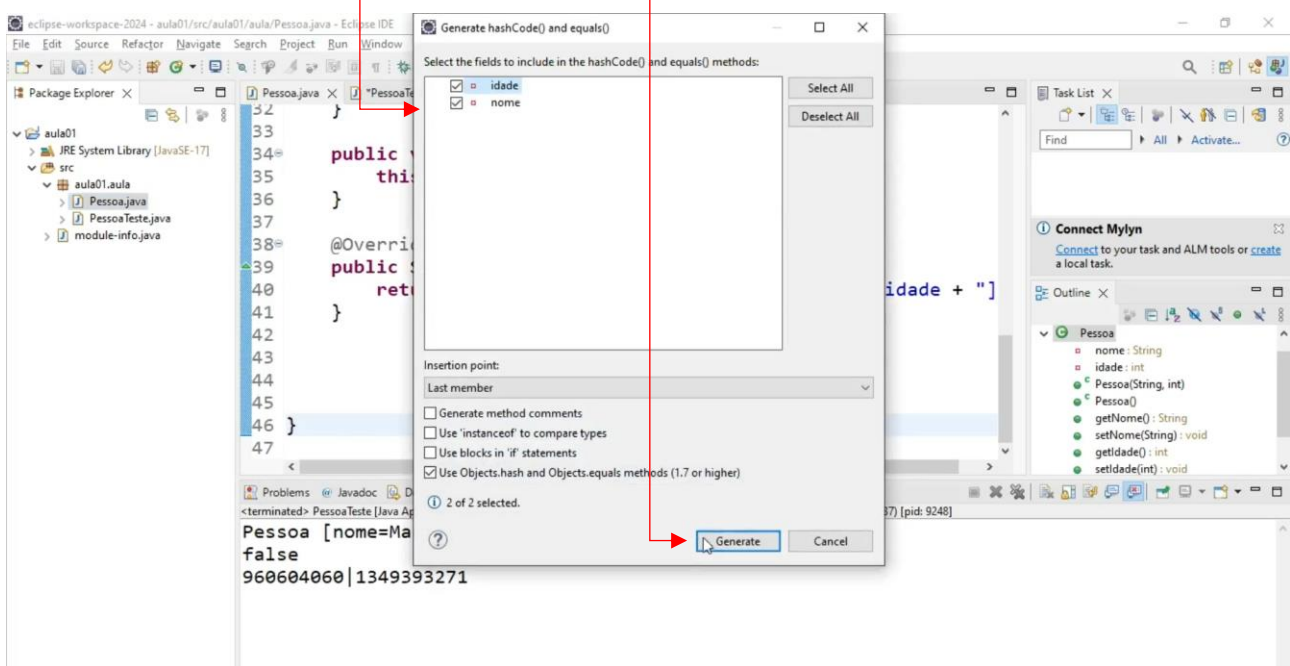
## Sobrescrevendo o método equals e hashCode

Selecione a classe Pessoa e clique com botão direito do mouse. No menu suspenso escolha Source > Generate hashCode() and equals().

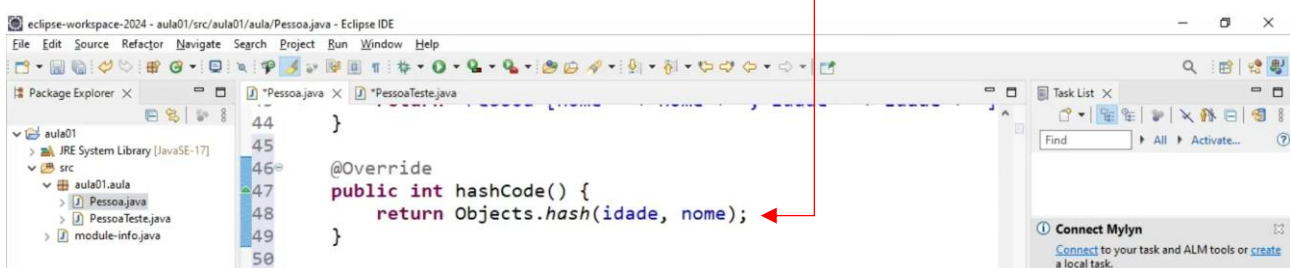




Selecione os campos(Fields) nome e idade e clique em Generate.

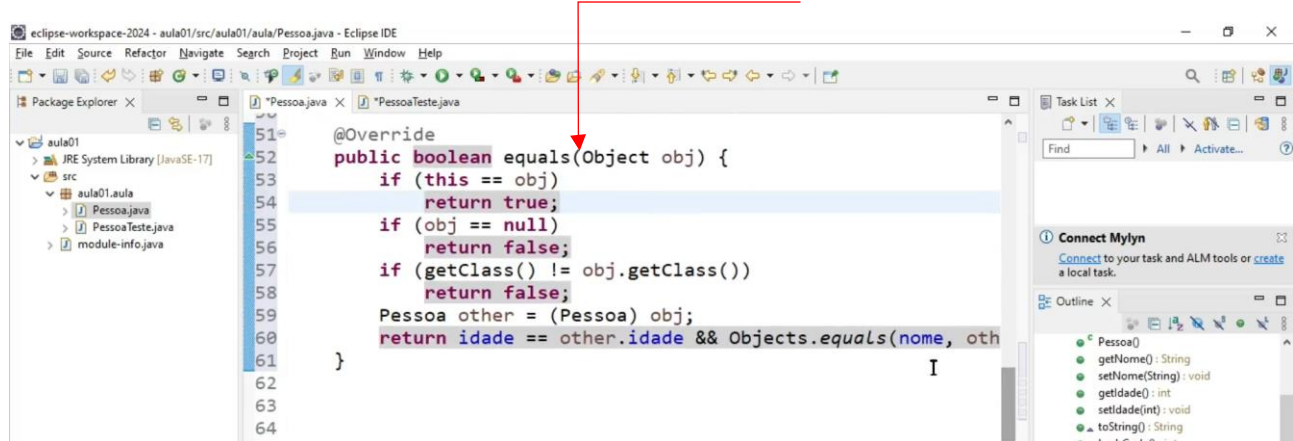


Observe que foi gerado um código referente ao método hashCode().



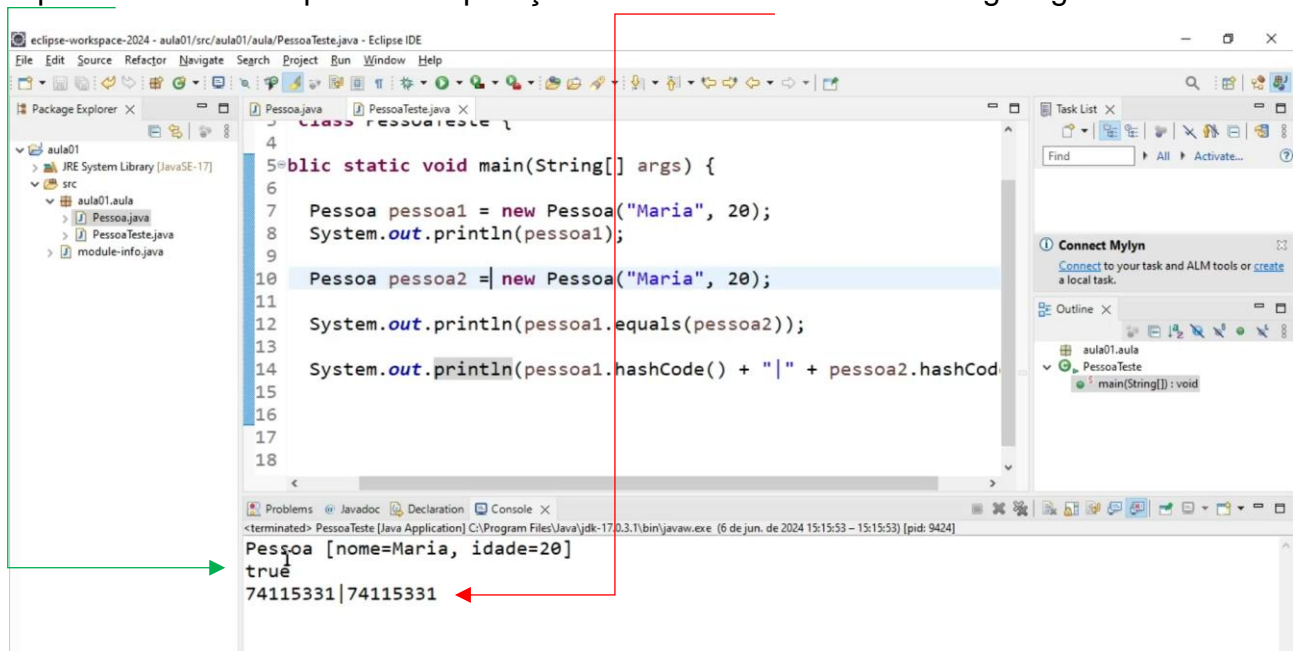


E também doi gerado um código referente ao método equals().



```
51 @Override
52 public boolean equals(Object obj) {
53     if (this == obj)
54         return true;
55     if (obj == null)
56         return false;
57     if (getClass() != obj.getClass())
58         return false;
59     Pessoa other = (Pessoa) obj;
60     return idade == other.idade && Objects.equals(nome, oth
61 }
62
63
64
```

Vamos executar novamente a classe de teste. Observe a saída, verifique que agora o equals retornou true para a comparação e o hashCode retornou códigos iguais.



```
4
5 public static void main(String[] args) {
6
7     Pessoa pessoa1 = new Pessoa("Maria", 20);
8     System.out.println(pessoa1);
9
10    Pessoa pessoa2 = new Pessoa("Maria", 20);
11
12    System.out.println(pessoa1.equals(pessoa2));
13
14    System.out.println(pessoa1.hashCode() + "|" + pessoa2.hashCode());
15
16
17
18

```

Pessoa [nome=Maria, idade=20]  
true  
74115331|74115331