

PROGRAMAÇÃO ORIENTADA A OBJETOS

Java

1995



ORIENTADA
A OBJETO

ROBUSTA

PORTATIL

OPERAÇÃO
EM REDE

SEGURANÇA

PROGRAMAÇÃO ORIENTADA A OBJETOS

Java

- O Java é uma *linguagem de programação de propósito geral, concorrente, baseada em classes e orientada a objetos.* Projetada para ser simples o bastante para que a maioria dos programadores se torne fluente na linguagem. Java tem relação com C e C++, porém é organizada de forma diferente, com vários aspectos de C e C++ omitidos e algumas ideias de outras linguagens incluídas.

PROGRAMAÇÃO ORIENTADA A OBJETOS

Java

1991



PROGRAMAÇÃO ORIENTADA A OBJETOS

Java



OAK



GREEN
OS

JAMES
GOSLING

PROGRAMAÇÃO ORIENTADA A OBJETOS

Java

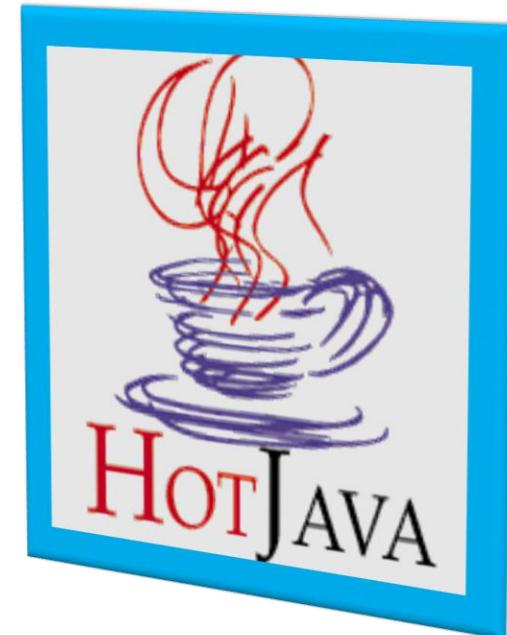


PROGRAMAÇÃO ORIENTADA A OBJETOS

Java



OAK



PROGRAMAÇÃO ORIENTADA A OBJETOS

Java



IBM



OS/2 *Warp*



PROGRAMAÇÃO ORIENTADA A OBJETOS
Java

JDK (JAVA DEVELOPMENT KIT)

JVM (JAVA VIRTUAL MACHINE)

JAVA SE(JAVA STANDARD EDITION)

JAVA EE(JAVA ENTERPRISE EDITION)

PROGRAMAÇÃO ORIENTADA A OBJETOS

Java

ORACLE

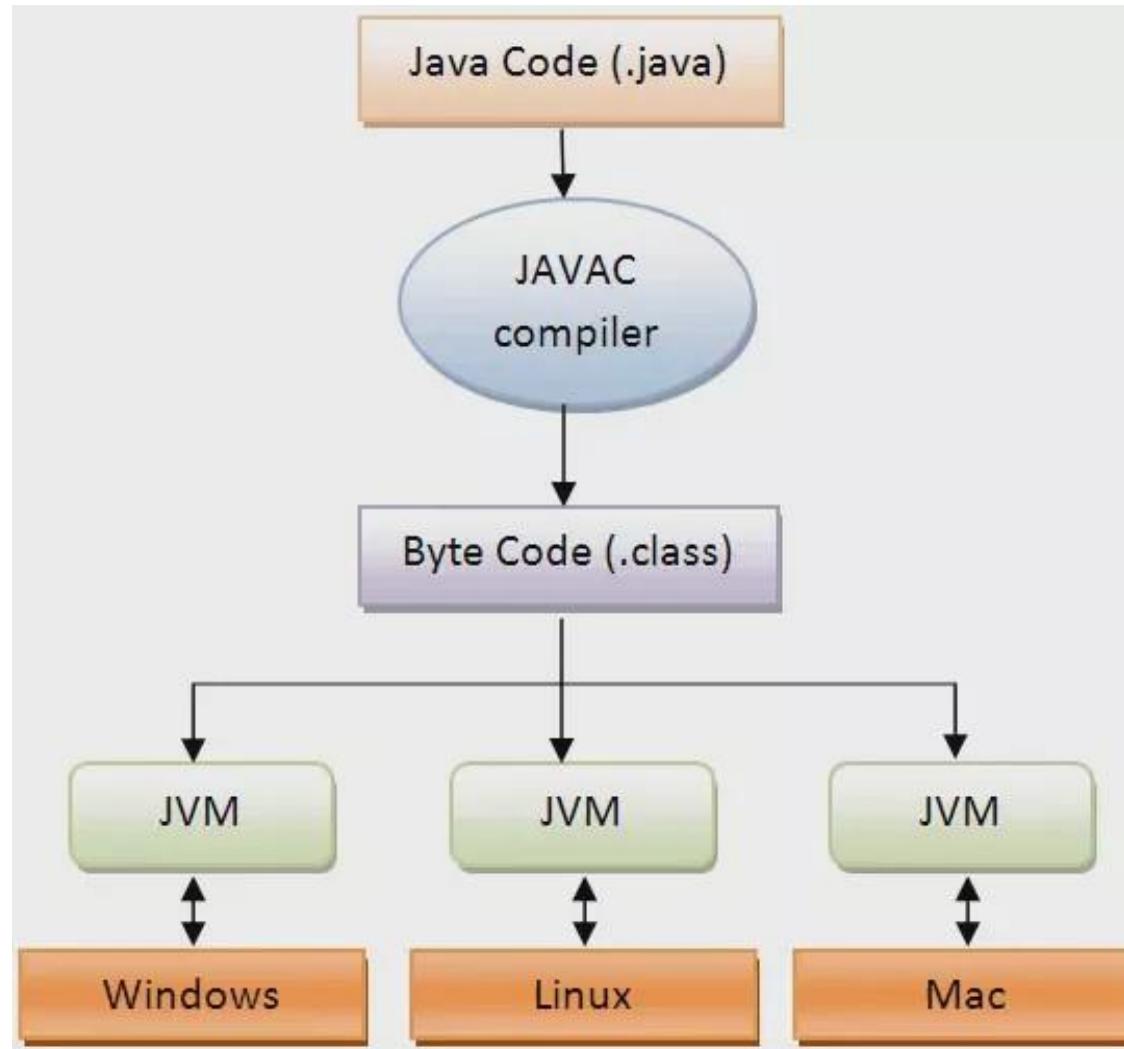


2009



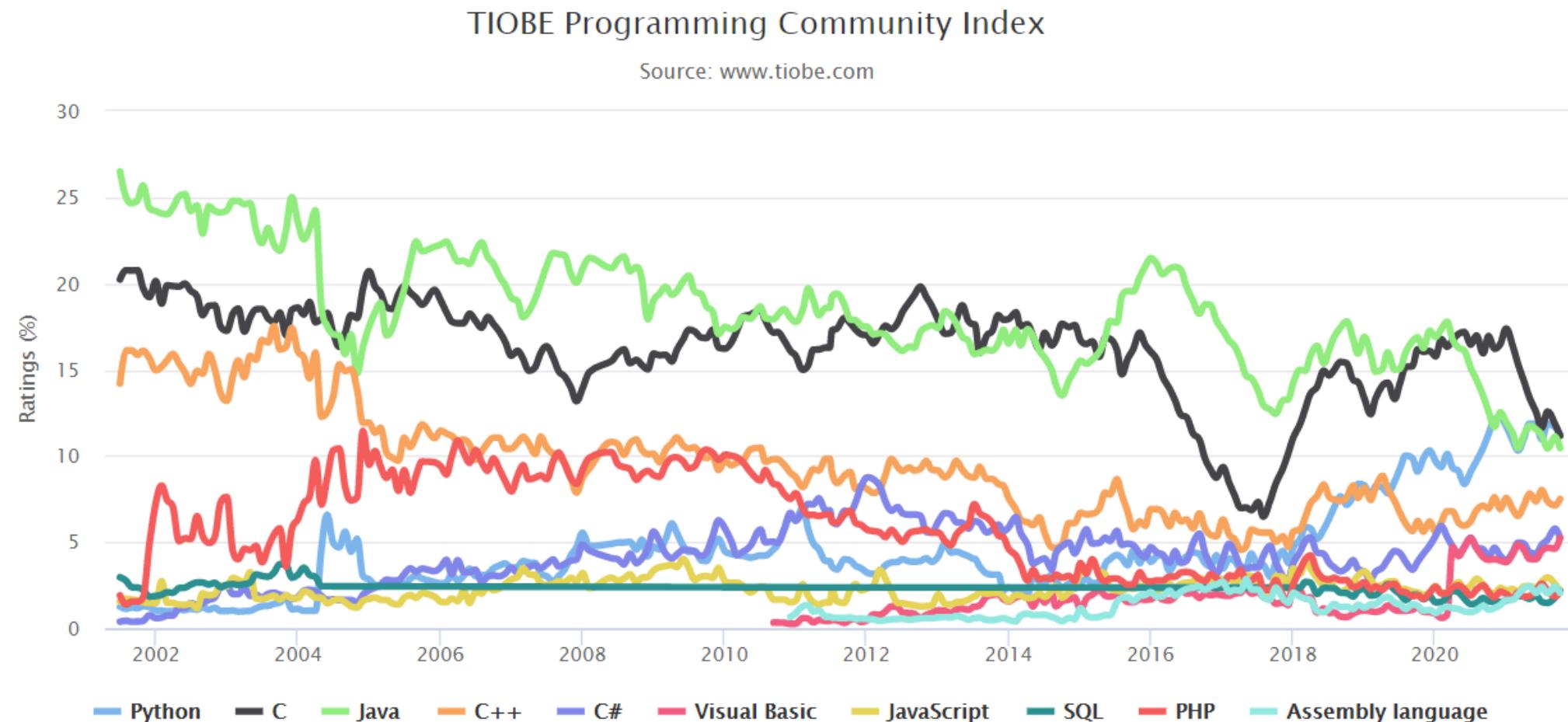
PROGRAMAÇÃO ORIENTADA A OBJETOS

Java - JVM



PROGRAMAÇÃO ORIENTADA A OBJETOS

Linguagens de Programação mais populares



PROGRAMAÇÃO ORIENTADA A OBJETOS

Linguagens de Programação mais populares

Oct 2021	Oct 2020	Change	Programming Language	Ratings
1	3	▲	 Python	11.27%
2	1	▼	 C	11.16%
3	2	▼	 Java	10.46%
4	4		 C++	7.50%
5	5		 C#	5.26%
6	6		 Visual Basic	5.24%
7	7		 JavaScript	2.19%
8	10	▲	 SQL	2.17%
9	8	▼	 PHP	2.10%
10	17	▲	 Assembly language	2.06%

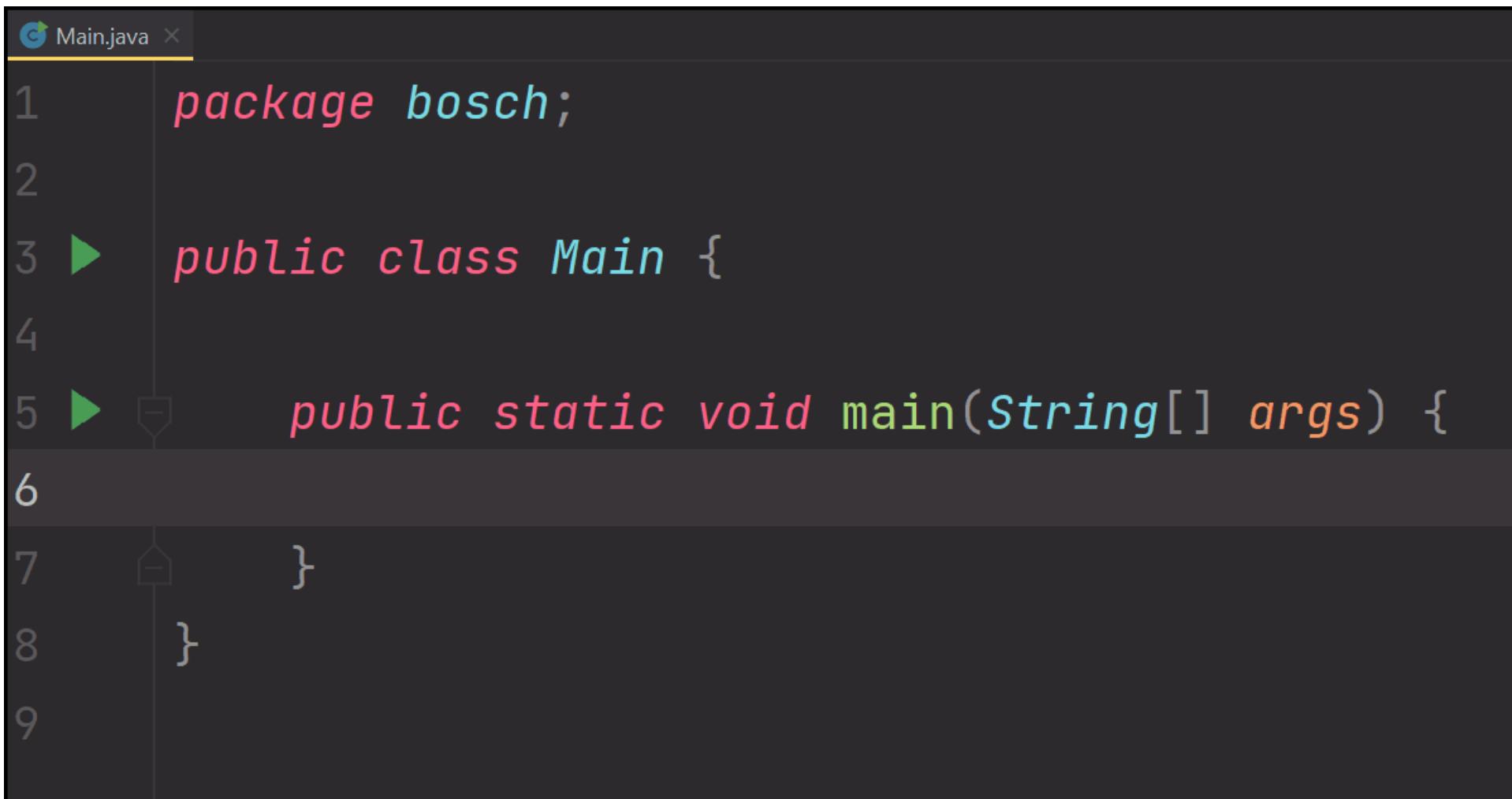
PROGRAMAÇÃO ORIENTADA A OBJETOS

Linguagens de Programação mais populares



PROGRAMAÇÃO ORIENTADA A OBJETOS

JAVA – Comentário de uma linha



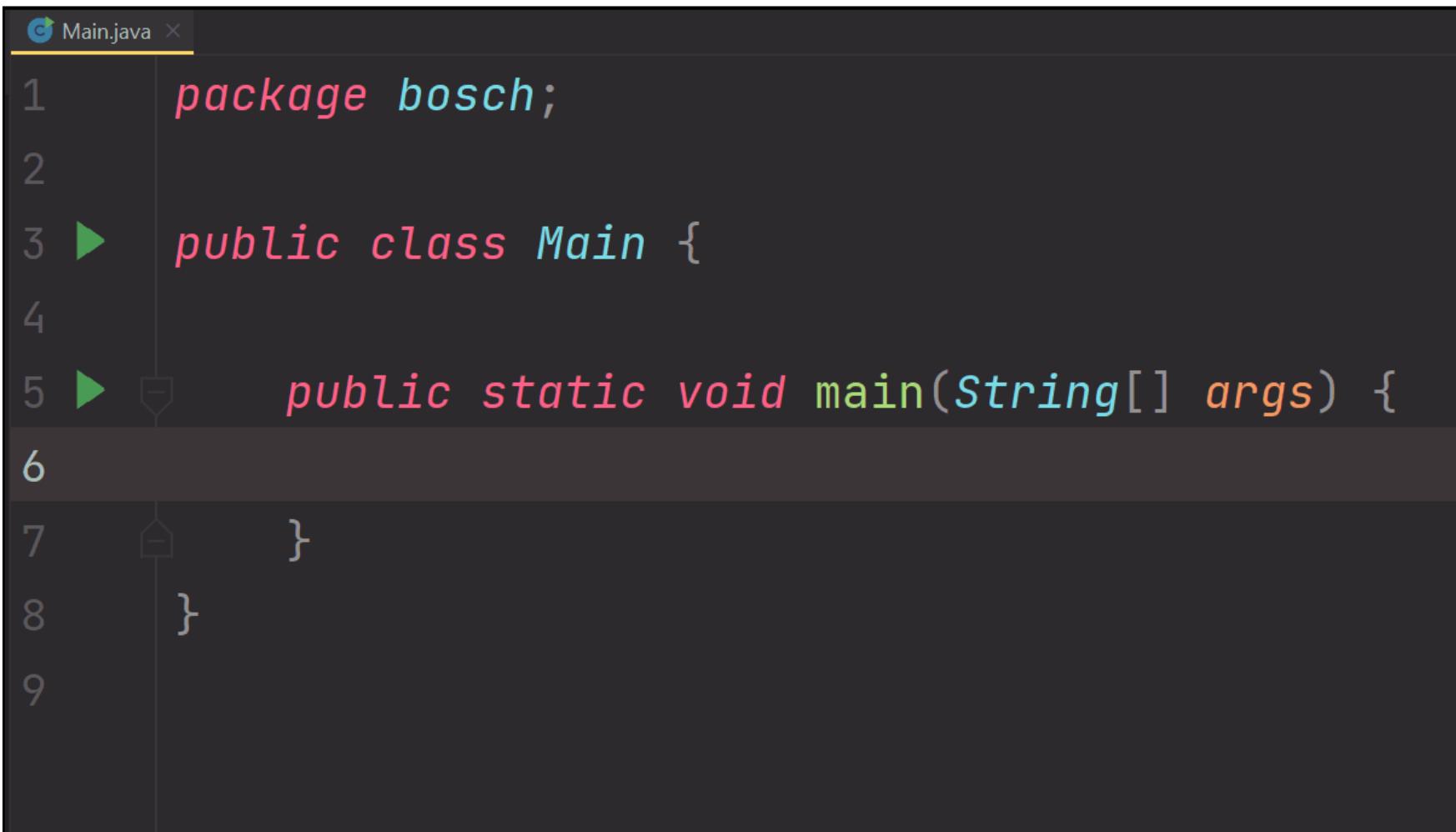
A screenshot of a Java code editor showing a single-line comment. The file is named Main.java. The code is as follows:

```
1 package bosch;
2
3 ► public class Main {
4
5 ►     public static void main(String[] args) {
6
7         }
8     }
9
```

The line 3, which starts with "► public class Main {", has a green triangle icon to its left, indicating it is a comment. The line 5, which starts with "► public static void main(String[] args) {", also has a green triangle icon to its left, indicating it is a comment.

PROGRAMAÇÃO ORIENTADA A OBJETOS

Java – Comentário de Múltiplas linhas



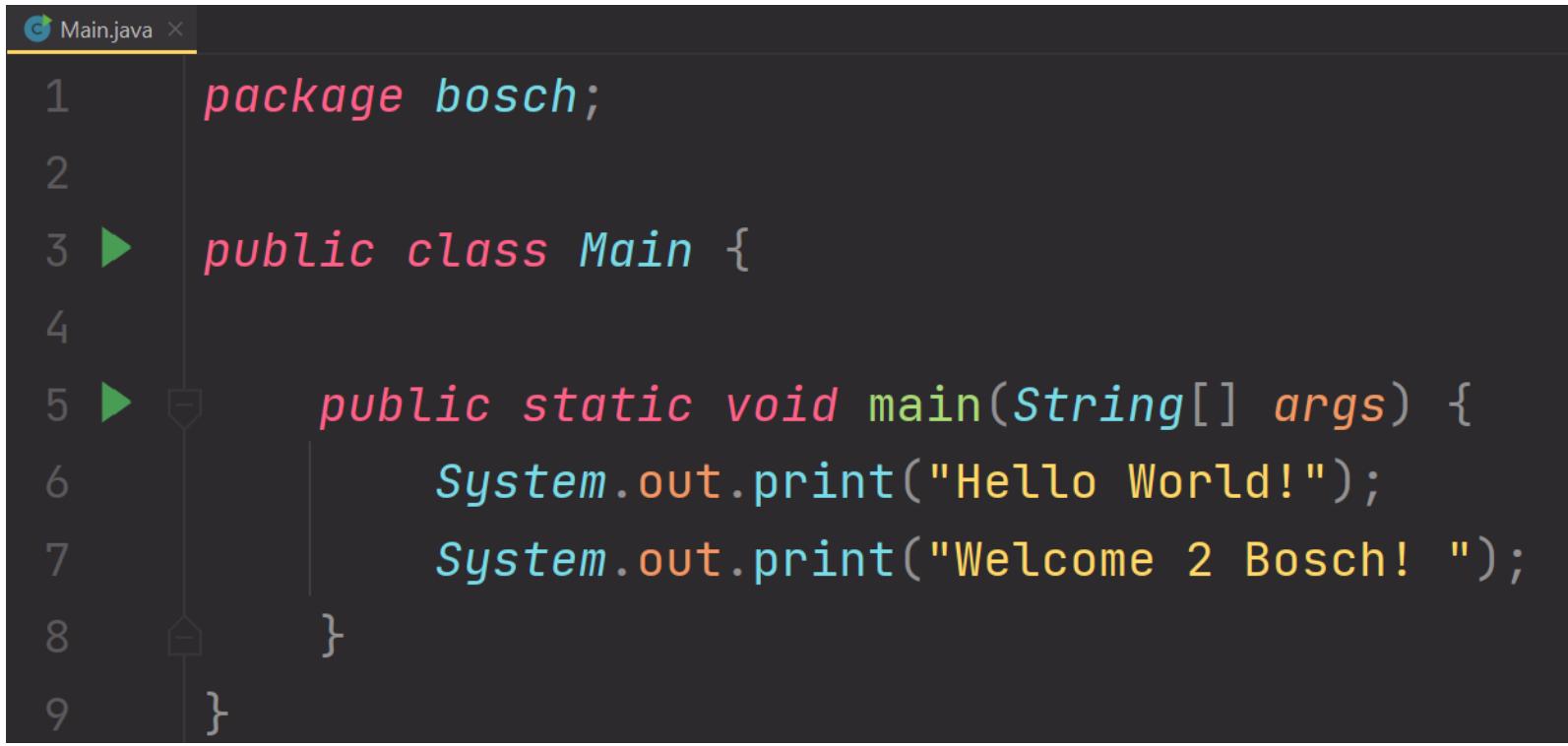
The screenshot shows a Java code editor window titled "Main.java". The code is a simple "Hello World" application:

```
1 package bosch;
2
3 > public class Main {
4
5 >     public static void main(String[] args) {
6
7         }
8     }
9 }
```

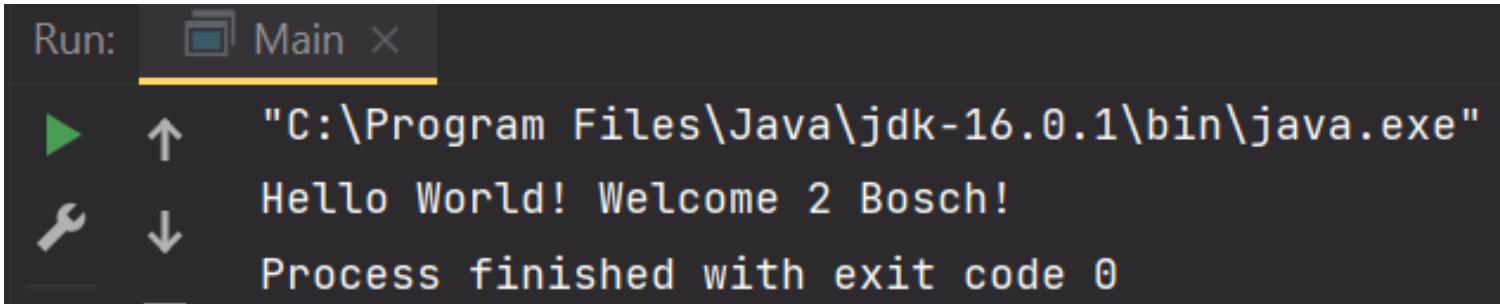
The code editor uses color coding for syntax: package, class, static, void, and String are in blue; main is in green; and args is in orange. Line numbers are on the left. The editor interface includes a tab bar at the top and a status bar at the bottom.

PROGRAMAÇÃO ORIENTADA A OBJETOS

Java – Saída não formatada



```
Main.java
1 package bosch;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         System.out.print("Hello World!");
7         System.out.print("Welcome 2 Bosch!");
8     }
9 }
```



```
Run: Main
▶ ↑ "C:\Program Files\Java\jdk-16.0.1\bin\java.exe"
▶ ↓ Hello World! Welcome 2 Bosch!
▶ ↓ Process finished with exit code 0
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Java – Saída não formatada

The image shows a Java code editor and a terminal window. The code editor displays a file named Main.java with the following content:

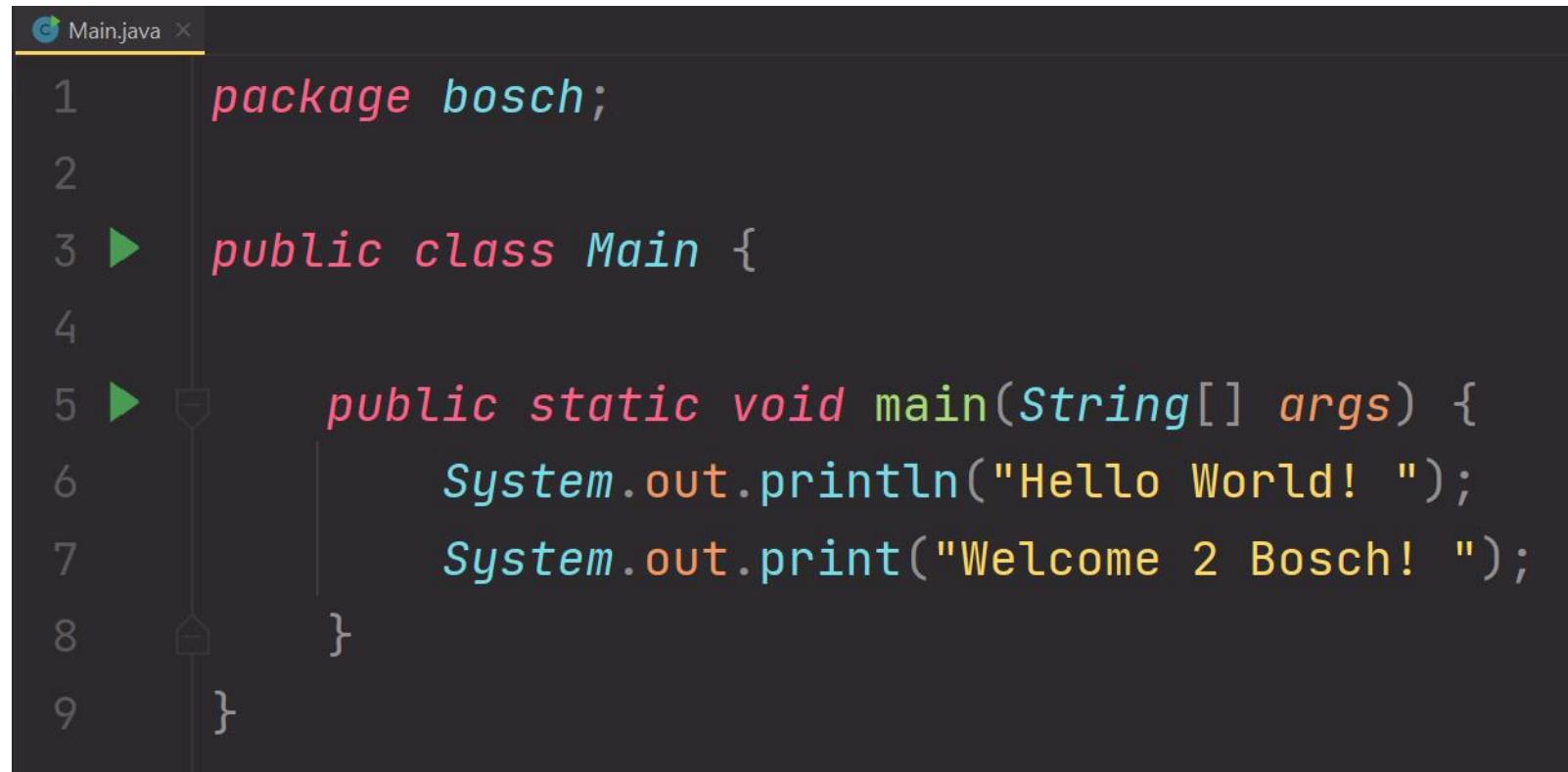
```
1 package bosch;
2
3 ► public class Main {
4
5 ►     public static void main(String[] args) {
6             System.out.print("Hello World! \n");
7             System.out.print("Welcome 2 Bosch! ");
8         }
9 }
```

The terminal window below shows the output of running the program:

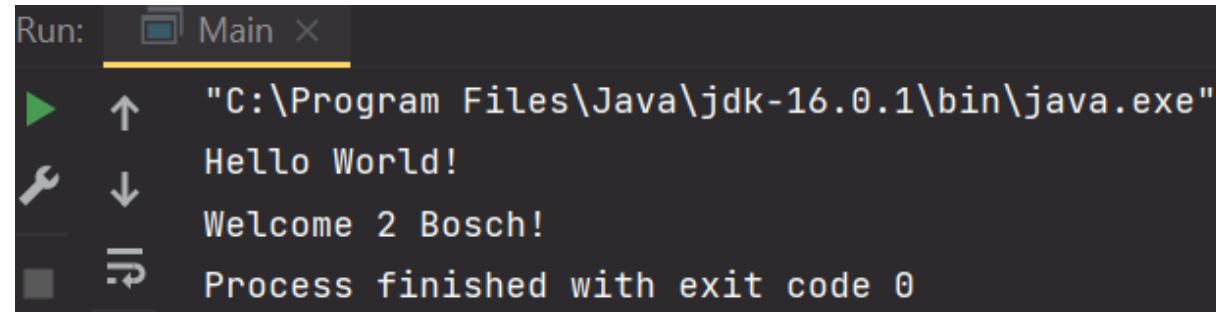
```
Run: Main
► ↑ "C:\Program Files\Java\jdk-16.0.1\bin\java.exe"
Hello World!
Welcome 2 Bosch!
Process finished with exit code 0
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Java – Saída não formatada



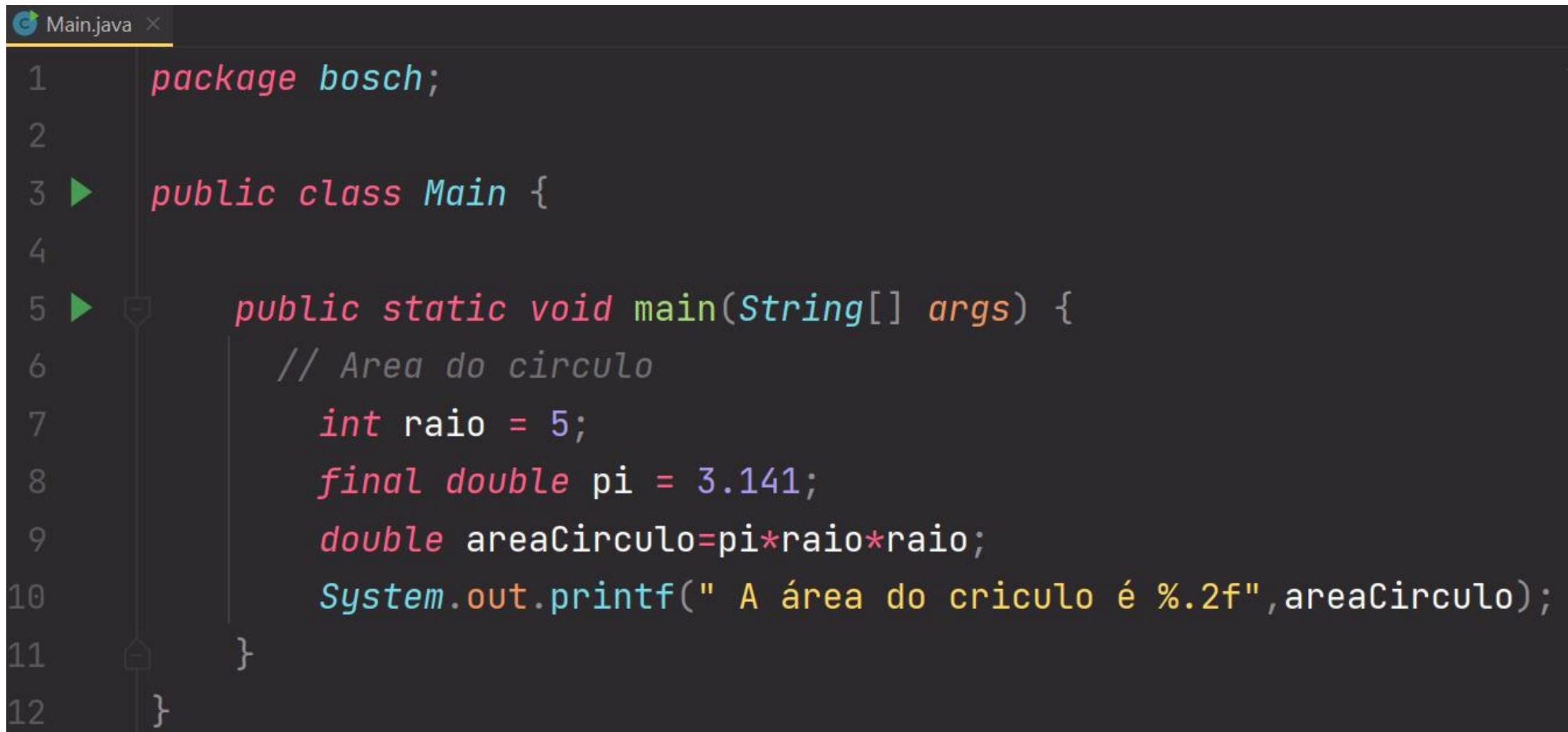
```
Main.java
1 package bosch;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         System.out.println("Hello World! ");
7         System.out.print("Welcome 2 Bosch! ");
8     }
9 }
```



```
Run: Main
1 "C:\Program Files\Java\jdk-16.0.1\bin\java.exe"
2 Hello World!
3 Welcome 2 Bosch!
4 Process finished with exit code 0
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Java – Saída formatada



The screenshot shows a Java code editor with a dark theme. The file is named "Main.java". The code calculates the area of a circle with a radius of 5, using pi as 3.141. The output is formatted to two decimal places.

```
1 package bosch;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         // Área do círculo
7         int raio = 5;
8         final double pi = 3.141;
9         double areaCirculo=pi*raio*raio;
10        System.out.printf(" A área do círculo é %.2f",areaCirculo);
11    }
12 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Java – Tipos de Dados Primitivos

Tipo Inteiro

byte

short

int

long

Tipo Real

float

double

Caractere

char

Lógico

boolean

PROGRAMAÇÃO ORIENTADA A OBJETOS

Categoria	Tipo	Bytes	bits	Faixa de Valores
Inteiro	Byte	1	8	De -128 a +127
	Short	2	16	De -32.768 a 32.767
	Int	4	32	De -2.147.483.648 a +2.147.483.647
	Long	8	64	De -9.223.372.036.854.775.808 a +9.223.372.036.854.775.807
Real	float	4	32	Valores Positivos: +1.40129846432481707e-45 a 3.402823466385288860e+38 Valores Negativos: -3.402823466385288860e+38 a +1.40129846432481707e-45
	double	8	64	Valores Positivos: +4.94065645841246544e-324 a 1.7976933486231570e+108 Valores Negativos: - 1.7976933486231570e+108 a -4.94065645841246544e-324
Caractere	char	2	16	De u\0000 a u\FFFF
Lógico	boolean	1	8	false e true

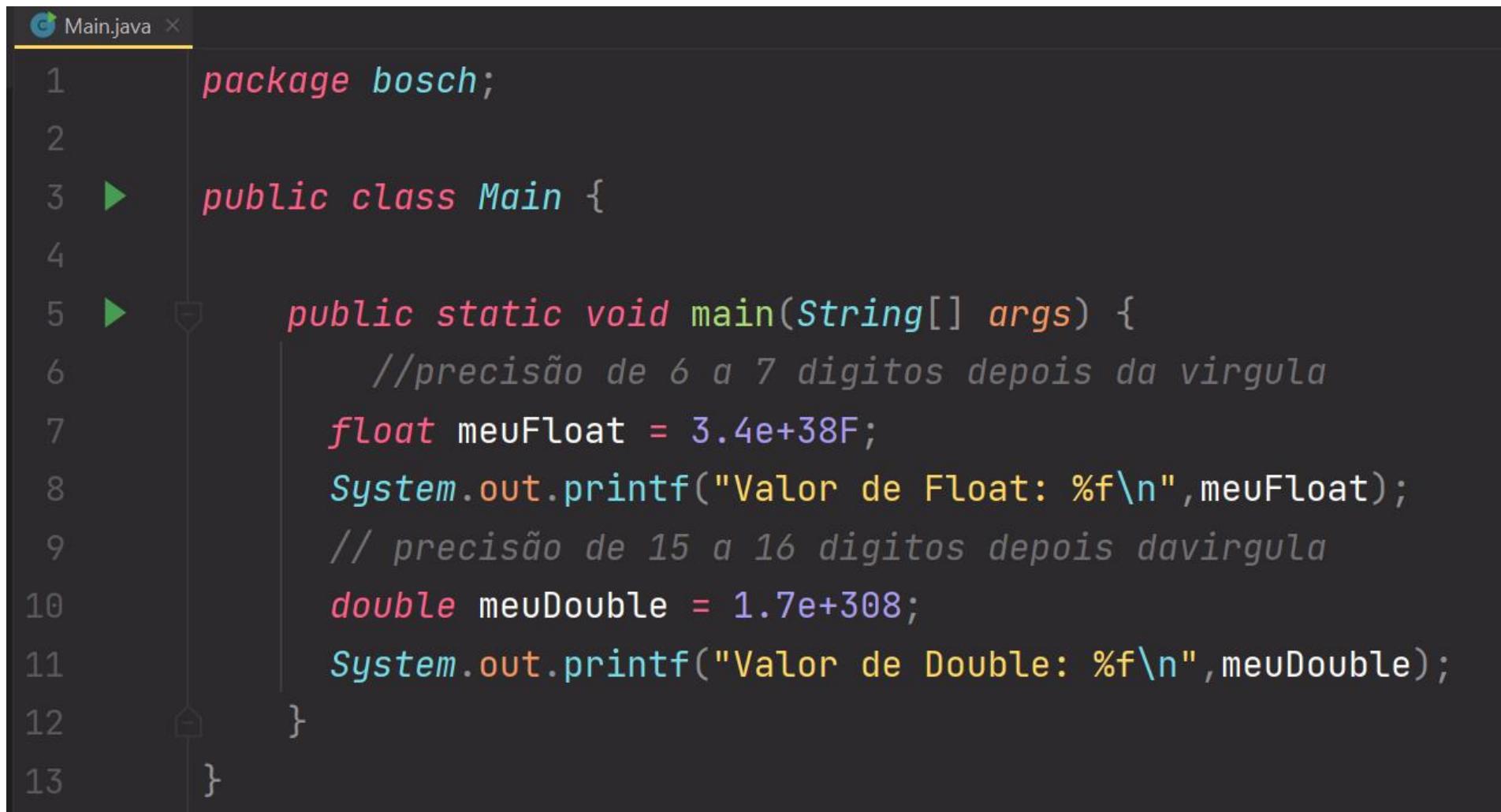
PROGRAMAÇÃO ORIENTADA A OBJETOS

Java – Tipos de Dados Primitivos

```
Main.java x
1 package bosch;
2
3 ► public class Main {
4
5 ►   ► public static void main(String[] args) {
6           // Tipos de Variaveis Inteiras
7           byte meuByte=127;
8           System.out.printf("Tamanho do Byte: %d\n",meuByte);
9           short meuShort=32767;
10          System.out.printf("Tamanho do Short: %d\n", meuShort);
11          int meuInt = 2_147_483_647;
12          System.out.printf("Tamanho do Int: %d\n", meuInt);
13          long meuLong = 9_223_372_036_854_775_807L;
14          System.out.printf("Tamanho do Long: %d\n",meuLong);
15      }
16 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Java – Tipos de Dados Primitivos

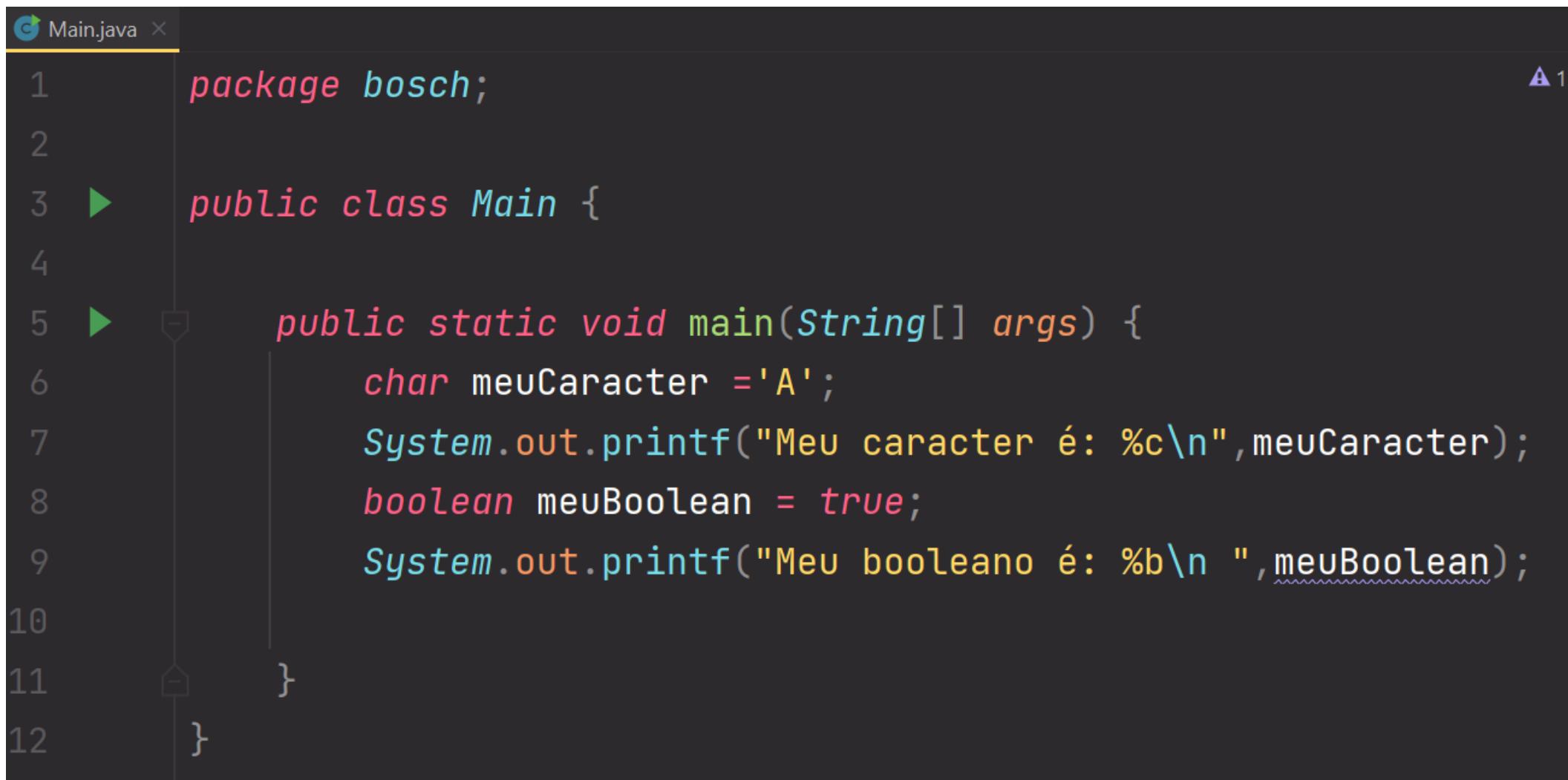


The image shows a screenshot of a Java code editor with a dark theme. The file being edited is named "Main.java". The code itself is as follows:

```
1 package bosch;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         //precisão de 6 a 7 dígitos depois da vírgula
7         float meuFloat = 3.4e+38F;
8         System.out.printf("Valor de Float: %f\n",meuFloat);
9         // precisão de 15 a 16 dígitos depois da vírgula
10        double meuDouble = 1.7e+308;
11        System.out.printf("Valor de Double: %f\n",meuDouble);
12    }
13}
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Java – Tipos de Dados Primitivos



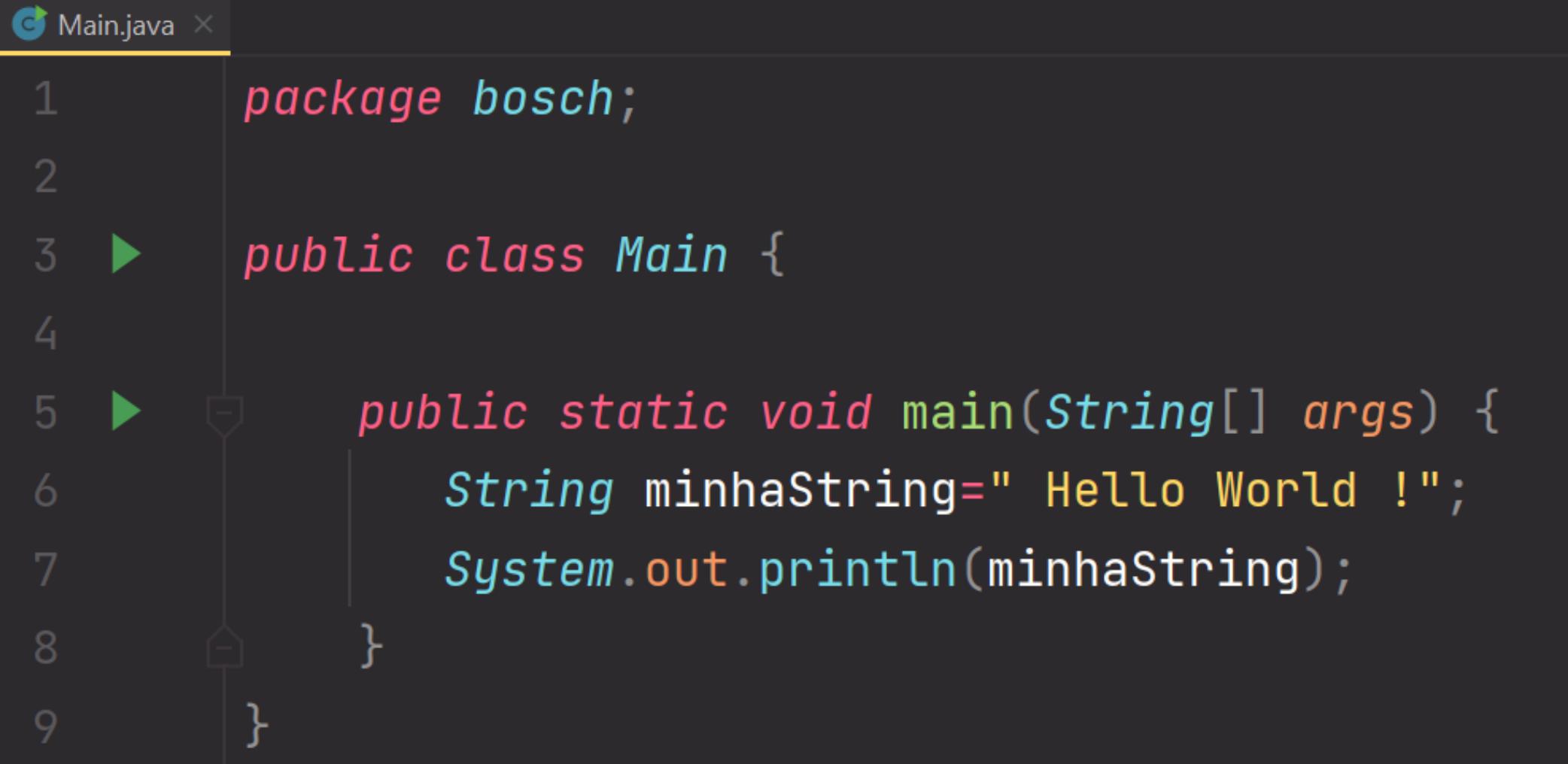
The screenshot shows a Java code editor with a dark theme. The file is named "Main.java". The code demonstrates the use of primitive data types:

```
1 package bosch;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         char meuCaracter = 'A';
7         System.out.printf("Meu caracter é: %c\n", meuCaracter);
8         boolean meuBoolean = true;
9         System.out.printf("Meu booleano é: %b\n ", meuBoolean);
10    }
11 }
12 }
```

The code defines a package named "bosch" and a class named "Main". The main method contains declarations for a character variable "meuCaracter" initialized to 'A', and a boolean variable "meuBoolean" initialized to true. It then prints these values using the `printf` method from the `System.out` class.

PROGRAMAÇÃO ORIENTADA A OBJETOS

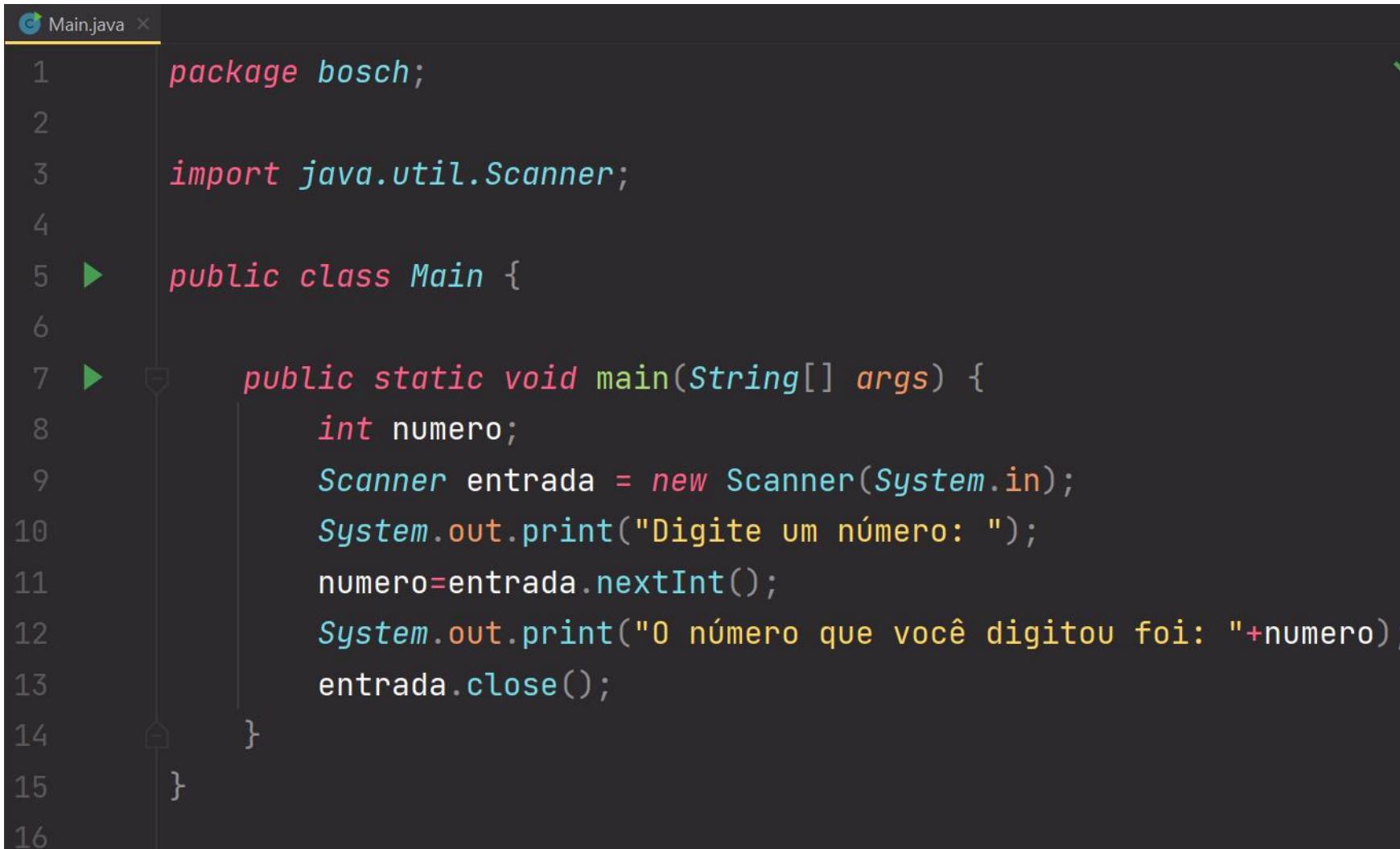
Java – String



```
Main.java
1 package bosch;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         String minhaString=" Hello World !";
7         System.out.println(minhaString);
8     }
9 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Java – Tipos de Dados Primitivos

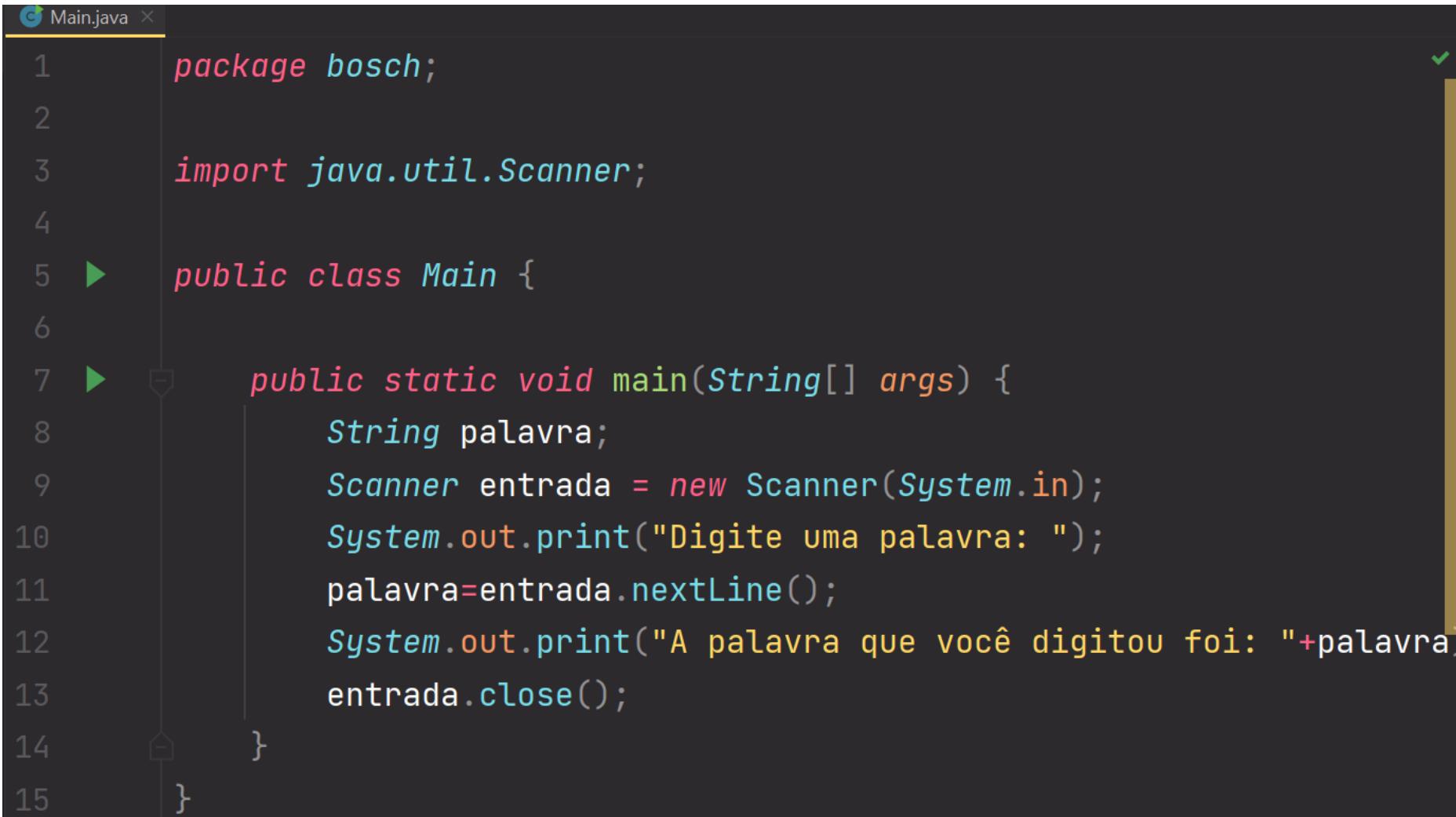


The image shows a screenshot of a Java code editor with a dark theme. The file being edited is named "Main.java". The code is as follows:

```
1 package bosch;
2
3 import java.util.Scanner;
4
5 public class Main {
6
7     public static void main(String[] args) {
8         int numero;
9         Scanner entrada = new Scanner(System.in);
10        System.out.print("Digite um número: ");
11        numero=entrada.nextInt();
12        System.out.print("O número que você digitou foi: "+numero);
13        entrada.close();
14    }
15
16 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Java – Entrada Formatada



```
>Main.java x
1 package bosch;
2
3 import java.util.Scanner;
4
5 public class Main {
6
7     public static void main(String[] args) {
8         String palavra;
9         Scanner entrada = new Scanner(System.in);
10        System.out.print("Digite uma palavra: ");
11        palavra=entrada.nextLine();
12        System.out.print("A palavra que você digitou foi: "+palavra);
13        entrada.close();
14    }
15 }
```

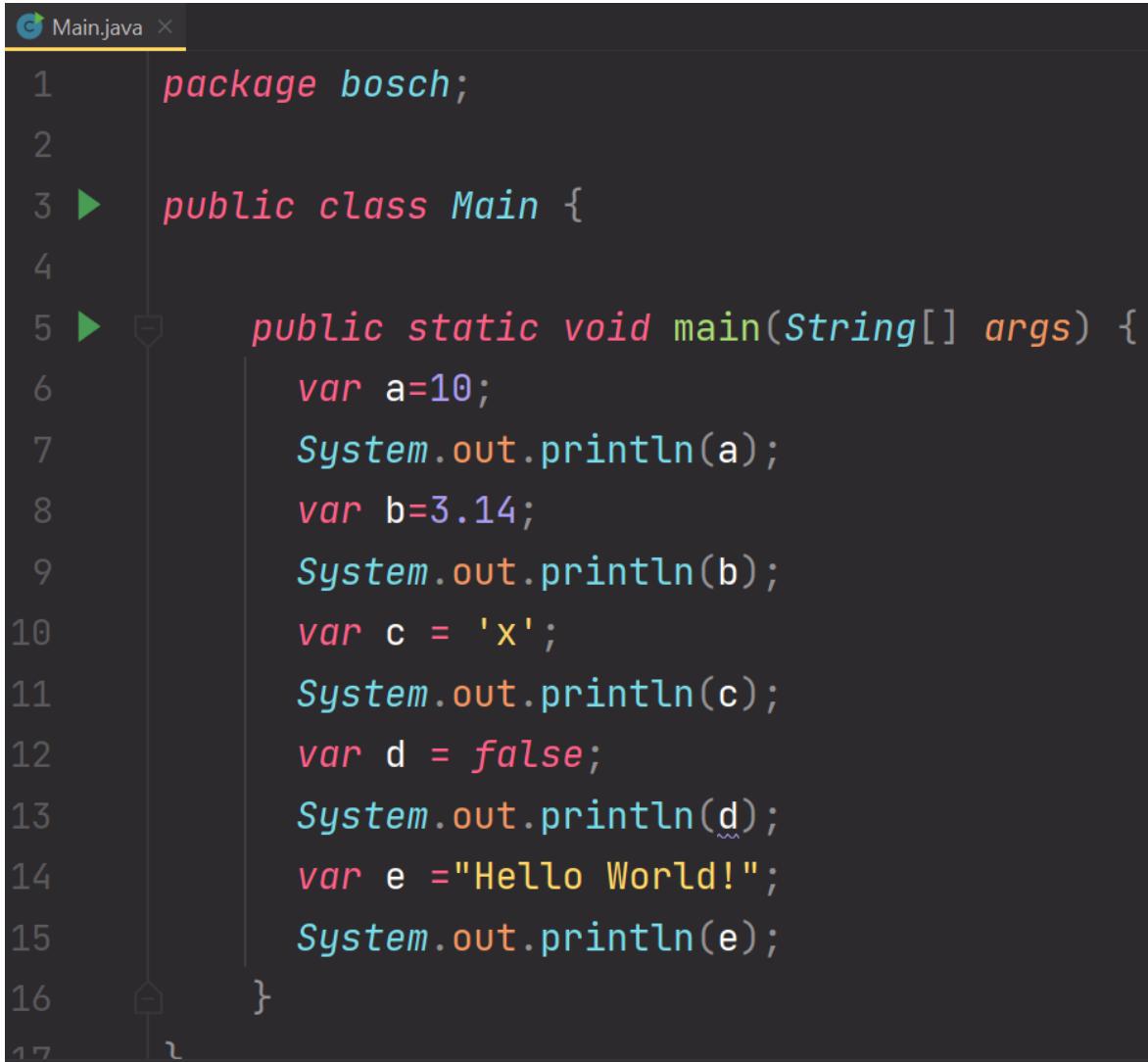
PROGRAMAÇÃO ORIENTADA A OBJETOS

Java – Entrada Formatada

```
5 ► public class Main {  
6  
7 ►   public static void main(String[] args) {  
8     Scanner entrada = new Scanner(System.in);  
9     String nome, sobrenome;  
10    int idade;  
11    nome=entrada.nextLine();  
12    System.out.println(nome);  
13    idade=entrada.nextInt();  
14    System.out.println(idade);  
15    entrada.nextLine();  
16    sobrenome=entrada.nextLine();  
17    System.out.println(sobrenome);  
18  }  
19 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Java - Inferência de Tipos



The screenshot shows a Java code editor with a dark theme. The file is named "Main.java". The code uses the new Java 10+ syntax for type inference with the `var` keyword. Annotations are present above each `var` declaration, showing the inferred type. The code prints the values of variables `a`, `b`, `c`, `d`, and `e` to the console.

```
1 package bosch;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         var a=10;
7         System.out.println(a);
8         var b=3.14;
9         System.out.println(b);
10        var c = 'x';
11        System.out.println(c);
12        var d = false;
13        System.out.println(d);
14        var e ="Hello World!";
15        System.out.println(e);
16    }
17}
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Java – Convertendo para Objeto e Verificando o Tipo

The screenshot shows a Java code editor with a dark theme. The file is named "Main.java". The code is as follows:

```
1 package bosch;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         var a = 10;
7         System.out.println(a+ " "+
8             ((Object)a).getClass().getSimpleName());
9     }
10}
```

The code demonstrates the use of the `var` keyword for variable declaration, the `System.out.println` method for output, and the `((Object)a)` cast to `Object` followed by `getClass()` and `getSimpleName()` to get the class name of the object `a`.

PROGRAMAÇÃO ORIENTADA A OBJETOS

Java – Classe Math



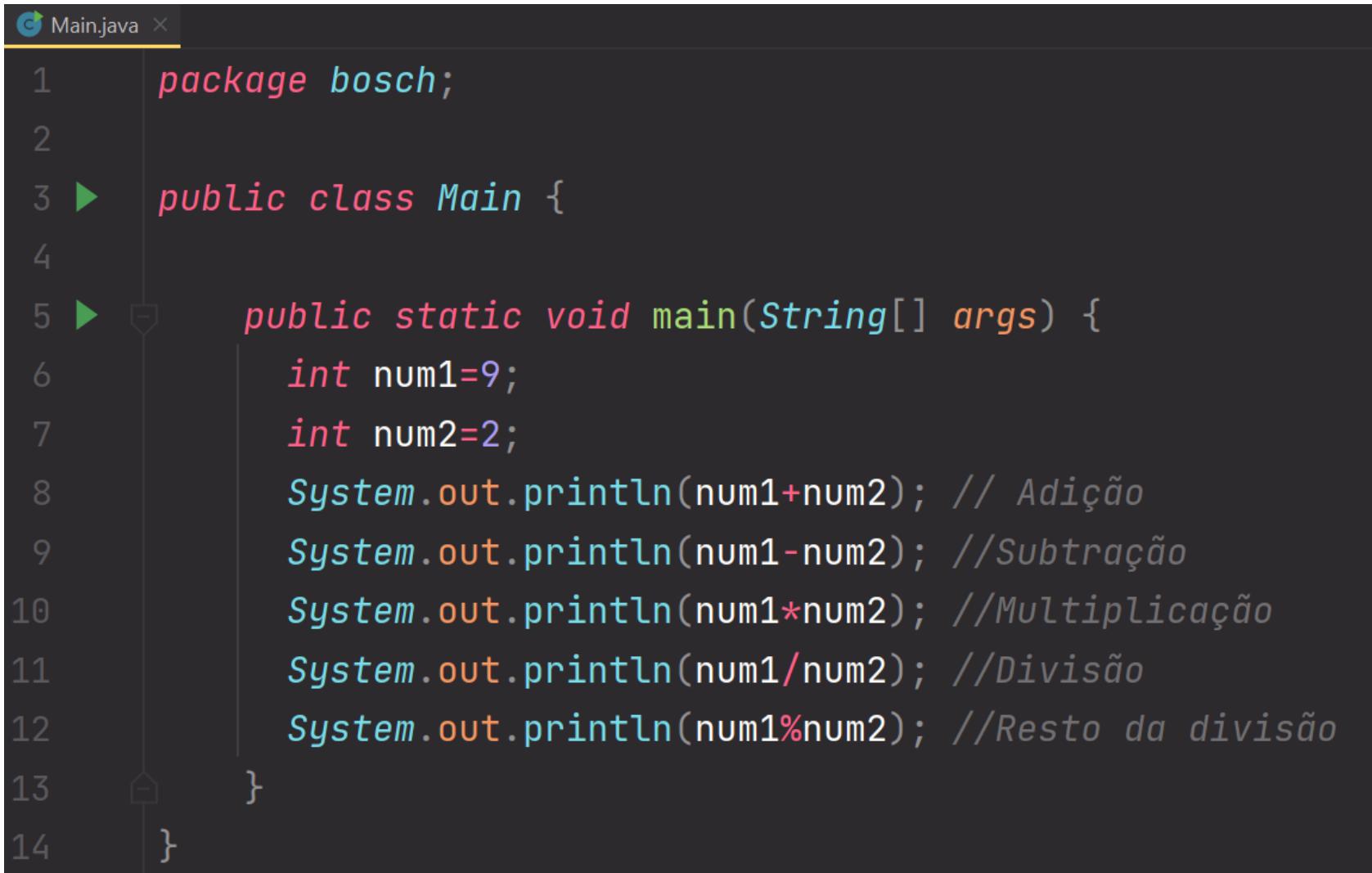
The image shows a screenshot of a Java code editor with a dark theme. The file being edited is named "Main.java". The code itself is as follows:

```
1 package bosch;
2
3
4 public class Main {
5
6     public static void main(String[] args) {
7         double raiz;
8         raiz=Math.sqrt(4);
9         System.out.println(raiz);
10
11         double potencia;
12         potencia= Math.pow(2,3);
13         System.out.println(potencia);
14
15     }
16 }
```

The code demonstrates the use of the `Math` class to calculate the square root of 4 and the cube of 2, and then prints the results.

PROGRAMAÇÃO ORIENTADA A OBJETOS

Java – Operadores Aritméticos

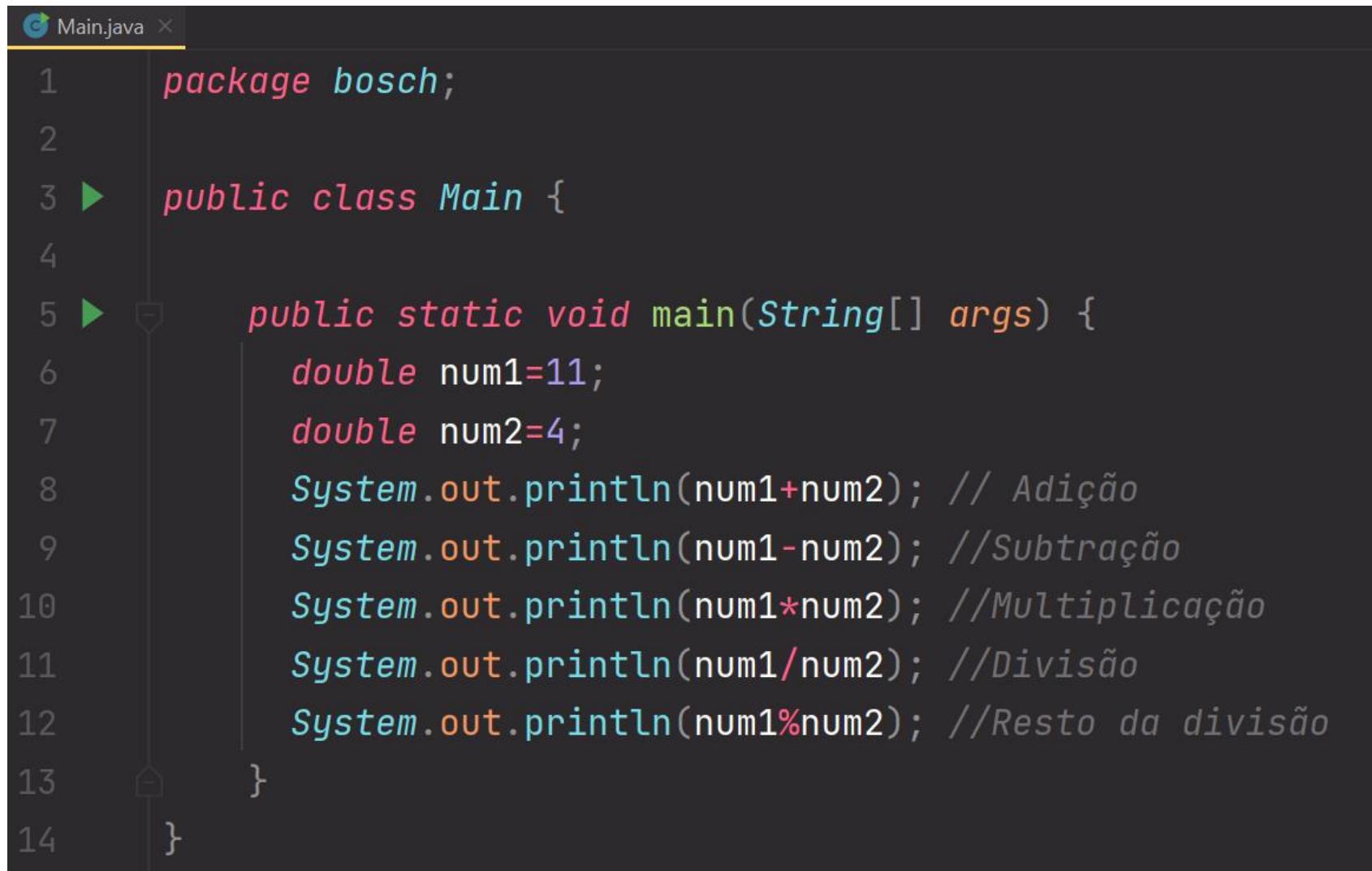


The image shows a screenshot of a Java code editor with a dark theme. The file being edited is named "Main.java". The code itself is as follows:

```
1 package bosch;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         int num1=9;
7         int num2=2;
8         System.out.println(num1+num2); // Adição
9         System.out.println(num1-num2); // Subtração
10        System.out.println(num1*num2); // Multiplicação
11        System.out.println(num1/num2); // Divisão
12        System.out.println(num1%num2); // Resto da divisão
13    }
14 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Java – Operadores Aritméticos



```
Main.java
1 package bosch;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         double num1=11;
7         double num2=4;
8         System.out.println(num1+num2); // Adição
9         System.out.println(num1-num2); // Subtração
10        System.out.println(num1*num2); // Multiplicação
11        System.out.println(num1/num2); // Divisão
12        System.out.println(num1%num2); // Resto da divisão
13    }
14 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Java – Exercícios Básicos

1. Criar um programa que leia a temperatura em Fahrenheit e converta para Celsius.
2. Criar um programa que leia a temperatura em Celsius e converta para Fahrenheit.
3. Criar um programa que leia o peso e a altura do usuário e imprima no console o IMC.
4. Criar um programa que leia um valor e apresente os resultados ao quadrado e ao cubo do valor.
5. Criar um programa que leia o valor da base e da altura de um triângulo e calcule a área.
6. Criar um programa que resolve equações do segundo grau ($ax^2 + bx + c = 0$) utilizando a fórmula de Bhaskara. Use como exemplo $a = 1$, $b = 12$ e $c = -13$. Encontre o delta

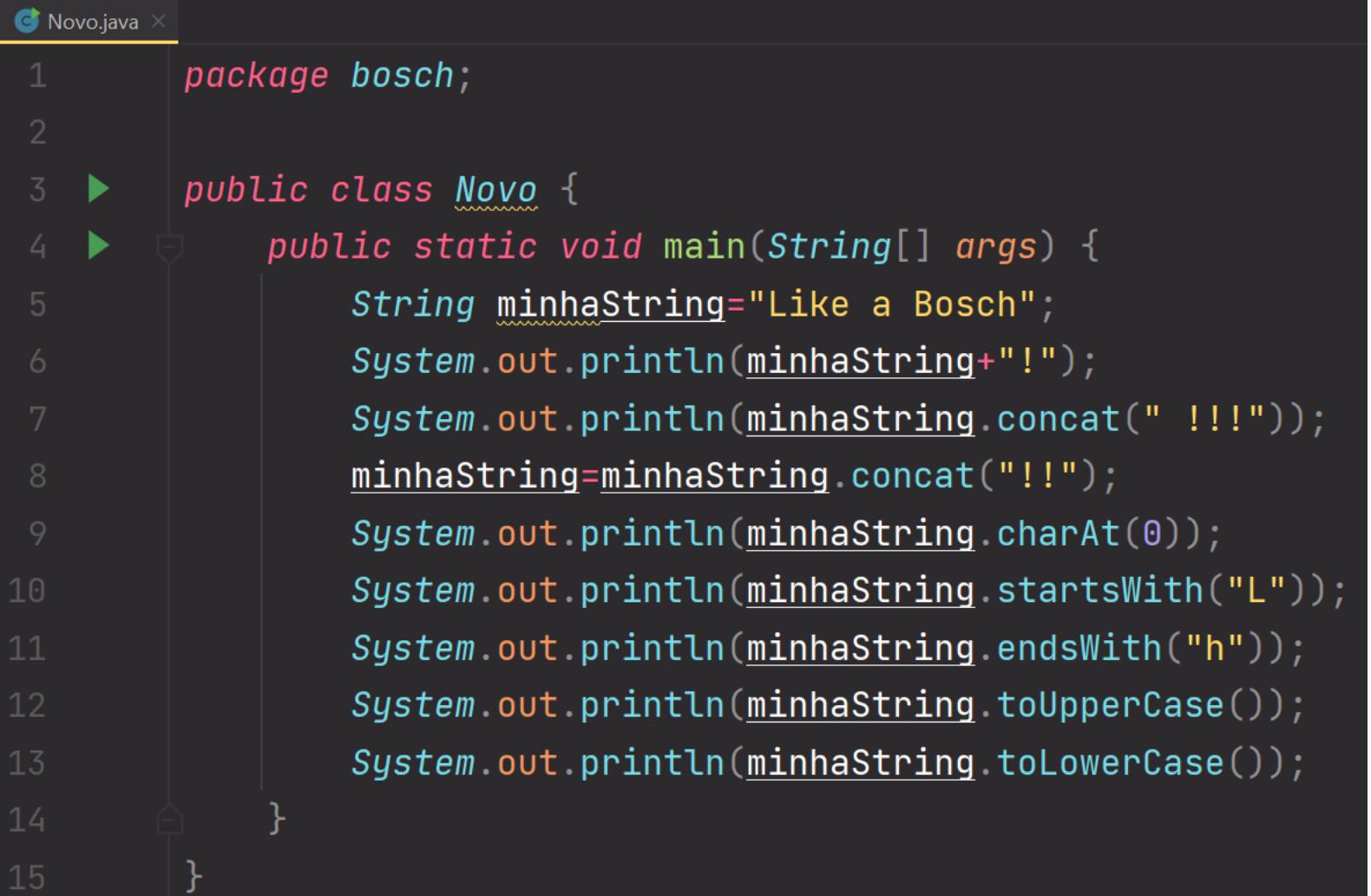
PROGRAMAÇÃO ORIENTADA A OBJETOS

Java – Exercícios Básicos

$$1) \frac{3 \cdot \left(\frac{-3}{4}\right)^{-2} + 6 \cdot \left(\frac{3^{-1}}{4}\right)^{-1} - 4}{7 \cdot \left(\frac{-3}{4}\right)^{-1} + 2} + 4 =$$

PROGRAMAÇÃO ORIENTADA A OBJETOS

Java – Classe String



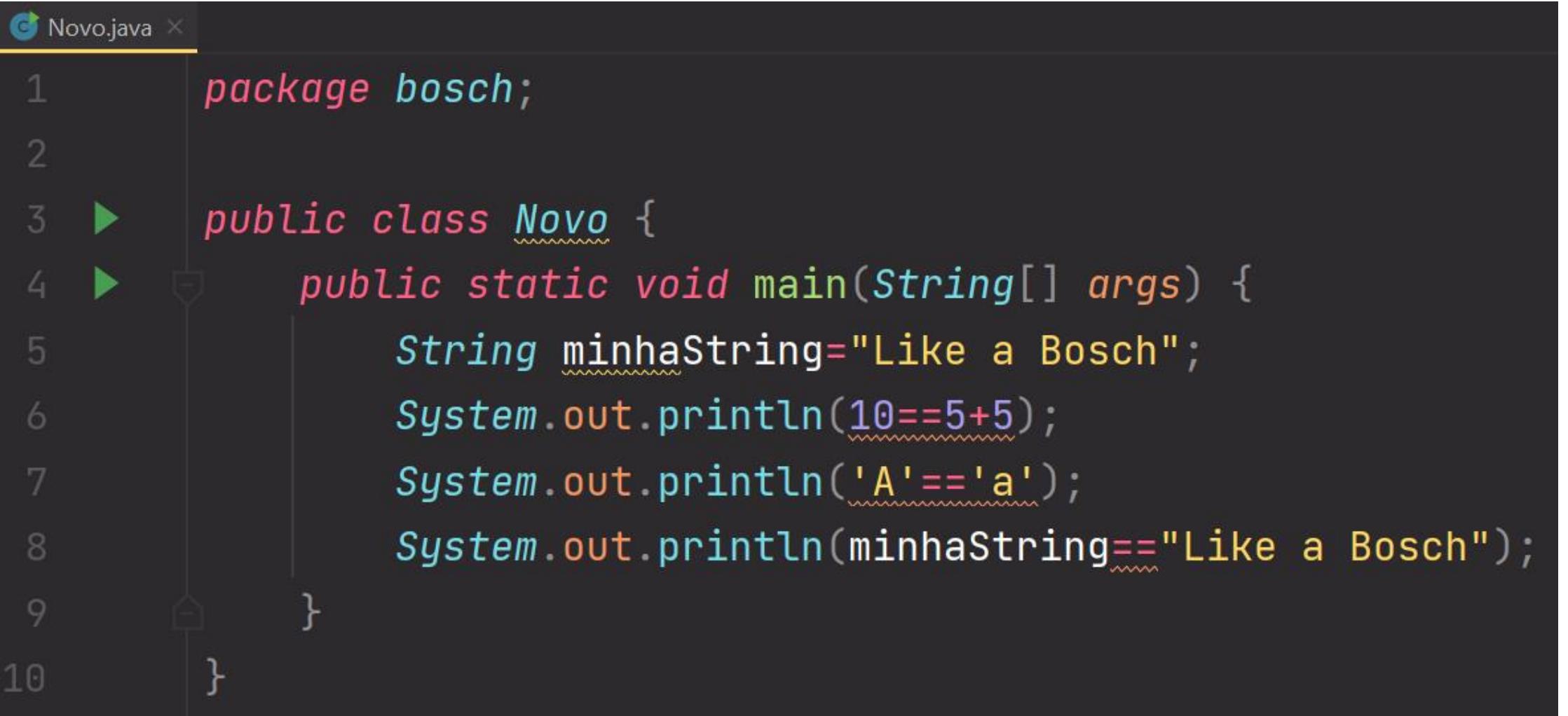
The screenshot shows a Java code editor with a dark theme. The file is named 'Novo.java'. The code defines a class 'Novo' with a main method. It creates a string 'minhaString' with the value 'Like a Bosch', then prints it followed by an exclamation mark. It concatenates three exclamation marks to the string and prints it again. It then concatenates two exclamation marks to the string and prints it. It prints the first character of the string. It checks if the string starts with 'L' and ends with 'h'. It converts the string to uppercase and then to lowercase.

```
Novo.java
package bosch;

public class Novo {
    public static void main(String[] args) {
        String minhaString="Like a Bosch";
        System.out.println(minhaString+"!");
        System.out.println(minhaString.concat(" !!!"));
        minhaString=minhaString.concat(" !! ");
        System.out.println(minhaString.charAt(0));
        System.out.println(minhaString.startsWith("L"));
        System.out.println(minhaString.endsWith("h"));
        System.out.println(minhaString.toUpperCase());
        System.out.println(minhaString.toLowerCase());
    }
}
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Java – Classe String



The screenshot shows a Java code editor with a dark theme. The file tab at the top left is labeled "Novo.java". The code itself is as follows:

```
1 package bosch;
2
3 public class Novo {
4     public static void main(String[] args) {
5         String minhaString="Like a Bosch";
6         System.out.println(10==5+5);
7         System.out.println('A'=='a');
8         System.out.println(minhaString=="Like a Bosch");
9     }
10 }
```

The code demonstrates basic Java syntax, including a package declaration, a class definition, a main method, string literals, and character comparisons.

PROGRAMAÇÃO ORIENTADA A OBJETOS

Java – Classe String

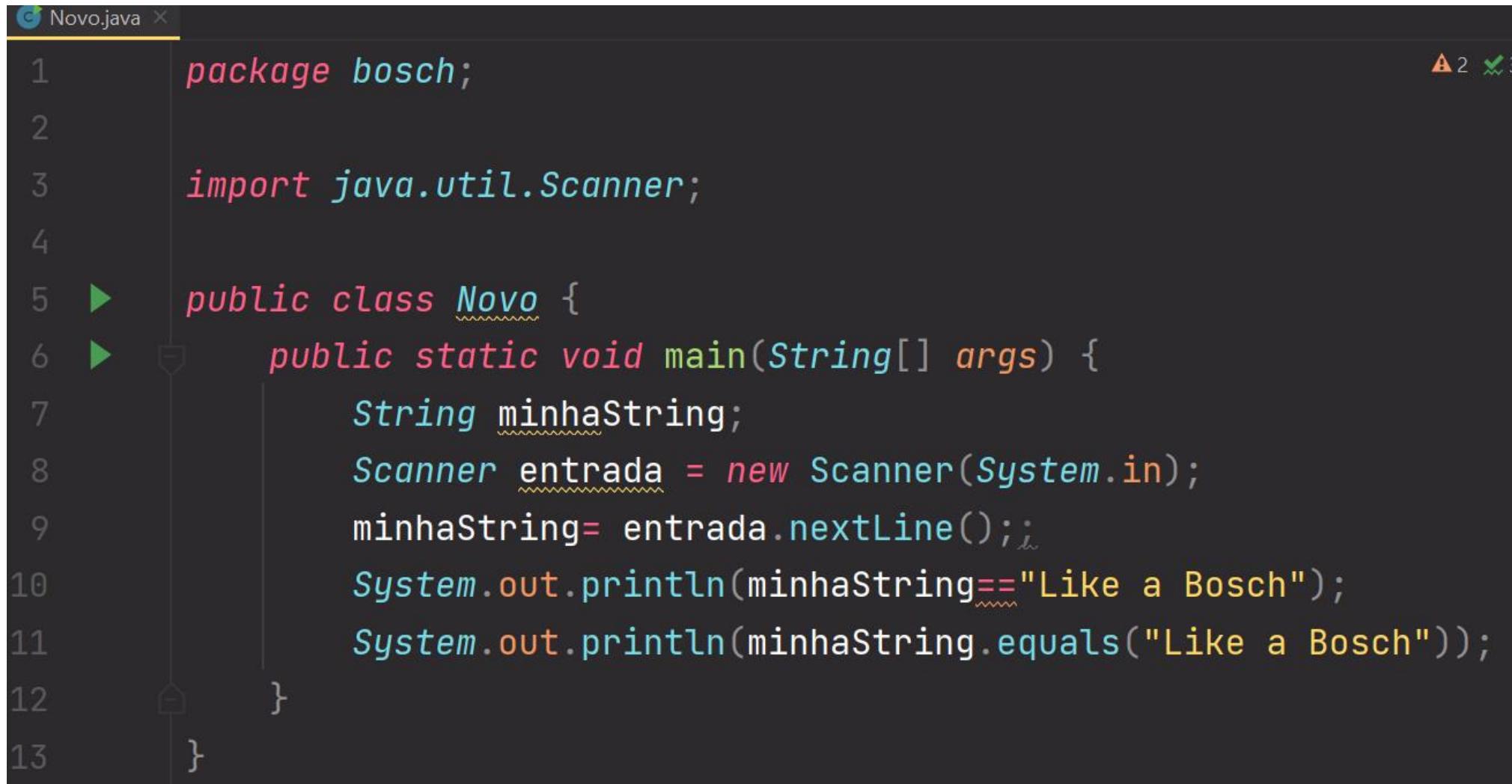
The screenshot shows a Java code editor with a dark theme. The file is named 'Novo.java'. The code is as follows:

```
1 package bosch;
2
3 ► public class Novo {
4     ►     public static void main(String[] args) {
5         String minhaString=new String( original: "Like a Bosch");
6         System.out.println(minhaString=="Like a Bosch");
7     }
8 }
```

The code editor highlights syntax: 'package' in red, 'String' in green, and strings in orange. It also shows code completion suggestions for 'minhaString' and 'original:'.

PROGRAMAÇÃO ORIENTADA A OBJETOS

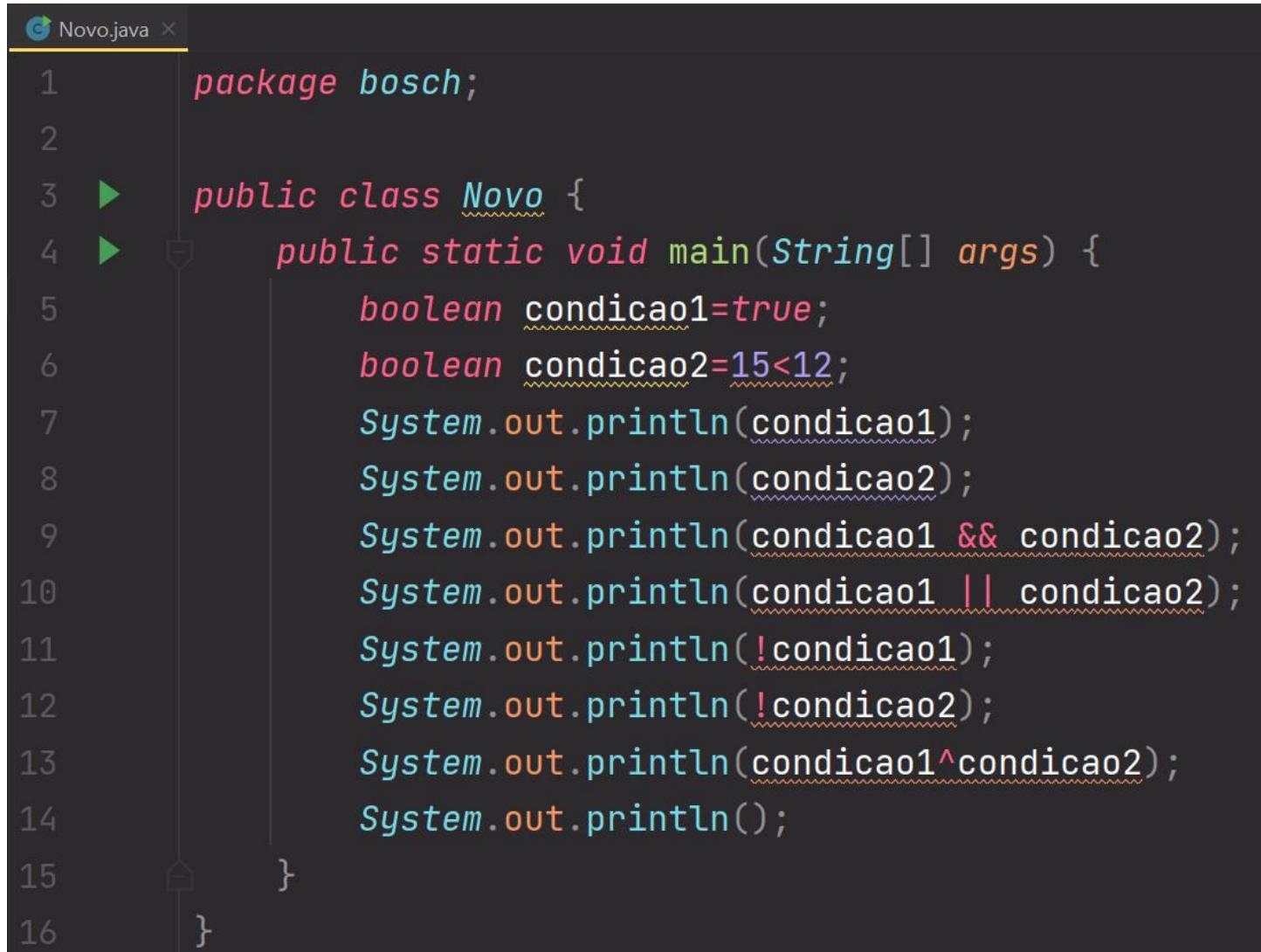
Java – Classe String



```
Novo.java x
1 package bosch;
2
3 import java.util.Scanner;
4
5 public class Novo {
6     public static void main(String[] args) {
7         String minhaString;
8         Scanner entrada = new Scanner(System.in);
9         minhaString = entrada.nextLine();
10        System.out.println(minhaString == "Like a Bosch");
11        System.out.println(minhaString.equals("Like a Bosch"));
12    }
13 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

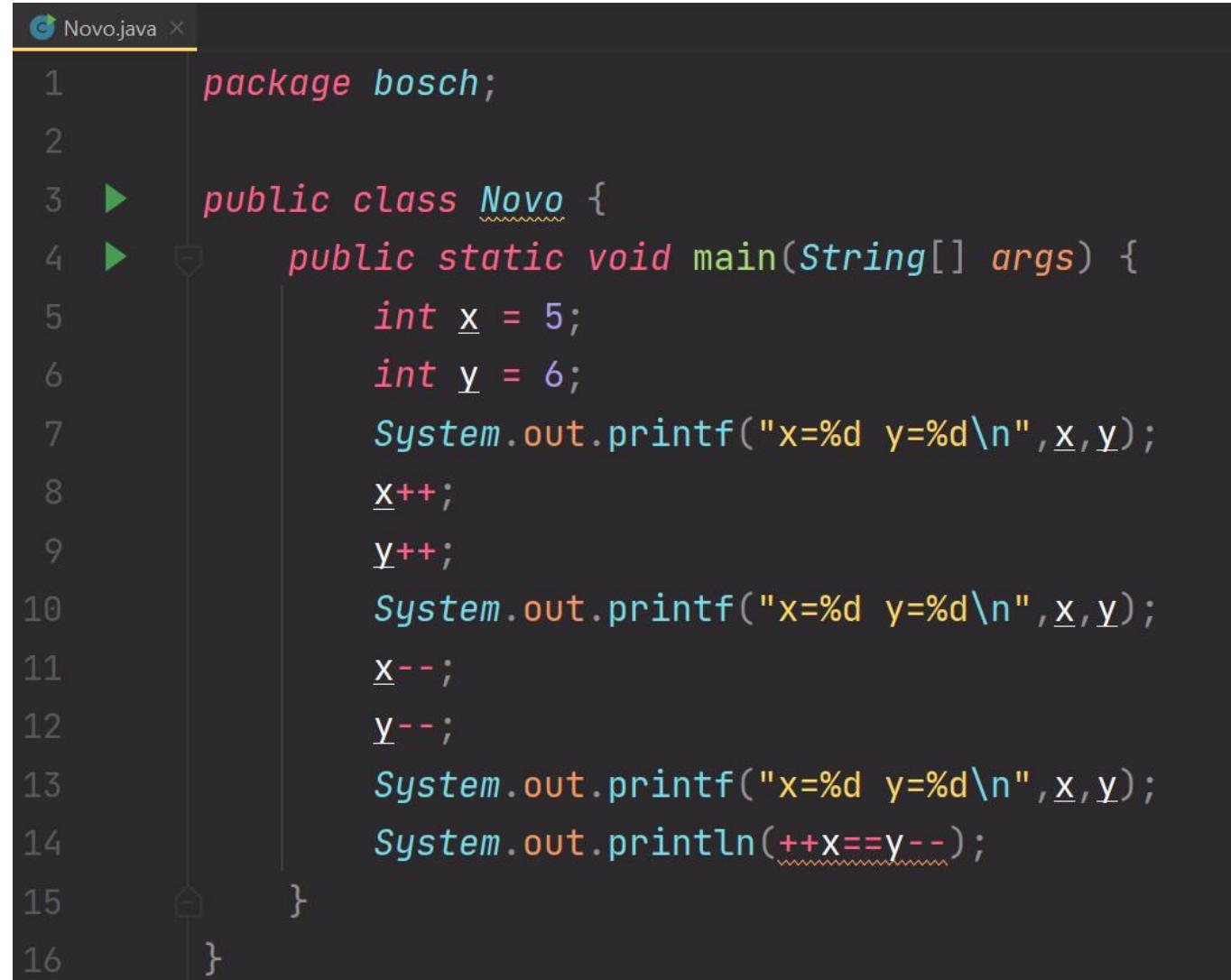
Java – Operadores Lógicos



```
Novo.java x
1 package bosch;
2
3 ► public class Novo {
4 ►     public static void main(String[] args) {
5         boolean condicao1=true;
6         boolean condicao2=15<12;
7         System.out.println(condicao1);
8         System.out.println(condicao2);
9         System.out.println(condicao1 && condicao2);
10        System.out.println(condicao1 || condicao2);
11        System.out.println(!condicao1);
12        System.out.println(!condicao2);
13        System.out.println(condicao1^condicao2);
14        System.out.println();
15    }
16 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Java – Operadores de Incremento e Decremento

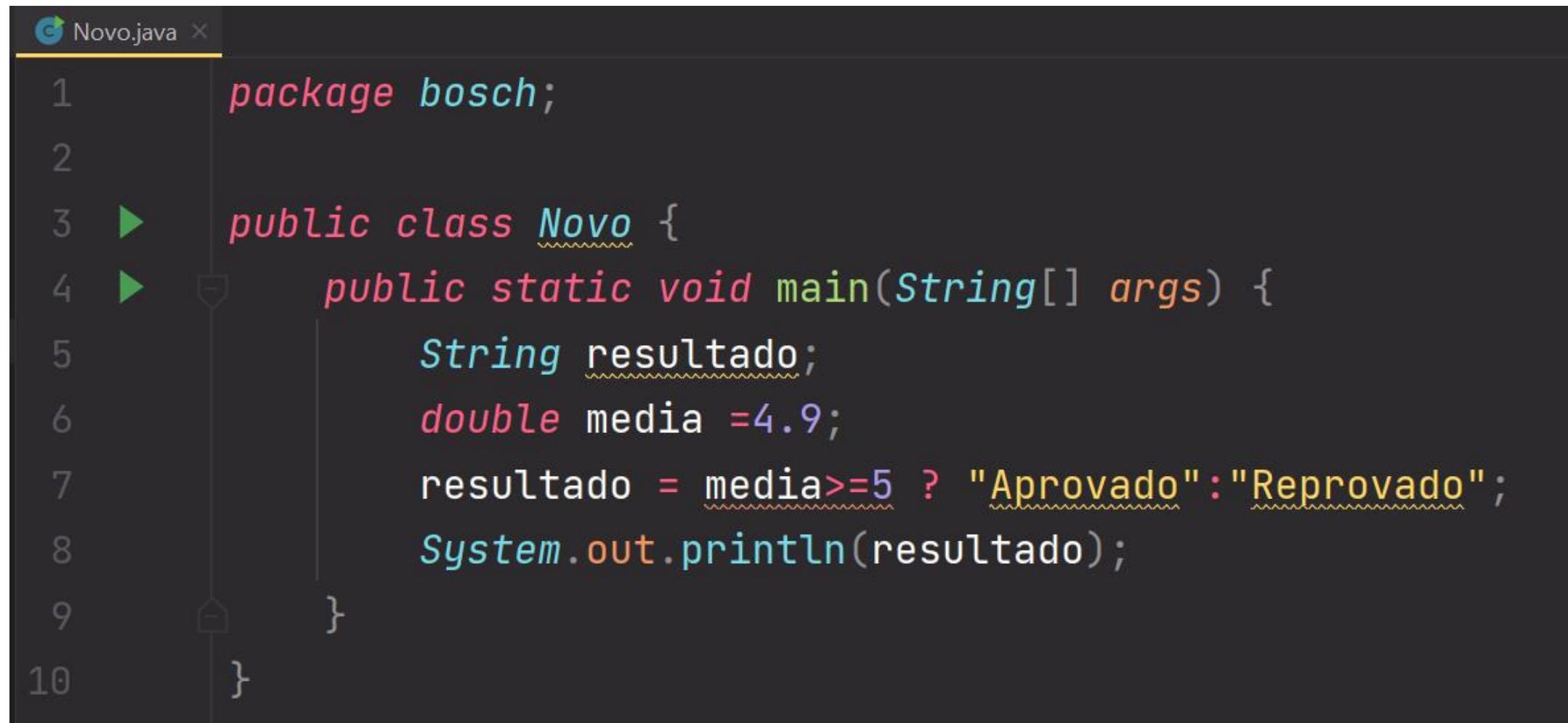


The image shows a screenshot of a Java code editor with a dark theme. The file being edited is named "Novo.java". The code contains several lines of Java code demonstrating the use of increment and decrement operators. The code is as follows:

```
1 package bosch;
2
3 public class Novo {
4     public static void main(String[] args) {
5         int x = 5;
6         int y = 6;
7         System.out.printf("x=%d y=%d\n", x, y);
8         x++;
9         y--;
10        System.out.printf("x=%d y=%d\n", x, y);
11        x--;
12        y--;
13        System.out.printf("x=%d y=%d\n", x, y);
14        System.out.println(++x == y--);
15    }
16 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

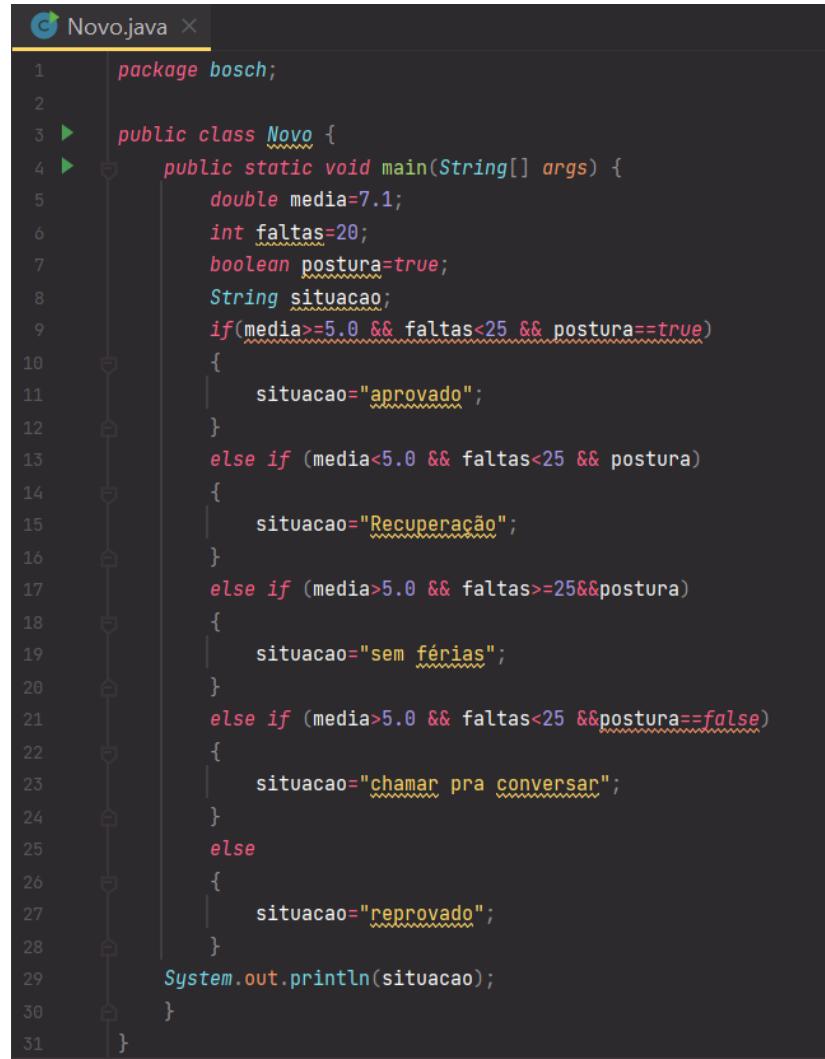
Java – Operador Ternário



```
Novo.java ×
1 package bosch;
2
3 ► public class Novo {
4     ►     public static void main(String[] args) {
5         String resultado;
6         double media = 4.9;
7         resultado = media >= 5 ? "Aprovado": "Reprovado";
8         System.out.println(resultado);
9     }
10 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Java – Comando Condicionais



```
Novo.java
1 package bosch;
2
3 public class Novo {
4     public static void main(String[] args) {
5         double media=7.1;
6         int faltas=20;
7         boolean postura=true;
8         String situacao;
9         if(media>=5.0 && faltas<25 && postura==true)
10        {
11            situacao="aprovado";
12        }
13        else if (media<5.0 && faltas<25 && postura)
14        {
15            situacao="Recuperação";
16        }
17        else if (media>5.0 && faltas>=25&&postura)
18        {
19            situacao="sem férias";
20        }
21        else if (media>5.0 && faltas<25 &&postura==false)
22        {
23            situacao="chamar pra conversar";
24        }
25        else
26        {
27            situacao="reprovado";
28        }
29        System.out.println(situacao);
30    }
31}
```

```
1 package bosch;  
2  
3 ► public class Novo {  
4 ►     public static void main(String[] args) {  
5         Byte b = 100;  
6         Short s =1000;  
7         Integer i = 10000;  
8         Long l = 100000L;  
9  
10            System.out.println(b.byteValue());  
11            System.out.println(s.toString());  
12            System.out.println(i*3);  
13            System.out.println(l/3);  
14  
15        }  
16    }
```

```
1 package bosch;  
2  
3 ► public class Novo {  
4 ►     public static void main(String[] args) {  
5         Float f = 123.10F;  
6         System.out.println(f);  
7  
8         Double d = 1234.5678;  
9         System.out.println(d);  
10  
11        Boolean bo = Boolean.parseBoolean(s: "true");  
12        System.out.println(bo);  
13        System.out.println(bo.toString().toUpperCase());  
14  
15    }  
16 }
```

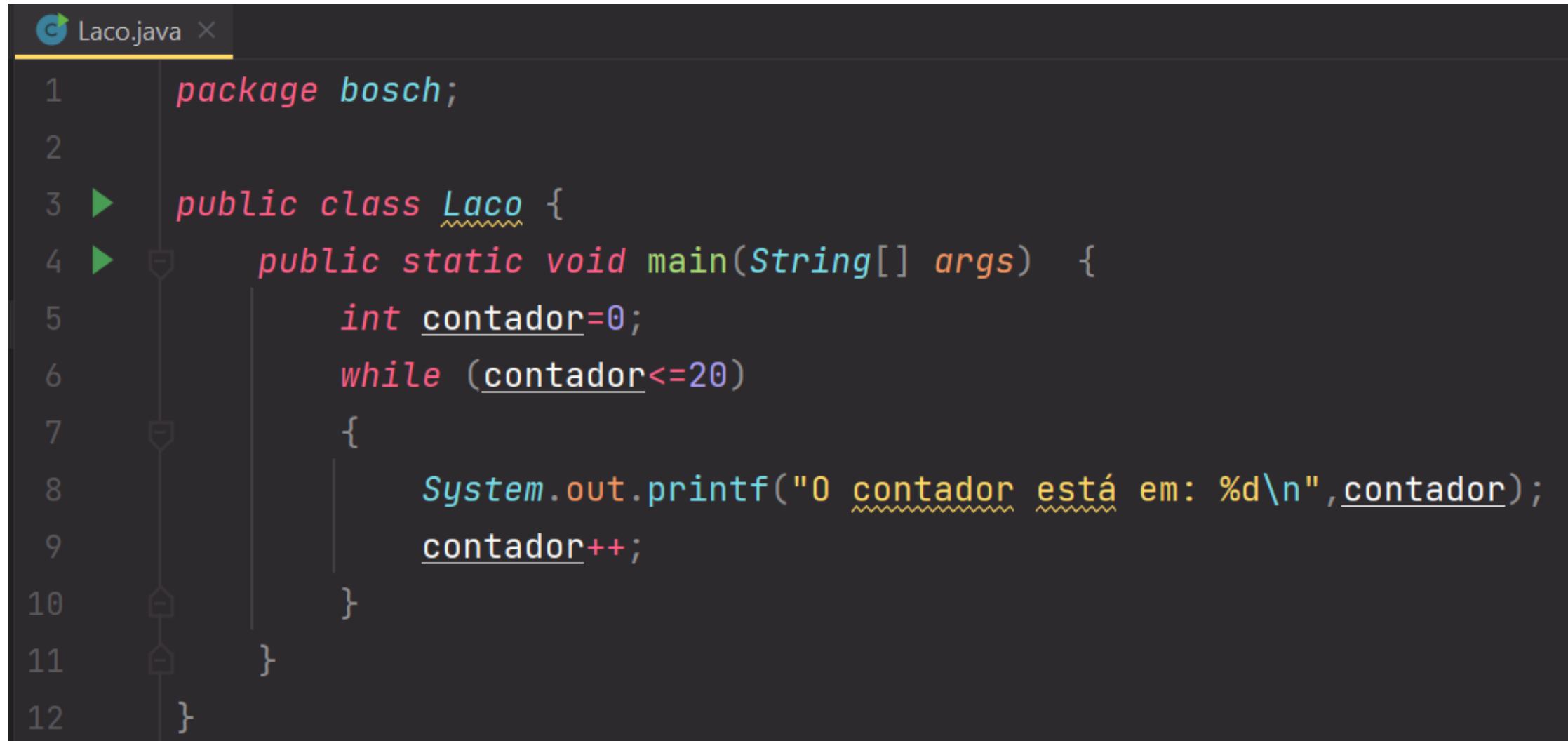
```
1 package bosch;  
2  
3 ► public class Novo {  
4 ►     public static void main(String[] args) {  
5         double a=1;  
6         System.out.println(a);  
7  
8         float b = (float) 1.12249999999;  
9         System.out.println(b);  
10  
11        int c=127;  
12        byte d = (byte) (c);  
13        System.out.println(d);  
14  
15        double e =1.99999999999;  
16        int f = (int) e;  
17        System.out.println(f);  
18    }  
19}
```

```
1 package bosch;
2
3 ► public class Novo {
4 ►   public static void main(String[] args) {
5     Integer num1=10000;
6     System.out.println(num1.toString().length());
7
8     int num2 = 1000000;
9     System.out.println(Integer.toString(num2).length());
10
11    System.out.println(""+num2).length());
12  }
13 }
```

```
1 package bosch;  
2  
3 ► public class Novo {  
4 ►     public static void main(String[] args) {  
5         String numero1= "12";  
6         String numero2="3.14";  
7  
8         int x = Integer.parseInt(numero1);  
9         double y = Double.parseDouble(numero2);  
10        double soma =x+y;  
11        System.out.println(x);  
12        System.out.println(y);  
13        System.out.println(soma);  
14    }  
15 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Java – Laços de Repetição



The image shows a screenshot of a Java code editor with a dark theme. The file being edited is named `Laco.java`. The code itself is a simple Java program that prints the value of a counter variable from 0 to 20. The code is color-coded: `package`, `public`, `class`, `int`, `while`, `System.out.printf`, and `++` are in blue; `bosch` is in red; and `main`, `args`, and `contador` are in green. The code is as follows:

```
1 package bosch;
2
3 public class Laco {
4     public static void main(String[] args) {
5         int contador=0;
6         while (contador<=20)
7         {
8             System.out.printf("O contador está em: %d\n",contador);
9             contador++;
10        }
11    }
12 }
```

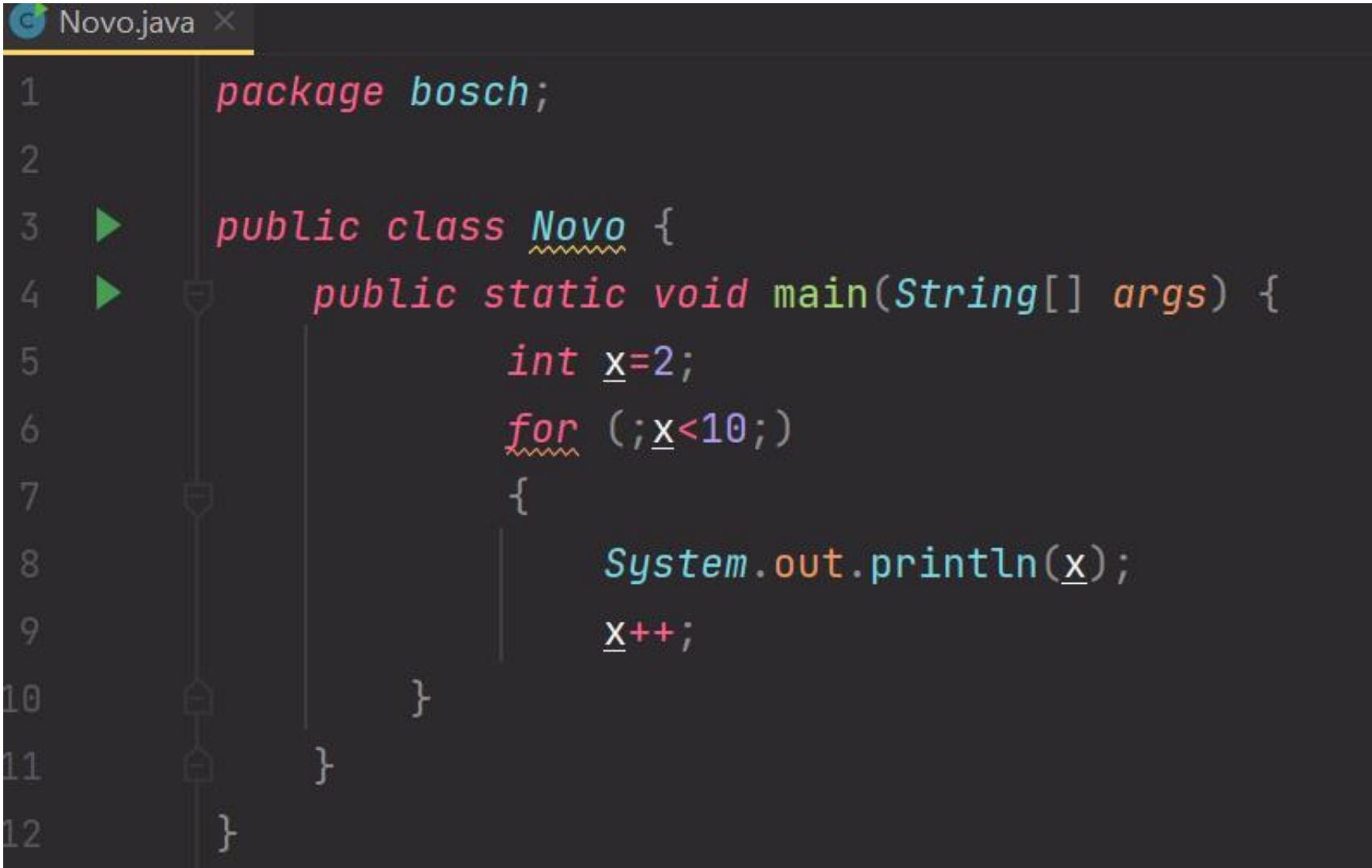
PROGRAMAÇÃO ORIENTADA A OBJETOS

Java – Laços de Repetição

```
Novo.java ×  
1 package bosch;  
2  
3 ► public class Novo {  
4 ►     public static void main(String[] args) throws InterruptedException {  
5         for (int i = 0; i < 10 ; i++) {  
6             System.out.println(i);  
7             Thread.sleep( millis: 1000);  
8         }  
9     }  
10 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Java – Laços de Repetição

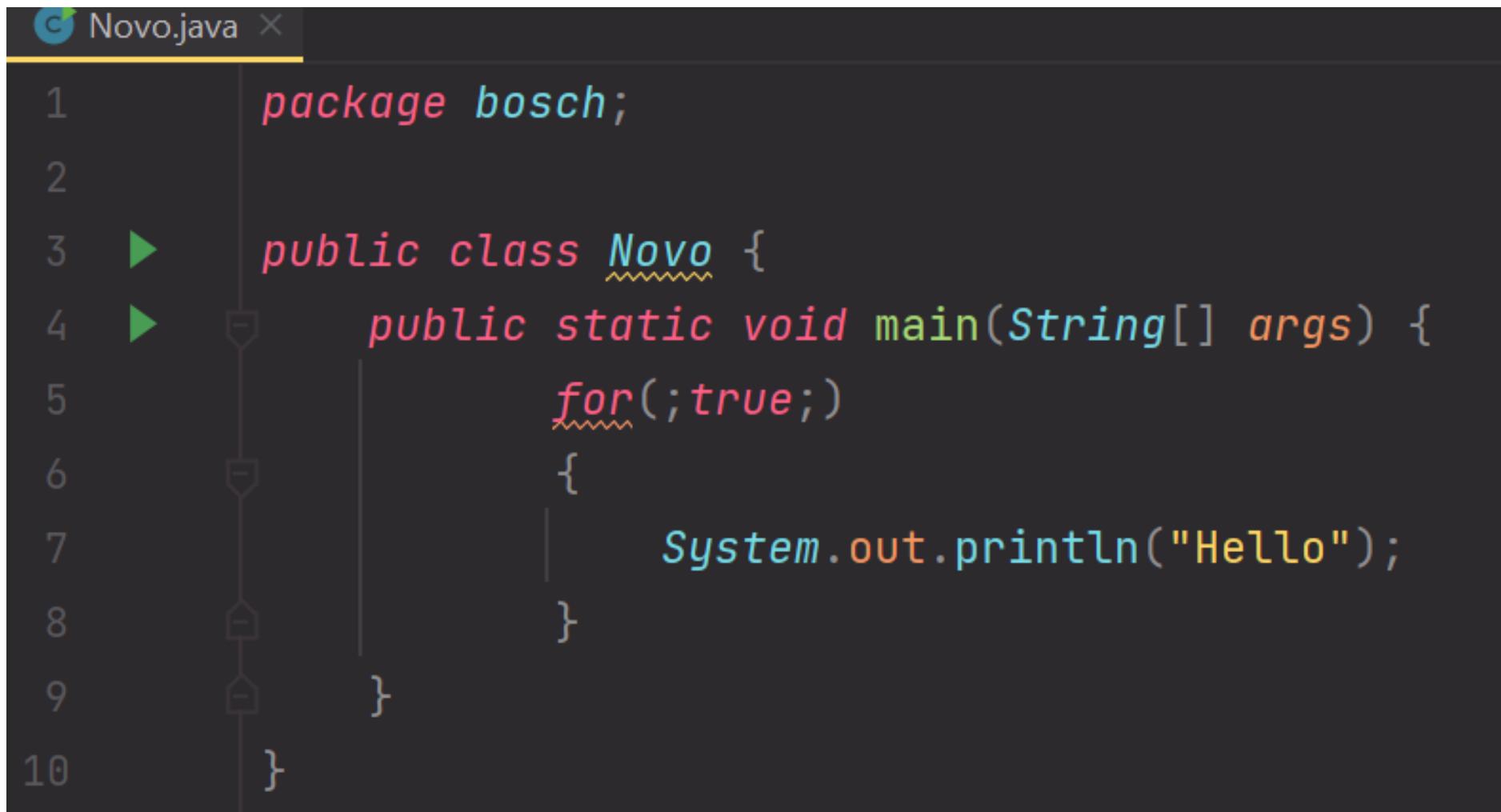


The image shows a screenshot of a Java code editor with a dark theme. The file being edited is named "Novo.java". The code contains a simple for loop that prints integers from 2 to 10 to the console.

```
1 package bosch;
2
3 public class Novo {
4     public static void main(String[] args) {
5         int x=2;
6         for (;x<10;) {
7             System.out.println(x);
8             x++;
9         }
10    }
11 }
12 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Java – Laços de Repetição



The image shows a screenshot of a Java code editor with a dark theme. The file being edited is named "Novo.java". The code contains a package declaration and a class definition named "Novo". Inside the class, there is a main method that prints "Hello" to the console in a continuous loop. The code is numbered from 1 to 10 on the left side.

```
1 package bosch;
2
3 public class Novo {
4     public static void main(String[] args) {
5         for(;true;)
6         {
7             System.out.println("Hello");
8         }
9     }
10 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Java – Laços de Repetição



The image shows a screenshot of a Java code editor with a dark theme. The file is named "Novo.java". The code implements a simple console application using the Scanner class to read input from the user. It features a while loop that continues to ask for input until the user types "sair". The code is color-coded: package, import, and class names are in purple; variable names and method parameters are in blue; and keywords like public, static, void, and while are in red. Brackets and operators are in black.

```
1 package bosch;
2
3 import java.util.Scanner;
4
5 public class Novo {
6     public static void main(String[] args) {
7         Scanner entrada = new Scanner(System.in);
8         String valor="";
9         while (!valor.equalsIgnoreCase("sair"))
10     {
11         System.out.print("Digite algo: ");
12         valor=entrada.nextLine();
13     }
14     entrada.close();
15 }
16 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Java – Laços de Repetição

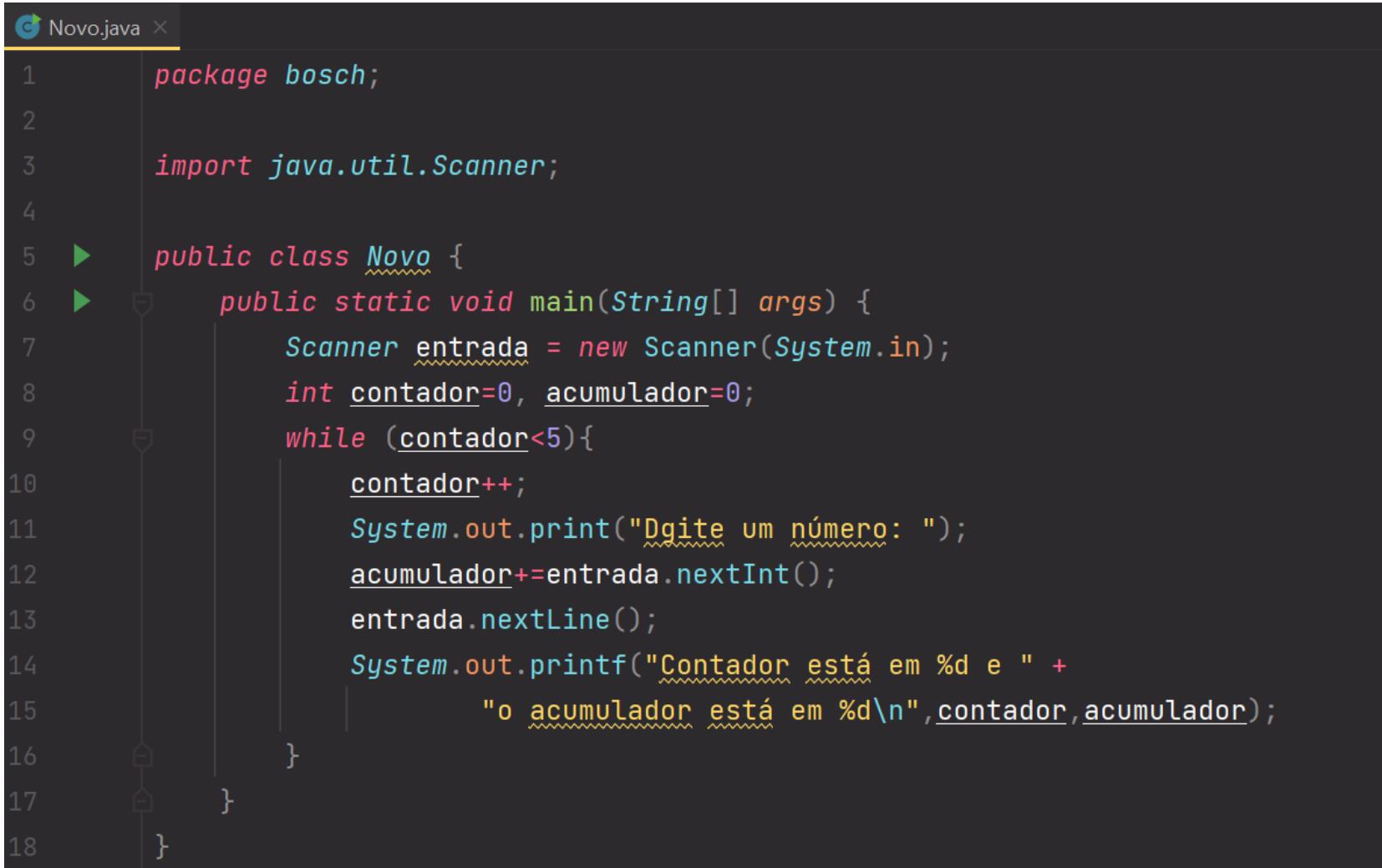


The image shows a screenshot of a Java code editor with a dark theme. The file is named "Novo.java". The code implements a do-while loop to read multiple lines of input from the standard input stream (System.in) using a Scanner object. The program prompts the user to enter text, reads it, and continues until the user enters the string "algo".

```
1 package bosch;
2
3 import java.util.Scanner;
4
5 public class Novo {
6     public static void main(String[] args) {
7         Scanner entrada = new Scanner(System.in);
8         String valor = "";
9         do{
10             System.out.print("Diga-me algo: ");
11             valor= entrada.nextLine();
12         }while (!valor.equalsIgnoreCase("algo"));
13         entrada.close();
14     }
15 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Java – Laços de Repetição



```
Novo.java
1 package bosch;
2
3 import java.util.Scanner;
4
5 public class Novo {
6     public static void main(String[] args) {
7         Scanner entrada = new Scanner(System.in);
8         int contador=0, acumulador=0;
9         while (contador<5){
10             contador++;
11             System.out.print("Dgite um número: ");
12             acumulador+=entrada.nextInt();
13             entrada.nextLine();
14             System.out.printf("Contador está em %d e " +
15                             "o acumulador está em %d\n",contador,acumulador);
16         }
17     }
18 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

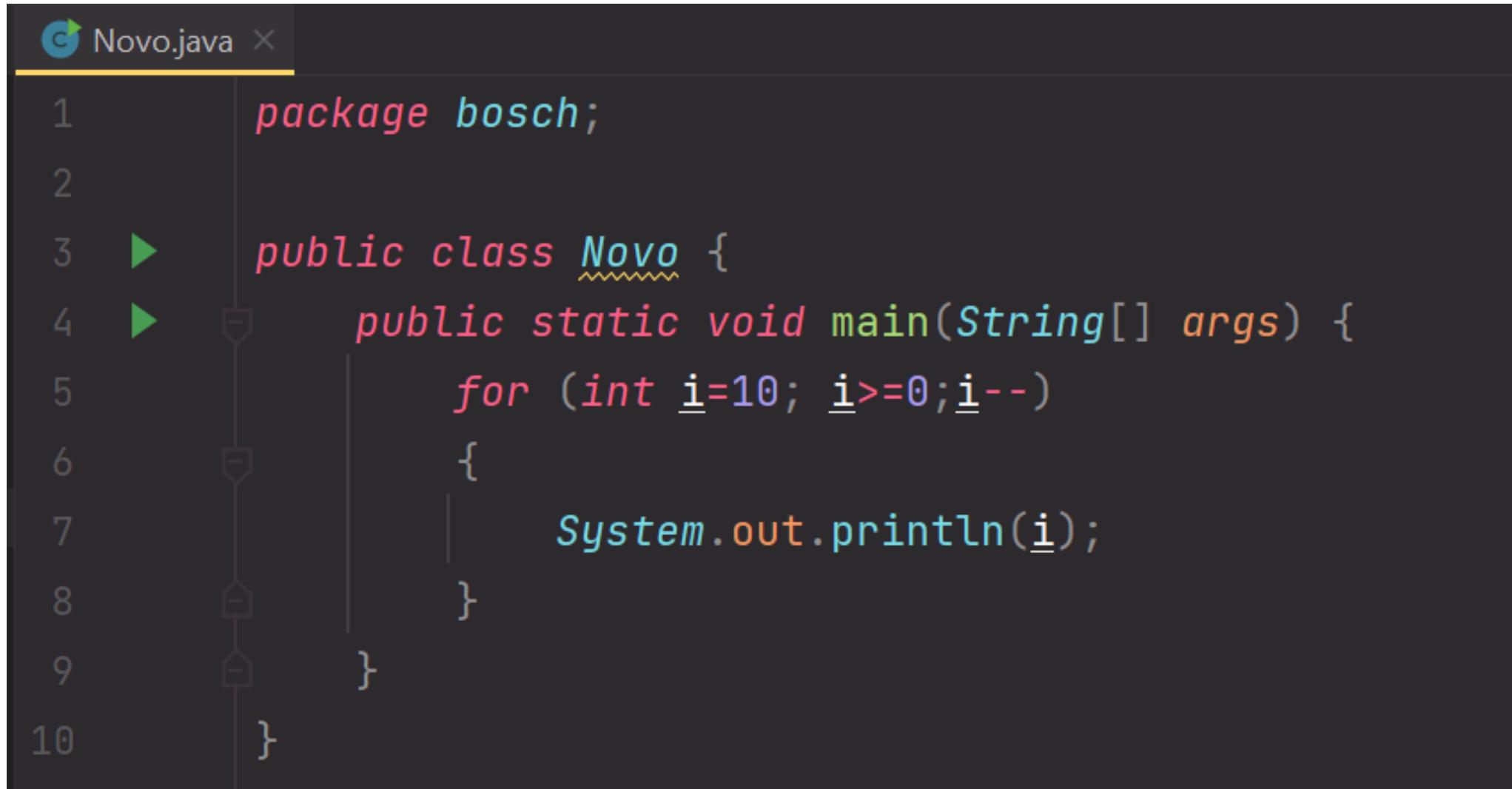
Java – Laços de Repetição

```
Novo.java
package bosch;

public class Novo {
    public static void main(String[] args) throws InterruptedException {
        int contador=0, acumulador=0;
        for(int i=0; i<5; i++){
            contador=i;
            acumulador+=i*i;
            System.out.printf("Contador está em %d e " +
                "o acumulador está em %d\n",contador,acumulador);
            Thread.sleep( millis: 2000 );
        }
    }
}
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Java – Laços de Repetição



The image shows a screenshot of a Java code editor with a dark theme. The file being edited is named "Novo.java". The code contains a single for loop that prints integers from 10 down to 0 to the console.

```
1 package bosch;
2
3 public class Novo {
4     public static void main(String[] args) {
5         for (int i=10; i>=0;i--)
6         {
7             System.out.println(i);
8         }
9     }
10 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Java – Laços de Repetição

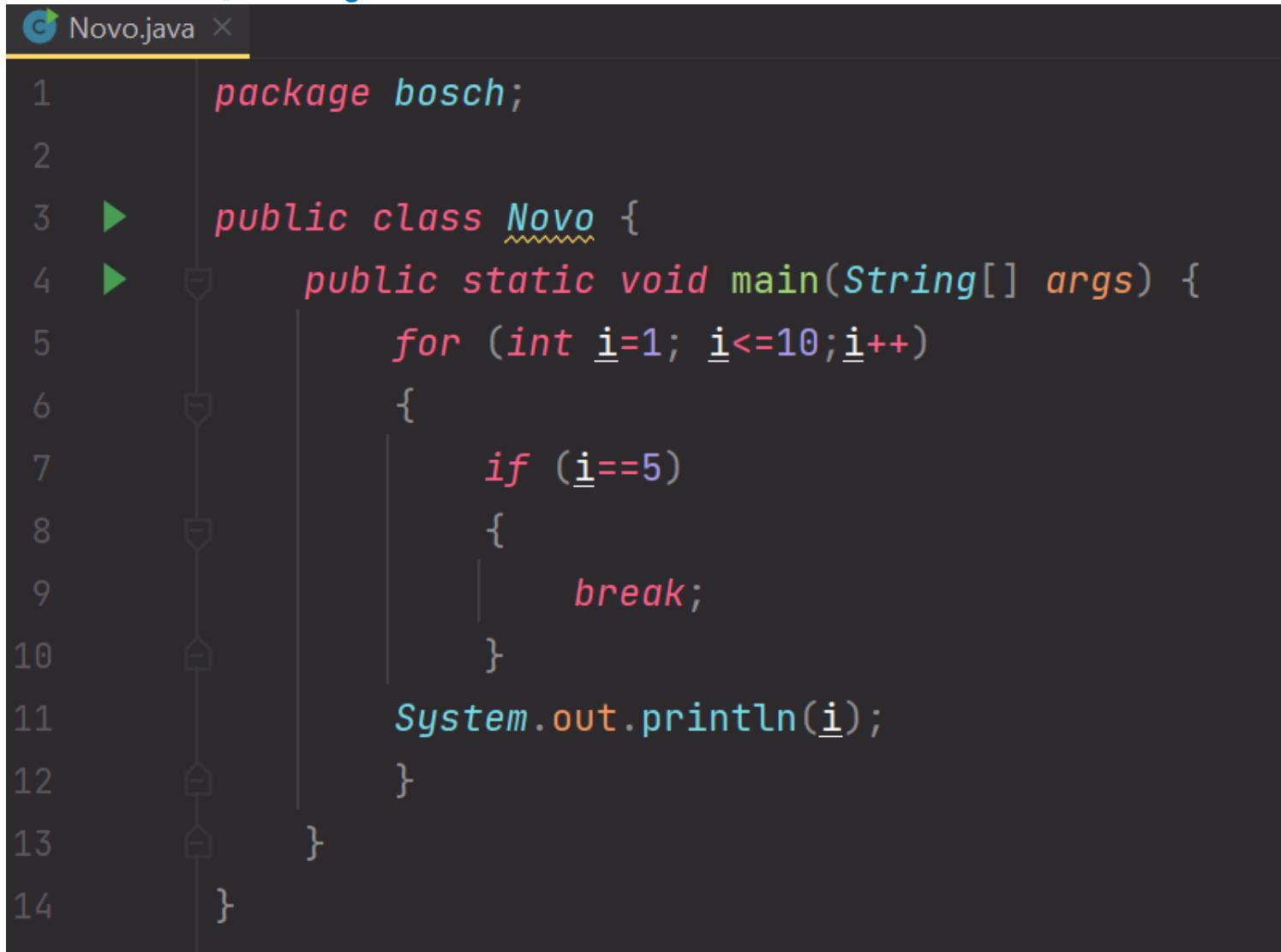
The screenshot shows a Java code editor window with a dark theme. The file tab at the top is labeled "Novo.java". The code itself is as follows:

```
1 package bosch;
2
3 public class Novo {
4     public static void main(String[] args) {
5         for (int i=1; i<=10;i++)
6         {
7             for (int j = 0; j <=10 ; j++) {
8                 System.out.printf("%d X %d = %d\n",i,j,i*j);
9             }
10            System.out.println();
11        }
12    }
13 }
```

The code uses nested for loops to print a multiplication table from 1 to 10. The outer loop iterates over 'i' (1 to 10), and the inner loop iterates over 'j' (0 to 10). The result is printed as $i \times j = \text{value}$ on each line, followed by a blank line after each row.

PROGRAMAÇÃO ORIENTADA A OBJETOS

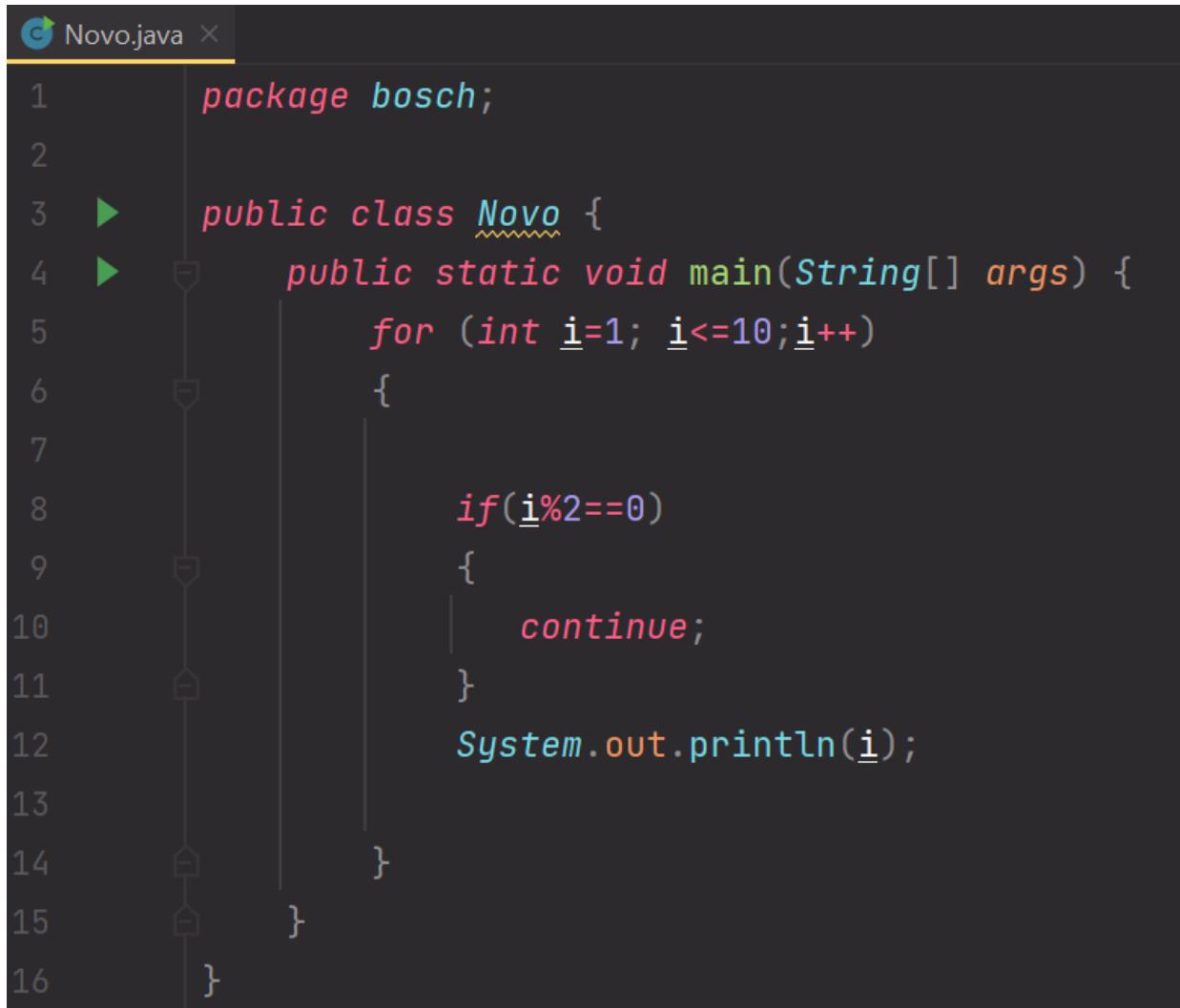
Java – Laços de Repetição



```
Novo.java
1 package bosch;
2
3 public class Novo {
4     public static void main(String[] args) {
5         for (int i=1; i<=10;i++)
6         {
7             if (i==5)
8             {
9                 break;
10            }
11            System.out.println(i);
12        }
13    }
14 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Java – Laços de Repetição



```
Novo.java
1 package bosch;
2
3 public class Novo {
4     public static void main(String[] args) {
5         for (int i=1; i<=10;i++)
6         {
7
8             if(i%2==0)
9             {
10                 continue;
11             }
12             System.out.println(i);
13
14         }
15     }
16 }
```

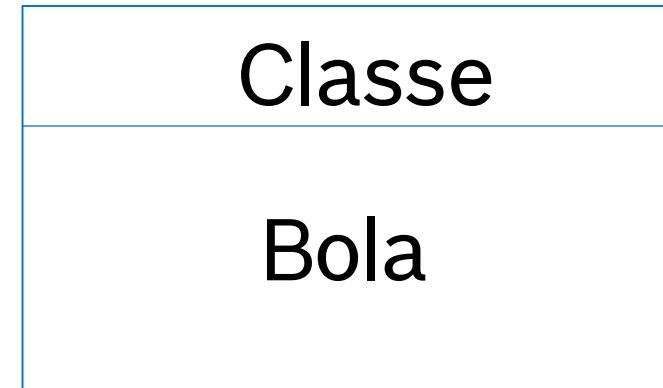
PROGRAMAÇÃO ORIENTADA A OBJETOS

Java – Laços de Repetição

```
Novo.java x Main.java x
1 package bosch;
2
3 import java.util.InputMismatchException;
4 import java.util.Scanner;
5
6 public class Main {
7
8     public static void main(String[] args) {
9         Scanner ler = new Scanner(System.in);
10        int numero;
11        while (true){
12
13            try {
14                System.out.print("Digite um numero: ");
15                numero = ler.nextInt();
16                if (numero<20)
17                {
18                    continue;
19                }
20                break;
21            } catch (InputMismatchException e) {
22                System.out.println("Ops... você digitou caracteres. Precisamos que digite apenas números.");
23            }
24            ler.nextLine();
25        }
26        System.out.println(numero);
27    }
28}
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Classes



Uma classe abstrai um conjunto de objetos semelhantes

Quais características em comum entre esses objetos?

PROGRAMAÇÃO ORIENTADA A OBJETOS

Classes



Class
e



Instância
(Objeto)

PROGRAMAÇÃO ORIENTADA A OBJETOS

Classe

Class
e



Objeto

PROGRAMAÇÃO ORIENTADA A OBJETOS

Classe

Uma classe pode ser considerada um molde através qual objetos de um certo tipo são criados.

Abstração (modelo) de um conjuntos de objetos com características semelhantes.

Mesma propriedades e comportamentos.

PROGRAMAÇÃO ORIENTADA A OBJETOS

Classe



- Uma classe pode ser considerada um molde através do qual objetos de um certo tipo são criados.
- Abstração (modelo) de um conjuntos de objetos com características semelhantes.
- Mesma propriedades e comportamentos.
- Existem muitas unidades de bicicleta, mas todas dotadas das mesmas propriedades e funcionalidades, cor, tamanho das rodas, pedalar, freiar...)

PROGRAMAÇÃO ORIENTADA A OBJETOS

Classe



Classe: Cachorro
Instância: Cachorro do João

Atributos	Comportamentos
Nome: Cometa	Latir
Raça: Golden Retriever	Deitar
Idade: 4 anos	Correr
Cor: Dourado	Brincar
Peso: 40 kg	Dar a pata
Fome: Sim	

PROGRAMAÇÃO ORIENTADA A OBJETOS

Classe



Classe: Ônibus

Instância: Ônibus para casa

Atributos	Comportamentos
Linha: 685	Acelerar
Bairro: Centro	Frear
Motorista: Jair	Abrir porta
Cor: Azul/Cinza	Ligar ar-condicionado
Marca: Mercedez	
Cheio: Sim	

PROGRAMAÇÃO ORIENTADA A OBJETOS

Classe

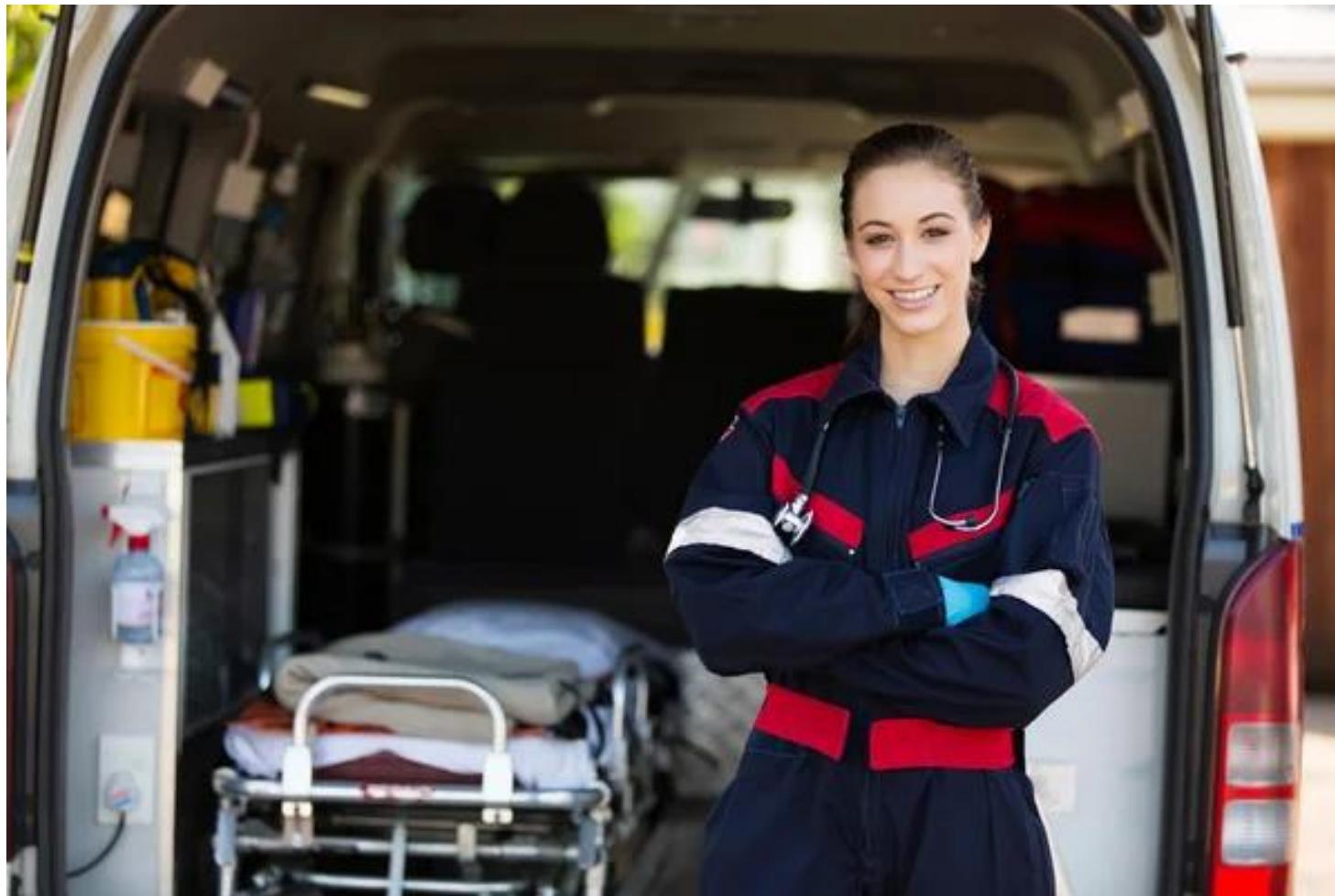


Classe: Avião
Instância: Avião da viagem

Atributos	Comportamentos
Modelo: Airbus	Decolar
Companhia: Azul	Pousar
Origem: Campinas	Voar
Destino: Fortaleza	Manobrar
Capacidade: 300	
Cheio: Sim	

PROGRAMAÇÃO ORIENTADA A OBJETOS

Classe



Tipo: Paramédico

Instância: Minha Socorrista

Atributos	Comportamentos
Nome: Beatriz	Ressuscitação
Idade: 22 anos	Desfibrilar
Sexo: Feminino	Aplicar injeções
Formação: Nível 3	
Instituição: Samu	
Missões: 200	

PROGRAMAÇÃO ORIENTADA A OBJETOS

Classe



Classe: Jogador de Basquete
Instância: Jogador Favorito

Atributos	Comportamentos
Nome: LeBron James	Driblar
Idade: 36 anos	Arremessar
Sexo: Masculino	Fazer Filme
Time: Lakers	
Altura: 2,06m	
Salário: 41,18 milhões USD	

PROGRAMAÇÃO ORIENTADA A OBJETOS

Classe



Classe: Bombeiro

Instância: Bombeiro da Cidade

Atributos	Comportamentos
Nome: Tony	Resgate em altura
Idade: 29 anos	Apagar incêndios
Sexo: Masculino	Resgate em acidente de transito
Posição: Tenente	
Instituição: Corpo de Bombeiros	
Medalhas: 200	

PROGRAMAÇÃO ORIENTADA A OBJETOS

Classe



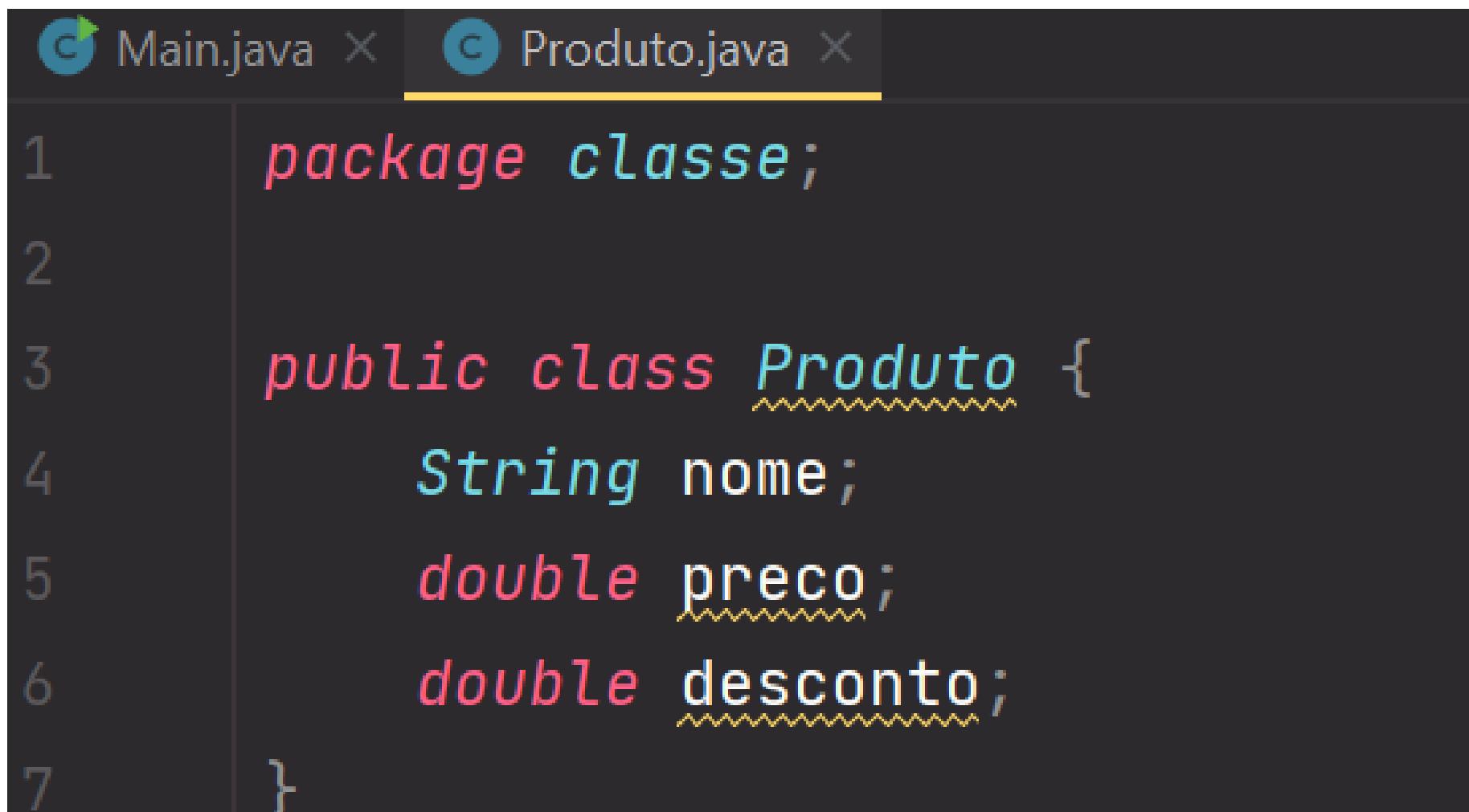
Classe: Carro

Identidade: Carro dos Sonhos

Atributos	Comportamentos
Modelo: Enzo	Acelerar
Marca: Ferrari	Frear
Velocidade Máxima: 320 km/h	Abrir Portas
Cor: Vermelho	

PROGRAMAÇÃO ORIENTADA A OBJETOS

Classe



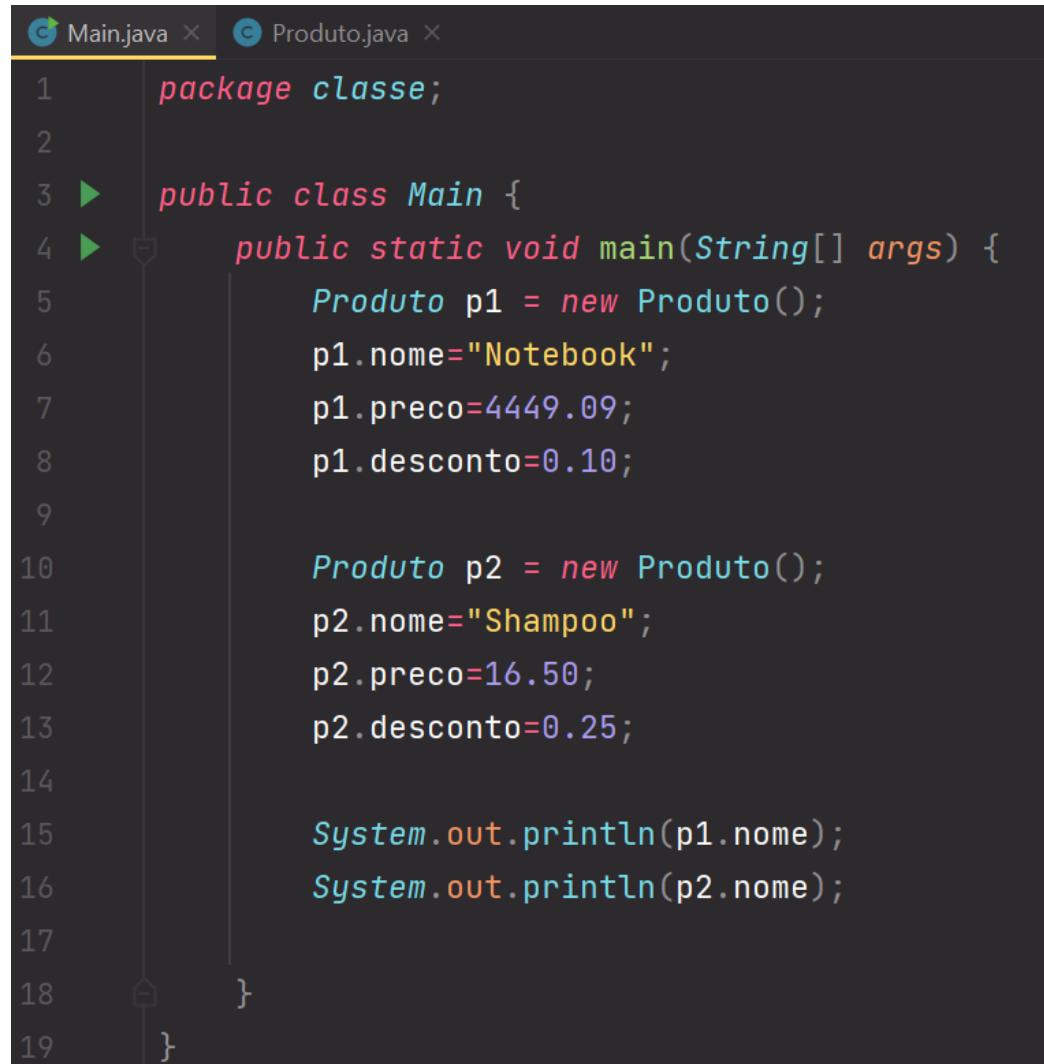
Main.java

Produto.java

```
1 package classe;
2
3 public class Produto {
4     String nome;
5     double preco;
6     double desconto;
7 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Classe



The image shows a screenshot of a Java code editor with two tabs: "Main.java" and "Produto.java". The "Main.java" tab is active, displaying the following code:

```
1 package classe;
2
3 public class Main {
4     public static void main(String[] args) {
5         Produto p1 = new Produto();
6         p1.nome="Notebook";
7         p1.preco=4449.09;
8         p1.desconto=0.10;
9
10        Produto p2 = new Produto();
11        p2.nome="Shampoo";
12        p2.preco=16.50;
13        p2.desconto=0.25;
14
15        System.out.println(p1.nome);
16        System.out.println(p2.nome);
17
18    }
19 }
```

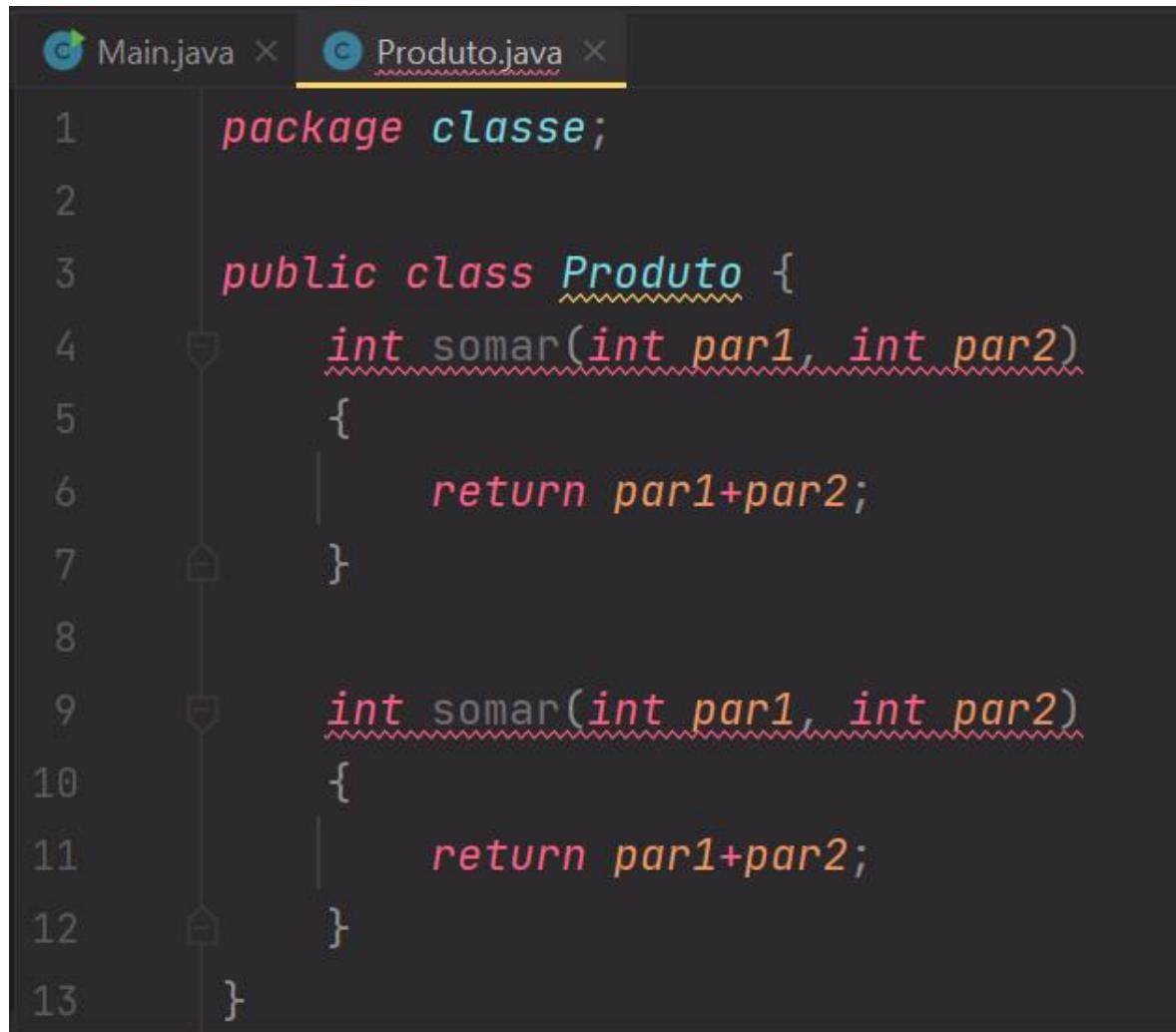
PROGRAMAÇÃO ORIENTADA A OBJETOS

Métodos

```
Main.java × Produto.java ×
1 package classe;
2
3 public class Produto {
4     int somar(int par1, int par2)
5     {
6         return par1+par2;
7     }
8
9     int somar(int par1, int par2, int par3)
10    {
11        return par1+par2+par3;
12    }
13
14 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

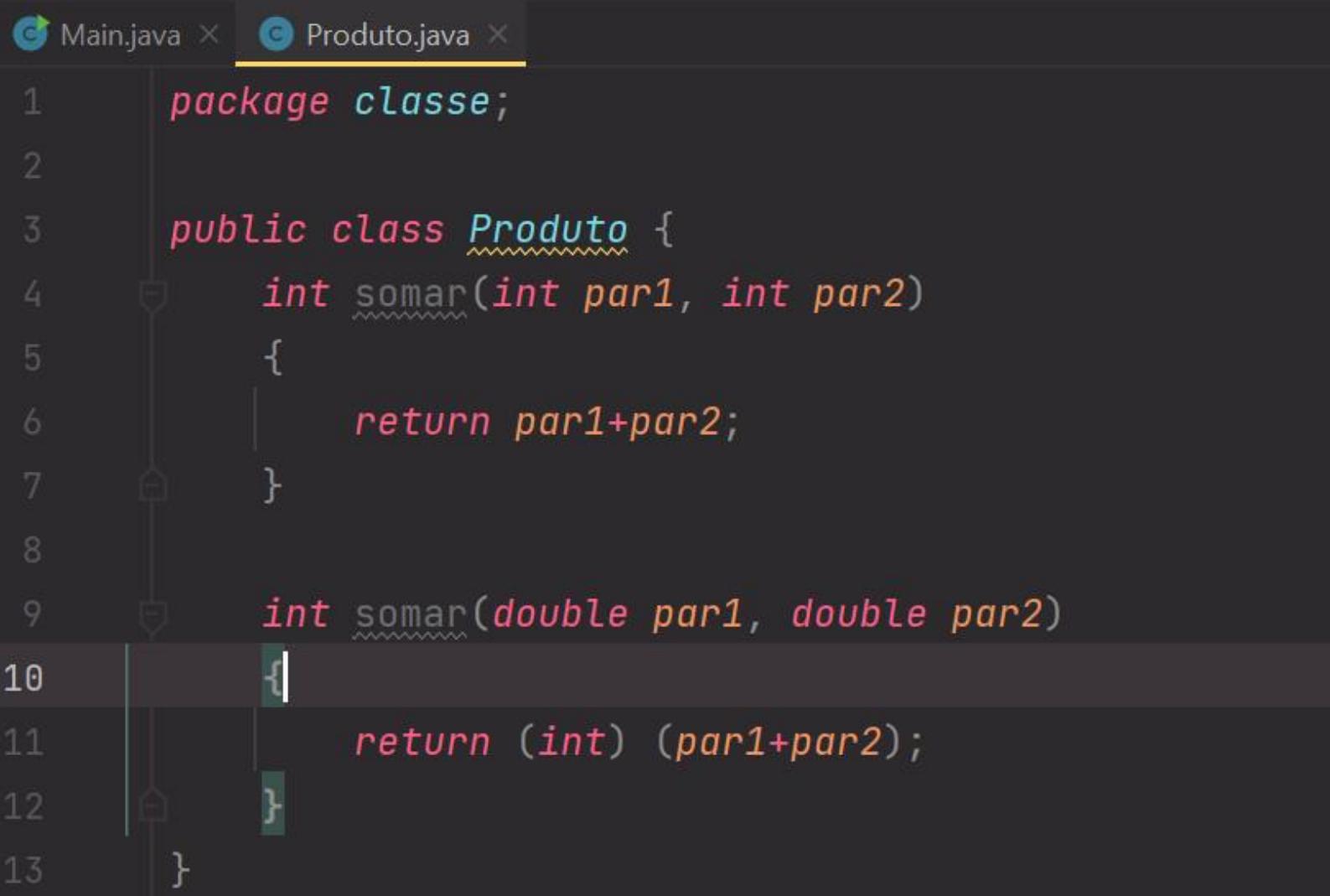
Métodos



```
Main.java x Produto.java x
1 package classe;
2
3 public class Produto {
4     int somar(int par1, int par2)
5     {
6         return par1+par2;
7     }
8
9     int somar(int par1, int par2)
10    {
11        return par1+par2;
12    }
13}
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Métodos



```
>Main.java ×  Produto.java ×
1 package classe;
2
3 public class Produto {
4     int somar(int par1, int par2)
5     {
6         return par1+par2;
7     }
8
9     int somar(double par1, double par2)
10    {
11        return (int) (par1+par2);
12    }
13 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Métodos

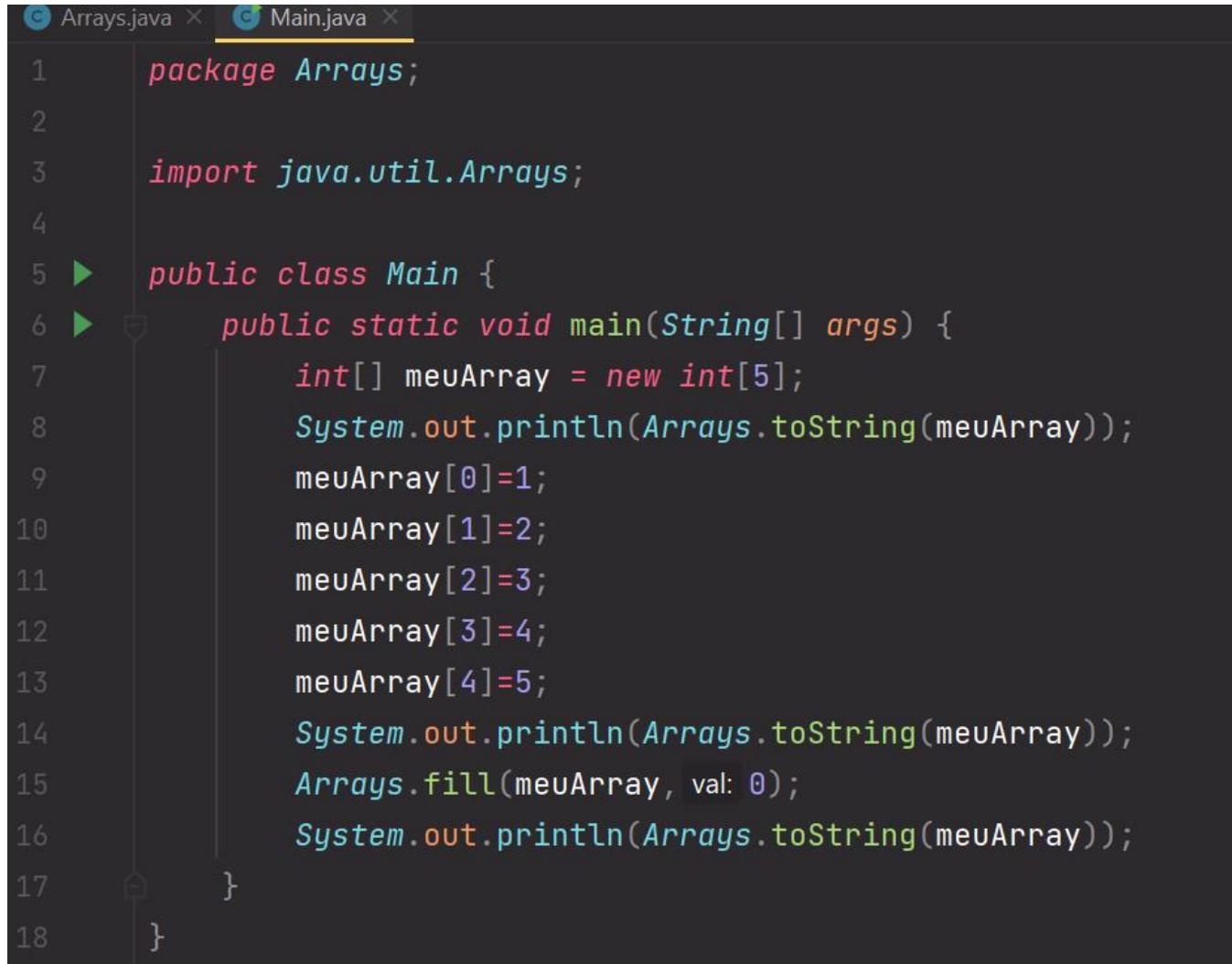


The screenshot shows a Java code editor with two tabs: 'Main.java' and 'Produto.java'. The 'Main.java' tab is active, displaying the following code:

```
1 package classe;
2
3 public class Main {
4     public static void main(String[] args) {
5         Produto p1 = new Produto();
6         p1.nome="Notebook";
7         p1.preco=4449.09;
8         p1.desconto=0.10;
9
10        Produto p2 = new Produto();
11        p2.nome="Shampoo";
12        p2.preco=16.50;
13        p2.desconto=0.25;
14
15        System.out.println(p1.precoComDesconto());
16        System.out.println(p1.precoComDescontoExtra(0.10));
17
18    }
19 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

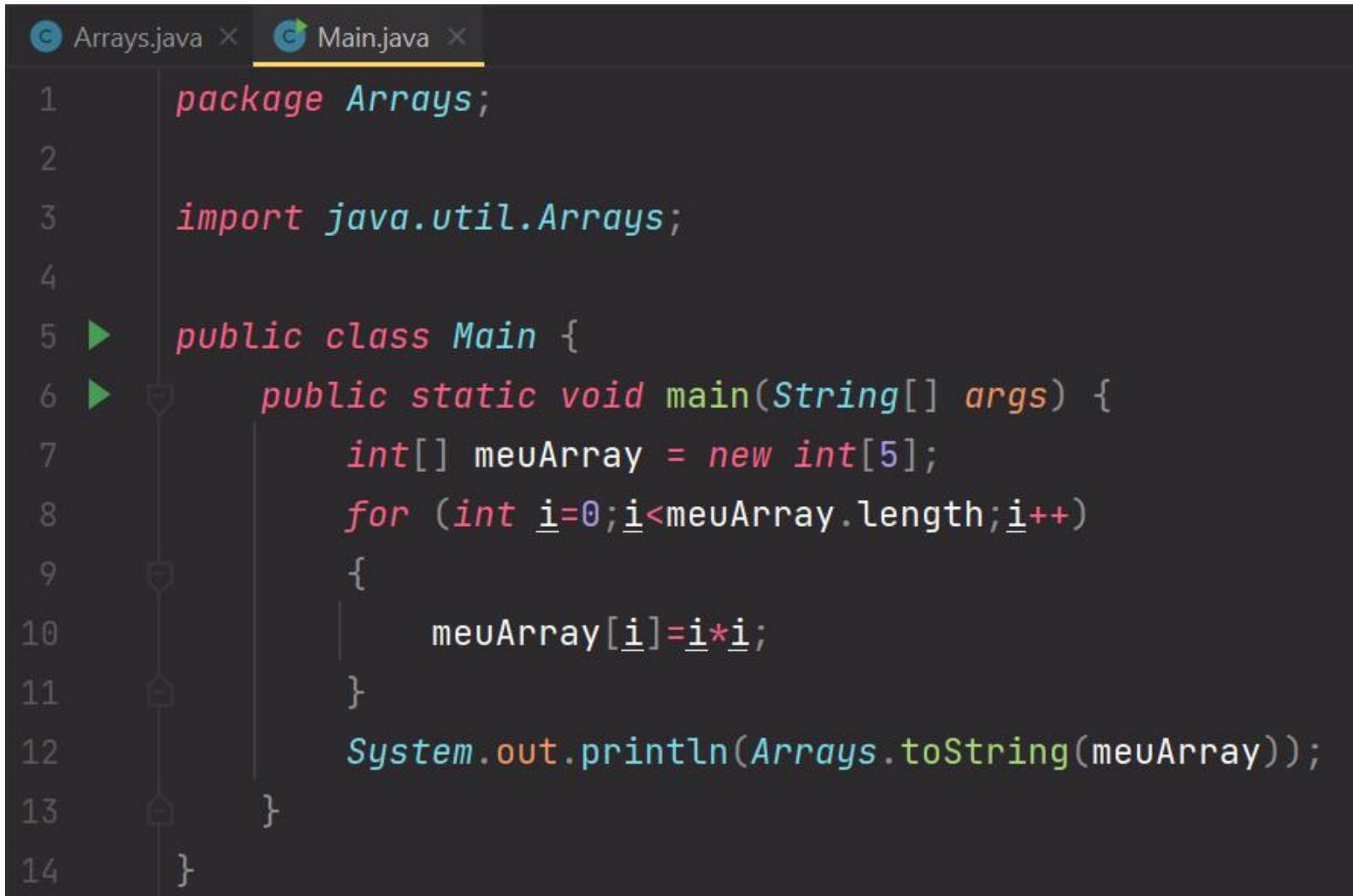
Array



```
Arrays.java X Main.java X
1 package Arrays;
2
3 import java.util.Arrays;
4
5 public class Main {
6     public static void main(String[] args) {
7         int[] meuArray = new int[5];
8         System.out.println(Arrays.toString(meuArray));
9         meuArray[0]=1;
10        meuArray[1]=2;
11        meuArray[2]=3;
12        meuArray[3]=4;
13        meuArray[4]=5;
14        System.out.println(Arrays.toString(meuArray));
15        Arrays.fill(meuArray, val: 0);
16        System.out.println(Arrays.toString(meuArray));
17    }
18 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

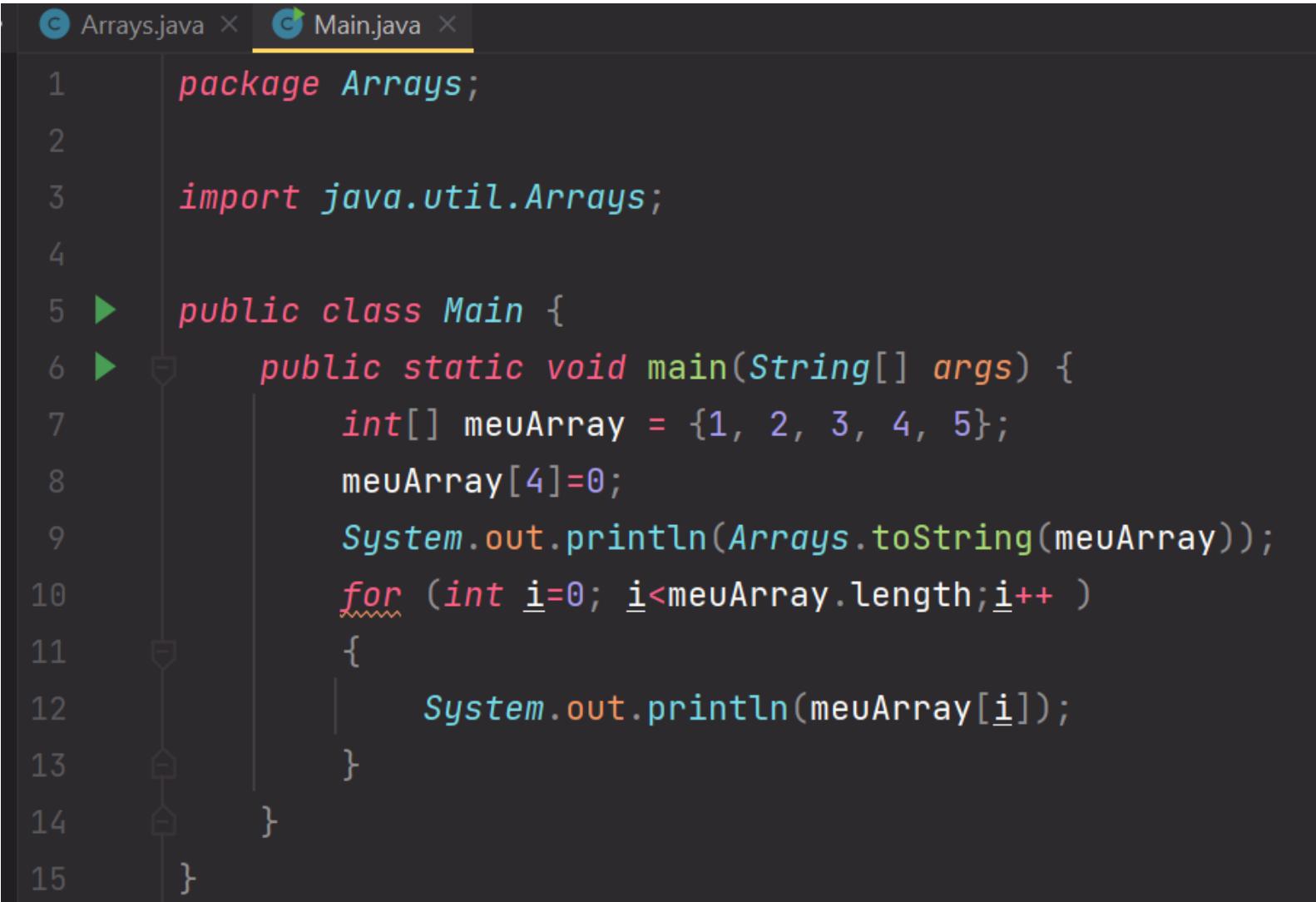
Array



```
Arrays.java × Main.java ×
1 package Arrays;
2
3 import java.util.Arrays;
4
5 ► public class Main {
6 ►     ► public static void main(String[] args) {
7         int[] meuArray = new int[5];
8         for (int i=0;i<meuArray.length;i++)
9         {
10             meuArray[i]=i*i;
11         }
12         System.out.println(Arrays.toString(meuArray));
13     }
14 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

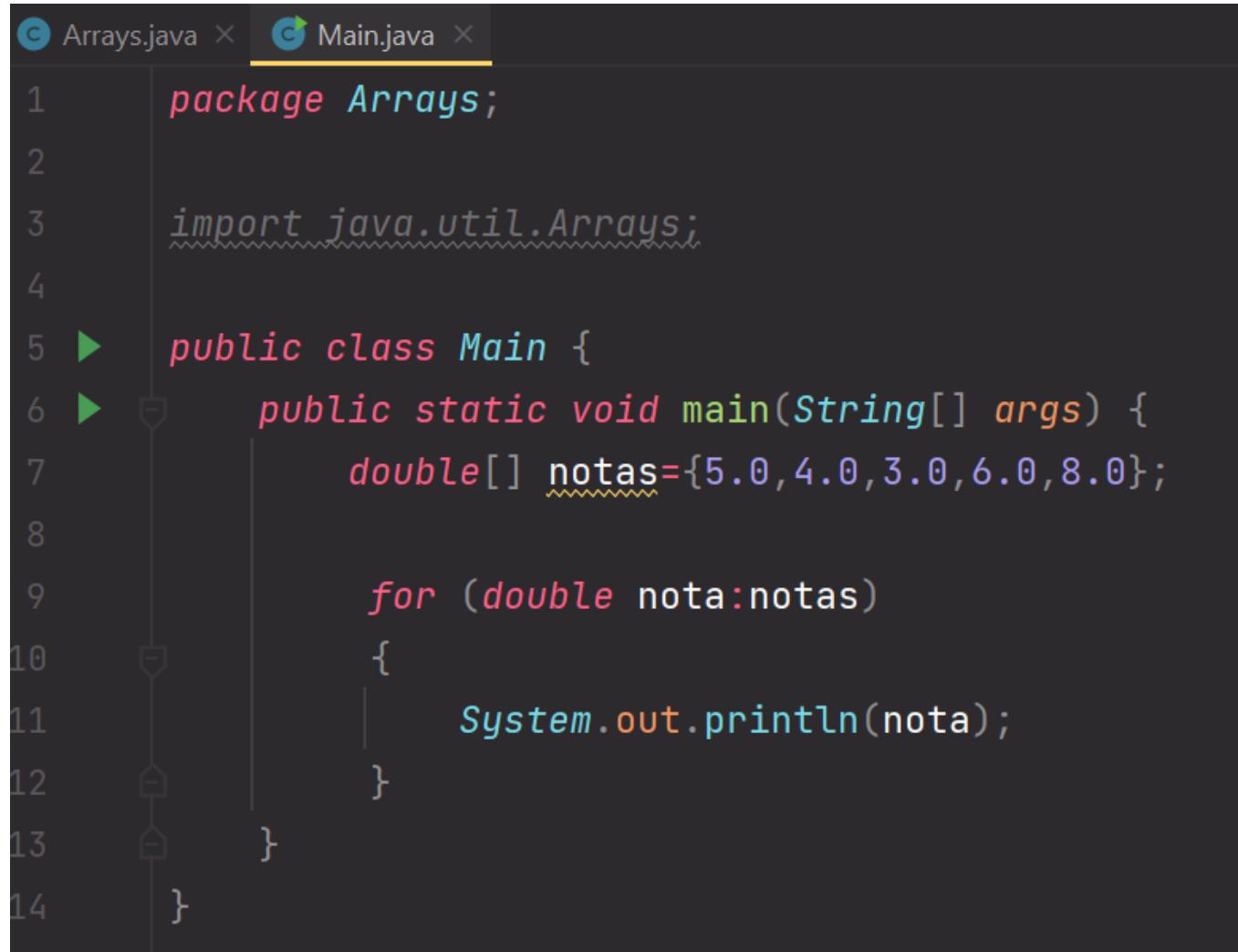
Array



```
Arrays.java x Main.java x
1 package Arrays;
2
3 import java.util.Arrays;
4
5 public class Main {
6     public static void main(String[] args) {
7         int[] meuArray = {1, 2, 3, 4, 5};
8         meuArray[4]=0;
9         System.out.println(Arrays.toString(meuArray));
10        for (int i=0; i<meuArray.length;i++)
11        {
12            System.out.println(meuArray[i]);
13        }
14    }
15 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

ForEach



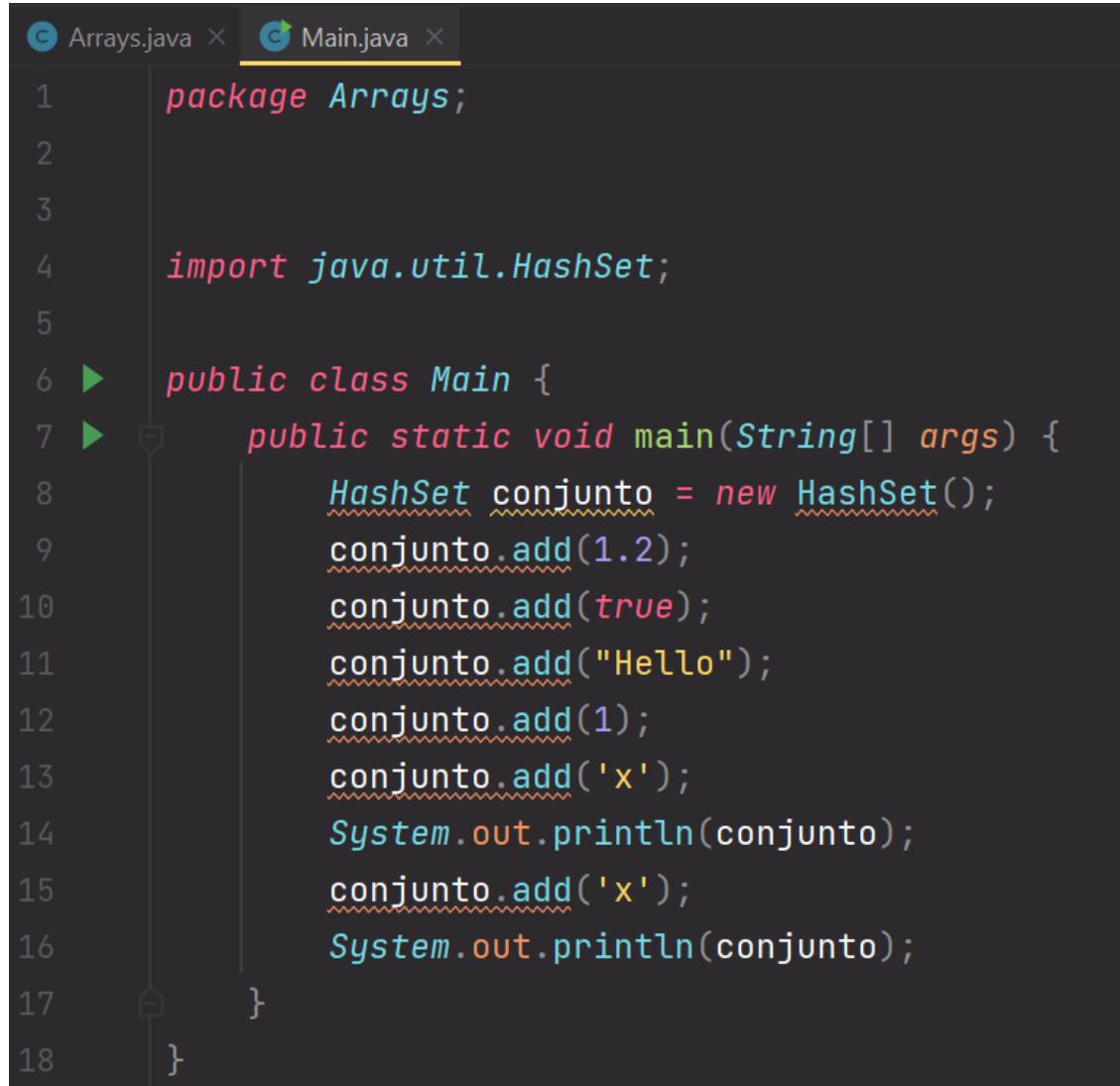
The screenshot shows a Java code editor with two tabs: 'Arrays.java' and 'Main.java'. The 'Main.java' tab is active, displaying the following code:

```
1 package Arrays;
2
3 import java.util.Arrays;
4
5 public class Main {
6     public static void main(String[] args) {
7         double[] notas={5.0,4.0,3.0,6.0,8.0};
8
9         for (double nota:notas)
10        {
11             System.out.println(nota);
12         }
13     }
14 }
```

The code defines a package named 'Arrays' and a class 'Main'. The 'main' method contains a 'for-each' loop that iterates over an array of doubles named 'notas', printing each value to the console.

PROGRAMAÇÃO ORIENTADA A OBJETOS

Sets



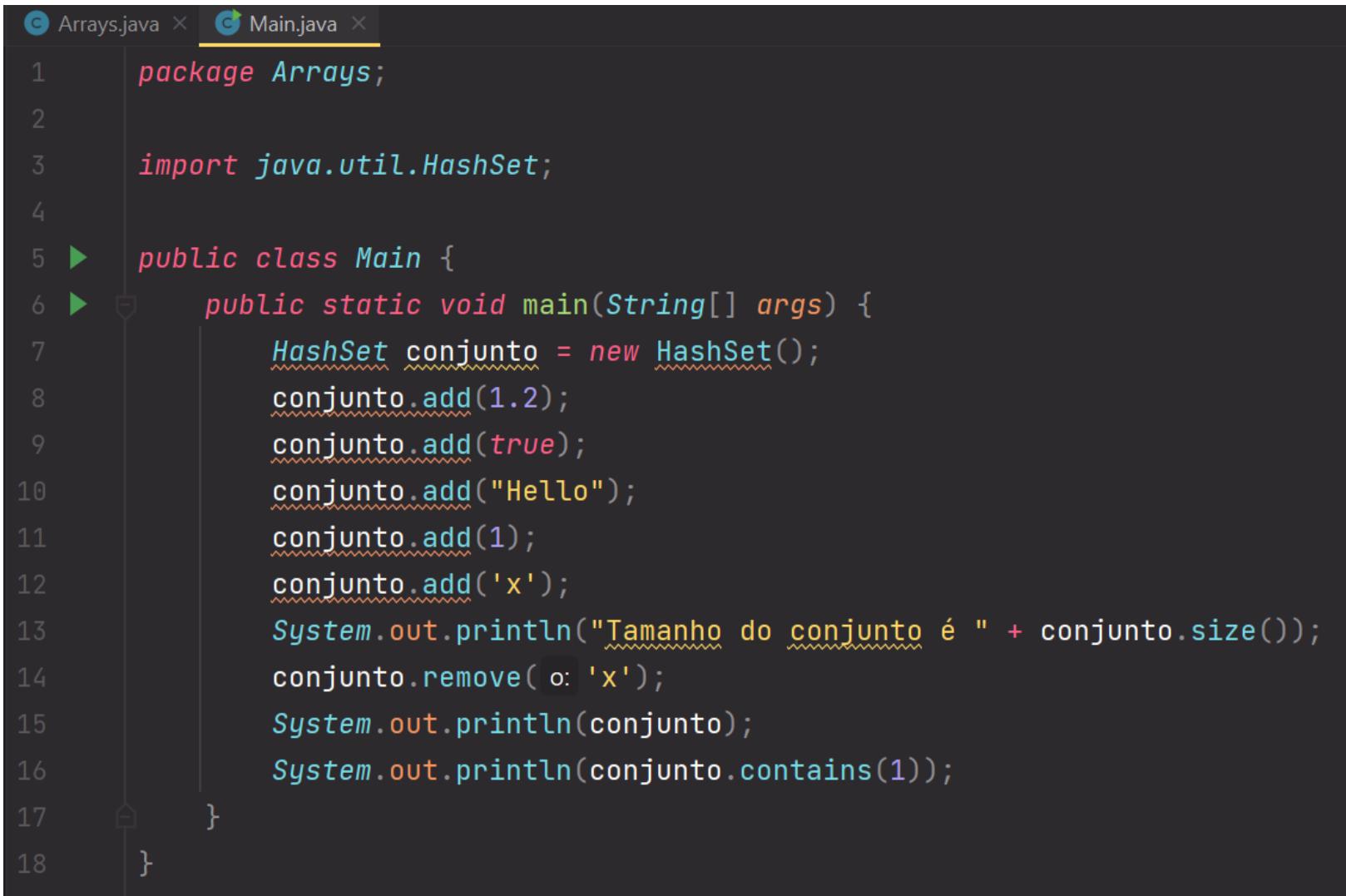
The image shows a screenshot of a Java code editor with two tabs: "Main.java" and "Arrays.java". The "Main.java" tab is active, displaying the following code:

```
1 package Arrays;
2
3
4 import java.util.HashSet;
5
6 public class Main {
7     public static void main(String[] args) {
8         HashSet conjunto = new HashSet();
9         conjunto.add(1.2);
10        conjunto.add(true);
11        conjunto.add("Hello");
12        conjunto.add(1);
13        conjunto.add('x');
14        System.out.println(conjunto);
15        conjunto.add('x');
16        System.out.println(conjunto);
17    }
18 }
```

The code demonstrates the use of a HashSet to store various objects of different types. The output of the program will show that the HashSet contains unique elements, even if they are of different types or have the same value.

PROGRAMAÇÃO ORIENTADA A OBJETOS

Sets

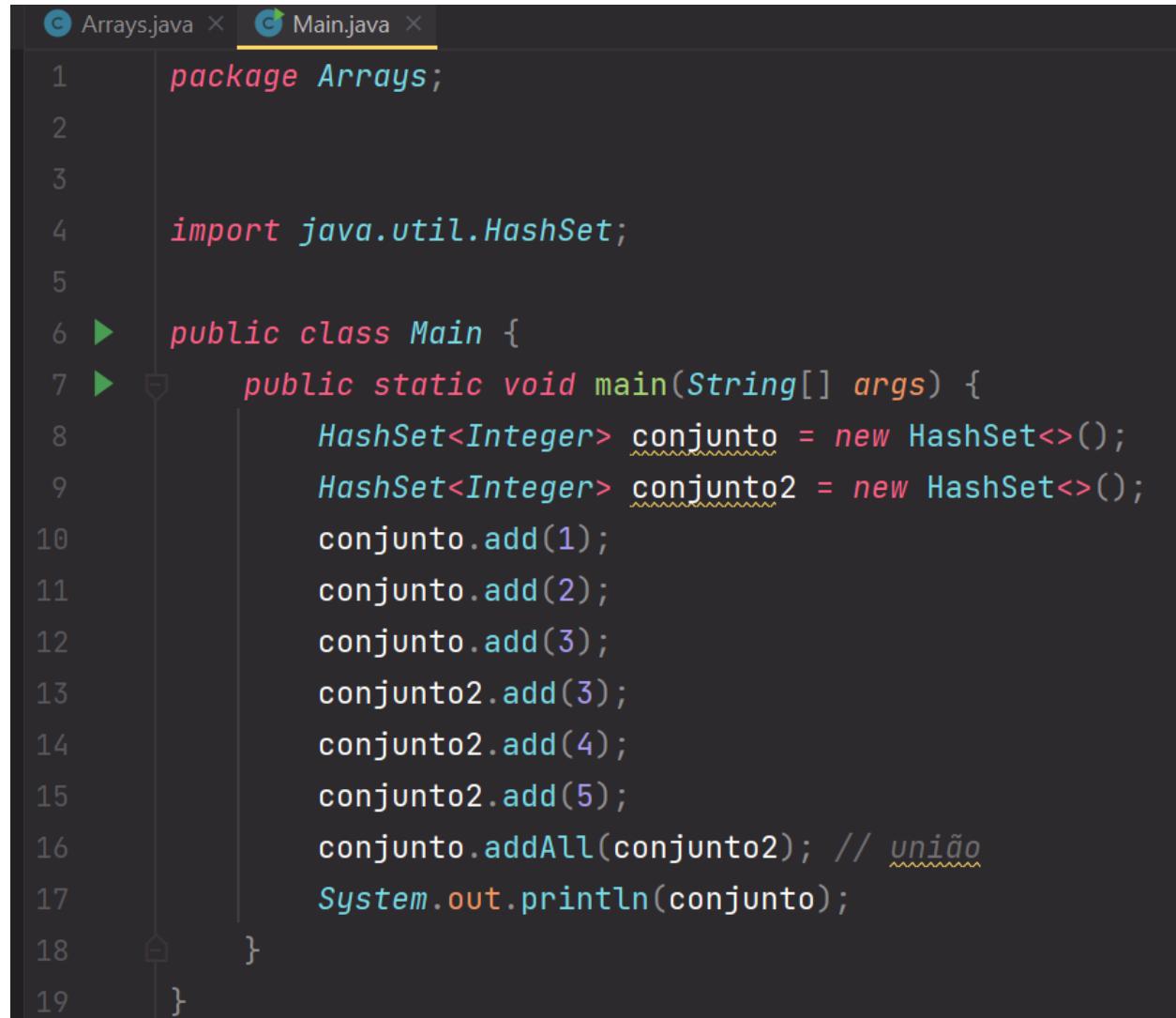


The screenshot shows a Java code editor with two tabs: 'Arrays.java' and 'Main.java'. The 'Main.java' tab is active, displaying the following code:

```
1 package Arrays;
2
3 import java.util.HashSet;
4
5 public class Main {
6     public static void main(String[] args) {
7         HashSet conjunto = new HashSet();
8         conjunto.add(1.2);
9         conjunto.add(true);
10        conjunto.add("Hello");
11        conjunto.add(1);
12        conjunto.add('x');
13        System.out.println("Tamanho do conjunto é " + conjunto.size());
14        conjunto.remove(o: 'x');
15        System.out.println(conjunto);
16        System.out.println(conjunto.contains(1));
17    }
18 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Sets

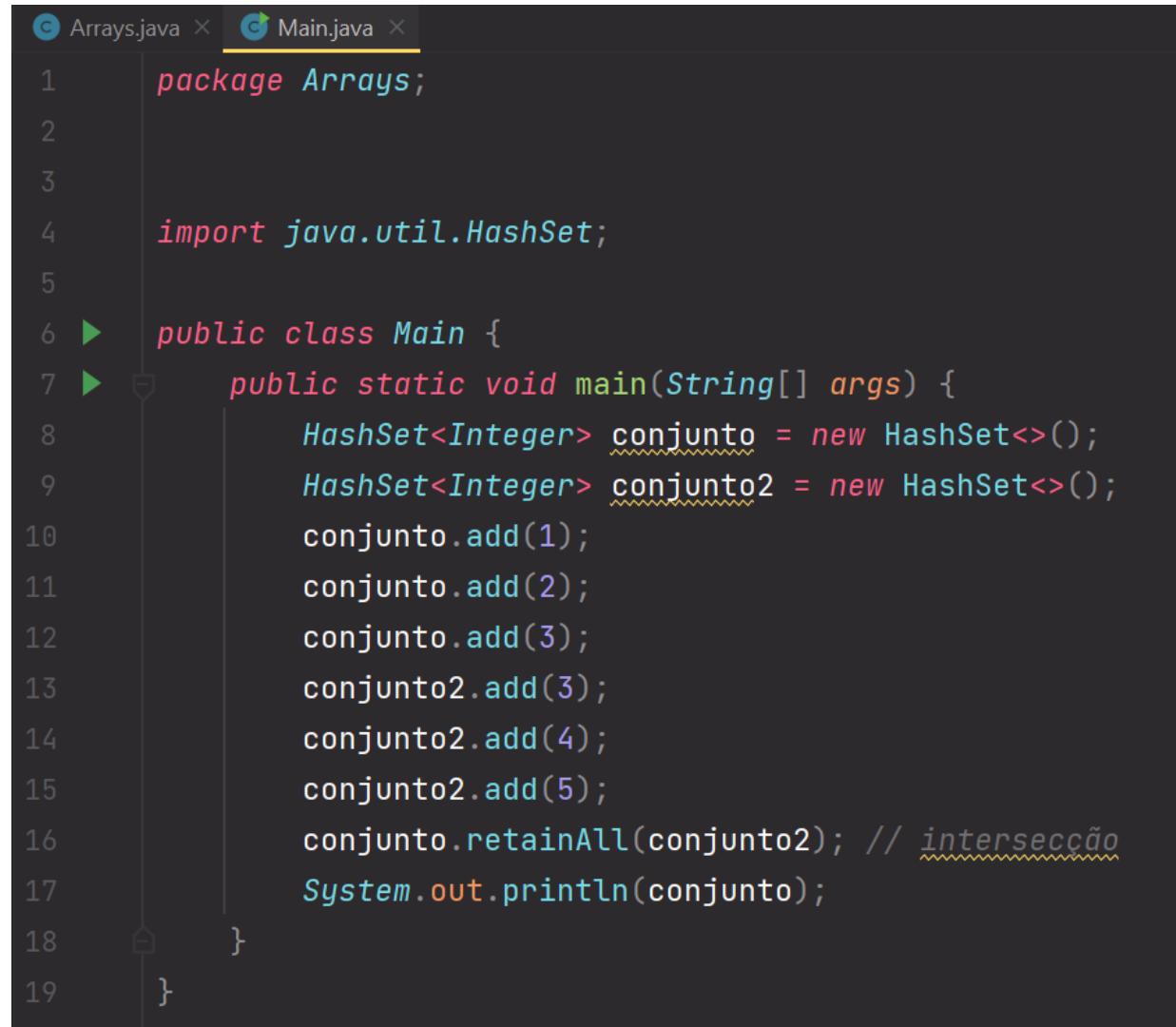


The image shows a screenshot of a Java code editor with two tabs: "Arrays.java" and "Main.java". The "Main.java" tab is active, displaying the following code:

```
1 package Arrays;
2
3
4 import java.util.HashSet;
5
6 public class Main {
7     public static void main(String[] args) {
8         HashSet<Integer> conjunto = new HashSet<>();
9         HashSet<Integer> conjunto2 = new HashSet<>();
10        conjunto.add(1);
11        conjunto.add(2);
12        conjunto.add(3);
13        conjunto2.add(3);
14        conjunto2.add(4);
15        conjunto2.add(5);
16        conjunto.addAll(conjunto2); // união
17        System.out.println(conjunto);
18    }
19 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Sets



```
Arrays.java × Main.java ×
1 package Arrays;
2
3
4 import java.util.HashSet;
5
6 ► public class Main {
7 ►   ►   public static void main(String[] args) {
8       HashSet<Integer> conjunto = new HashSet<>();
9       HashSet<Integer> conjunto2 = new HashSet<>();
10      conjunto.add(1);
11      conjunto.add(2);
12      conjunto.add(3);
13      conjunto2.add(3);
14      conjunto2.add(4);
15      conjunto2.add(5);
16      conjunto.retainAll(conjunto2); // interseccão
17      System.out.println(conjunto);
18   }
19 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Sets



```
Arrays.java × Main.java ×
1 package Arrays;
2
3 import java.util.HashSet;
4
5 ► public class Main {
6 ►     public static void main(String[] args) {
7         HashSet <String> conjunto = new HashSet<>();
8         conjunto.add("João");
9         conjunto.add("Maria");
10        conjunto.add("José");
11
12        for (String nome:conjunto) {
13            System.out.println(nome);
14        }
15    }
16}
```

The image shows a screenshot of a Java code editor with two tabs: "Arrays.java" and "Main.java". The "Main.java" tab is active, displaying the following Java code. The code defines a class named Main with a main method. Inside the main method, a HashSet of strings is created and populated with three names: João, Maria, and José. A for loop then iterates over the elements in the set and prints each name to the console using System.out.println.

PROGRAMAÇÃO ORIENTADA A OBJETOS

Sets

```
5     public class Usuário {
6         String nome;
7         int idade;
8
9
10        Usuário(String nome, int idade) {
11            this.nome = nome;
12            this.idade = idade;
13        }
14
15        public int getIdade() {
16            return idade;
17        }
18    }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Sets

```
20      @Override  
21  ⚡  public int hashCode() {  
22      |   return Objects.hash(nome);  
23  ⚡ }  
24  
25      @Override  
26  ⚡  public String toString() {  
27      |   return this.nome;  
28  ⚡ }  
29 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

ArrayList



The screenshot shows a Java code editor with two tabs: "Main.java" and "Usuario.java". The "Main.java" tab is active, displaying the following code:

```
1 package Arrays;
2
3 import java.util.ArrayList;
4 import java.util.Comparator;
5
6
7 public class Main {
8     public static void main(String[] args) {
9         ArrayList<Usuario> listaDeUsuario = new ArrayList<>();
10        Usuario u1= new Usuario( nome: "João", idade: 16);
11        listaDeUsuario.add(u1);
12        listaDeUsuario.add(new Usuario( nome: "Maria", idade: 17));
13        listaDeUsuario.add(new Usuario( nome: "José", idade: 18));
14        listaDeUsuario.add(new Usuario( nome: "Pedro", idade: 20));
15        listaDeUsuario.sort(Comparator.comparingInt(Usuario::getIdade).reversed());
16        System.out.println(listaDeUsuario);
17    }
18 }
```

The "Usuario.java" tab is visible in the background, indicating it is part of the same project.

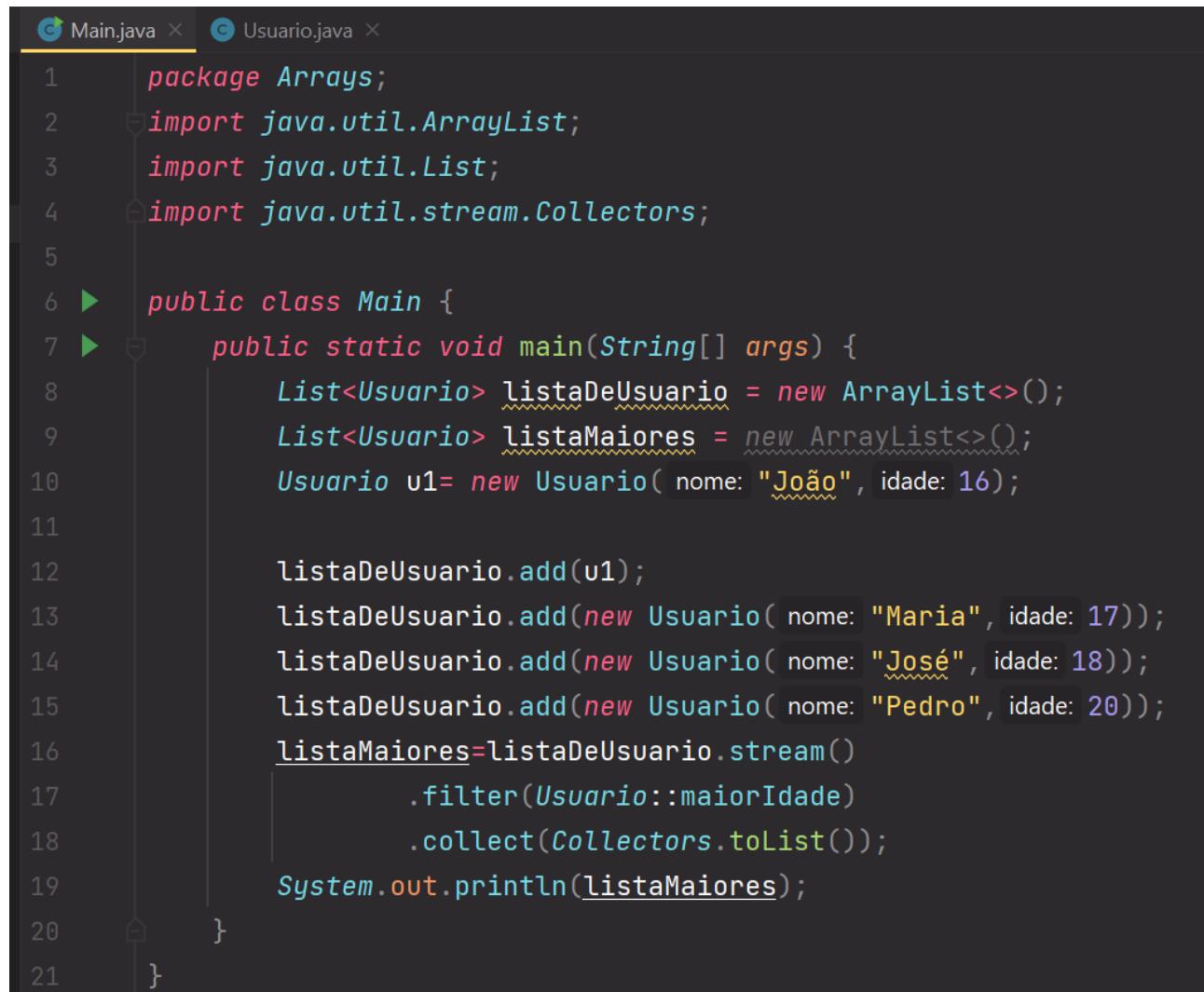
PROGRAMAÇÃO ORIENTADA A OBJETOS

ArrayList

```
1 package Arrays;  
2  
3 import java.util.ArrayList;  
4 import java.util.Comparator;  
5  
6  
7 ► public class Main {  
8 ►     public static void main(String[] args) {  
9         ArrayList<Usuario> listaDeUsuario = new ArrayList<>();  
10        Usuario u1= new Usuario( nome: "João", idade: 16);  
11        listaDeUsuario.add(u1);  
12        listaDeUsuario.add(new Usuario( nome: "Maria", idade: 17));  
13        listaDeUsuario.add(new Usuario( nome: "José", idade: 18));  
14        listaDeUsuario.add(new Usuario( nome: "Pedro", idade: 20));  
15        Usuario maisVelho = listaDeUsuario.stream().max(Comparator.  
16                    comparingInt(Usuario::getIdade)).get();  
17        System.out.println(maisVelho);  
18    }  
19 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

ArrayList



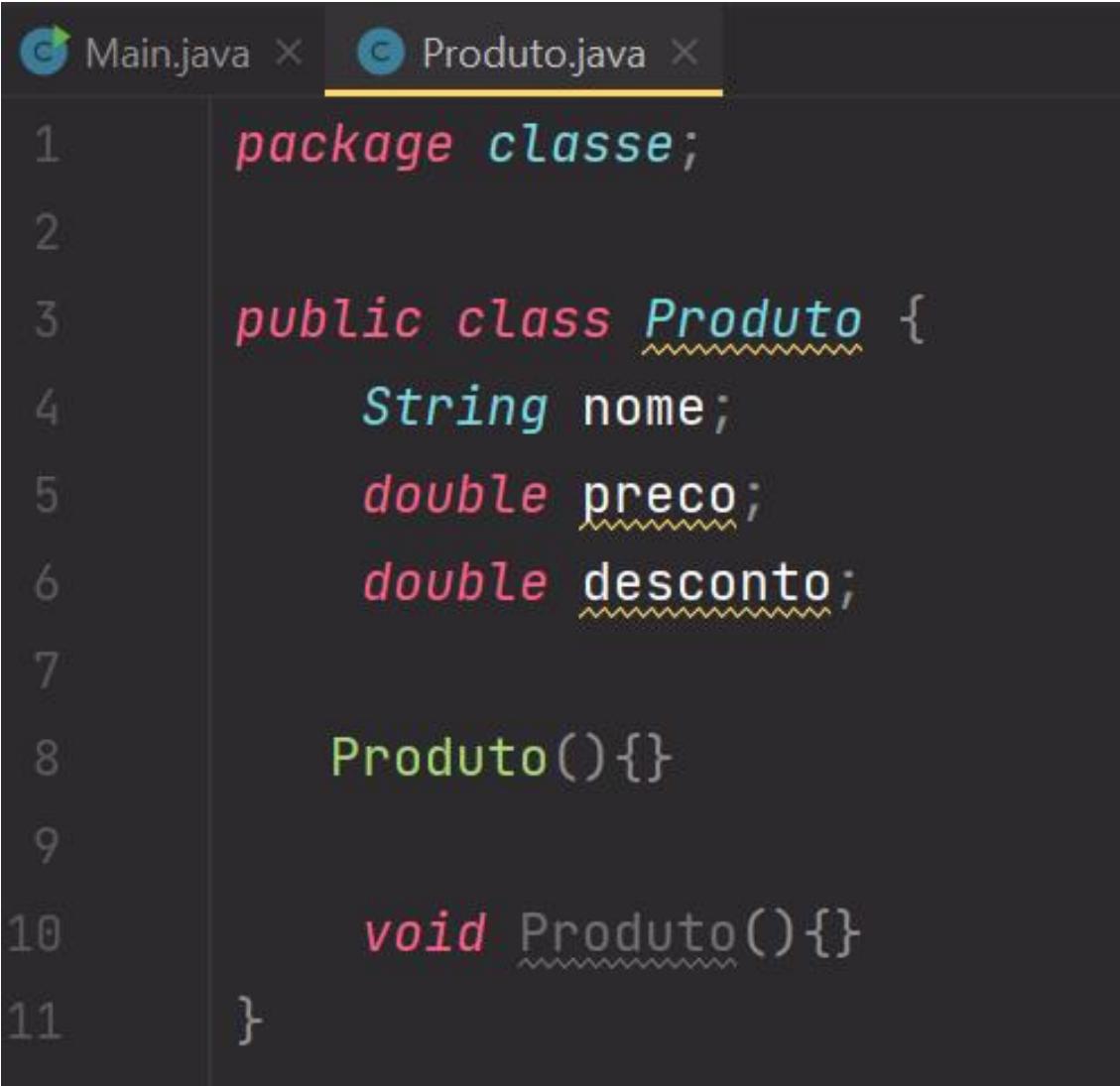
The screenshot shows a Java code editor with two tabs: "Main.java" and "Usuario.java". The "Main.java" tab is active, displaying the following code:

```
1 package Arrays;
2 import java.util.ArrayList;
3 import java.util.List;
4 import java.util.stream.Collectors;
5
6 public class Main {
7     public static void main(String[] args) {
8         List<Usuario> listaDeUsuario = new ArrayList<>();
9         List<Usuario> listaMaiores = new ArrayList<>();
10        Usuario u1= new Usuario( nome: "João", idade: 16);
11
12        listaDeUsuario.add(u1);
13        listaDeUsuario.add(new Usuario( nome: "Maria", idade: 17));
14        listaDeUsuario.add(new Usuario( nome: "José", idade: 18));
15        listaDeUsuario.add(new Usuario( nome: "Pedro", idade: 20));
16        listaMaiores=listaDeUsuario.stream()
17            .filter(Usuario::maiorIdade)
18            .collect(Collectors.toList());
19        System.out.println(listaMaiores);
20    }
21 }
```

The "Usuario.java" tab is visible but contains no code.

PROGRAMAÇÃO ORIENTADA A OBJETOS

Construtor



```
Main.java × Produto.java ×
1 package classe;
2
3 public class Produto {
4     String nome;
5     double preco;
6     double desconto;
7
8     Produto(){}
9
10    void Produto(){}
11 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Construtor

The screenshot shows a Java code editor with two tabs: 'Main.java' and 'Produto.java'. The 'Main.java' tab is active, displaying the following code:

```
1 package classe;
2
3 public class Main {
4     public static void main(String[] args) {
5         Produto p1 = new Produto();
6         p1.nome="Notebook";
7         Produto p2 = new Produto( nomeProduto: "Notebook");
8         Produto p3 = new Produto( nomeProduto: "Geladeira", precoProduto: 3000.25);
9         Produto p4 = new Produto( nomeProduto: "Celular,", precoProduto: 4000, descontoProduto: 0.10);
10
11        System.out.println(p1.nome);
12        System.out.println(p2.nome);
13        System.out.println(p3.nome);
14        System.out.println(p4.nome);
15    }
16 }
```

The 'Produto.java' tab is visible in the background, showing the definition of the `Produto` class.

PROGRAMAÇÃO ORIENTADA A OBJETOS

Construtor

- ▶ **Método especial para a inicialização de um objeto**
 - ▶ Deve ter obrigatoriamente o mesmo nome da classe
- ▶ **O que o construtor faz?**
 - ▶ Aloca memória no computador para armazenar o objeto
 - ▶ Inicializa o objeto
 - ▶ “Retorna” uma referência para o objeto



PROGRAMAÇÃO ORIENTADA A OBJETOS

Construtor

- **Construtor padrão:**

- Se não for explicitamente declarado um construtor, o compilador fornece um construtor padrão
- Sem parâmetros
- Configura as variáveis de instância para seus valores padrões (dependendo do tipo, ex. o valor 0 para inteiros)

- **Construtores podem ter parâmetros:**

- Valores a serem usados para inicializar atributos ou em geral, opções de configuração do objeto

- **Construtores podem ser sobrecarregados**

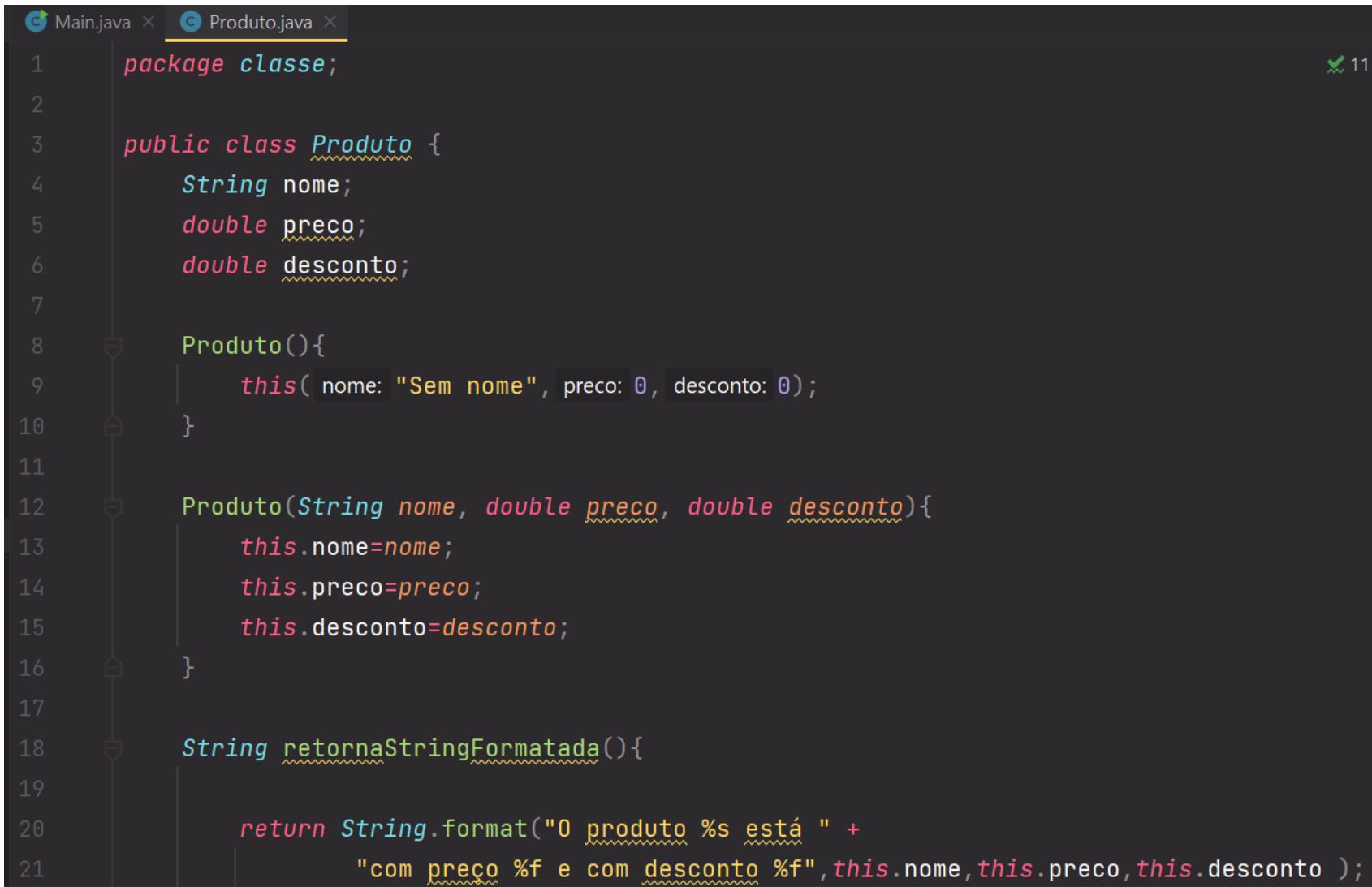
- **Pode existir mais de um**

- **Sobrecarga do construtor é permitida**

- Construtores são chamados com a palavra-chave **new**

PROGRAMAÇÃO ORIENTADA A OBJETOS

This

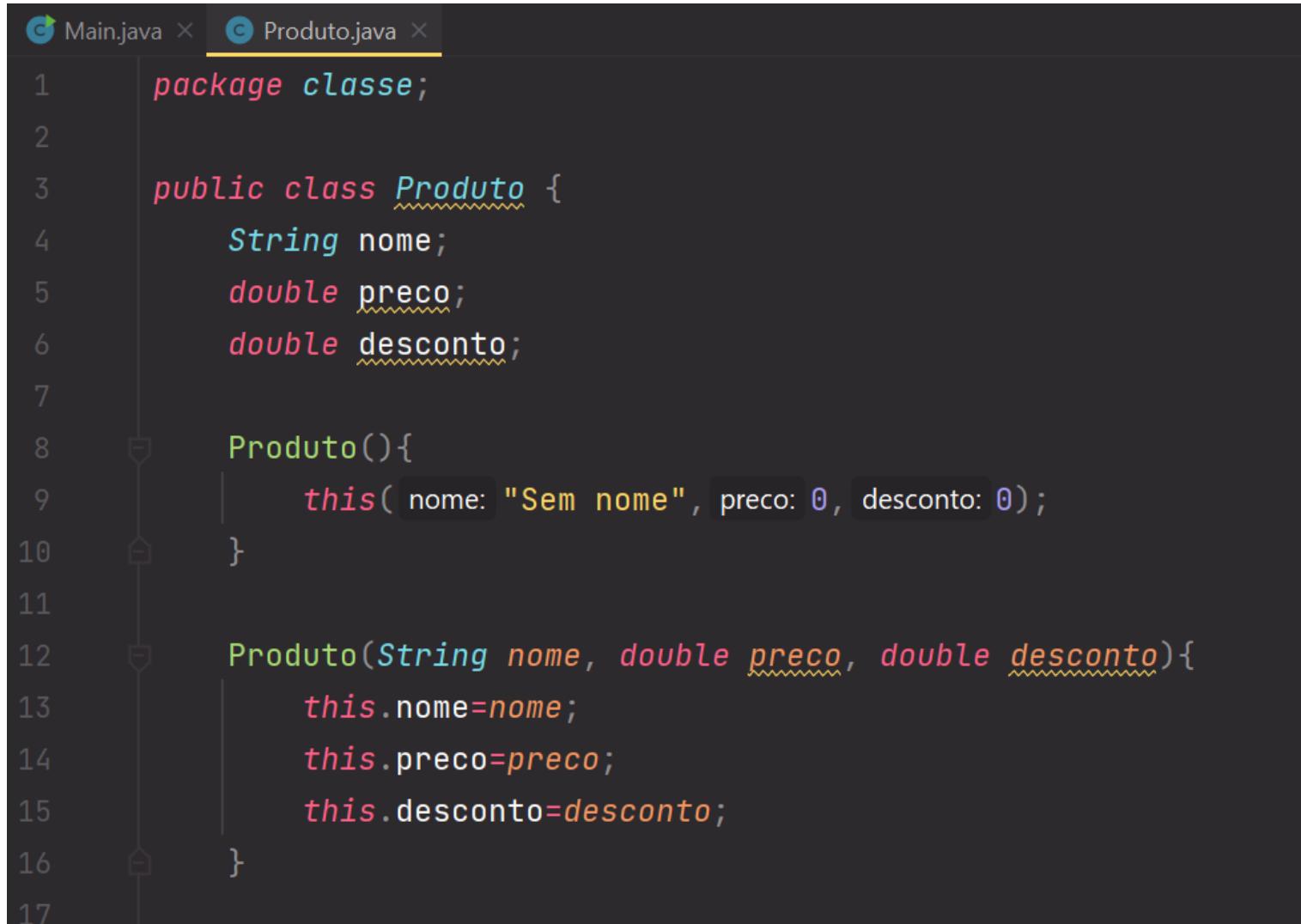


The image shows a screenshot of a Java code editor with two tabs: Main.java and Produto.java. The Produto.java tab is active, displaying the following code:

```
1 package classe;
2
3 public class Produto {
4     String nome;
5     double preco;
6     double desconto;
7
8     Produto(){
9         this( nome: "Sem nome" , preco: 0 , desconto: 0 );
10    }
11
12    Produto(String nome, double preco, double desconto){
13        this.nome=nome;
14        this.preco=preco;
15        this.desconto=desconto;
16    }
17
18    String retornaStringFormatada(){
19
20        return String.format("O produto %s está " +
21                           "com preço %f e com desconto %f",this.nome,this.preco,this.desconto );
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

This



```
Main.java x Produto.java x
1 package classe;
2
3 public class Produto {
4     String nome;
5     double preco;
6     double desconto;
7
8     Produto(){
9         this( "Sem nome" , 0 , 0 );
10    }
11
12    Produto(String nome, double preco, double desconto){
13        this.nome=nome;
14        this.preco=preco;
15        this.desconto=desconto;
16    }
17}
```

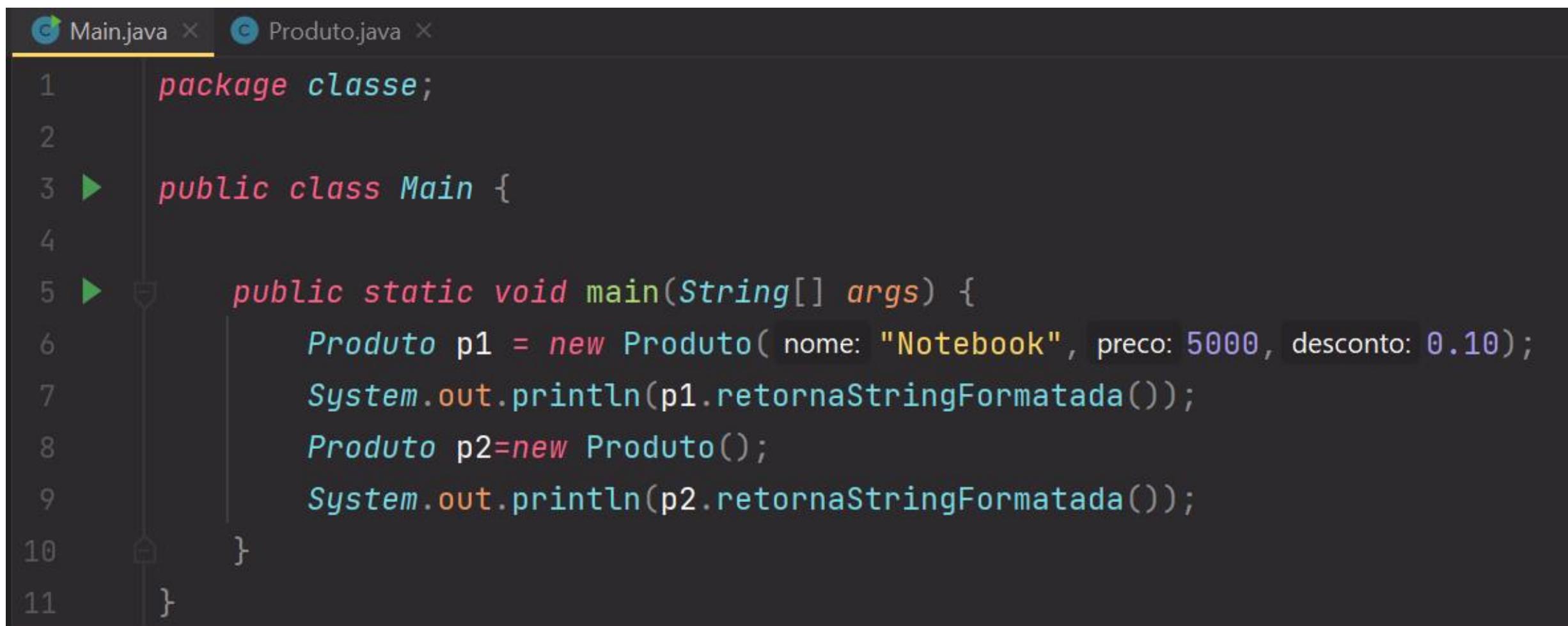
PROGRAMAÇÃO ORIENTADA A OBJETOS

This

```
    Produto(String nome, double preco, double desconto){  
        this.nome=nome;  
        this.preco=preco;  
        this.desconto=desconto;  
    }  
  
    String retornaStringFormatada(){  
  
        return String.format("O produto %s está " +  
            "com preço %f e com desconto %f",this.nome,this.preco,this.desconto );  
    }  
}
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

This



The screenshot shows a Java code editor with two tabs: 'Main.java' and 'Produto.java'. The 'Main.java' tab is active, displaying the following code:

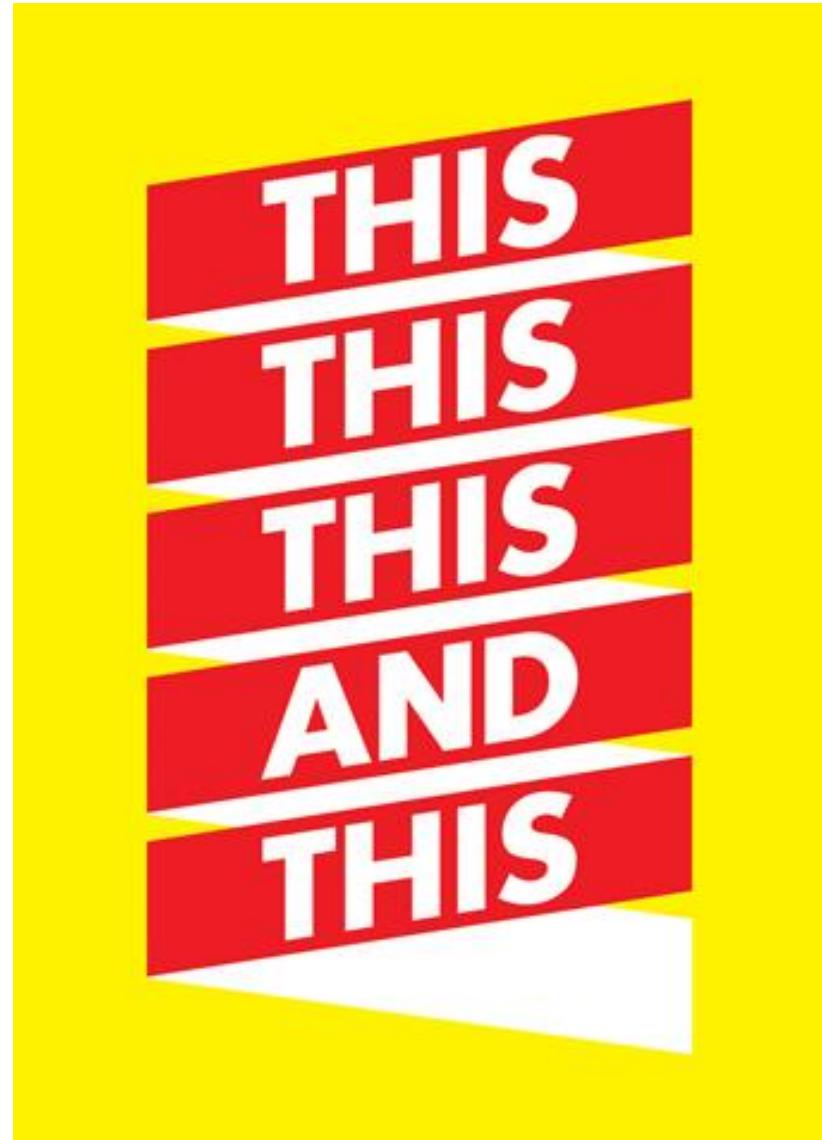
```
1 package classe;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         Produto p1 = new Produto( nome: "Notebook", preco: 5000, desconto: 0.10 );
7         System.out.println(p1.retornaStringFormatada());
8         Produto p2=new Produto();
9         System.out.println(p2.retornaStringFormatada());
10    }
11 }
```

The 'Produto.java' tab is visible but its content is not shown.

PROGRAMAÇÃO ORIENTADA A OBJETO

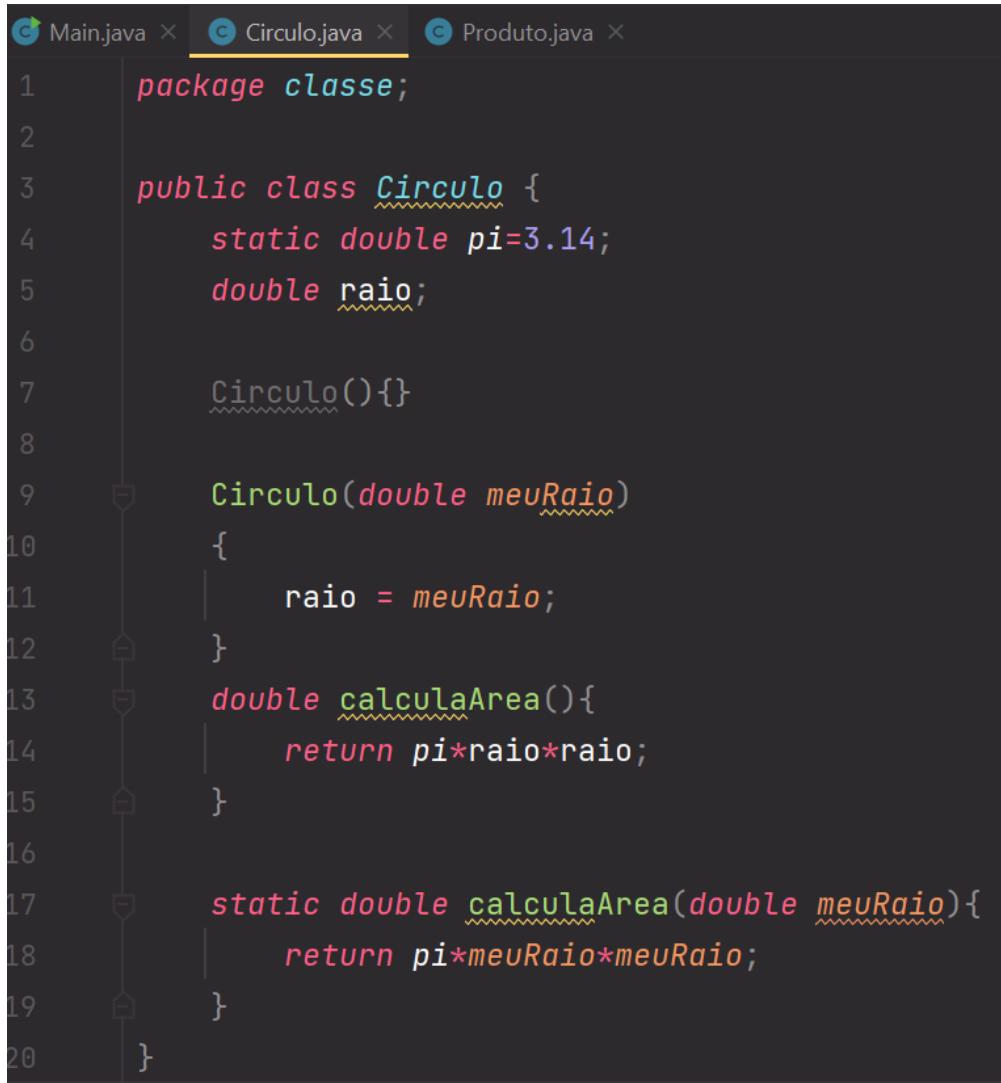
This

- ▶ Outras linguagens: this, self etc.
- ▶ Utilizada como referência ao objeto corrente:
 - ▶ Dentro dos métodos de uma classe
 - ▶ Pode ser omitida quando não existe ambiguidade
- ▶ Lembrando
 - ▶ Os métodos são executados no contexto de um objeto
 - ▶ O objeto recebe uma mensagem e executa o método
- ▶ Outra forma de entender o this
 - ▶ O objeto envia uma mensagem para si mesmo



PROGRAMAÇÃO ORIENTADA A OBJETOS

Static

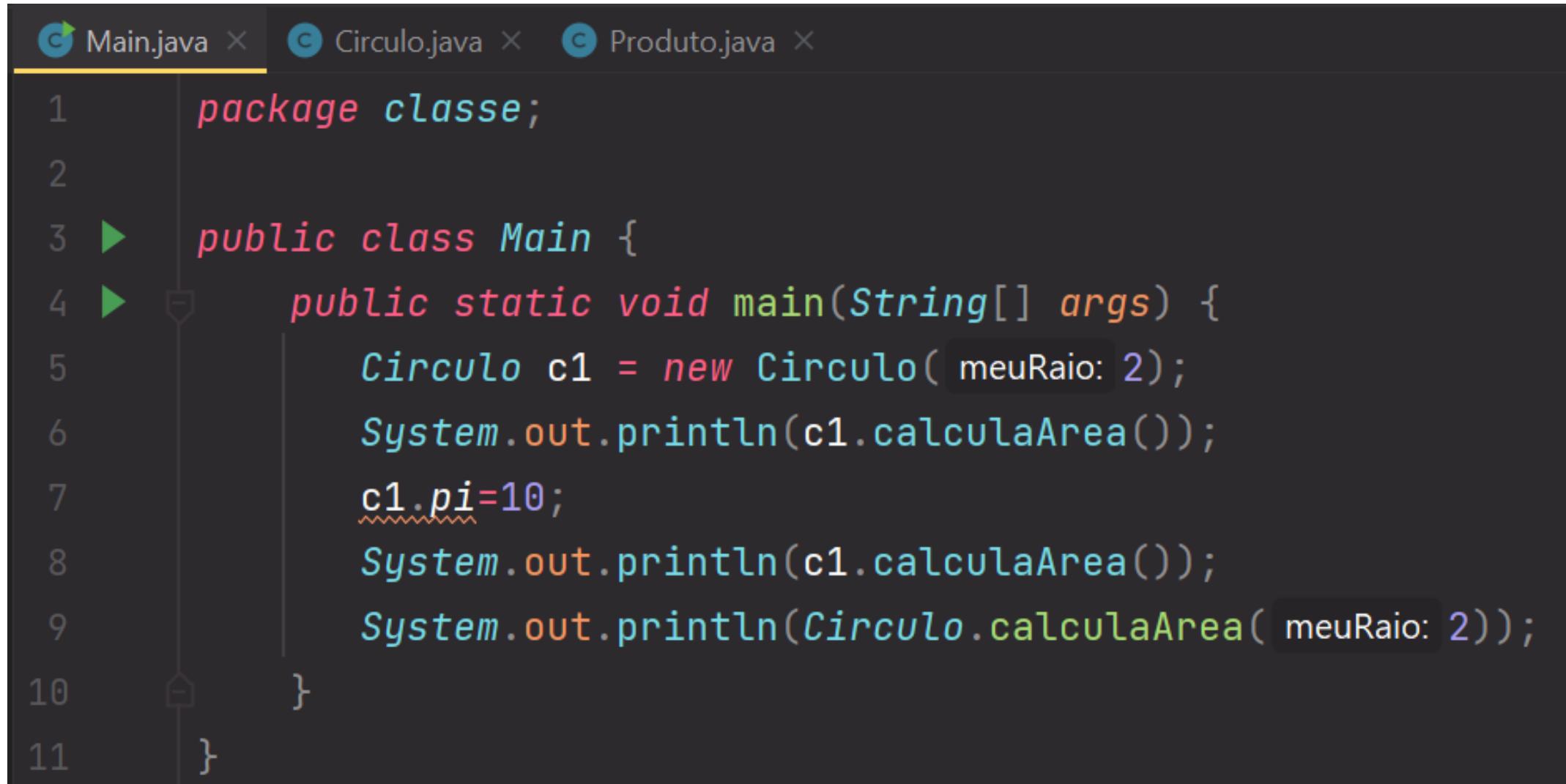


The screenshot shows a Java code editor with three tabs at the top: Main.java, Circulo.java (which is selected), and Produto.java. The Circulo.java tab has a yellow underline. The code in Circulo.java is as follows:

```
1 package classe;
2
3 public class Circulo {
4     static double pi=3.14;
5     double raio;
6
7     Circulo(){}
8
9     Circulo(double meuRaio)
10    {
11         raio = meuRaio;
12    }
13    double calculaArea(){
14        return pi*raio*raio;
15    }
16
17    static double calculaArea(double meuRaio){
18        return pi*meuRaio*meuRaio;
19    }
20 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

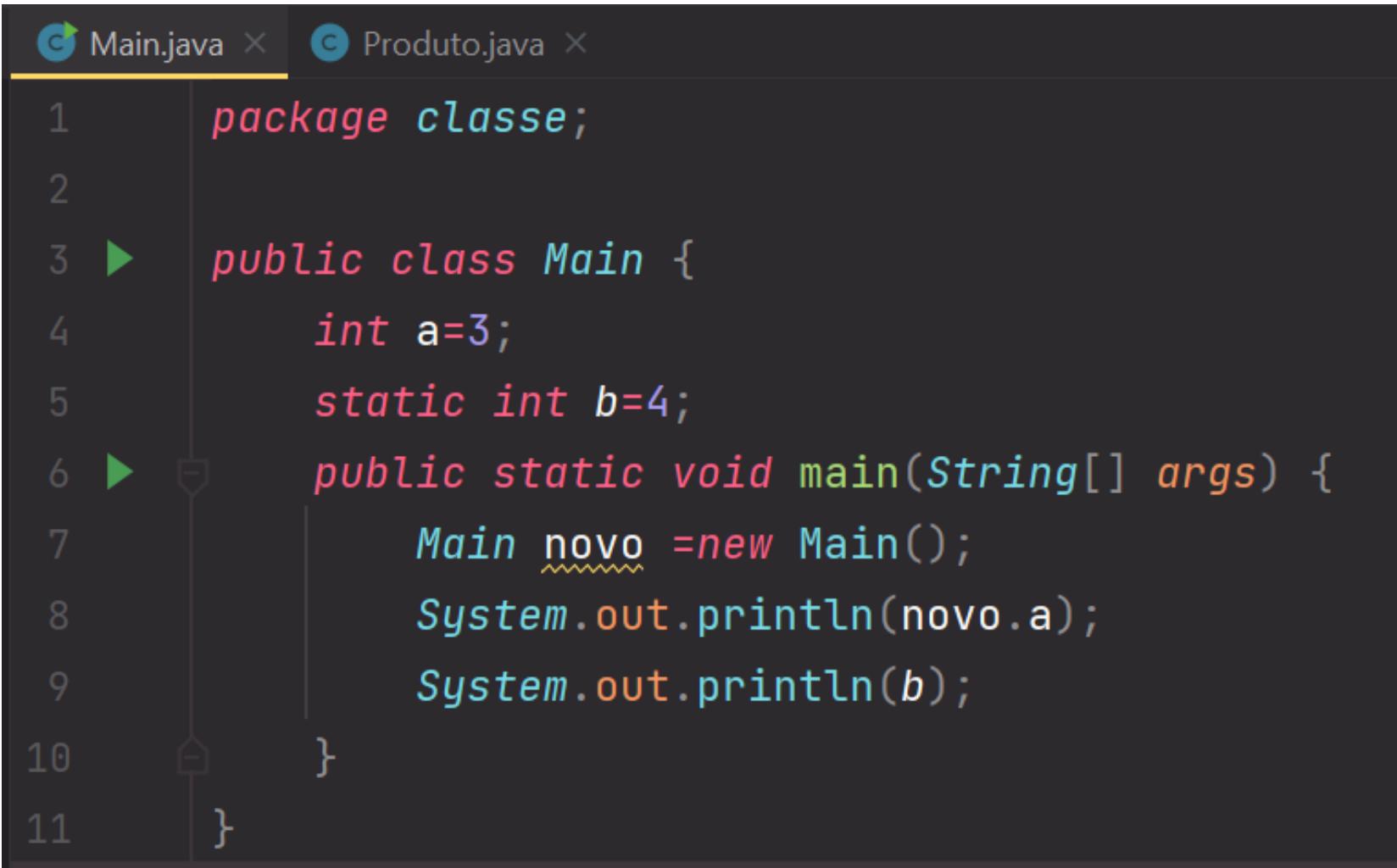
Static



```
1 package classe;
2
3 public class Main {
4     public static void main(String[] args) {
5         Circulo c1 = new Circulo( meuRaio: 2);
6         System.out.println(c1.calculaArea());
7         c1.pi=10;
8         System.out.println(c1.calculaArea());
9         System.out.println(Circulo.calculaArea( meuRaio: 2));
10    }
11 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Static



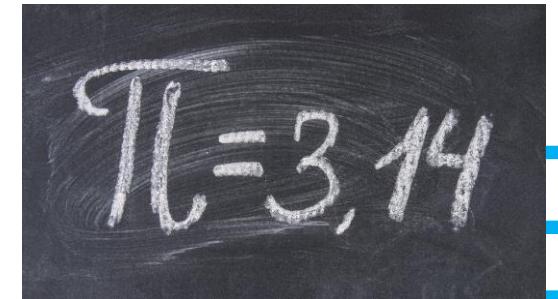
The image shows a screenshot of a Java code editor with two files open: Main.java and Produto.java. The Main.java file is the active tab, indicated by a yellow underline. The code in Main.java is as follows:

```
1 package classe;
2
3 public class Main {
4     int a=3;
5     static int b=4;
6     public static void main(String[] args) {
7         Main novo =new Main();
8         System.out.println(novo.a);
9         System.out.println(b);
10    }
11 }
```

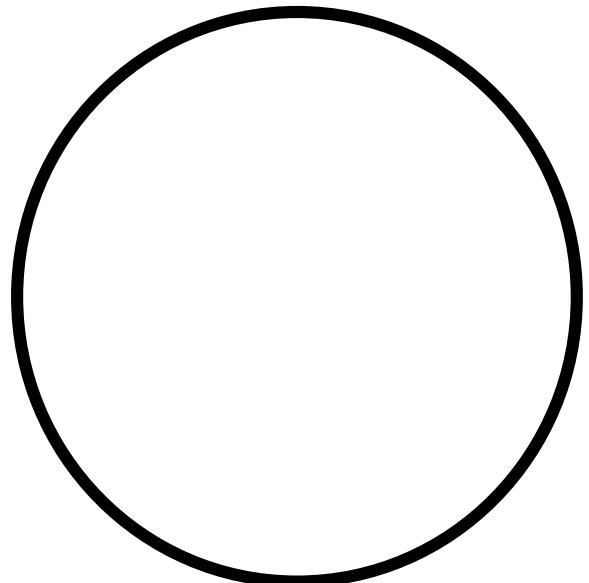
The Produto.java file is visible in the background, showing its header line: package classe;

PROGRAMAÇÃO ORIENTADA A OBJETOS

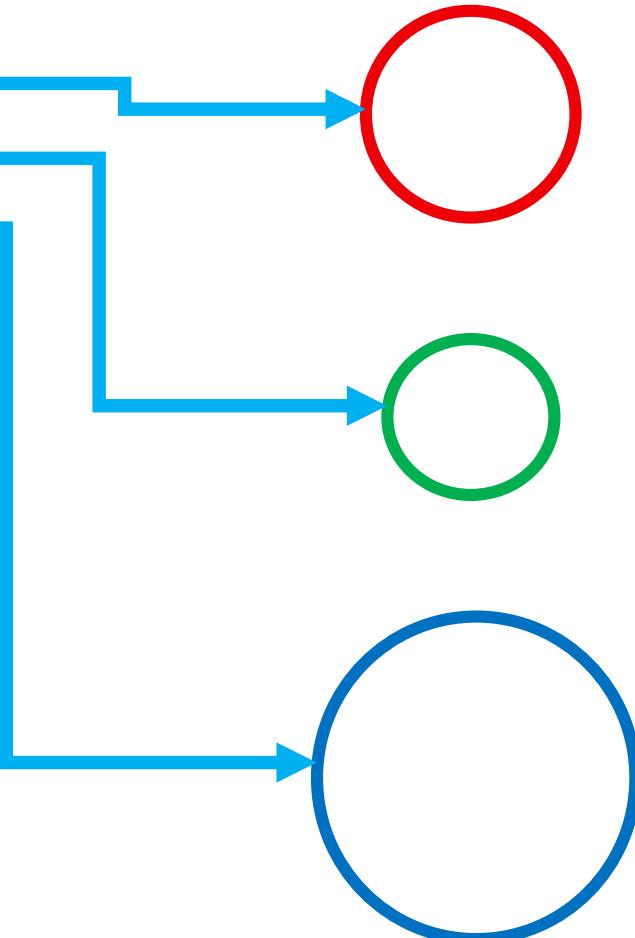
static



static



CIRCUL
O



CIRCULO A

COR: VERMELHO

RAIO=2

CIRCULO

COR:

VERDE

RAIO=1

CIRCULO

COR:

AZUL

RAIO=3

PROGRAMAÇÃO ORIENTADA A OBJETOS

static

- ▶ Os métodos static ou métodos da classe são funções que não dependem de nenhuma variável de instância, quando invocados executam uma função sem a dependência do conteúdo de um objeto ou a execução da instância de uma classe, conseguindo chamar direto qualquer método da classe e também manipulando alguns campos da classe.
- ▶ Os métodos static tem um relacionamento com uma classe como um todo, enquanto os métodos que não são static são associados a uma instância de classe específica (objeto) e podem manipular as variáveis de instância do objeto, como pode ser visto nos exemplos de declarações de métodos.

PROGRAMAÇÃO ORIENTADA A OBJETOS

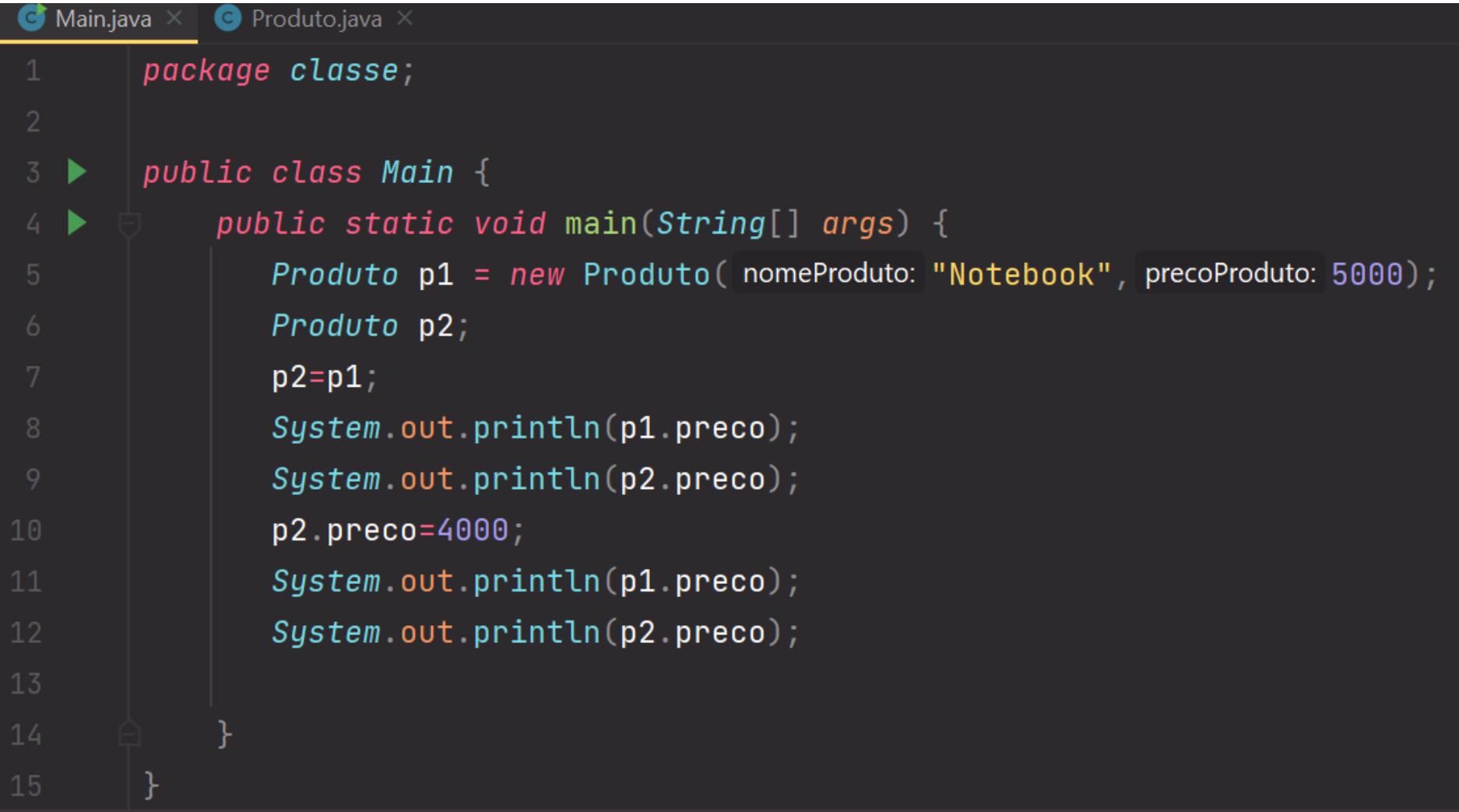
Atribuição por Valor vs Atribuição por referência



```
Main.java × Produto.java ×
1 package classe;
2
3 public class Produto {
4     String nome;
5     double preco;
6     double desconto;
7
8     Produto(){}
9
10    Produto(String nomeProduto, double precoProduto){
11        nome=nomeProduto;
12        preco=precoProduto;
13    }
14}
15
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Atribuição por Valor vs Atribuição por referência



The screenshot shows a Java code editor with two tabs: 'Main.java' and 'Produto.java'. The 'Main.java' tab is active, displaying the following code:

```
1 package classe;
2
3 public class Main {
4     public static void main(String[] args) {
5         Produto p1 = new Produto( nomeProduto: "Notebook", precoProduto: 5000);
6         Produto p2;
7         p2=p1;
8         System.out.println(p1.preco);
9         System.out.println(p2.preco);
10        p2.preco=4000;
11        System.out.println(p1.preco);
12        System.out.println(p2.preco);
13
14    }
15 }
```

The 'Produto.java' tab is visible but contains no code.

PROGRAMAÇÃO ORIENTADA A OBJETOS

Atribuição por Valor vs Atribuição por referência

```
1 package classe;  
2  
3 public class Produto implements Cloneable {  
4     String nome;  
5     double preco;  
6     double desconto;  
7  
8     Produto(){  
9  
10    Produto(String nomeProduto, double precoProduto){  
11        nome=nomeProduto;  
12        preco=precoProduto;  
13    }  
14}
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Atribuição por Valor vs Atribuição por referência

```
15     @Override  
16     public Produto clone() {  
17         try {  
18             Produto clone = (Produto) super.clone();  
19             // TODO: copy mutable state here, so the clone can't change the intern  
20             return clone;  
21         } catch (CloneNotSupportedException e) {  
22             throw new AssertionError();  
23         }  
24     }  
25 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Atribuição por Valor vs Atribuição por referência

Nos dados primitivos os dados são passados por **valor**, isso significa que é criado uma cópia e acontece uma nova alocação de memória para aquela informação. Pense nisso como criar vários **clones** de uma mesma coisa



PROGRAMAÇÃO ORIENTADA A OBJETOS

Atribuição por Valor vs Atribuição por referência



Quando falamos que os dados são passados por referência significa que irão apontar para o mesmo objeto na memória – o novo objeto será apenas uma referência ao caminho do objeto original na memória
pense como se fosse um espelho ele apenas “imita” O objeto real

PROGRAMAÇÃO ORIENTADA A OBJETOS

Valor Padrão

```
1 package classe;  
2  
3 public class Produto {  
4     String nome;  
5     double preco;  
6     double desconto;  
7 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Valor Padrão

```
1  package classe;  
2  
3 ► public class Main {  
4  
5 ►   ► public static void main(String[] args) {  
6       Produto p1 = new Produto();  
7       System.out.println(p1.nome);  
8       System.out.println(p1.preco);  
9       System.out.println(p1.desconto);  
10      }  
11  }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Variáveis Locais

```
1  package classe;  
2  
3  public class Produto {  
4      String nome;  
5      double preco;  
6      double desconto;  
7  
8      void defineString(){  
9          String novaString="Hello";  
10     }  
11     System retornaString(){  
12         return novaString();  
13     }  
14  
15  
16     }  
17
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Variáveis Locais

```
1     package classe;  
2  
3 ►  public class Main {  
4  
5 ►    public static void main(String[] args) {  
6        Produto p1 = new Produto();  
7        System.out.println(p1.nome);  
8        System.out.println(p1.preco);  
9        System.out.println(p1.desconto);  
10       p1.defineString();  
11       System.out.println(p1.retornaString());  
12    }  
13 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Valor Null

```
1 package classe;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         int a=4;
7         String s1 = "Hello";
8         System.out.println(s1.concat(" World!"));
9         String s2= null;
10        //System.out.println(s2.concat(" World"));
11        String s = (a>=5)? null: new String( original: "");
12        System.out.println(s.concat("Hello World!"));
13    }
14 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Instanciação



Class
e



Instância
(Objeto)

PROGRAMAÇÃO ORIENTADA A OBJETOS

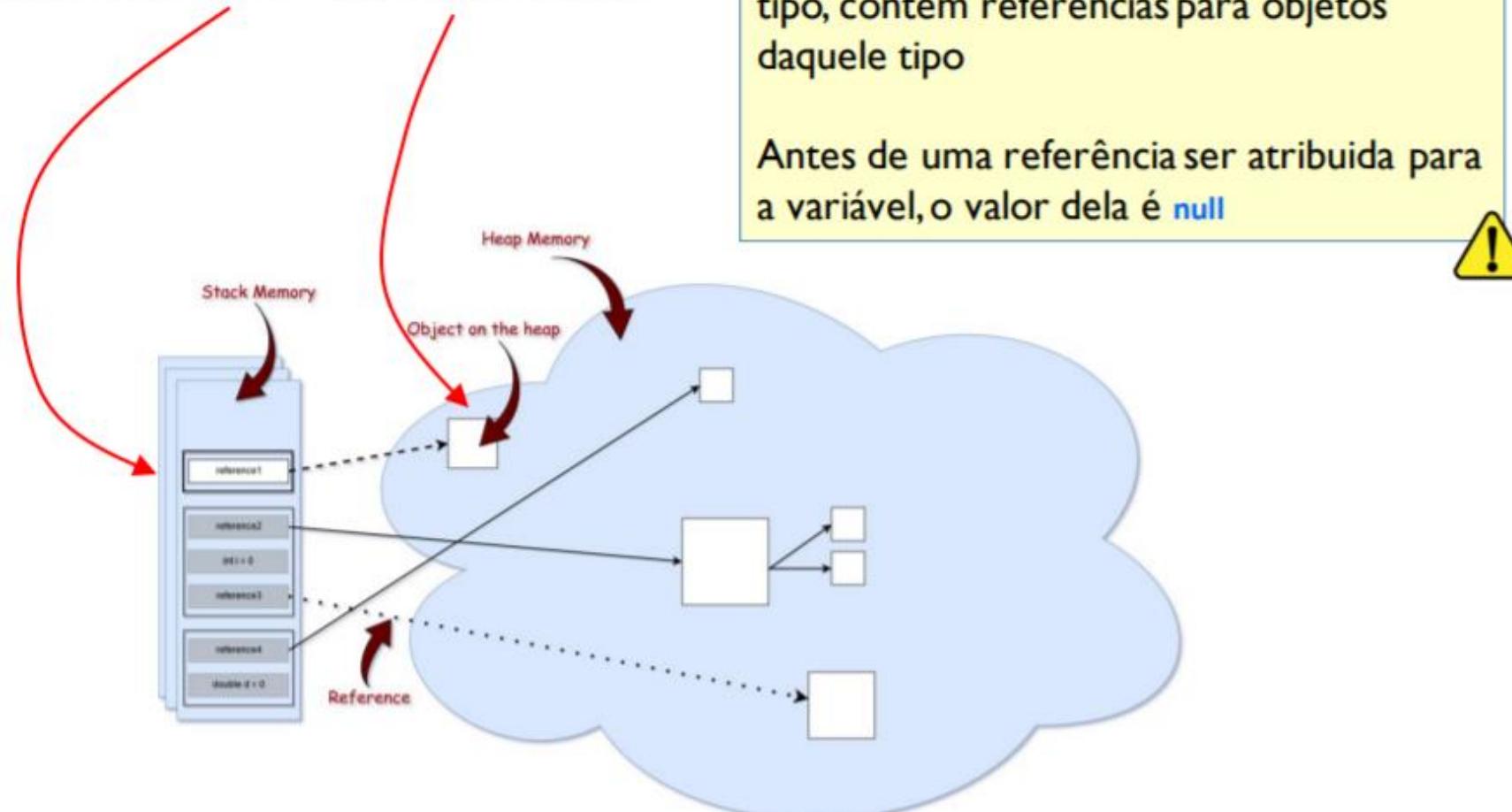
Instanciação

- Objetos devem ser instanciados (criados) para ser utilizados
- Isso é feito por um tipo especial de métodos, chamados de **construtores**

PROGRAMAÇÃO ORIENTADA A OBJETOS

Instanciação

```
BankAccount conta = new BankAccount();
```



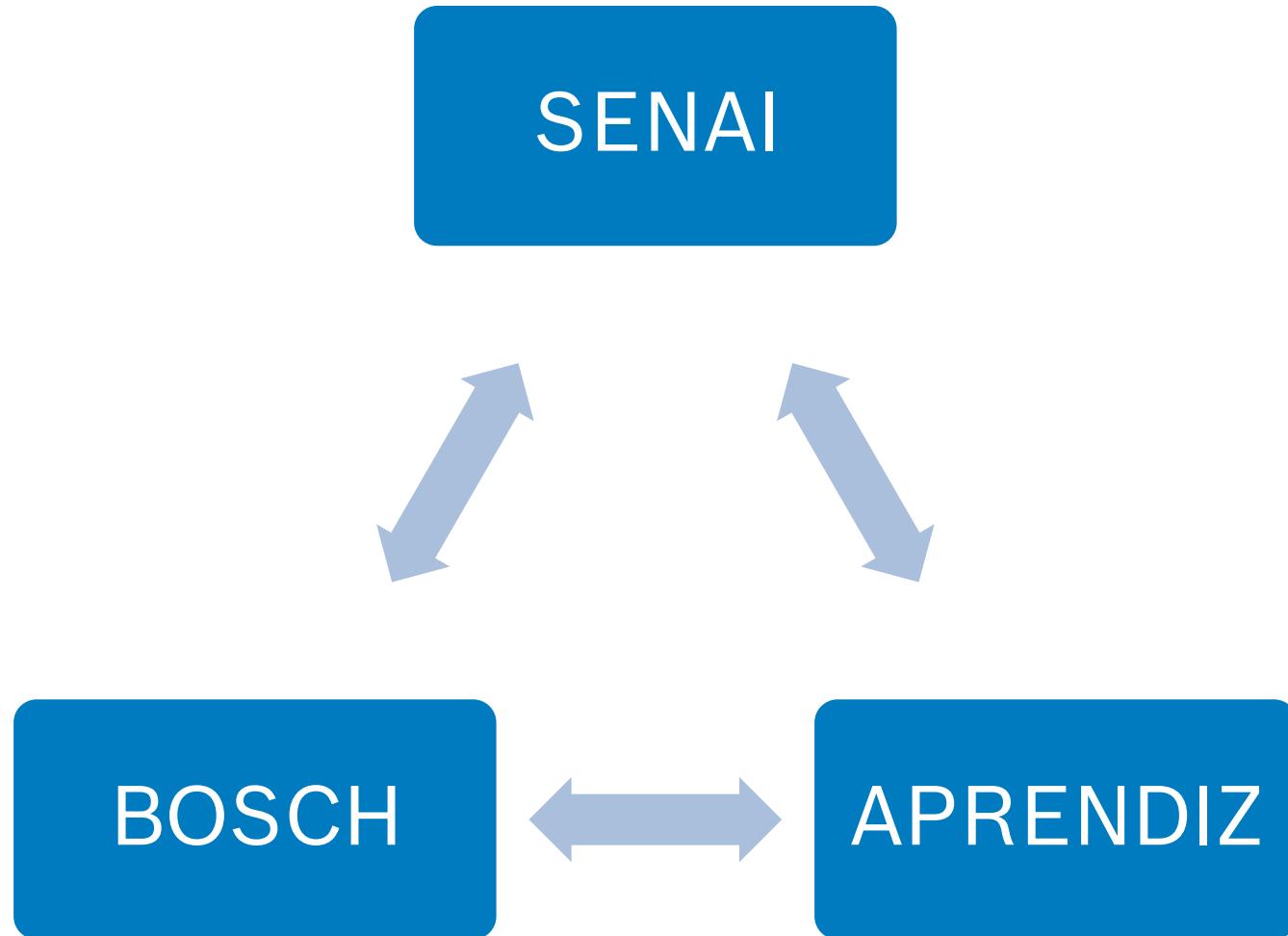
Uma variável que tem uma classe como tipo, contém referências para objetos daquele tipo

Antes de uma referência ser atribuída para a variável, o valor dela é **null**



PROGRAMAÇÃO ORIENTADA A OBJETOS

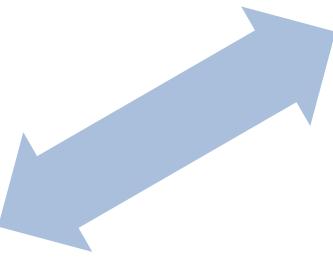
Relacionamento



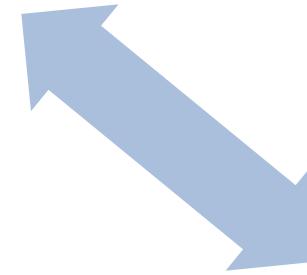
PROGRAMAÇÃO ORIENTADA A OBJETOS

Relacionamento

ENFERMEI
RA



PACIEN
TE



MÉDICO



PROGRAMAÇÃO ORIENTADA A OBJETOS

Composição



PNEU



CARRO



MOTOR



CHASSI

PROGRAMAÇÃO ORIENTADA A OBJETOS

Relacionamento de um para um

1



1



CARR
O

MOTO
R

PROGRAMAÇÃO ORIENTADA A OBJETOS

Relacionamento de um para muitos

1



n



CARR
O

MOTO
R

PROGRAMAÇÃO ORIENTADA A OBJETOS

Relacionamento de muitos para muitos



TIO



SOBRINH
O

PROGRAMAÇÃO ORIENTADA A OBJETOS

Relacionamento de um para um

```
1 package bosch;  
2  
3 public class Motor {  
4     double fatorInjecao=1;  
5     boolean ligado=false;  
6  
7     int giros()  
8     {  
9         if(!ligado) {  
10             return 0;  
11         }  
12         else{  
13             return (int) Math.round(fatorInjecao * 3000);  
14         }  
15     }  
16 }  
17 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Relacionamento de um para um

```
1 package bosch;
2
3 public class Carro {
4     Motor motor=new Motor();
5
6     void acelerar(){
7         motor.fatorInjecao+=0.4;
8     }
9     void frear(){
10    {
11        motor.fatorInjecao-=0.4;
12    }
13    void ligar(){
14        motor.ligado=true;
15    }
16    void desligar(){
17    {
18        motor.ligado=false;
19    }
20    boolean estaLigado(){
21        return motor.ligado;
22    }
23
24 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Relacionamento de um para um

```
1 package bosch;  
2  
3 ► public class Main {  
4  
5 ►   ► public static void main(String[] args) {  
6     Carro carro = new Carro();  
7     System.out.println(carro.estaLigado());  
8     carro.ligar();  
9     System.out.println(carro.estaLigado());  
10    System.out.println(carro.motor.giros());  
11    carro.acelerar();  
12    carro.acelerar();  
13    System.out.println(carro.motor.giros());  
14    carro.frear();  
15    System.out.println(carro.motor.giros());  
16  }  
17 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Relacionamento de um para muitos

```
1 package bosch;  
2  
3 public class Item {  
4     String nome;  
5     int quantidade;  
6     double preco;  
7     Compra compra;  
8  
9     Item(String nome, int quantidade, double preco){  
10        this.nome=nome;  
11        this.quantidade=quantidade;  
12        this.preco=preco;  
13    }  
14}
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Relacionamento de um para muitos

```
1 package bosch;
2
3 import java.util.ArrayList;
4
5 public class Compra {
6     String cliente;
7     ArrayList<Item> itens=new ArrayList<Item>();
8
9     void adicionarItem(Item item){
10         this.itens.add(item);
11         item.compra=this;
12     }
13     double obterValorTotal() {
14         double total=0;
15         for (Item item : itens) {
16             total += item.quantidade * item.preco;
17         }
18         return total;
19     }
20 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Relacionamento de um para muitos

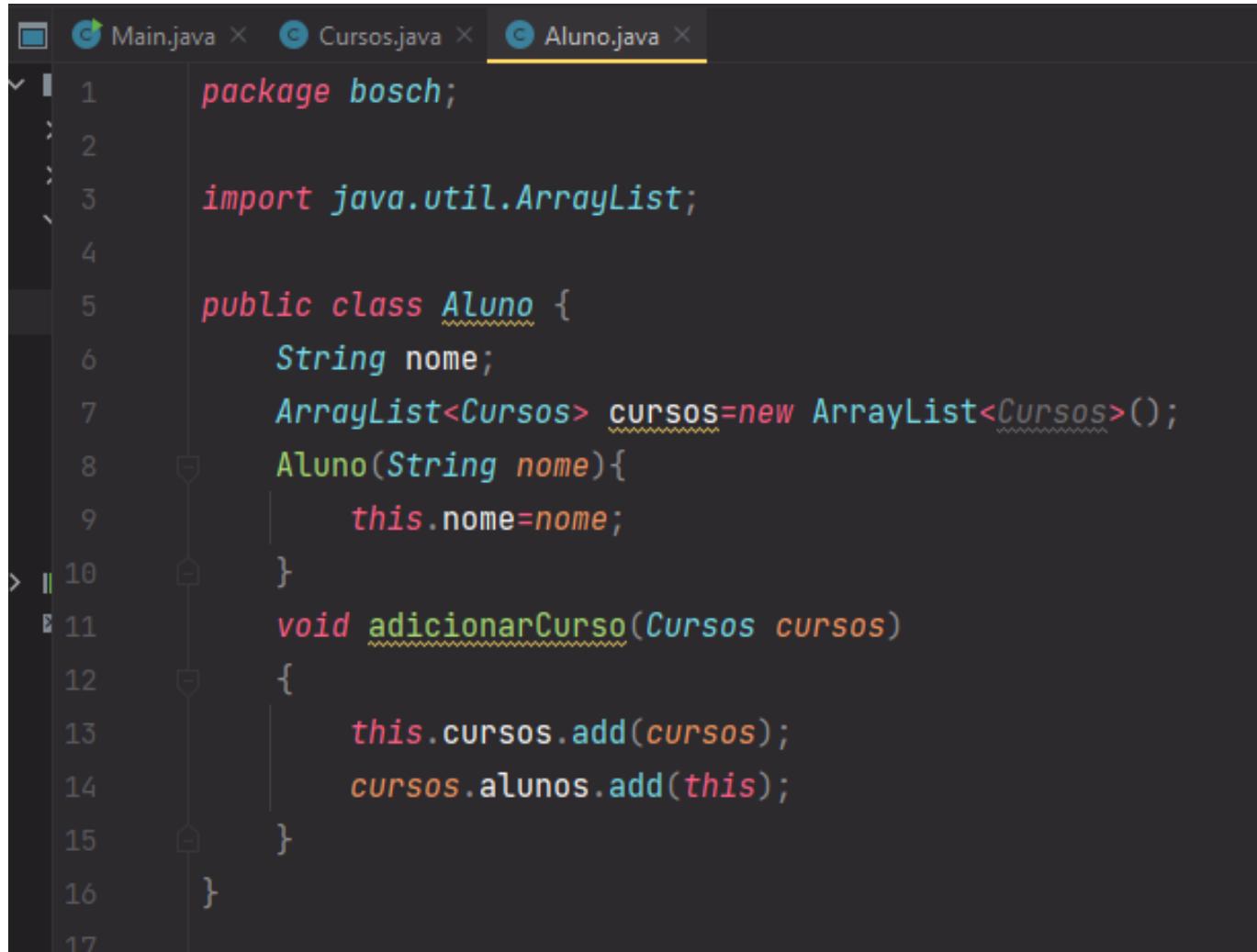


The screenshot shows a Java code editor with three tabs at the top: Main.java (selected), Item.java, and Compra.java. The Main.java tab contains the following code:

```
1 package bosch;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         Compra compra1 = new Compra();
7         compra1.cliente="João Pedro";
8         compra1.adicionarItem(new Item( nome: "Notebook", quantidade: 1, preco: 5000));
9         compra1.adicionarItem(new Item( nome: "Iphone", quantidade: 2, preco: 6000));
10        compra1.adicionarItem(new Item( nome: "Amazon Echo", quantidade: 3, preco: 1200));
11        System.out.println(compra1.itens.size());
12        System.out.println(compra1.obterValorTotal());
13    }
14 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Relacionamento de muitos para muitos



The screenshot shows a Java code editor with three tabs at the top: Main.java, Cursos.java, and Aluno.java. The Aluno.java tab is active, displaying the following code:

```
package bosch;

import java.util.ArrayList;

public class Aluno {
    String nome;
    ArrayList<Cursos> cursos=new ArrayList<Cursos>();
    Aluno(String nome){
        this.nome=nome;
    }
    void adicionarCurso(Cursos cursos)
    {
        this.cursos.add(cursos);
        cursos.alunos.add(this);
    }
}
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Relacionamento de muitos para muitos

```
1 package bosch;
2
3 import java.util.ArrayList;
4
5 public class Cursos {
6     String nome;
7     ArrayList<Aluno> alunos=new ArrayList<Aluno>();
8     Cursos(String nome) { this.nome=nome; }
9
10    void adicionarAluno(Aluno aluno){
11        this.alunos.add(aluno);
12        aluno.cursos.add(this);
13    }
14
15    ArrayList<String>obeterporCurso(){
16
17        ArrayList<String>cadastrados=new ArrayList<~>();
18        for (Aluno aluno:alunos) {
19            cadastrados.add(aluno.nome);
20        }
21        return cadastrados;
22    }
23
24}
25}
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Relacionamento de muitos para muitos

```
1 package bosch;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         Aluno aluno1=new Aluno( nome: "João");
7         Aluno aluno2=new Aluno( nome: "Maria");
8         Aluno aluno3=new Aluno( nome: "Pedro");
9         Cursos cursos1=new Cursos( nome: "Java");
10        Cursos cursos2=new Cursos( nome: "Python");
11        Cursos cursos3=new Cursos( nome: "VsDia");
12        cursos1.adicionarAluno(aluno1);
13        cursos1.adicionarAluno(aluno2);
14        cursos2.adicionarAluno(aluno1);
15        cursos2.adicionarAluno(aluno3);
16        aluno1.adicionarCurso(cursos3);
17        aluno2.adicionarCurso(cursos3);
18        aluno3.adicionarCurso(cursos3);
19        for (Aluno aluno: cursos3.alunos)
20        {
21            System.out.printf("Meu nome é %s e estou matriculado" +
22                " no curso %s\n",aluno.nome,cursos3.nome);
23        }
24        System.out.println(cursos3.alunos.get(0).nome);
25        System.out.println(cursos3.obeterporCurso());
26    }
27}
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Aplicativo de Delivery

- Um aplicativo de delivery é uma solução de gerenciamento de pedidos para restaurantes, visando ser uma opção acessível para os clientes pedirem comida em casa ou no trabalho. O aplicativo é configurado pelo gerente do restaurante e implementa as seguintes funcionalidades.



PROGRAMAÇÃO ORIENTADA A OBJETOS

Aplicativo de Delivery

- ▶ O aplicativo é configurado com os detalhes do restaurante: nome, CNPJ, e posição no mapa. Por simplicidade considere que a posição é um par de coordenadas (X, Y).
- ▶ O aplicativo permite o cadastro de usuários, que é feito informando nome, CPF, e endereço de entrega. Por simplicidade considere que o endereço é armazenado em forma de posição (X, Y).
- ▶ O aplicativo permite adicionar e remover itens ao cardápio do restaurante. Cada item contém nome, preço.
- ▶ O aplicativo permite imprimir o cardápio completo do restaurante.
- ▶ O aplicativo permite registrar pedidos dos usuários. Cada pedido pode incluir um ou mais itens do cardápio do restaurante
- ▶ O valor total do pedido é a soma dos preços atuais de cada item.

PROGRAMAÇÃO ORIENTADA A OBJETOS

Aplicativo de Delivery

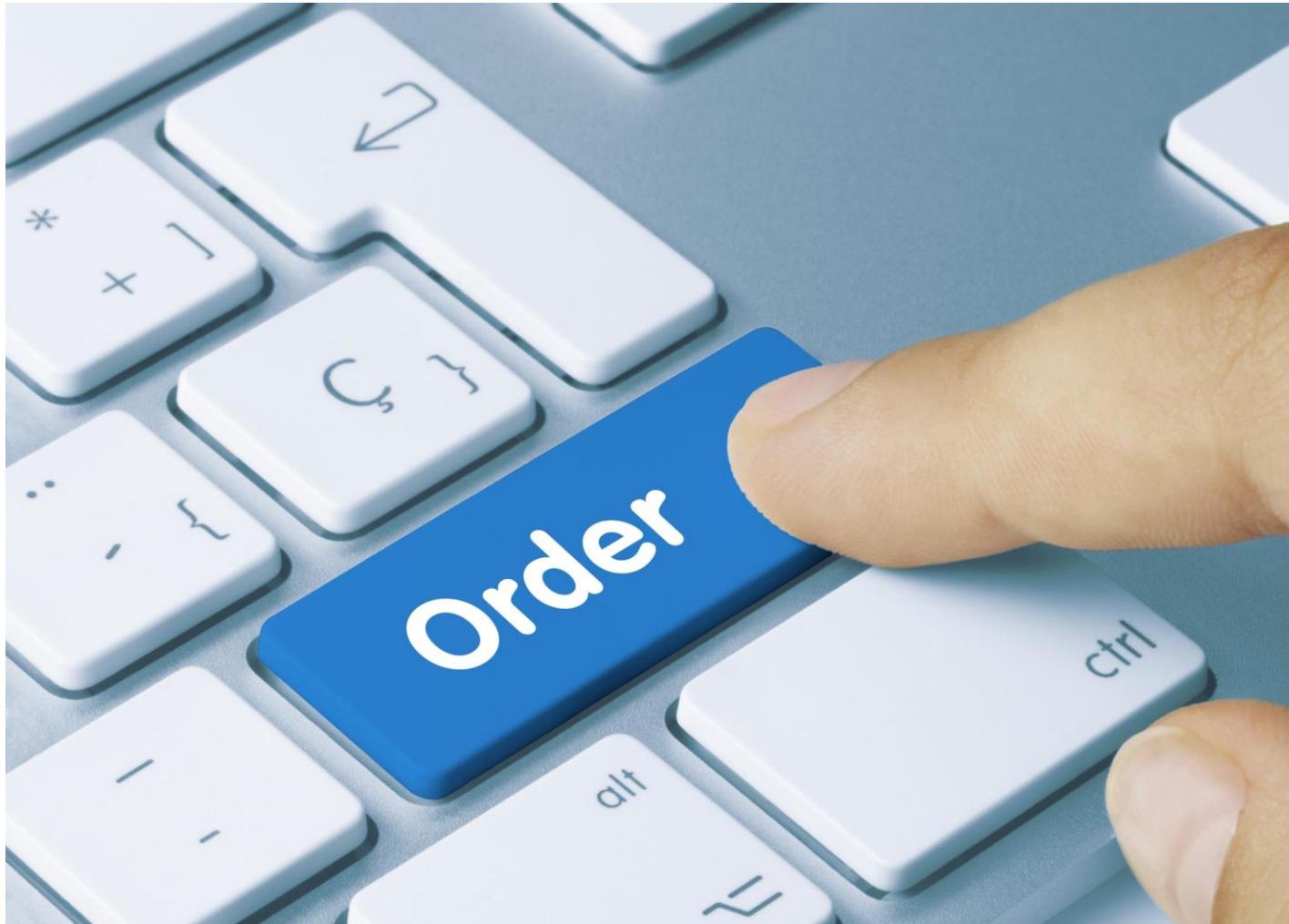


Classe: Aplicativo
Instância:

Atributos	Comportamentos
Restaurantes	cadastrarRestaurante()
Usuários	cadastrarUsuario()
Pedidos	

PROGRAMAÇÃO ORIENTADA A OBJETOS

Aplicativo de Delivery



Classe: Pedido	
Instância:	
Atributos	Comportamentos
Restaurantes	fazerPedido()
Usuários	imprimirPedido()

PROGRAMAÇÃO ORIENTADA A OBJETOS

Aplicativo Delivery

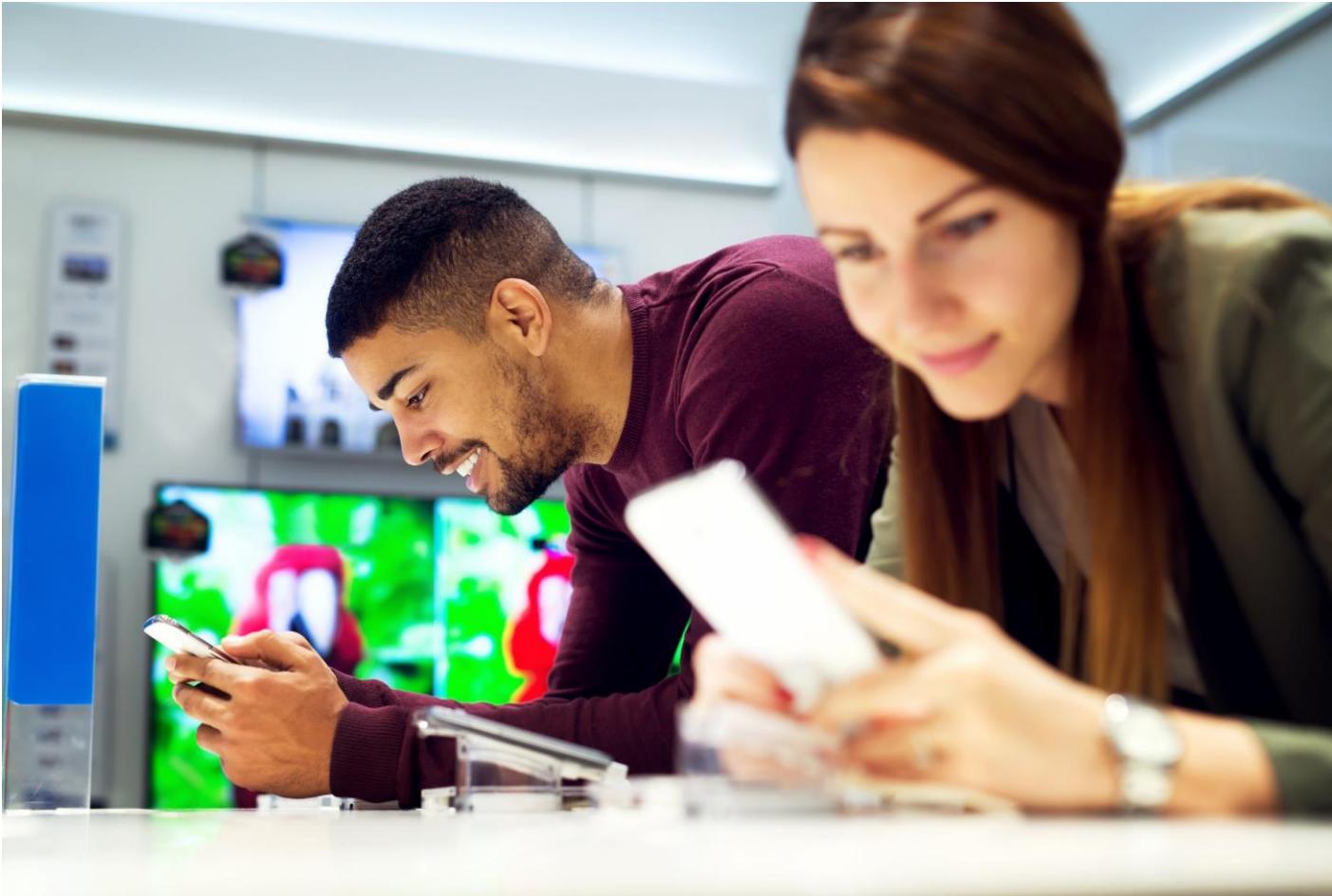


Classe: Restaurante
Instância:

Atributos	Comportamentos
Nome	imprimirCardapio()
Preço	adicionarLanche()
	removerLanche()

PROGRAMAÇÃO ORIENTADA A OBJETOS

Aplicativo Delivery



Classe: Usuário
Instância: nome do Usuário

Atributos	Comportamentos
Nome	
Endereço	
CPF	

PROGRAMAÇÃO ORIENTADA A OBJETOS

Aplicativo Delivery



Classe: Lanche
Instância:

Atributos	Comportamentos
Nome	
Preço	

PROGRAMAÇÃO ORIENTADA A OBJETOS

Herança



MÉDICO



PEDIATRA



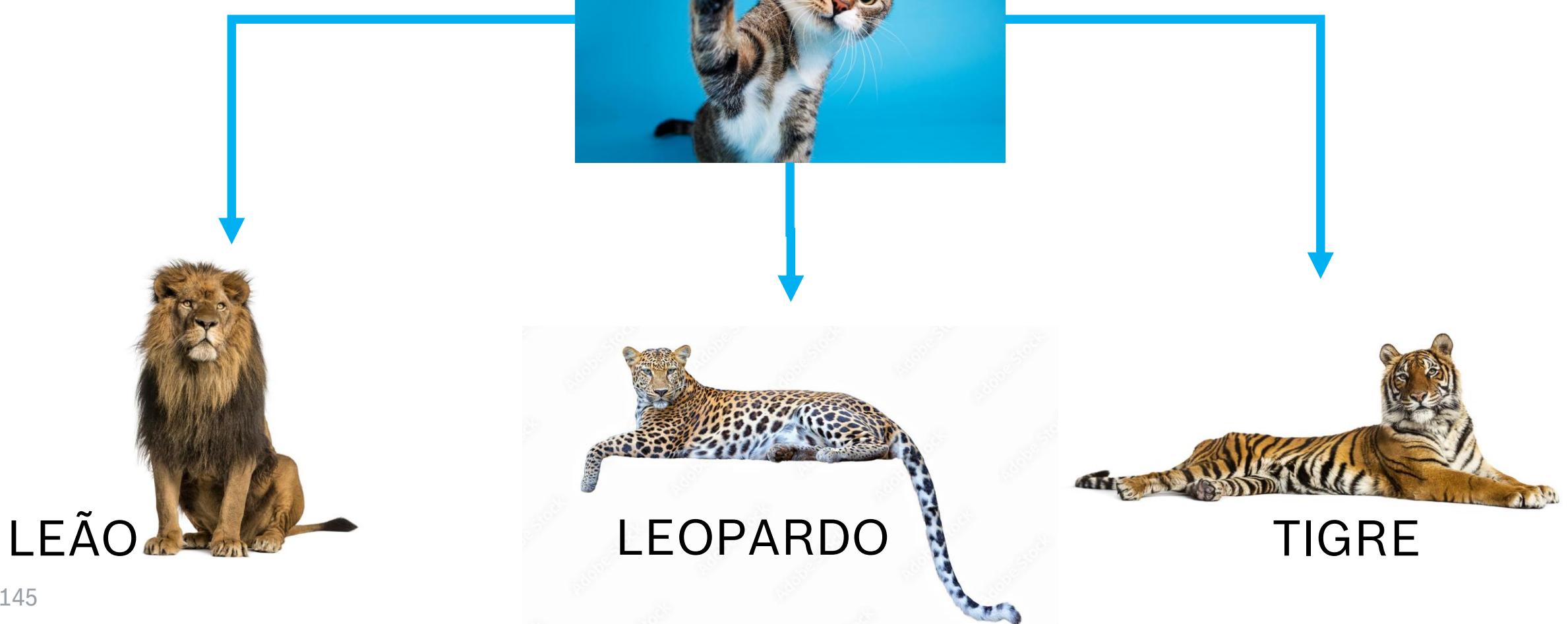
CARDIOLOGISTA



OFTALMOLOGISTA

PROGRAMAÇÃO ORIENTADA A OBJETOS

Herança

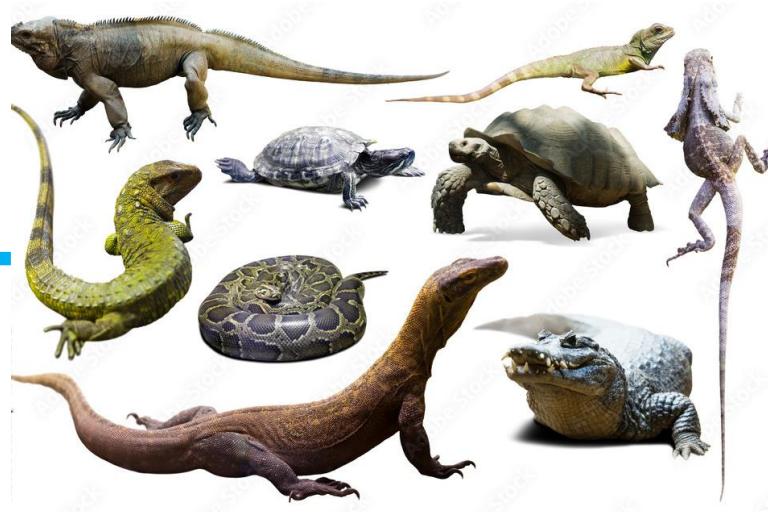


PROGRAMAÇÃO ORIENTADA A OBJETOS

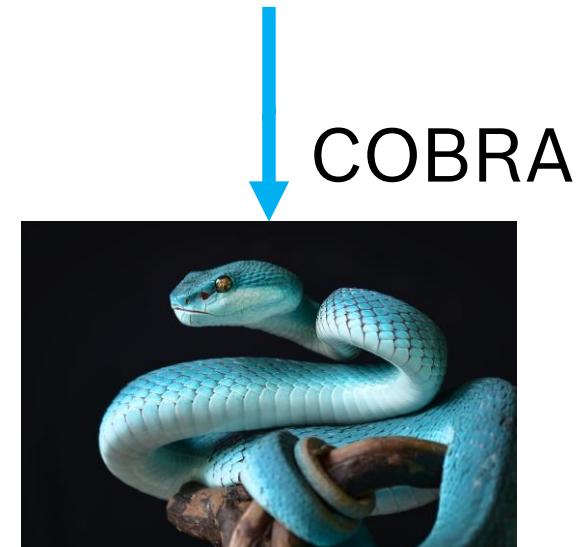
Herança



146



RÉPTEIS



COBRA



TARTARUGA

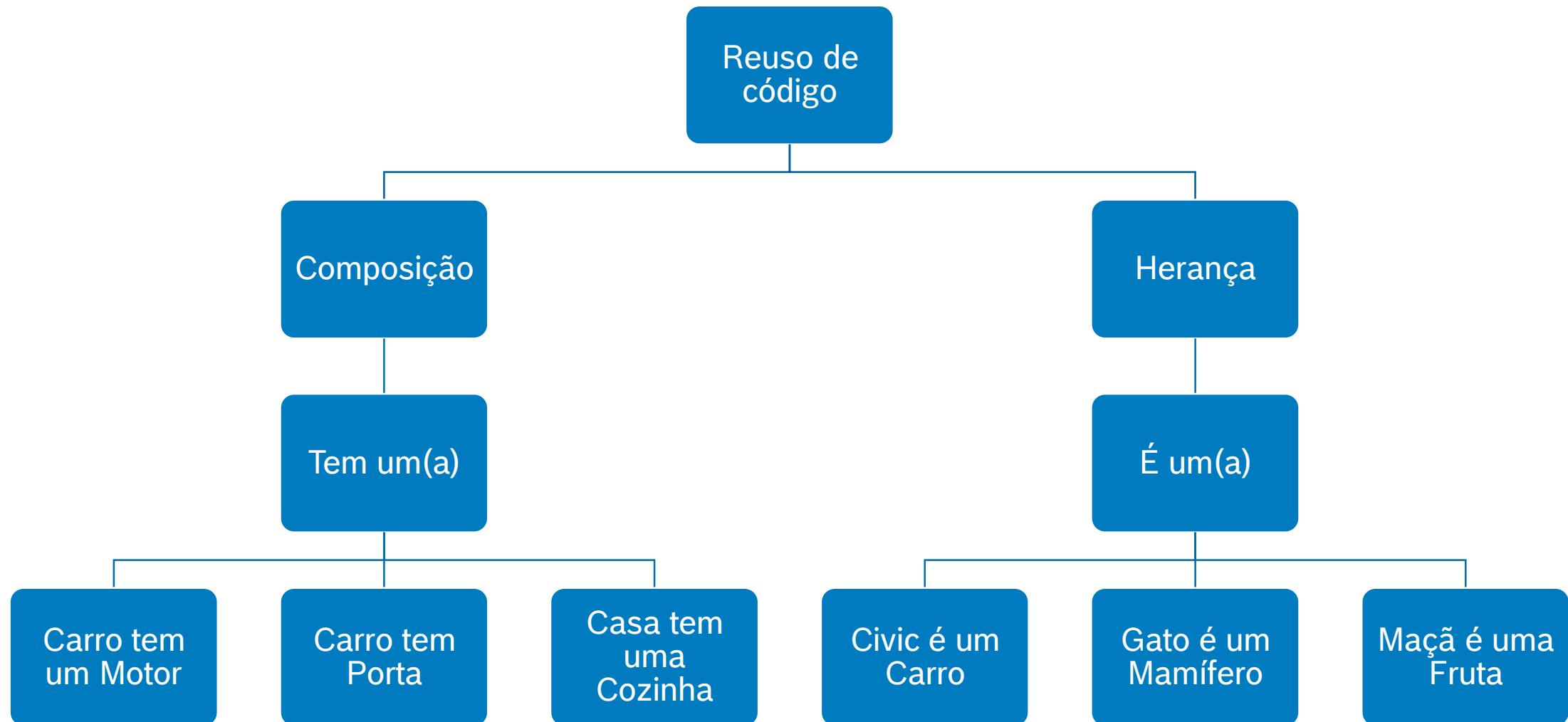
PROGRAMAÇÃO ORIENTADA A OBJETOS

Herança



PROGRAMAÇÃO ORIENTADA A OBJETOS

Herança



PROGRAMAÇÃO ORIENTADA A OBJETOS

Herança

```
1 package bosch;
2
3 public class Jogador {
4     int x;
5     int y;
6
7     boolean andar(String direcao){
8         if("norte".equalsIgnoreCase(direcao))
9         {
10             y++;
11         }
12         else if("sul".equalsIgnoreCase(direcao))
13         {
14             y++;
15         }
16         else if("leste".equalsIgnoreCase(direcao))
17         {
18             x++;
19         }
20         else if("oeste".equalsIgnoreCase(direcao))
21         {
22             x--;
23         }
24         else {
25             return false;
26         }
27         return true;
28     }
29 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Herança

```
1 package bosch;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         Jogador j1 = new Jogador();
7         j1.x=10;
8         j1.y=20;
9         j1.andar( direcao: "norte");
10        j1.andar( direcao: "norte");
11        j1.andar( direcao: "norte");
12        j1.andar( direcao: "norte");
13        System.out.println(j1.y);
14    }
15 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Herança

```
1 package bosch;  
2  
3 public class Jogador {  
4     int x;  
5     int y;  
6  
7     @    boolean andar(Direcao direcao){  
8         switch (direcao) {  
9             case NORTE -> y++;  
10            case SUL -> y--;  
11            case LESTE -> x++;  
12            case OESTE -> x--;  
13        }  
14        return true;  
15    }  
16}  
17 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Herança

```
1  package bosch;  
2  
3  ►  public class Main {  
4  
5    ►   public static void main(String[] args) {  
6        Jogador j1 = new Jogador();  
7        j1.x=10;  
8        j1.y=20;  
9        j1.andar(Direcao.NORTE);  
10       j1.andar(Direcao.NORTE);  
11       j1.andar(Direcao.NORTE);  
12       j1.andar(Direcao.NORTE);  
13       System.out.println(j1.x);  
14       System.out.println(j1.y);  
15  
16    }  
17 }
```

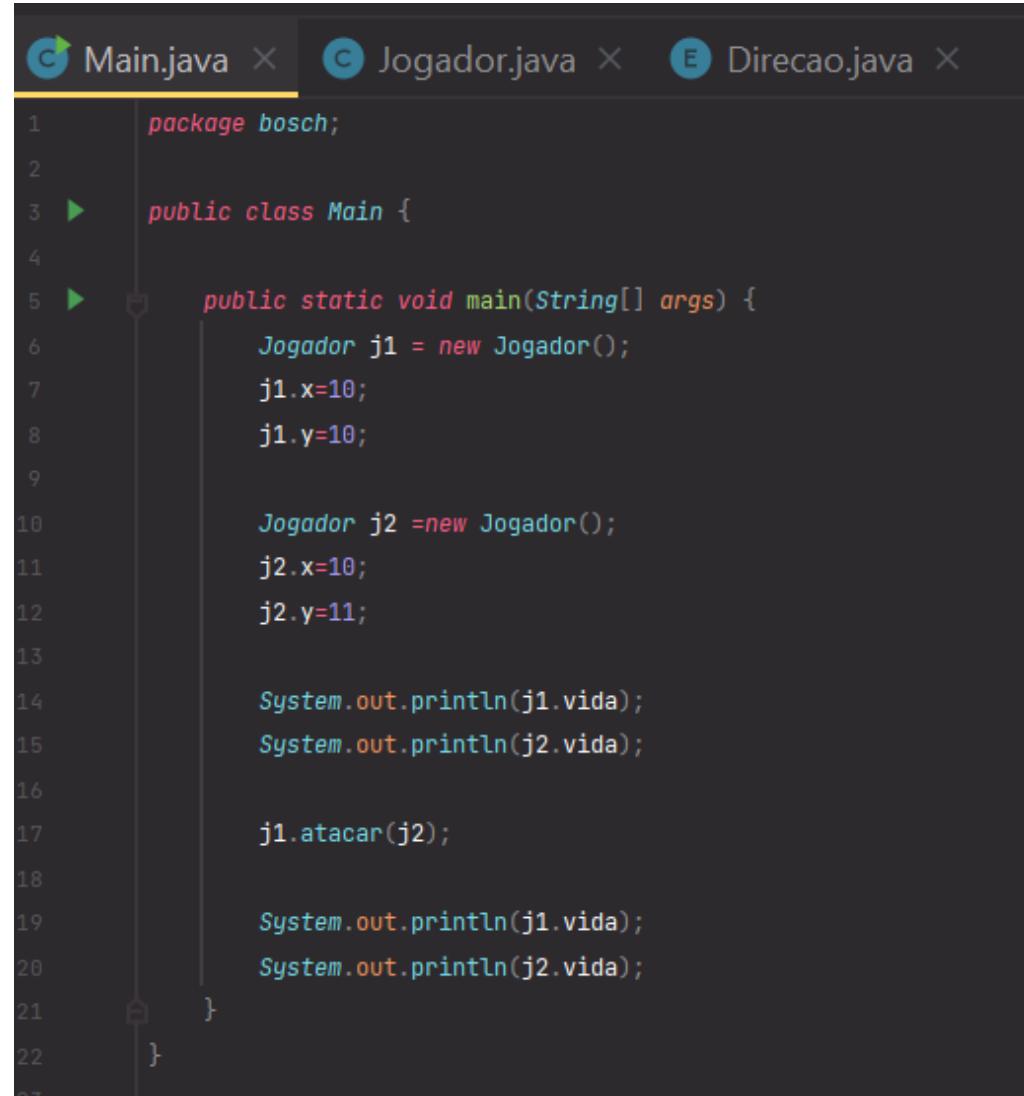
PROGRAMAÇÃO ORIENTADA A OBJETOS

Herança

```
1 package bosch;
2
3 public class Jogador {
4     int x;
5     int y;
6     int vida=100;
7
8     @ boolean andar(Direcao direcao){...}
18
19     @ boolean atacar(Jogador oponente){
20         int deltaX=Math.abs(x-oponente.x);
21         int deltaY=Math.abs(y-oponente.y);
22
23         if (deltaX==0 && deltaY==1){
24             oponente.vida-=10;
25             return true;
26         }
27         else if( deltaX==1 && deltaY==0)
28         {
29             oponente.vida-=10;
30             return true;
31         }
32         else {
33             return false;
34         }
35     }
36 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Herança



The image shows a screenshot of a Java code editor with three tabs: Main.java, Jogador.java, and Direcao.java. The Main.java tab is active, displaying the following code:

```
1 package bosch;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         Jogador j1 = new Jogador();
7         j1.x=10;
8         j1.y=10;
9
10        Jogador j2 =new Jogador();
11        j2.x=10;
12        j2.y=11;
13
14        System.out.println(j1.vida);
15        System.out.println(j2.vida);
16
17        j1.atacar(j2);
18
19        System.out.println(j1.vida);
20        System.out.println(j2.vida);
21    }
22}
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Herança

```
3 ► public class Main {  
4  
5 ►   public static void main(String[] args) {  
6       Guerreiro g1 = new Guerreiro();  
7       g1.x=10;  
8       g1.y=10;  
9  
10      Fera f2 =new Fera();  
11      f2.x=10;  
12      f2.y=11;  
13  
14      System.out.println(g1.vida);  
15      System.out.println(f2.vida);  
16  
17      g1.atacar(f2);  
18  
19      System.out.println(g1.vida);  
20      System.out.println(f2.vida);  
21    }  
22 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Herança

```
1 package bosch;
2
3 public class Guerreiro extends Jogador{
4
5     boolean atacar(Jogador oponente) {
6         int deltaX = Math.abs(x - oponente.x);
7         int deltay = Math.abs(y - oponente.y);
8
9         if (deltaX == 0 && deltay == 1) {
10             oponente.vida -= 20;
11             return true;
12         } else if (deltaX == 1 && deltay == 0) {
13             oponente.vida -= 20;
14             return true;
15         } else {
16             return false;
17         }
18     }
19 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Herança

```
1 package bosch;  
2  
3 public class Fera extends Jogador{  
4  
5     @Override  
6     boolean atacar(Jogador oponente){  
7         super.atacar(oponente);  
8         super.atacar(oponente);  
9  
10    return true;  
11}
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Sobrecarga

```
1 package bosch;  
2  
3 public class Conta {  
4     double saldo;  
5  
6     void deposito(double valor)  
7     {  
8         this.saldo=this.saldo+valor;  
9     }  
10    void deposito(double valor, String texto)  
11    {  
12        this.saldo=this.saldo+valor;  
13    }  
14}
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Sobrescrita

```
1 package bosch;  
2  
3 public class ContaCorrente extends Conta{  
4  
5     @Override  
6     void deposito(double valor)  
7     {  
8         this.saldo=this.saldo+valor+1;  
9     }  
10 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Sobrecarga vs Sobrescrita

```
1 package bosch;
2
3
4 ► public class Main {
5
6 ►     public static void main(String[] args) {
7         Conta conta =new Conta();
8         conta.deposito( valor: 20);
9         conta.deposito( valor: 30, texto: "Pagamento");
10        System.out.println(conta.saldo);
11        ContaCorrente contaCorrente = new ContaCorrente();
12        contaCorrente.deposito( valor: 20);
13        contaCorrente.deposito( valor: 30);
14        System.out.println(contaCorrente.saldo);
15    }
16 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Sobrecarga vs Sobrescrita

► Sobrescrita de Métodos

- Uma subclasse pode customizar métodos que ela herda da superclasse
- Processo denominado sobreposição ou sobrescrita (overriding).
- Isto é, redefinir um método da superclasse com uma implementação própria especializada
- Se diz que a subclasse sobrescreve (overrides) a implementação de um método quando proporciona um método com a mesma assinatura da superclasse.

PROGRAMAÇÃO ORIENTADA A OBJETOS

Sobrecarga vs Sobrescrita

► Sobrecarga (overload)

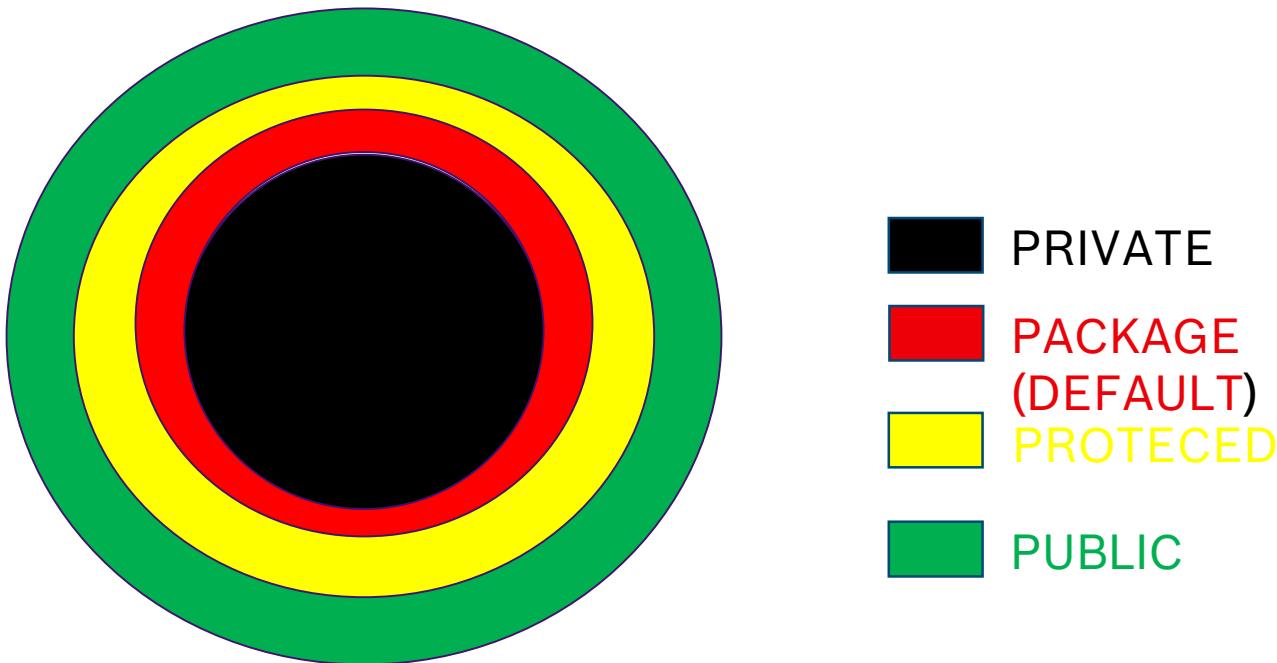
- ▶ Na mesma classe (geralmente)
- ▶ A implementação preexistente é mantida
- ▶ Métodos com assinaturas diferentes

► Sobrescrita (override)

- ▶ Apenas entre subclasse e superclasse
- ▶ A implementação preexistente é “escondida”
- ▶ Métodos com a mesma assinatura

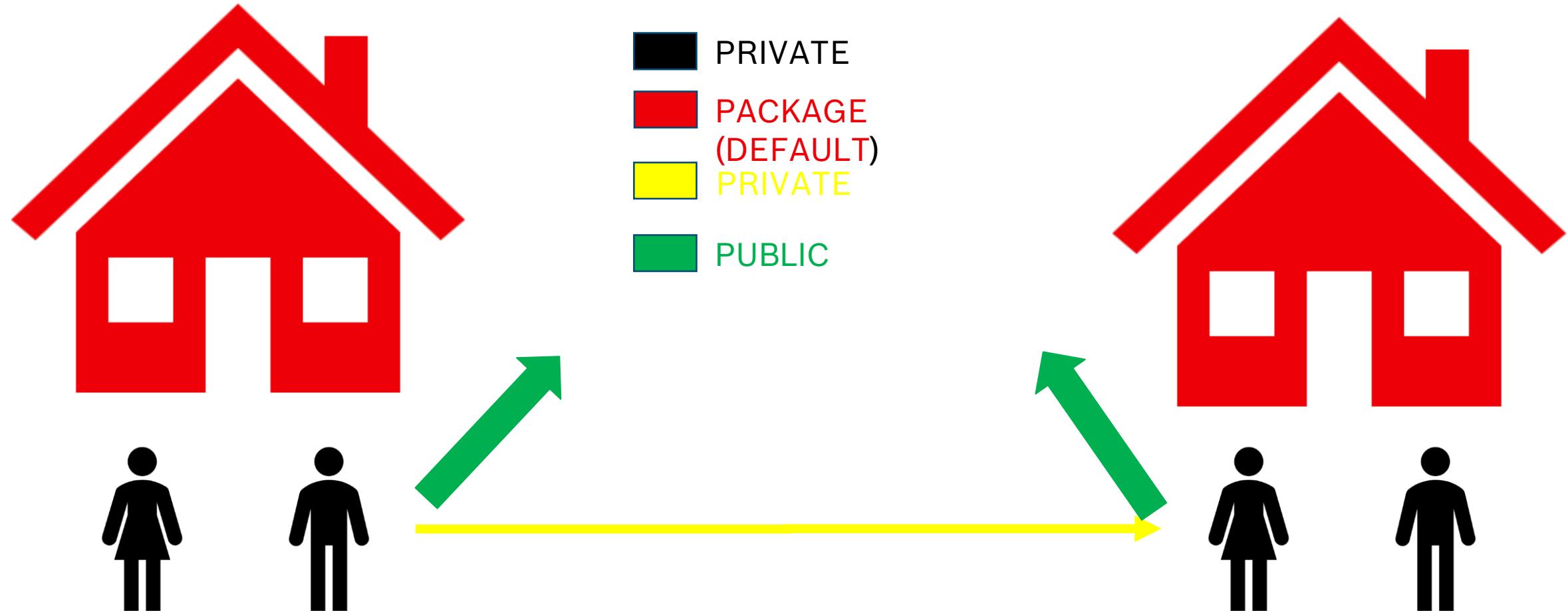
PROGRAMAÇÃO ORIENTADA A OBJETOS

Encapsulamento



PROGRAMAÇÃO ORIENTADA A OBJETOS

Encapsulamento



PROGRAMAÇÃO ORIENTADA A OBJETOS

Encapsulamento



The screenshot shows a Java code editor with five tabs at the top: Main.java, Jose.java, Maria.java (underlined in yellow), Joao.java, and Leticia.java. The Maria.java tab has a blue circular icon with a white arrow pointing down to its right. The code editor displays the following Java code:

```
1 package Casa1;
2
3 public class Maria {
4     private String segredo="Roubei um carro";
5     String facoDentroDeCasa="Ronco durante o sono";
6     protected String familiaSabe="Devo no banco";
7     public String todoMundoSabe="Vou no bar";
8 }
9
10
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Encapsulamento

```
>Main.java × Jose.java × Maria.java × Joao.java × Leticia.java ×
1 package Casa1;
2
3 public class Jose {
4     Maria esposa =new Maria();
5
6     void teste(){
7         //System.out.println(esposa.secreto);
8         System.out.println(esposa.facoDentroDeCasa);
9         System.out.println(esposa.familiaSabe);
10        System.out.println(esposa.todoMundoSabe);
11    }
12 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Encapsulamento

The screenshot shows a Java code editor with five tabs at the top: Main.java, Jose.java, Maria.java, Joao.java (which is selected), and Leticia.java. The code in the Joao.java tab is as follows:

```
1 package Casa2;
2
3 import Casa1.Maria;
4
5 public class Joao extends Maria{
6
7     void teste(){
8         //System.out.println(super.secreto);
9         //System.out.println(super.facoDentroDeCasa);
10        System.out.println(super.familiaSabe);
11        System.out.println(super.todoMundoSabe);
12    }
13}
```

The code illustrates inheritance where the Joao class extends the Maria class. It also demonstrates encapsulation through the use of the super keyword to access protected members of the base class.

PROGRAMAÇÃO ORIENTADA A OBJETOS

Encapsulamento



The screenshot shows a Java code editor with multiple tabs at the top: Main.java, Jose.java, Maria.java, Joao.java, and Leticia.java. The Leticia.java tab is active, highlighted with a yellow bar. The code in the editor is:

```
1 package Casa2;
2
3 import Casa1.Maria;
4
5 public class Leticia {
6     Maria sogra = new Maria();
7
8     void teste(){
9         //System.out.println(sogra.secreto);
10        //System.out.println(sogra.facoDentroDeCasa);
11        //System.out.println(sogra.familiaSabe);
12        System.out.println(sogra.todoMundoSabe);
13    }
14}
```

The code demonstrates encapsulation by creating a private attribute `Maria sogra` and providing a public method `teste()` that prints the value of `sogra.todoMundoSabe`.

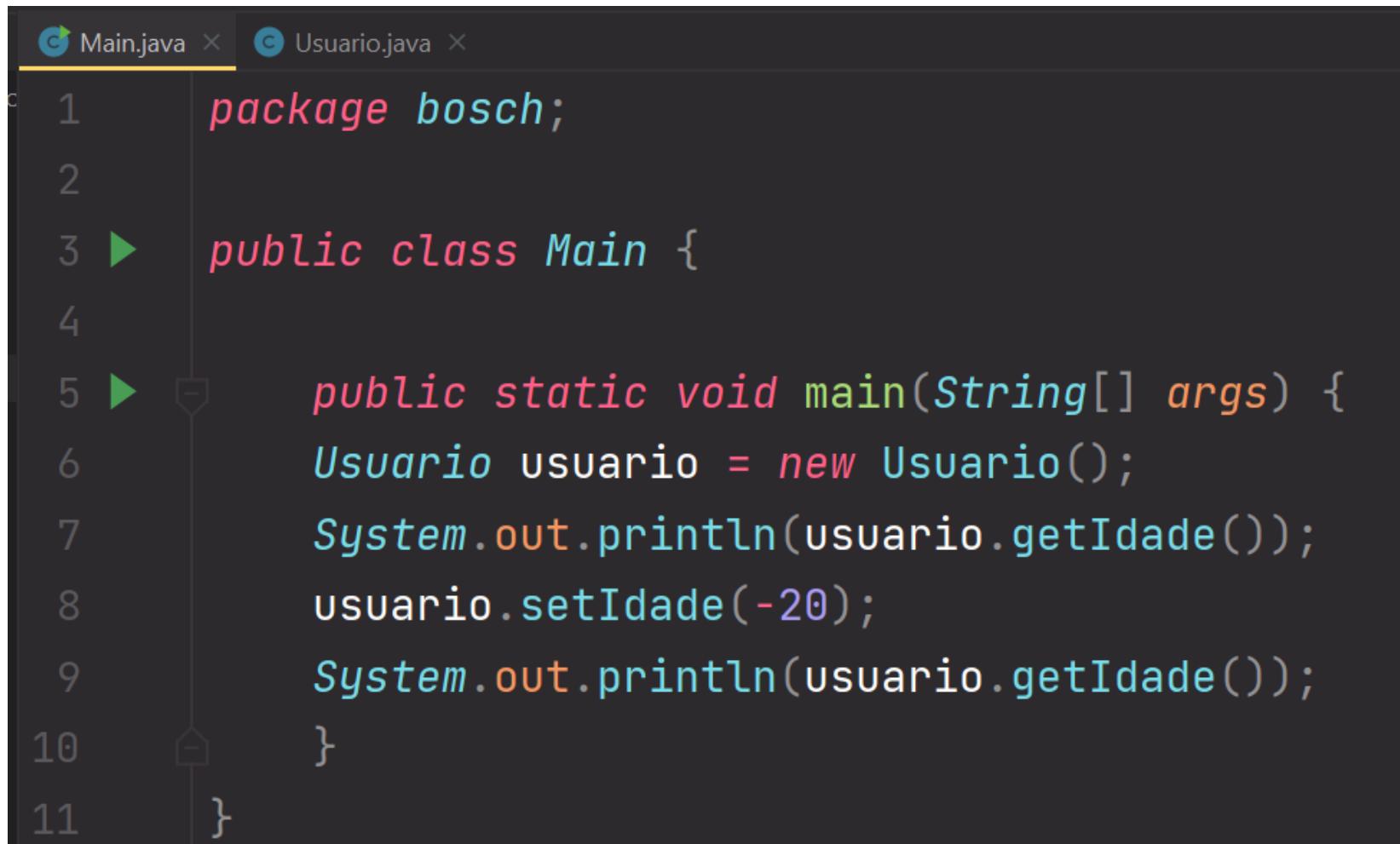
PROGRAMAÇÃO ORIENTADA A OBJETOS

Encapsulamento

```
1 package bosch;  
2  
3 public class Usuario {  
4     private int idade;  
5  
6     public int getIdade() {  
7         return idade;  
8     }  
9  
10    public void setIdade(int idade)  
11    {    idade=Math.abs(idade);  
12        this.idade = idade;  
13    }  
14}
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Encapsulamento



The image shows a screenshot of a Java code editor. The current file is `Main.java`, indicated by the yellow underline. The `Usuario.java` file is visible in the background. The code in `Main.java` is as follows:

```
1 package bosch;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         Usuario usuario = new Usuario();
7         System.out.println(usuario.getIdade());
8         usuario.setIdade(-20);
9         System.out.println(usuario.getIdade());
10    }
11 }
```

PROGRAMAÇÃO ORIENTADA A OBJETOS

Aplicativo Universidade

- Você deverá criar abstração de uma universidade no seu programa em Java.
- Com relação à composição:
 - Uma Universidade é composta por Institutos
 - Um Instituto é composto por Disciplinas
 - Uma Disciplina é composta por n Alunos, 1 Monitor, e 1 Docente.
- Com relação à Herança:
 - Existe a classe Membro
 - A classe Discente herda características da classe Membro
 - A classe Docente herda características da classe Membro
 - As classes Aluno e Monitor herdaram da classe Discente
 - As classes Professor, Coordenador e Diretor herdaram da classe Docente



PROGRAMAÇÃO ORIENTADA A OBJETOS

Aplicativo Universidade

- Funcionalidades
 - Todos as classes devem sobrescreve o método `toString` com uma resumo dos elementos dentro da classe.
 - Para cada aluno serão adicionadas notas presença e será calculada a média.
 - Para o aluno ser aprovado a média dever ser maior ou igual a 5 e presença maior que 75
 - Caso mais de 80% da turma for aprovada:
 - O monitor recebe 5% de aumento na bolsa e recebe a avaliação boa
 - Se o docente for um professor ele recebe 10% de aumento
 - Se o docente for um coordenador ele recebe 15% de aumento
 - Se o docente for um diretor ele recebe 20% de aumento
 - Caso menos de 50% da turma for aprovada:
 - O monitor recebe 5% de redução de salario e recebe a avaliação ruim
 - Se o docente for um professor ele recebe 10% de desconto no salario
 - Se o docente for um coordenador ele recebe 15% de desconto no salário
 - Se o docente for um diretor ele recebe 20% de desconto no salario.

PROGRAMAÇÃO ORIENTADA A OBJETOS

Aplicativo Universidade

- Salários Iniciais:
 - Monitor: 2.000 R\$
 - Professor: 7.000 R\$
 - Coordenador: 11.000 R\$
 - Diretor: 16.000 R\$

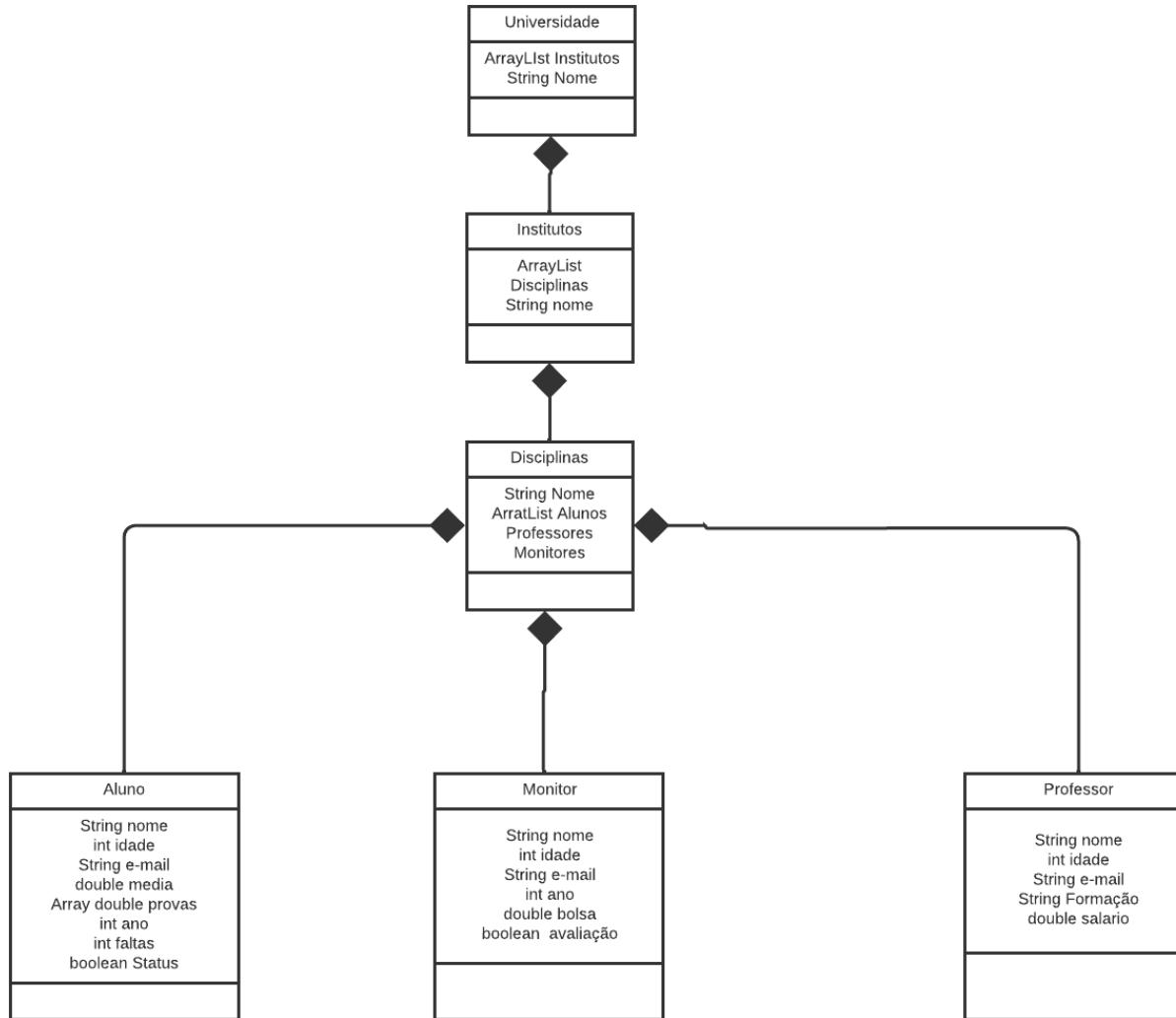
PROGRAMAÇÃO ORIENTADA A OBJETOS

Aplicativo Universidade

- Desafios para a classe disciplina
 - Para cada turma exiba a média da sala
 - O nome e a média do aluno com maior média
 - O nome e a média aluno com menor média
 - O nome e a idade do aluno mais velho
 - O nome e a idade do aluno mais novo

PROGRAMAÇÃO ORIENTADA A OBJETOS

Aplicativo Universidade



PROGRAMAÇÃO ORIENTADA A OBJETOS

Aplicativo Universidade

