

Programação Orientada a Objetos – Aula 03

Prof. Dr. Eduardo Takeo Ueda
eduardo.tueda@sp.senac.br

Encapsulamento

- Ocultar detalhes internos de uma classe para proteger a implementação de riscos externos
- O encapsulamento permite visualizar um objeto como uma caixa preta
- “Não mostrar suas cartas em um jogo de baralho!!!”

Criação de classes em Java

```
[package nomeDoPacote]
```

```
[import * ou classe]
```

```
public class NomeDaClasse {
```

```
    [Construtores]
```

```
    [Atributos]
```

```
    [Métodos]
```

```
}
```

**Membros
da classe**

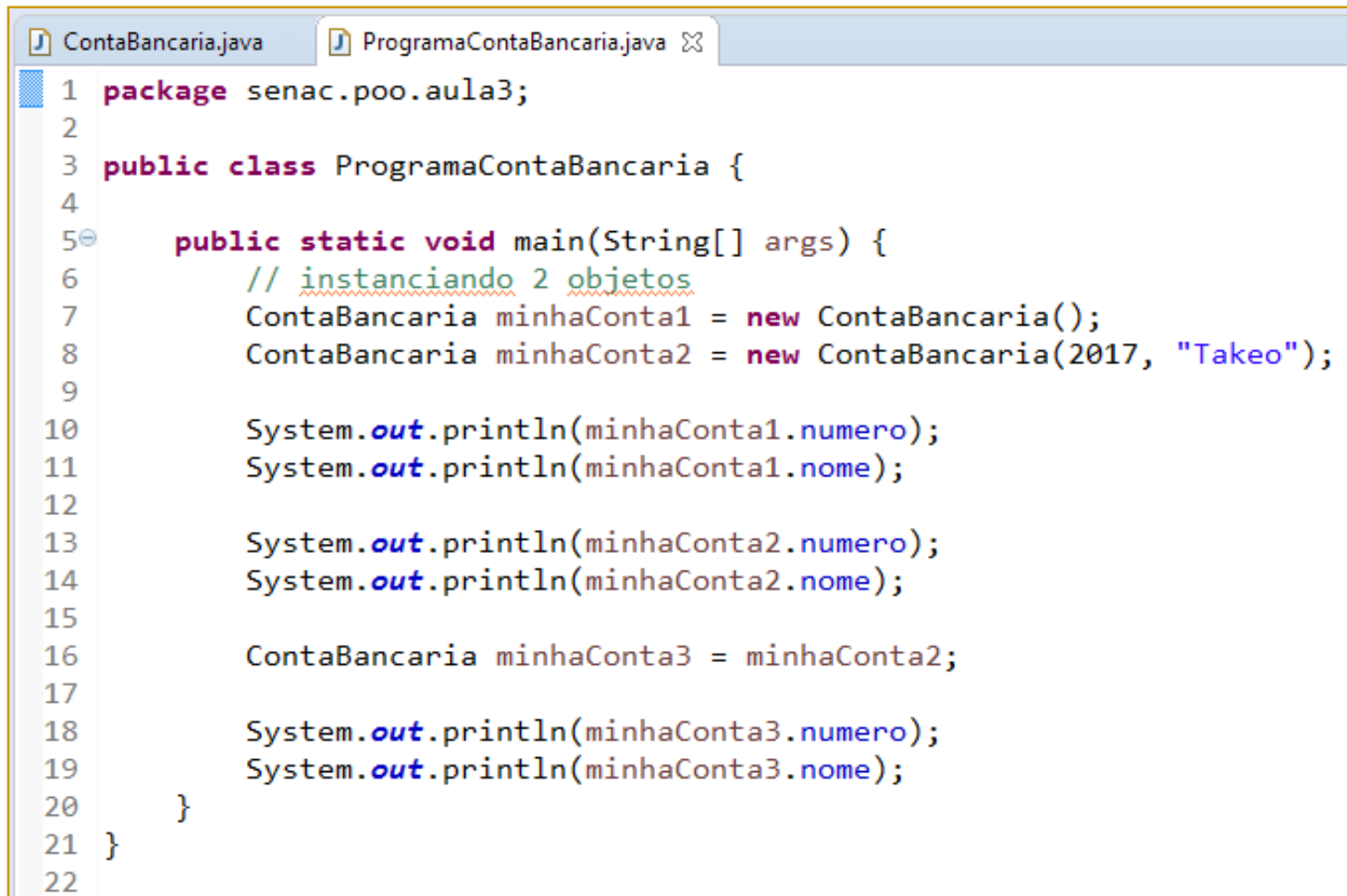
Construtores

- Determina que ações devem ser executadas na criação de um objeto
- Em Java, o construtor é definido como um método cujo nome deve ser o mesmo nome da classe e sem indicação do tipo de retorno (nem mesmo void)
- Por padrão, o compilador fornece um construtor default sem parâmetros em qualquer classe que não inclua explicitamente um construtor
- Uma classe pode ter um ou mais construtores

Exemplo de construtores

```
ContaBancaria.java ✕
1 package senac.poo.aula3;
2
3 public class ContaBancaria {
4     // declaração de atributos
5     int numero;
6     String nome;
7     double saldo;
8     double limite;
9
10    // construtor padrão (default)
11    // ContaBancaria() {
12    // }
13
14    // 1o construtor da classe
15    ContaBancaria() {
16        System.out.println("Construindo uma conta bancária!");
17    }
18
19    // 1o construtor da classe
20    ContaBancaria(int numero, String nome) {
21        this.numero = numero;
22        this.nome = nome;
23    }
24 }
25
```

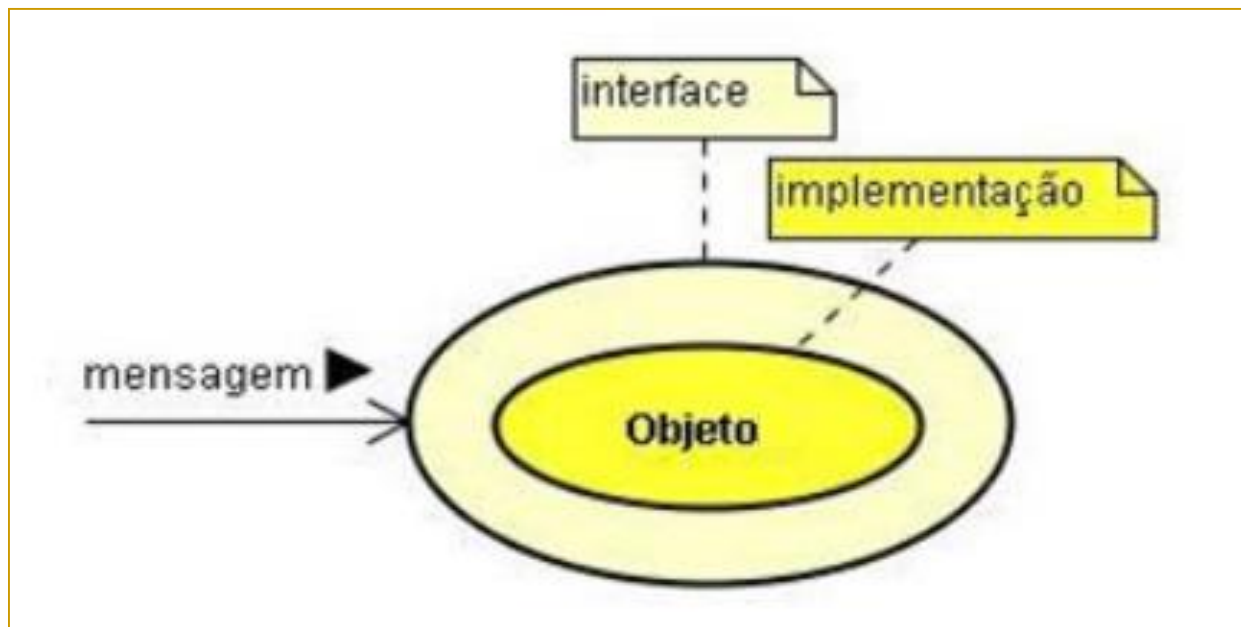
Exemplo de construtores



```
1 package senac.poo.aula3;
2
3 public class ProgramaContaBancaria {
4
5     public static void main(String[] args) {
6         // instanciando 2 objetos
7         ContaBancaria minhaConta1 = new ContaBancaria();
8         ContaBancaria minhaConta2 = new ContaBancaria(2017, "Takeo");
9
10        System.out.println(minhaConta1.numero);
11        System.out.println(minhaConta1.nome);
12
13        System.out.println(minhaConta2.numero);
14        System.out.println(minhaConta2.nome);
15
16        ContaBancaria minhaConta3 = minhaConta2;
17
18        System.out.println(minhaConta3.numero);
19        System.out.println(minhaConta3.nome);
20    }
21 }
22
```

Voltando a falar de encapsulamento

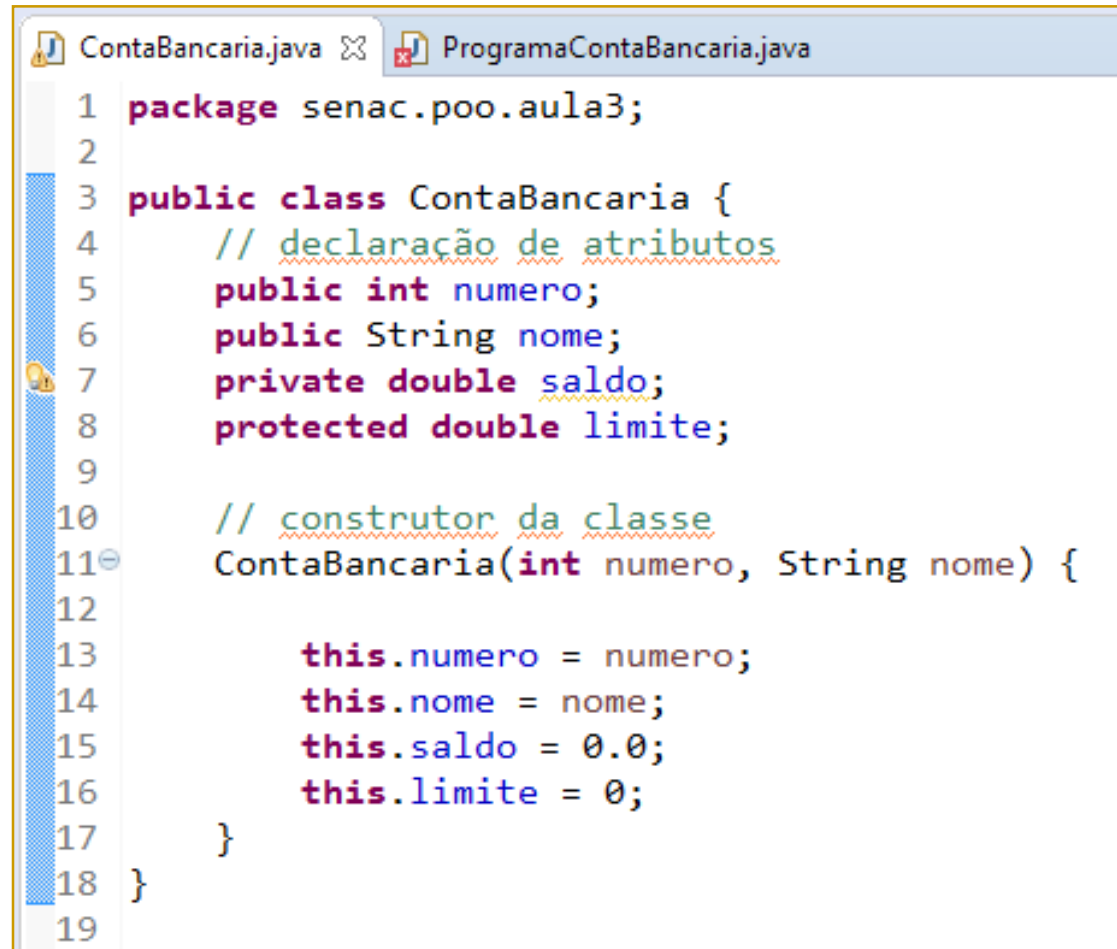
- **Encapsulamento** é a capacidade de ocultar/proteger partes (dados e detalhes) da implementação de uma classe



Modificadores de Acesso

- São palavras chave ou reservadas da linguagem Java que permitem ou proíbem o acesso aos atributos e/ou métodos das classes
- **public**: garante que um atributo ou método da classe seja acessado ou executado a partir de qualquer outra classe (pode modificar classes também)
- **private**: assegura que um atributo ou método da classe seja acessado ou executado apenas por métodos da mesma classe (**Cuidado: não se aplica a classes!**)
- **protected**: funciona como o **private** exceto que as subclasses (classes filhas ou derivadas) também terão acesso ao atributo ou método (usado com **herança**)

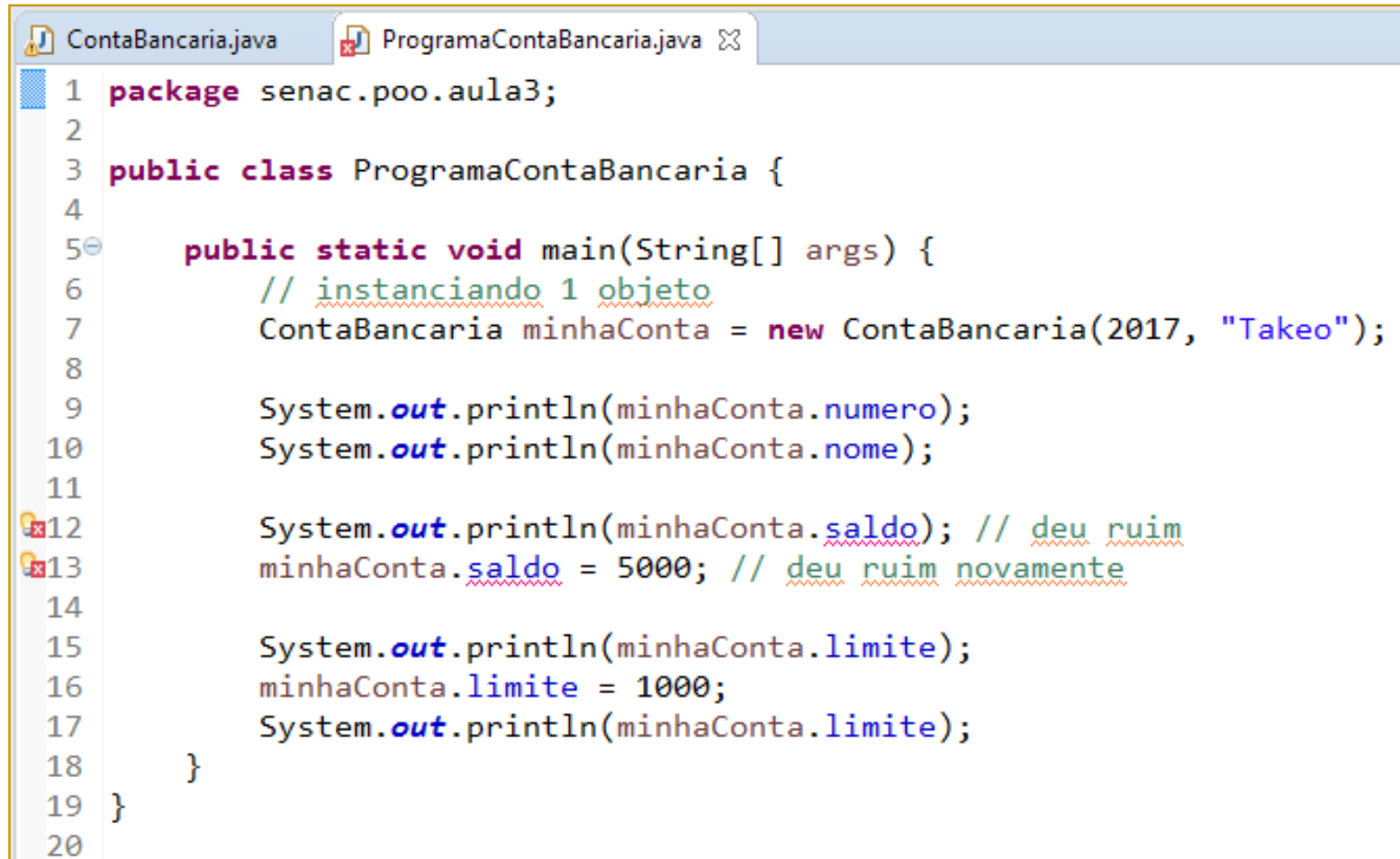
Encapsulando a classe ContaBancaria



The screenshot shows a Java IDE with two tabs: 'ContaBancaria.java' and 'ProgramaContaBancaria.java'. The 'ContaBancaria.java' tab is active, displaying the following code:

```
1 package senac.poo.aula3;
2
3 public class ContaBancaria {
4     // declaração de atributos
5     public int numero;
6     public String nome;
7     private double saldo;
8     protected double limite;
9
10    // construtor da classe
11    ContaBancaria(int numero, String nome) {
12
13        this.numero = numero;
14        this.nome = nome;
15        this.saldo = 0.0;
16        this.limite = 0;
17    }
18 }
19
```

Encapsulando a classe ContaBancaria

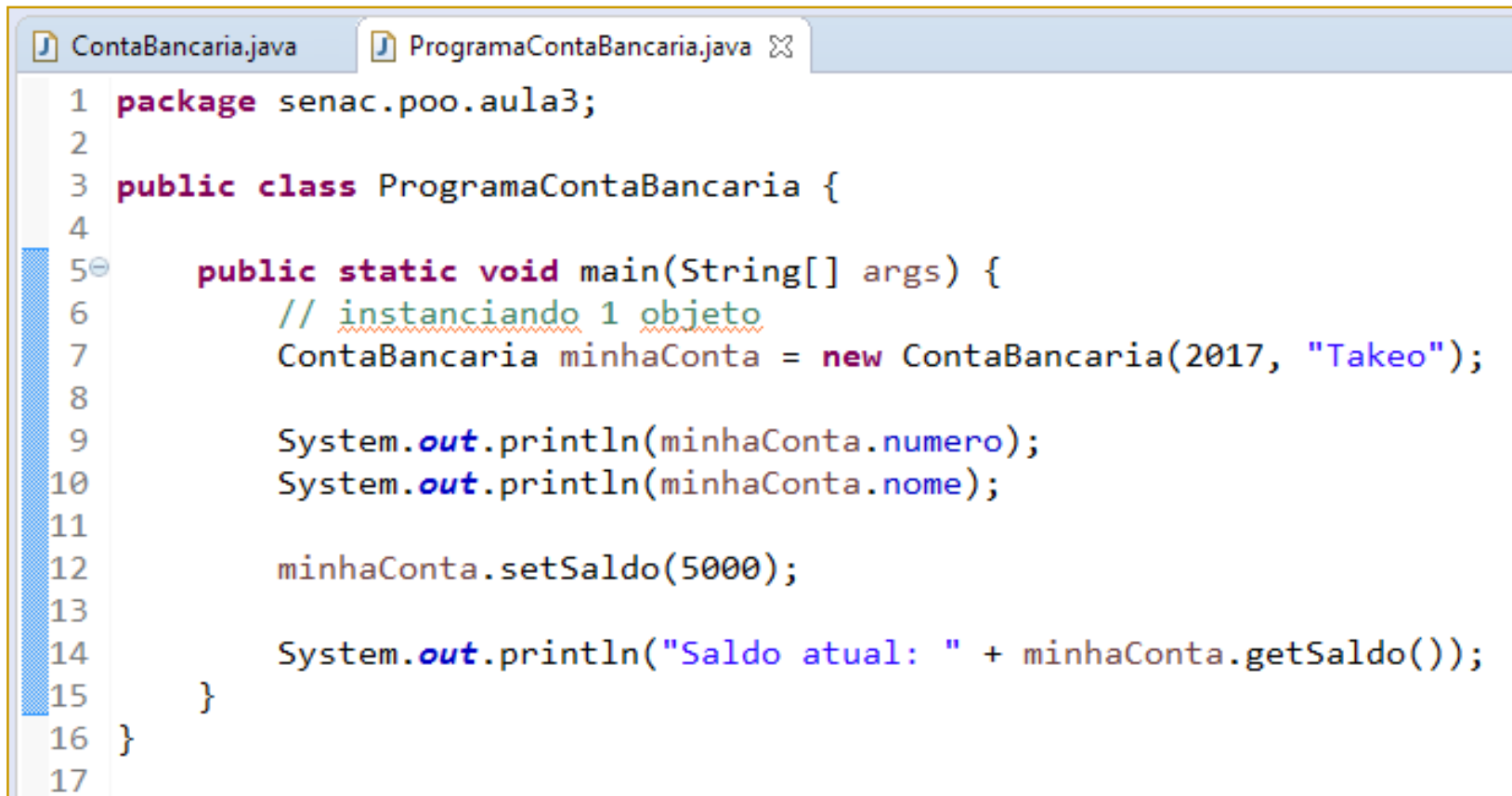


```
ContaBancaria.java  ProgramaContaBancaria.java ✕
1 package senac.poo.aula3;
2
3 public class ProgramaContaBancaria {
4
5     public static void main(String[] args) {
6         // instanciando 1 objeto
7         ContaBancaria minhaConta = new ContaBancaria(2017, "Takeo");
8
9         System.out.println(minhaConta.numero);
10        System.out.println(minhaConta.nome);
11
12        System.out.println(minhaConta.saldo); // deu ruim
13        minhaConta.saldo = 5000; // deu ruim novamente
14
15        System.out.println(minhaConta.limite);
16        minhaConta.limite = 1000;
17        System.out.println(minhaConta.limite);
18    }
19 }
20
```

Setters e Getters

```
ContaBancaria.java ProgramaContaBancaria.java
1 package senac.poo.aula3;
2
3 public class ContaBancaria {
4     // declaração de atributos
5     public int numero;
6     public String nome;
7     private double saldo;
8     protected double limite;
9
10    // construtor da classe
11    ContaBancaria(int numero, String nome) {
12
13        this.numero = numero;
14        this.nome = nome;
15        this.saldo = 0.0;
16        this.limite = 0;
17    }
18    // método setter
19    public void setSaldo(double saldo) {
20        this.saldo = saldo;
21    }
22    // método getter
23    public double getSaldo() {
24        return this.saldo;
25    }
26 }
```

Setters e Getters



```
1 package senac.poo.aula3;
2
3 public class ProgramaContaBancaria {
4
5     public static void main(String[] args) {
6         // instanciando 1 objeto
7         ContaBancaria minhaConta = new ContaBancaria(2017, "Takeo");
8
9         System.out.println(minhaConta.numero);
10        System.out.println(minhaConta.nome);
11
12        minhaConta.setSaldo(5000);
13
14        System.out.println("Saldo atual: " + minhaConta.getSaldo());
15    }
16 }
17
```

Setters e Getters

```
ContaBancaria.java ProgramaContaBancaria.java
20 // método setter
21 public void setSaldo(double saldo) {
22
23     if(pediSenha())
24         this.saldo = saldo;
25     else
26         System.out.println("Acesso negado!");
27 }
28
29 private boolean pediSenha() {
30
31     Scanner entrada = new Scanner(System.in);
32     System.out.println("Digite uma senha numérica: ");
33     int senha = entrada.nextInt();
34     if(senha == 42)
35         return true;
36     else
37         return false;
38 }
39
40 // método getter
41 public double getSaldo() {
42     return this.saldo;
43 }
44 }
45
```

Modificadores e elementos de uma classe

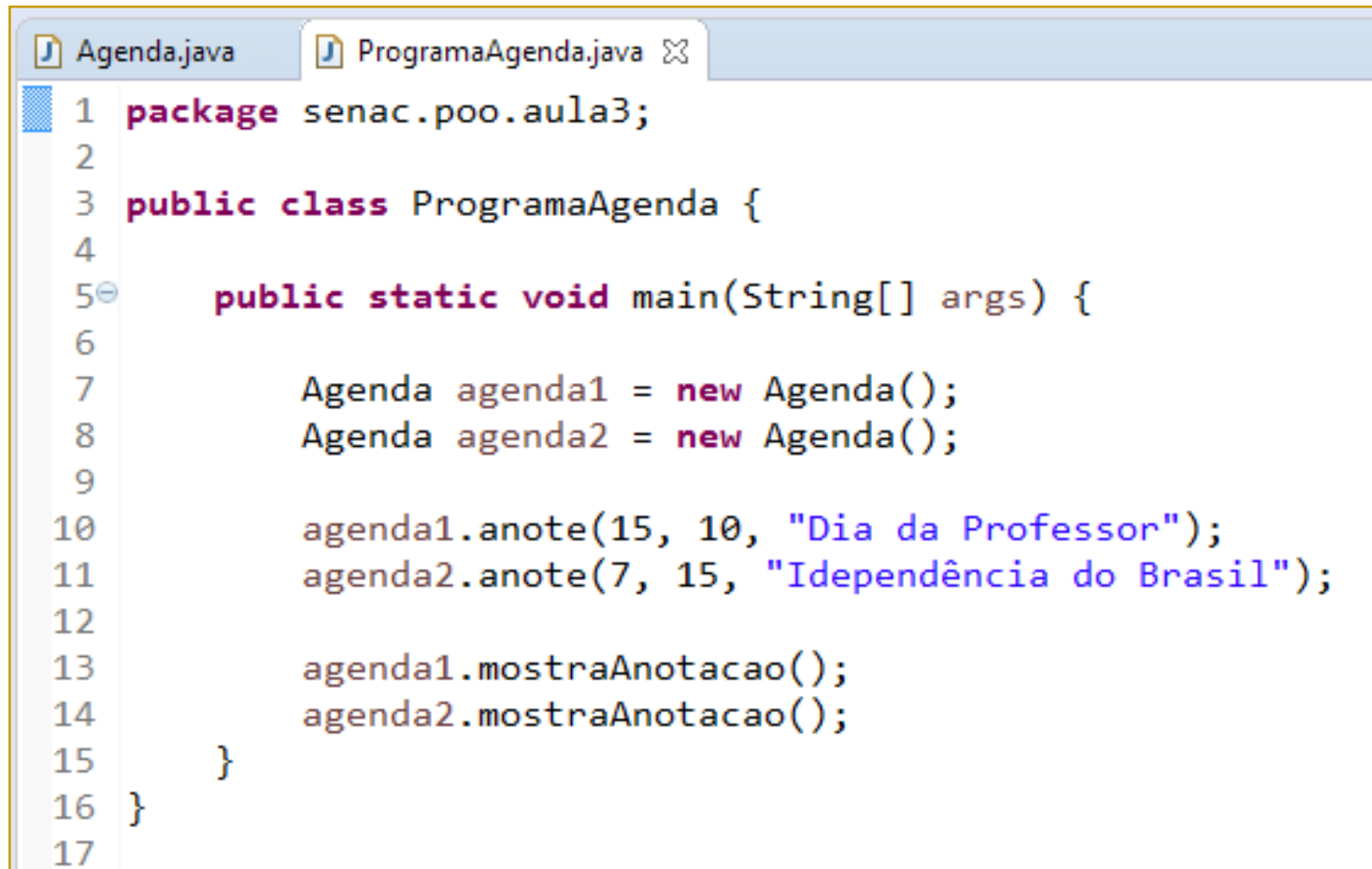
	Classe	Atributos	Construtores	Métodos
public	sim	sim	sim	sim
protected	não	sim	sim*	sim
default	sim	sim	sim	sim
private	não	sim	sim*	sim

* Pouco utilizado nesses casos.

Exemplo de uma agenda

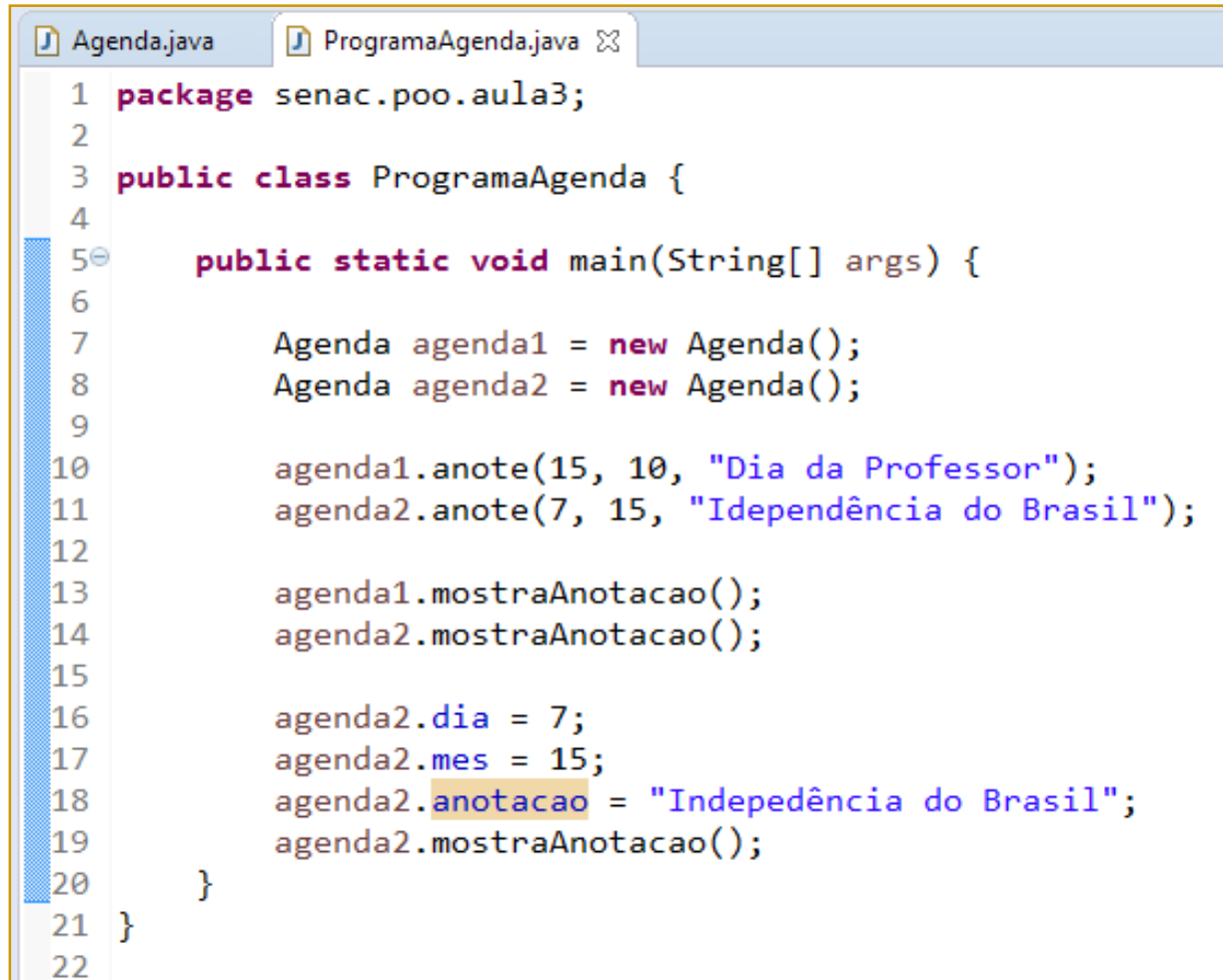
```
Agenda.java
1 package senac.poo.aula3;
2
3 public class Agenda {
4
5     int dia;
6     int mes;
7     String anotacao;
8
9     void anote(int d, int m, String nota) {
10         dia = d;
11         mes = m;
12         anotacao = nota;
13         validaData();
14     }
15     void validaData() {
16         if((dia < 1) || (dia > 31) || (mes < 1) || (mes > 12)) {
17             dia = 0;
18             mes = 0;
19             anotacao = "data inválida";
20         }
21     }
22     void mostraAnotacao() {
23         System.out.println(dia + "/" + mes + " : " + anotacao);
24     }
25 }
26
```

Agenda sem encapsulamento



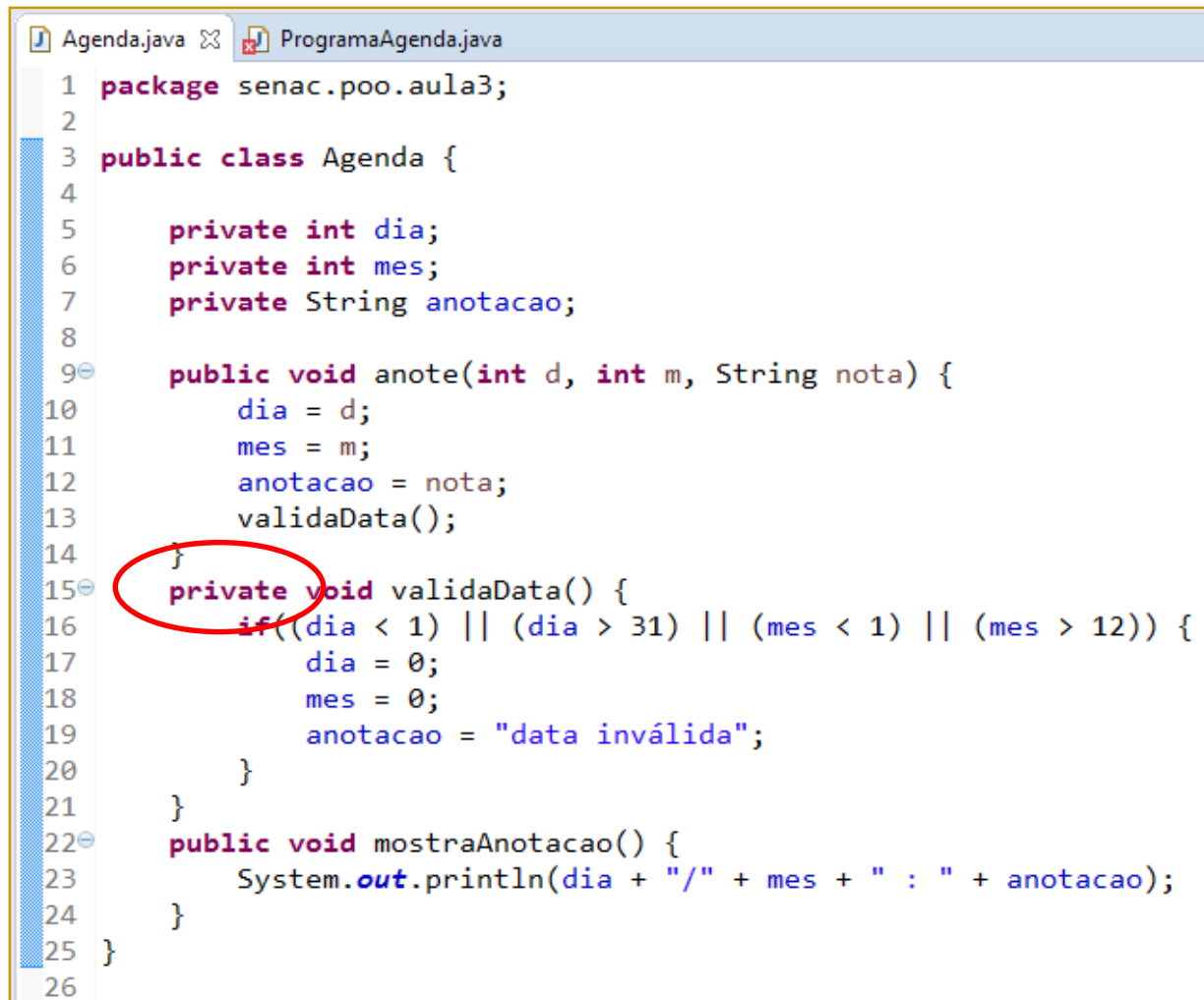
```
1 package senac.poo.aula3;
2
3 public class ProgramaAgenda {
4
5     public static void main(String[] args) {
6
7         Agenda agenda1 = new Agenda();
8         Agenda agenda2 = new Agenda();
9
10        agenda1.anote(15, 10, "Dia da Professor");
11        agenda2.anote(7, 15, "Independência do Brasil");
12
13        agenda1.mostraAnotacao();
14        agenda2.mostraAnotacao();
15    }
16 }
17
```


Injetando uma data inválida na agenda



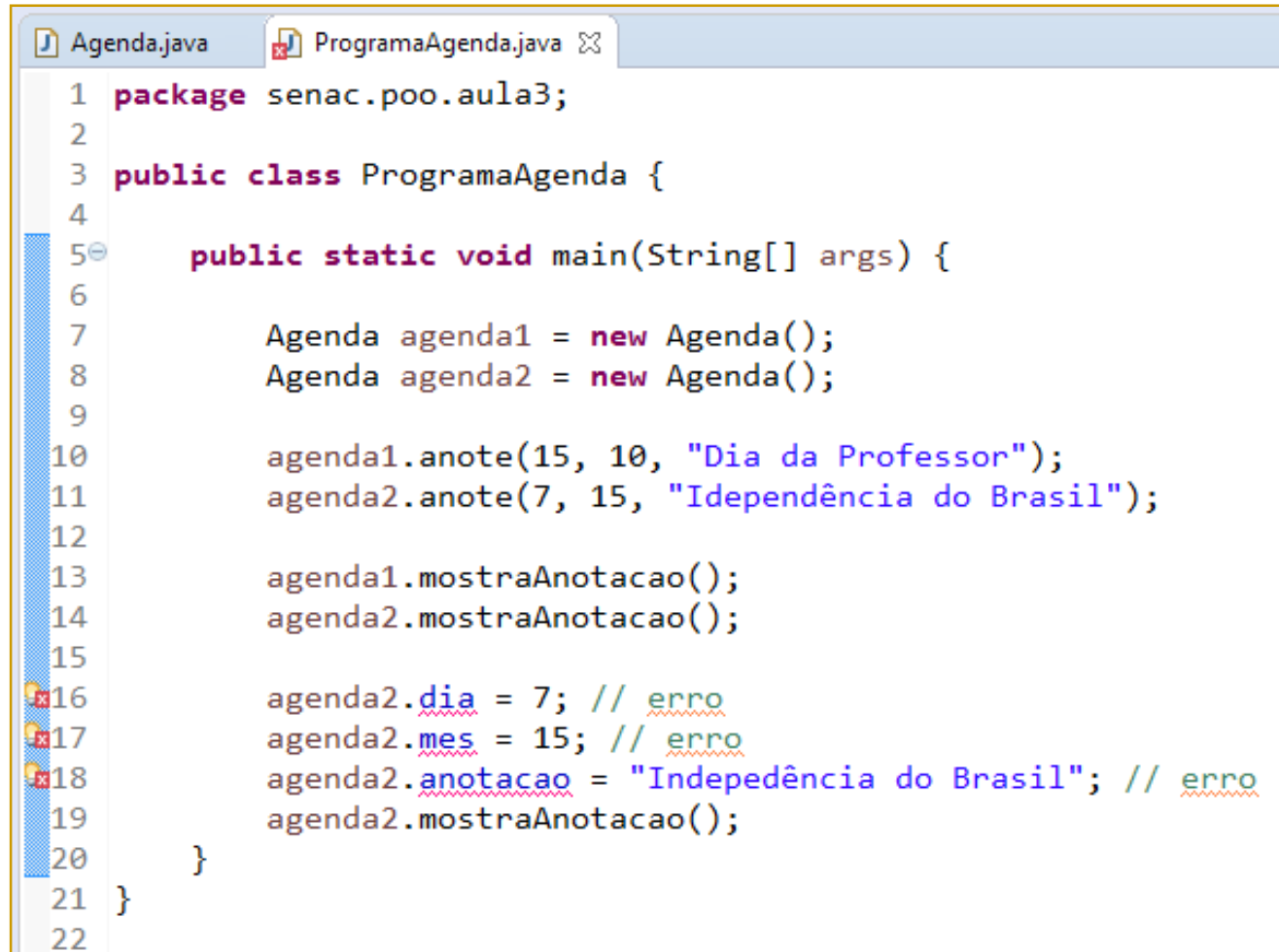
```
1 package senac.poo.aula3;
2
3 public class ProgramaAgenda {
4
5     public static void main(String[] args) {
6
7         Agenda agenda1 = new Agenda();
8         Agenda agenda2 = new Agenda();
9
10        agenda1.anote(15, 10, "Dia da Professor");
11        agenda2.anote(7, 15, "Independência do Brasil");
12
13        agenda1.mostraAnotacao();
14        agenda2.mostraAnotacao();
15
16        agenda2.dia = 7;
17        agenda2.mes = 15;
18        agenda2.anotacao = "Independência do Brasil";
19        agenda2.mostraAnotacao();
20    }
21 }
22
```

Solução do problema com encapsulamento



```
1 package senac.poo.aula3;
2
3 public class Agenda {
4
5     private int dia;
6     private int mes;
7     private String anotacao;
8
9     public void anote(int d, int m, String nota) {
10         dia = d;
11         mes = m;
12         anotacao = nota;
13         validaData();
14     }
15     private void validaData() {
16         if((dia < 1) || (dia > 31) || (mes < 1) || (mes > 12)) {
17             dia = 0;
18             mes = 0;
19             anotacao = "data inválida";
20         }
21     }
22     public void mostraAnotacao() {
23         System.out.println(dia + "/" + mes + " : " + anotacao);
24     }
25 }
26
```

Solução do problema com encapsulamento



```
1 package senac.poo.aula3;
2
3 public class ProgramaAgenda {
4
5     public static void main(String[] args) {
6
7         Agenda agenda1 = new Agenda();
8         Agenda agenda2 = new Agenda();
9
10        agenda1.anote(15, 10, "Dia da Professor");
11        agenda2.anote(7, 15, "Independência do Brasil");
12
13        agenda1.mostraAnotacao();
14        agenda2.mostraAnotacao();
15
16        agenda2.dia = 7; // erro
17        agenda2.mes = 15; // erro
18        agenda2.anotacao = "Independência do Brasil"; // erro
19        agenda2.mostraAnotacao();
20    }
21 }
22
```

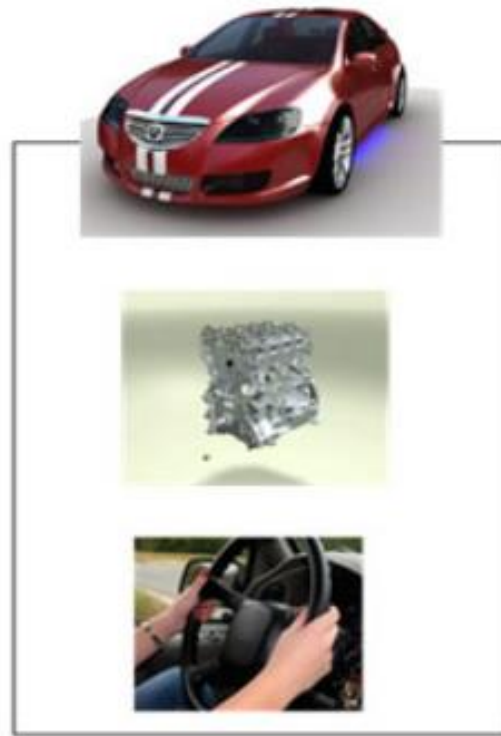
Composição (ou Agregação)

- Composição ou Agregação é um mecanismo de reaproveitamento (reutilização) de classes utilizado pela Orientação a Objetos (OO) para aumentar a produtividade e a qualidade no desenvolvimento de software

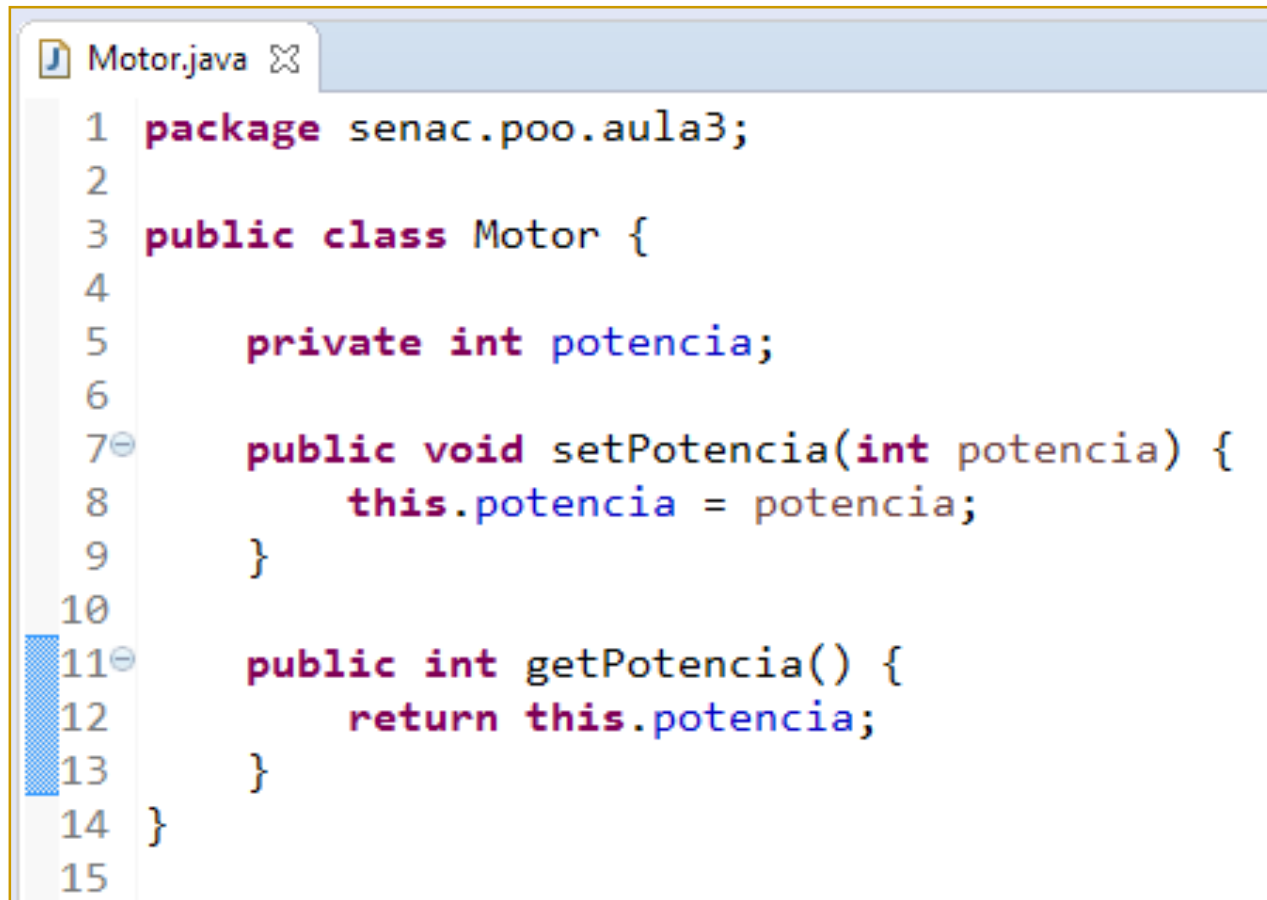


Exemplo de composição na prática

Composição da classe Automovel: Motor e Direcao

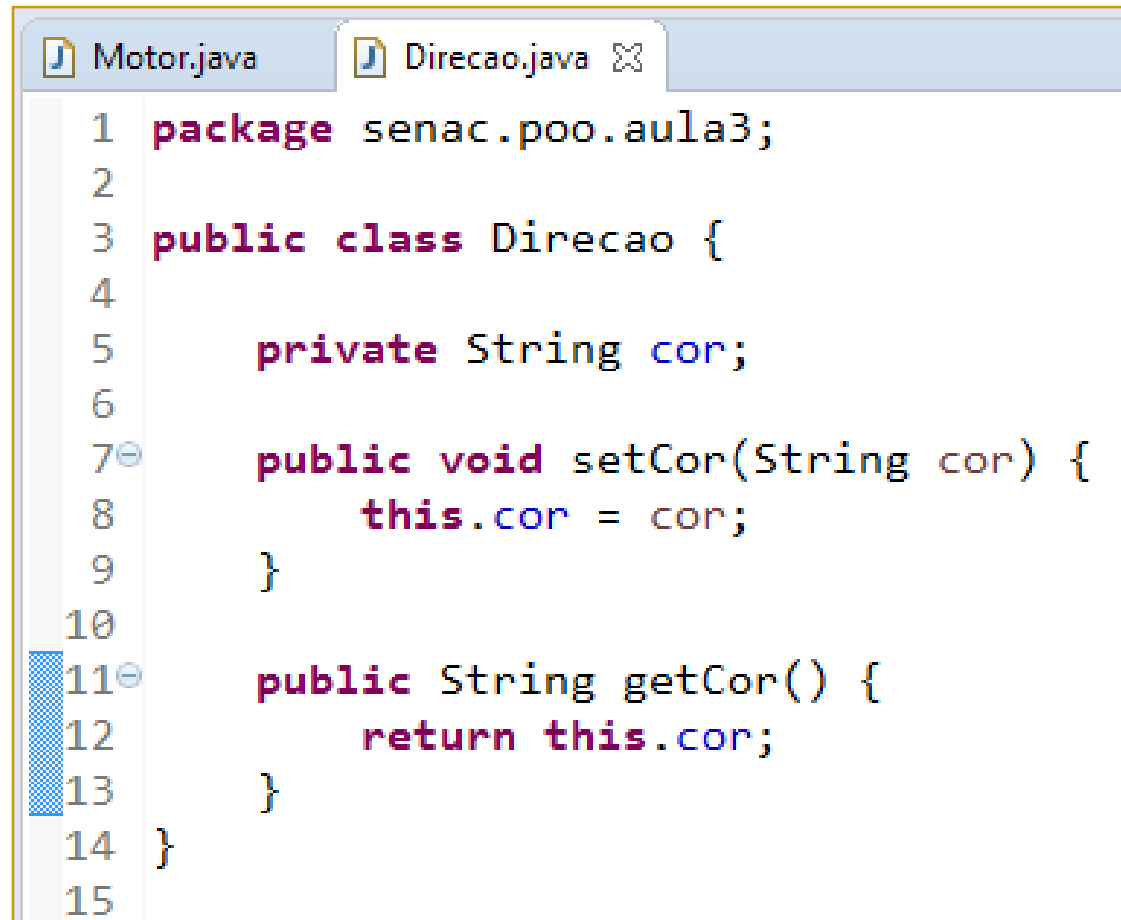


Exemplo de composição na prática



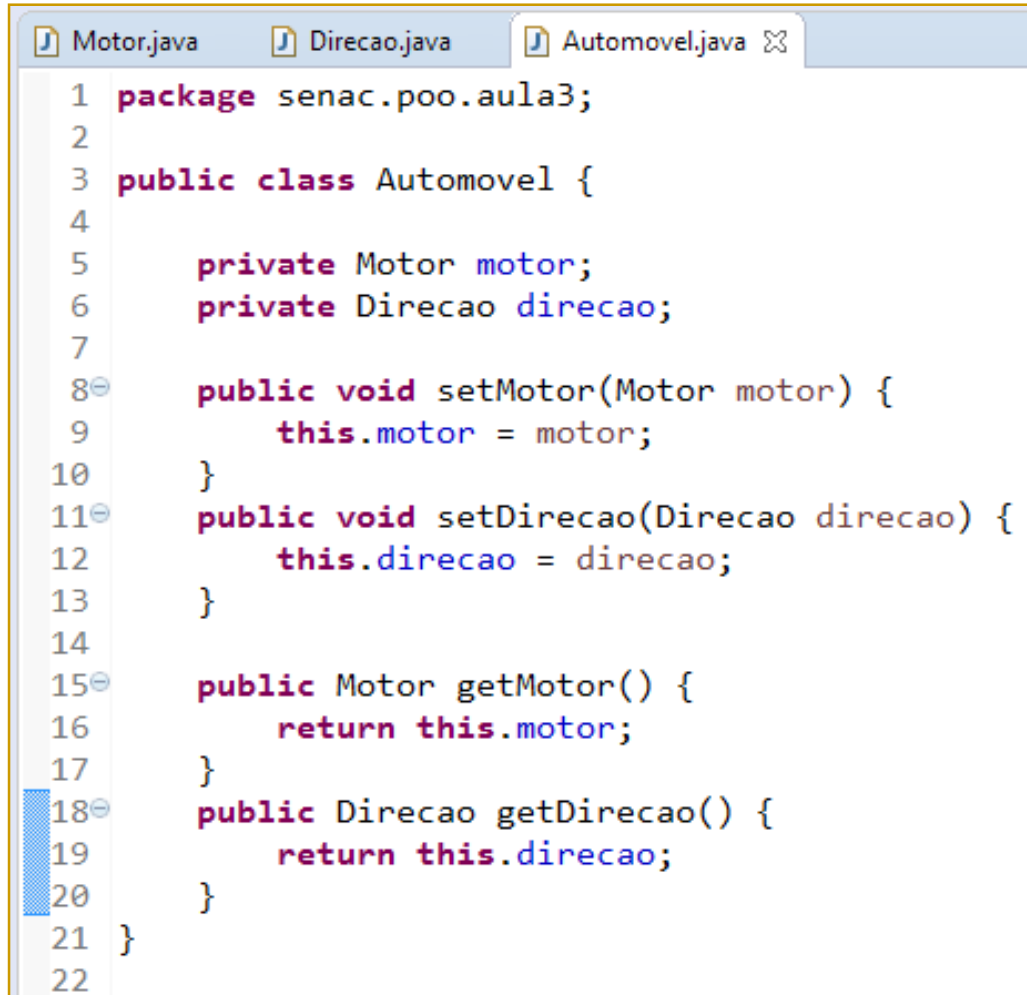
```
Motor.java ✕
1 package senac.poo.aula3;
2
3 public class Motor {
4
5     private int potencia;
6
7     public void setPotencia(int potencia) {
8         this.potencia = potencia;
9     }
10
11     public int getPotencia() {
12         return this.potencia;
13     }
14 }
15
```

Exemplo de composição na prática



```
Motor.java Direcao.java ✕
1 package senac.poo.aula3;
2
3 public class Direcao {
4
5     private String cor;
6
7     public void setCor(String cor) {
8         this.cor = cor;
9     }
10
11     public String getCor() {
12         return this.cor;
13     }
14 }
15
```

Exemplo de composição na prática

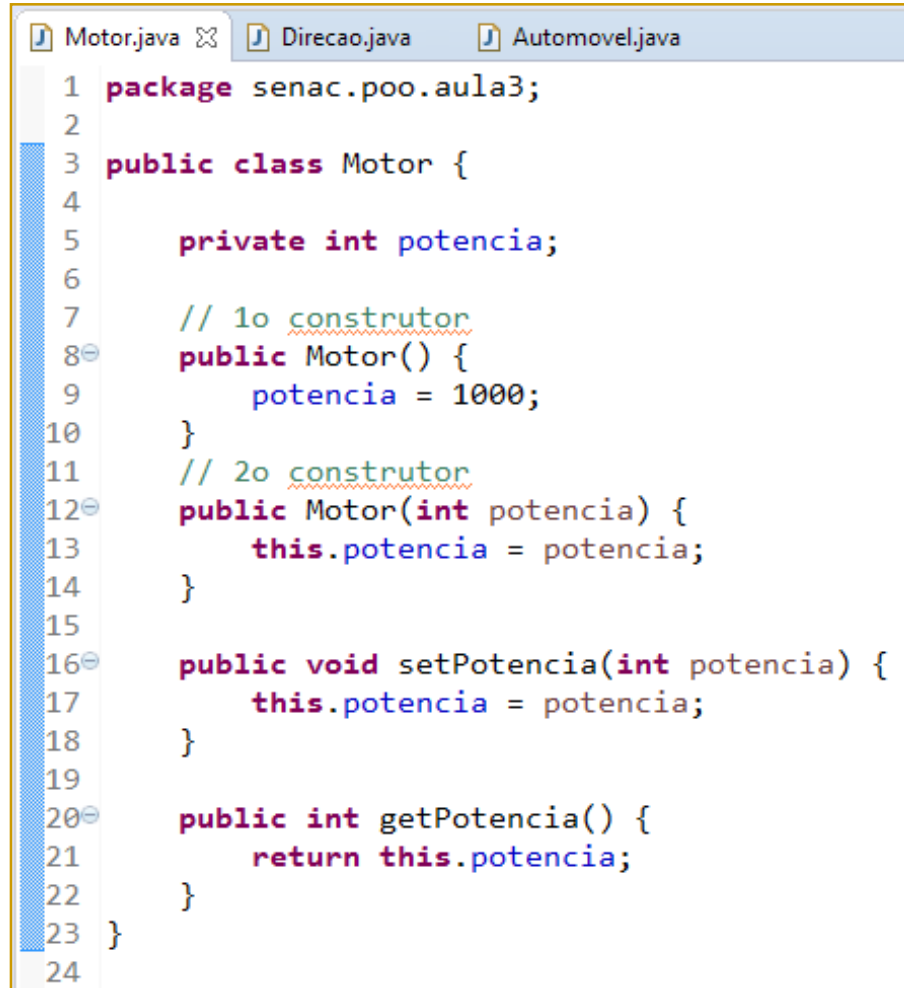


```
1 package senac.poo.aula3;
2
3 public class Automovel {
4
5     private Motor motor;
6     private Direcao direcao;
7
8     public void setMotor(Motor motor) {
9         this.motor = motor;
10    }
11    public void setDirecao(Direcao direcao) {
12        this.direcao = direcao;
13    }
14
15    public Motor getMotor() {
16        return this.motor;
17    }
18    public Direcao getDirecao() {
19        return this.direcao;
20    }
21 }
22
```


Composição e o método construtor

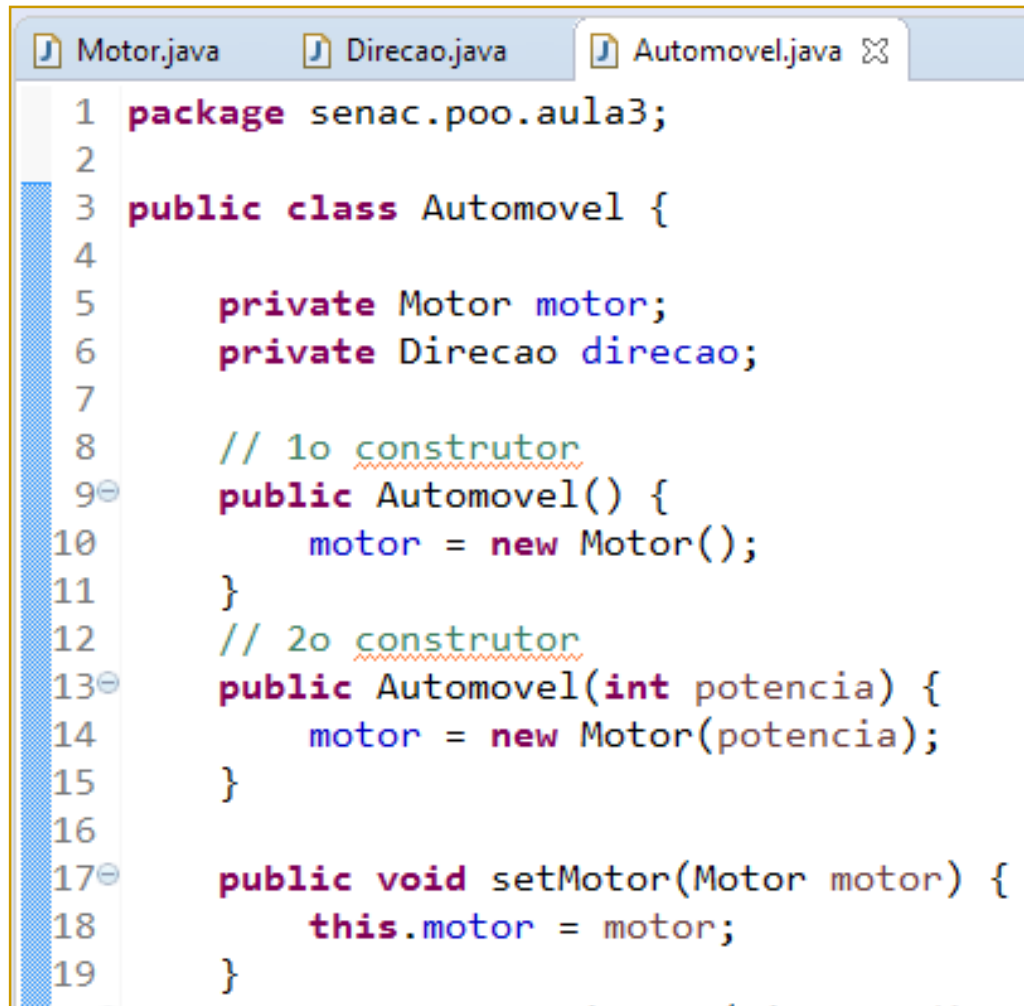
- Os métodos construtores das classes componentes (Motor e Direcao) que fazem parte da classe composta (Automovel), podem ser chamados de três maneiras diferentes
 - **CASO 1:** chamadas nos construtores da classe que é composta
 - **CASO 2:** chamadas em qualquer método da classe que é composta
 - **CASO 3:** chamadas fora da classe que é composta

Alteração da classe Motor



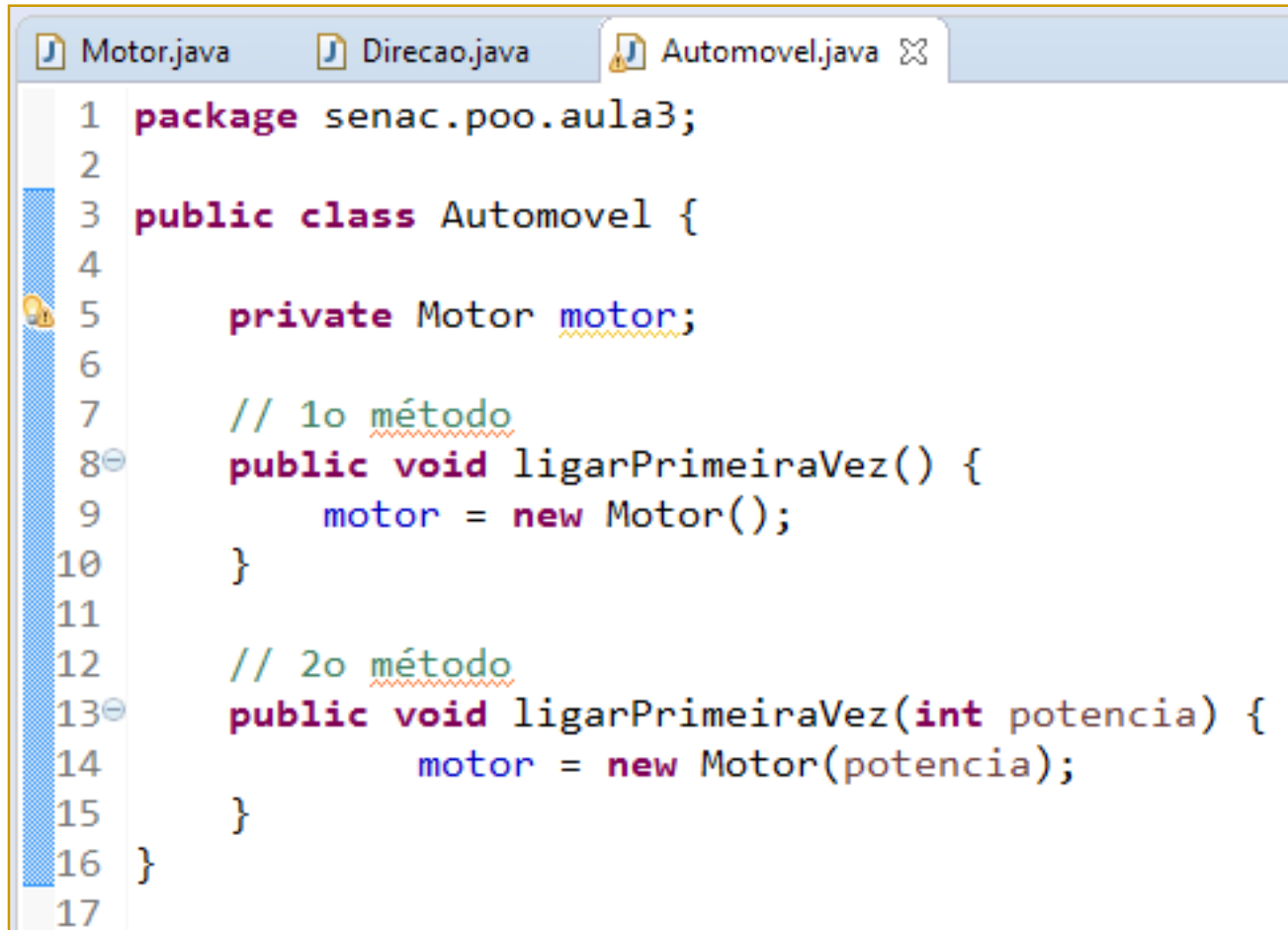
```
1 package senac.poo.aula3;
2
3 public class Motor {
4
5     private int potencia;
6
7     // 1o construtor
8     public Motor() {
9         potencia = 1000;
10    }
11    // 2o construtor
12    public Motor(int potencia) {
13        this.potencia = potencia;
14    }
15
16    public void setPotencia(int potencia) {
17        this.potencia = potencia;
18    }
19
20    public int getPotencia() {
21        return this.potencia;
22    }
23 }
24
```

Caso 1 – Classe Automovel com 2 métodos construtores



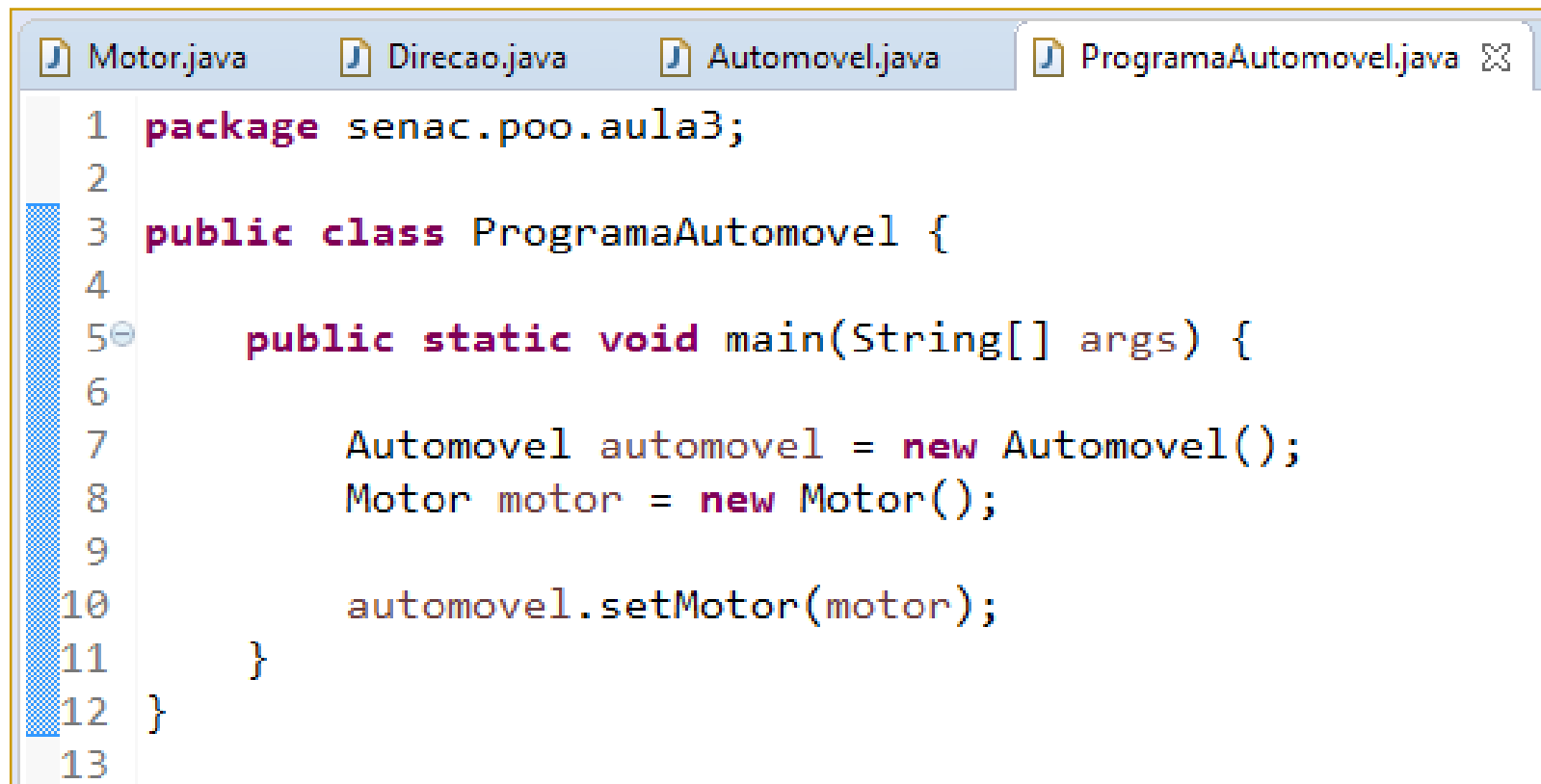
```
Motor.java  Direcao.java  Automovel.java ✖
1 package senac.poo.aula3;
2
3 public class Automovel {
4
5     private Motor motor;
6     private Direcao direcao;
7
8     // 1o construtor
9     public Automovel() {
10         motor = new Motor();
11     }
12     // 2o construtor
13     public Automovel(int potencia) {
14         motor = new Motor(potencia);
15     }
16
17     public void setMotor(Motor motor) {
18         this.motor = motor;
19     }
```

Caso 2 – Chamada em qualquer método da classe que é composta



```
1 package senac.poo.aula3;
2
3 public class Automovel {
4
5     private Motor motor;
6
7     // 1o método
8     public void ligarPrimeiraVez() {
9         motor = new Motor();
10    }
11
12    // 2o método
13    public void ligarPrimeiraVez(int potencia) {
14        motor = new Motor(potencia);
15    }
16 }
17
```

Caso 3 – Chamadas fora da classe que é composta



```
Motor.java  Direcao.java  Automovel.java  ProgramaAutomovel.java ✕
1  package senac.poo.aula3;
2
3  public class ProgramaAutomovel {
4
5      public static void main(String[] args) {
6
7          Automovel automovel = new Automovel();
8          Motor motor = new Motor();
9
10         automovel.setMotor(motor);
11     }
12 }
13
```