

# PROGRAMAÇÃO ORIENTADA A OBJETOS

# JAVA

# PROGRAMAÇÃO ORIENTADA A OBJETOS

## Java

1995



ORIENTADA  
A OBJETO

ROBUSTA

PORTATIL

OPERAÇÃO  
EM REDE

SEGURANÇA

# PROGRAMAÇÃO ORIENTADA A OBJETOS

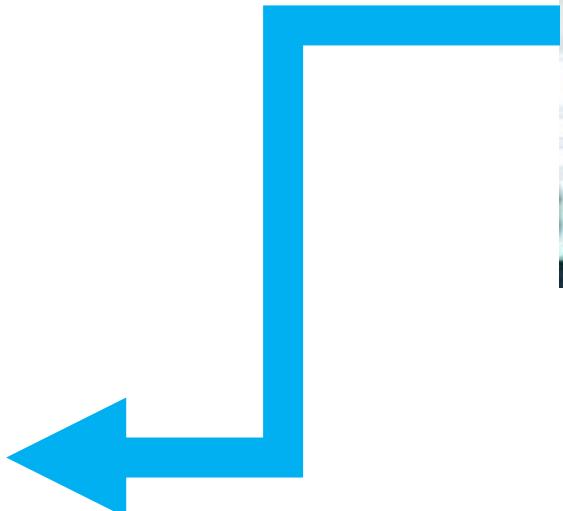
## Java

► 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

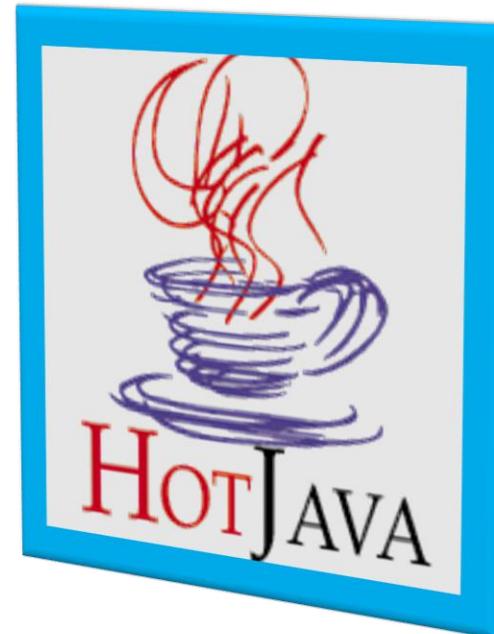


# PROGRAMAÇÃO ORIENTADA A OBJETOS

## 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

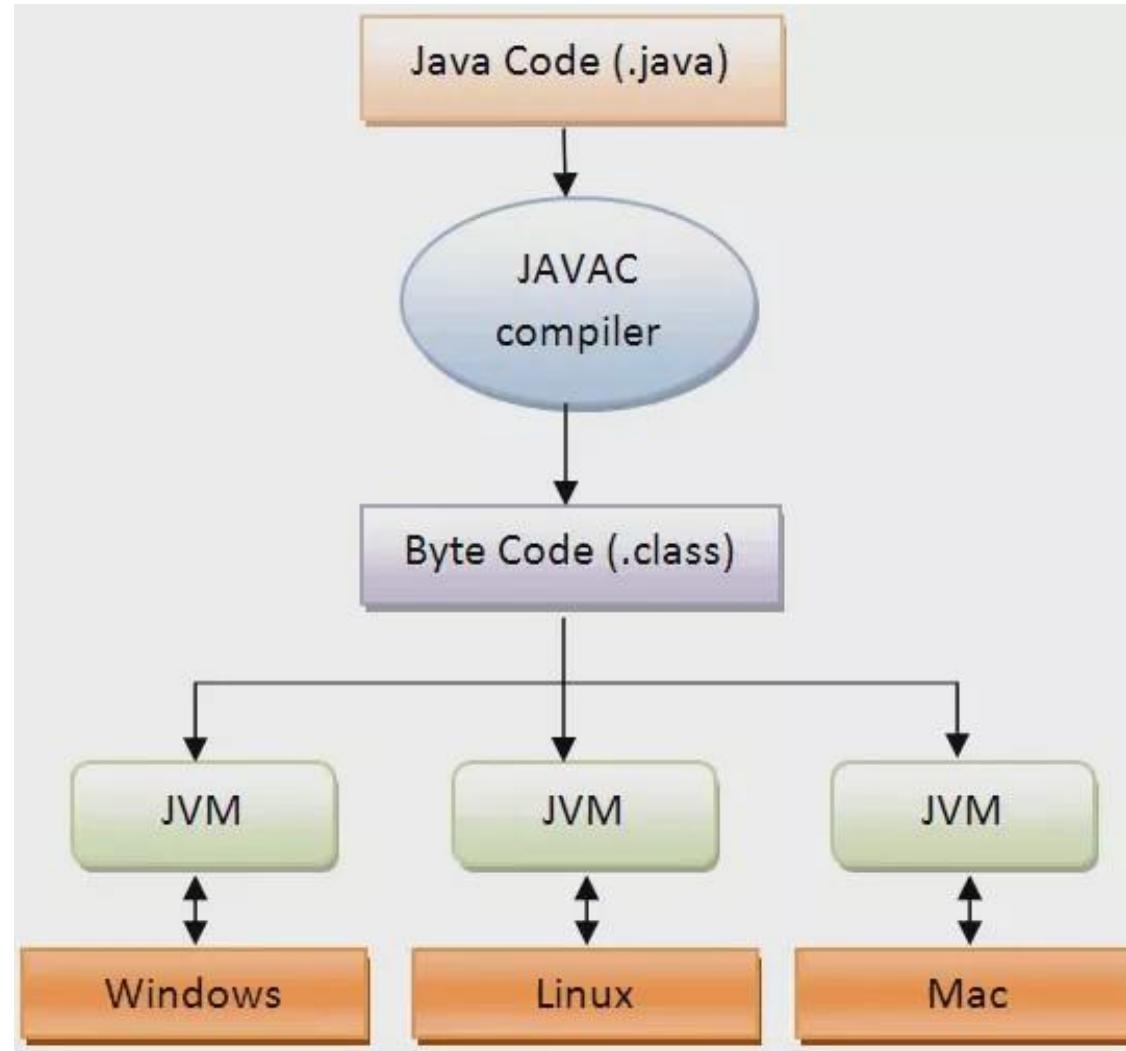


2009



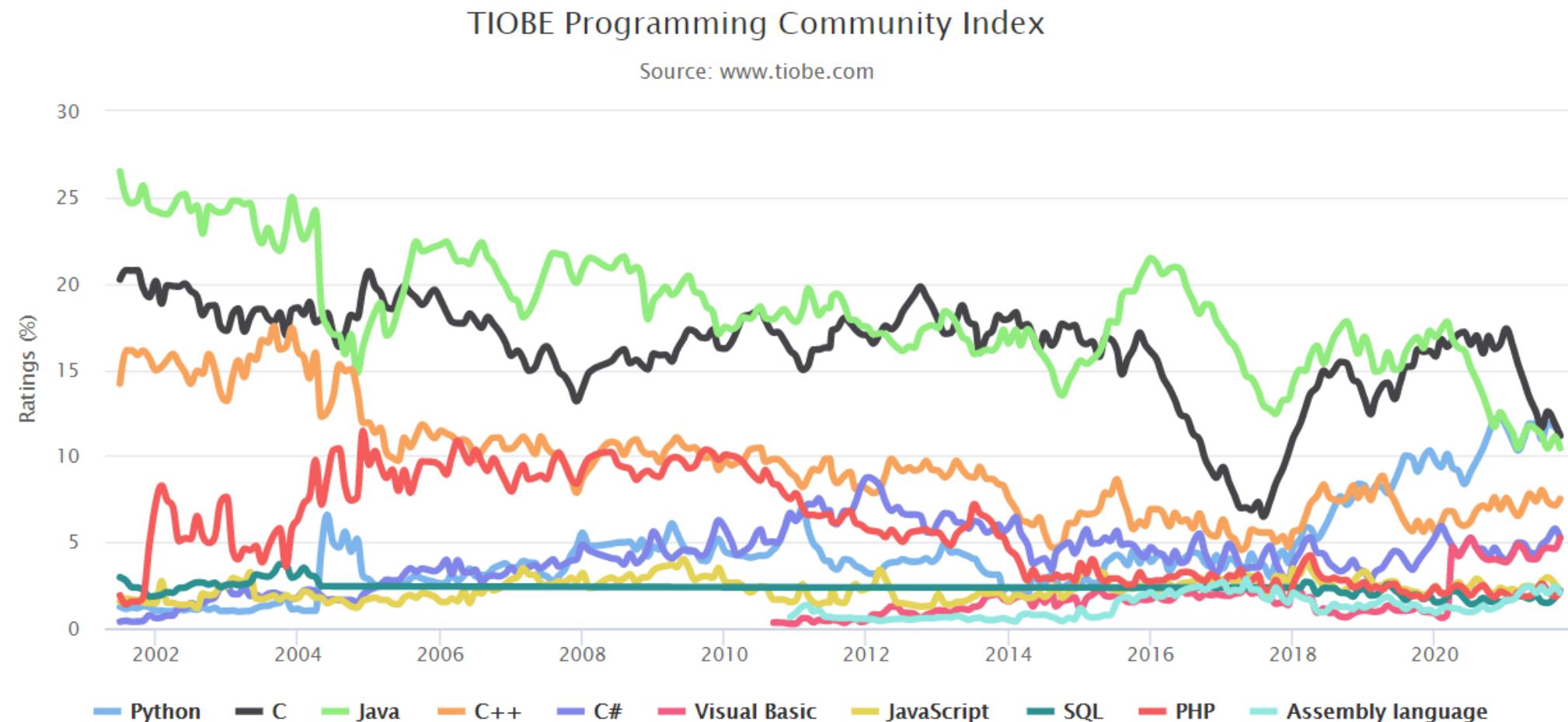
# PROGRAMAÇÃO ORIENTADA A OBJETOS

## 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%

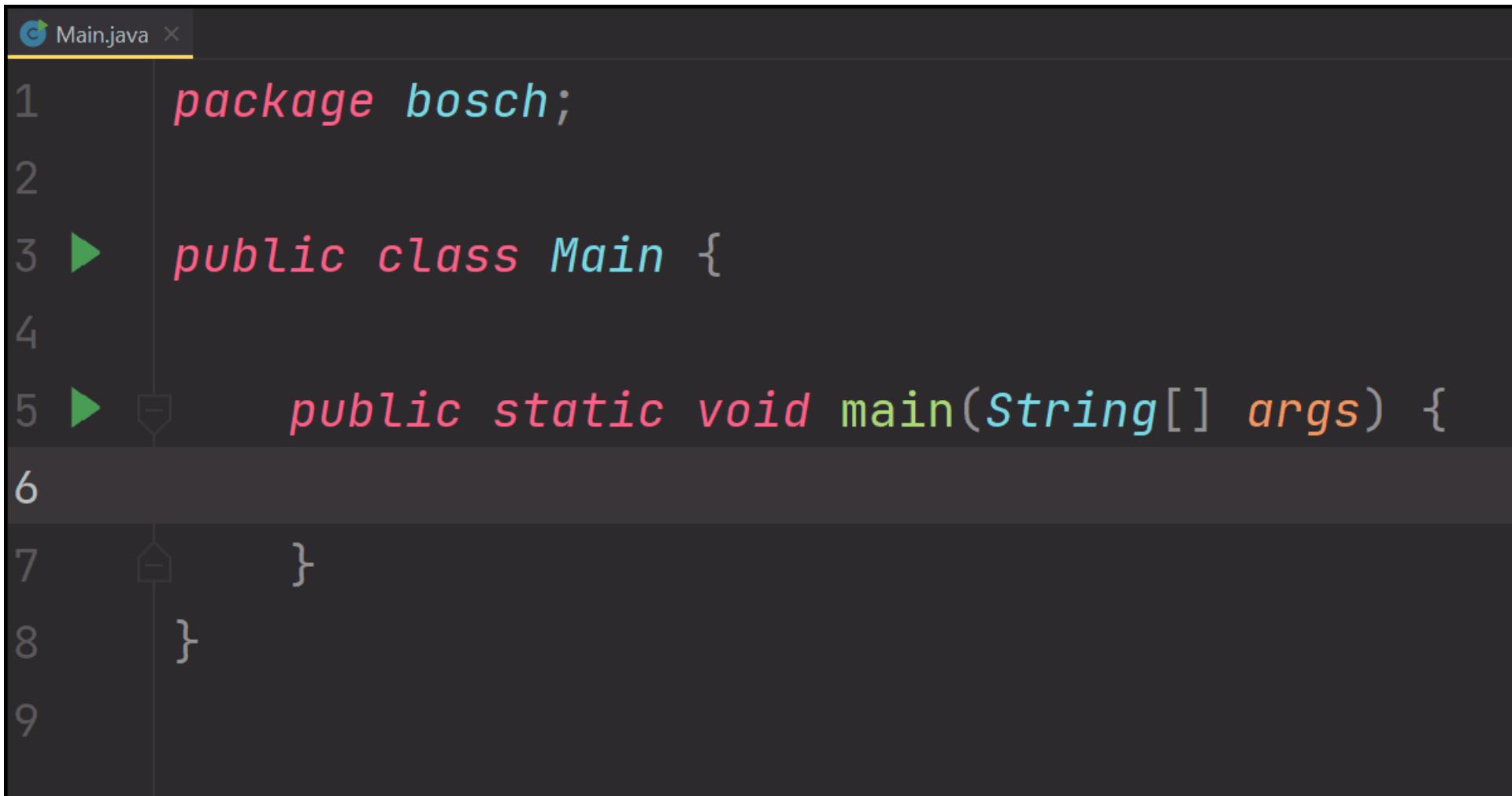
# PROGRAMAÇÃO ORIENTADA A OBJETOS

## Linguagens de Programação mais populares



# PROGRAMAÇÃO ORIENTADA A OBJETOS

## JAVA – Comentário de uma linha



The screenshot shows a Java code editor with a dark theme. The file is named "Main.java". The code contains a single-line comment starting at line 3:

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

The comment starts with a double slash (//) and spans from line 3 to line 8.

# PROGRAMAÇÃO ORIENTADA A OBJETOS

## Java – Comentário de Múltiplas linhas

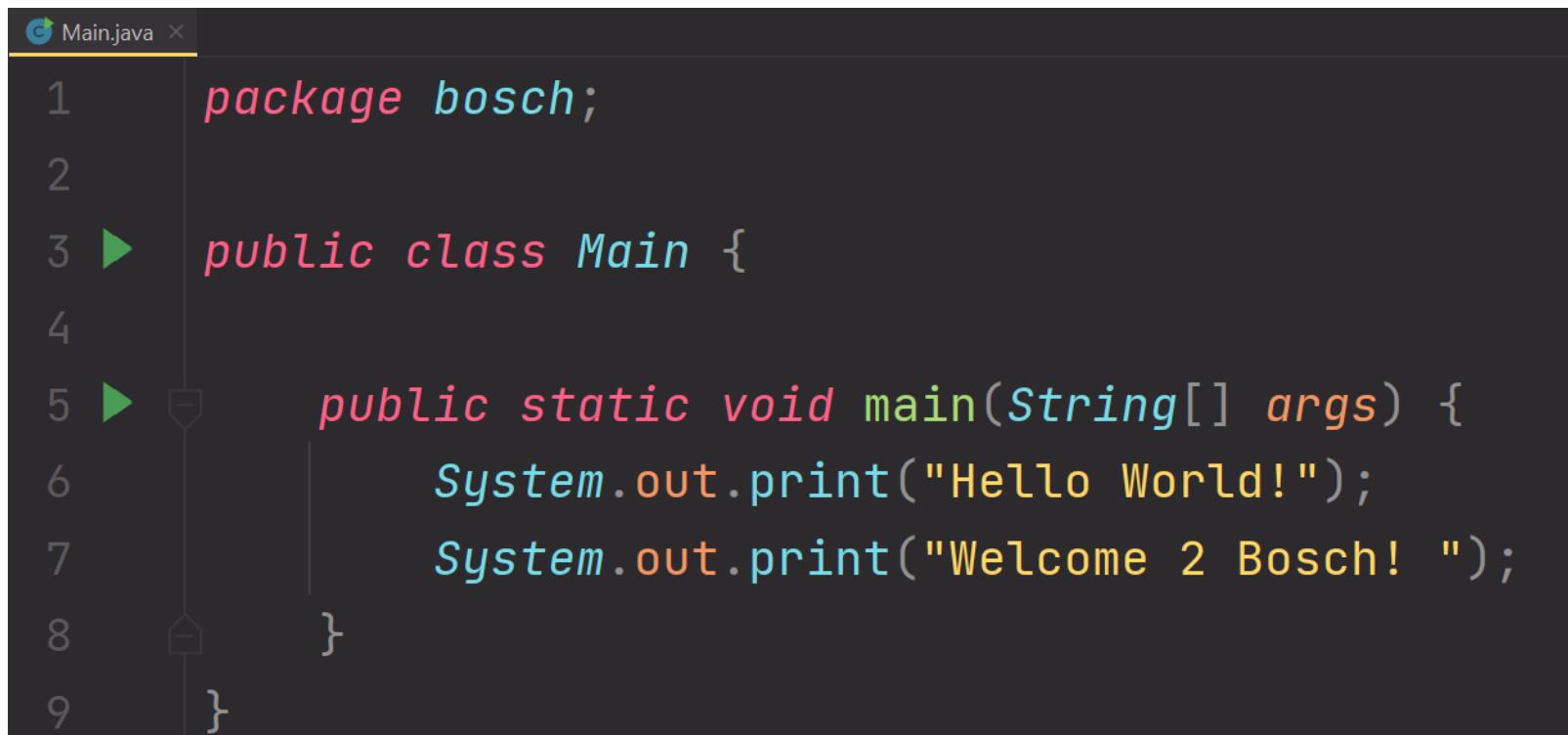
The screenshot shows a code editor window titled "Main.java". The code is as follows:

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

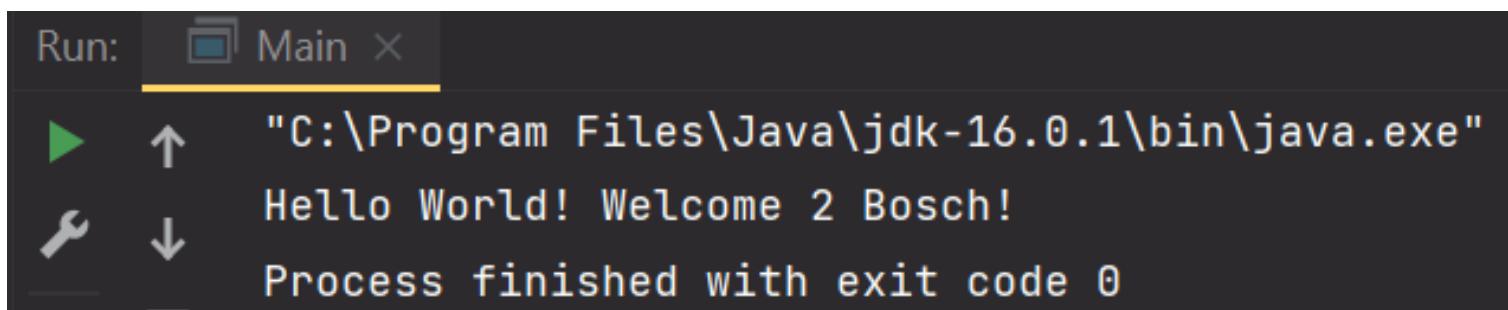
The code uses syntax highlighting where "package", "public", "class", "static", "void", and "String" are in blue, and "args" is in green. The "main" method signature is preceded by a green triangle icon. The opening brace of the class definition is preceded by a green triangle icon, and the closing brace of the class definition is preceded by a yellow square icon. The opening brace of the "main" method is preceded by a green triangle icon, and its closing brace is preceded by a yellow square icon.

# 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:

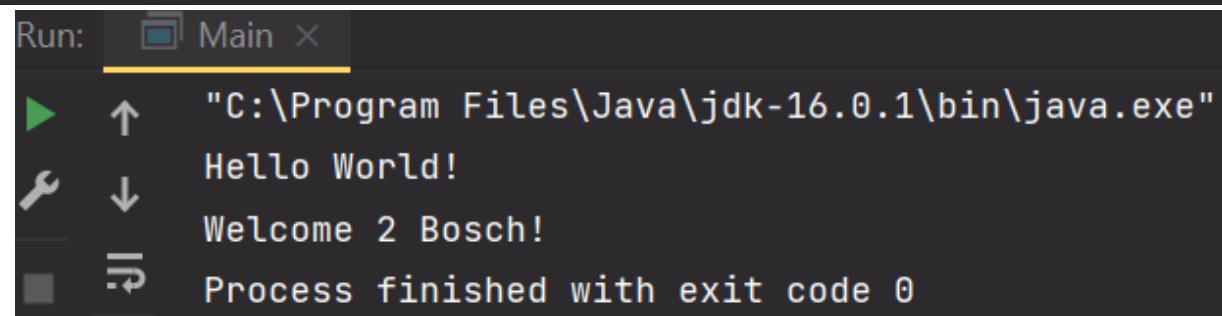
```
Run: Main
▶ ↑ "C:\Program Files\Java\jdk-16.0.1\bin\java.exe"
Hello World!
Welcome 2 Bosch!
Process finished with exit code 0
```

# PROGRAMAÇÃO ORIENTADA A OBJETOS

## Java – Saída não formatada



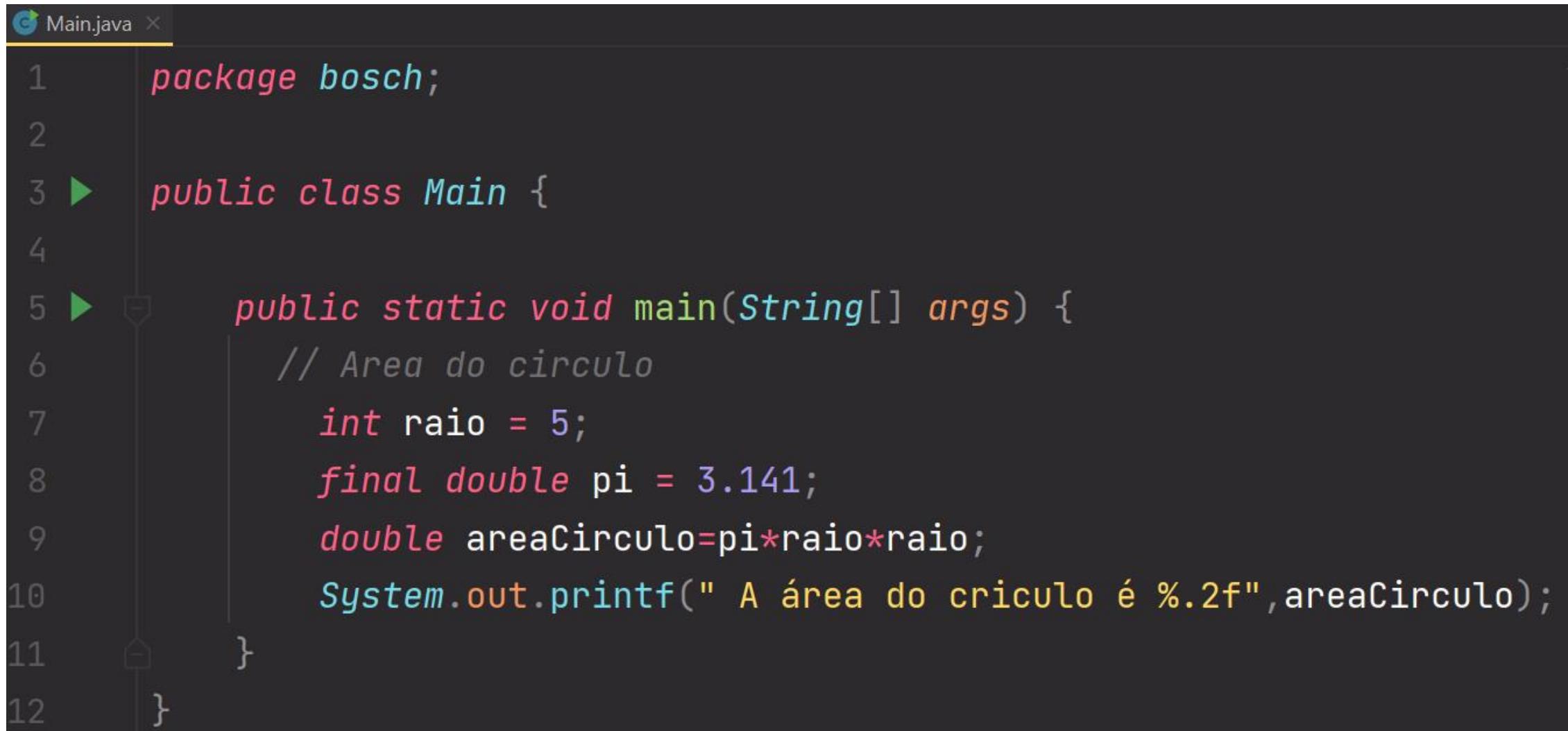
```
Main.java
1 package bosch;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         System.out.println("Hello World! ");
7         System.out.print("Welcome 2 Bosch! ");
8     }
9 }
```



```
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



```
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

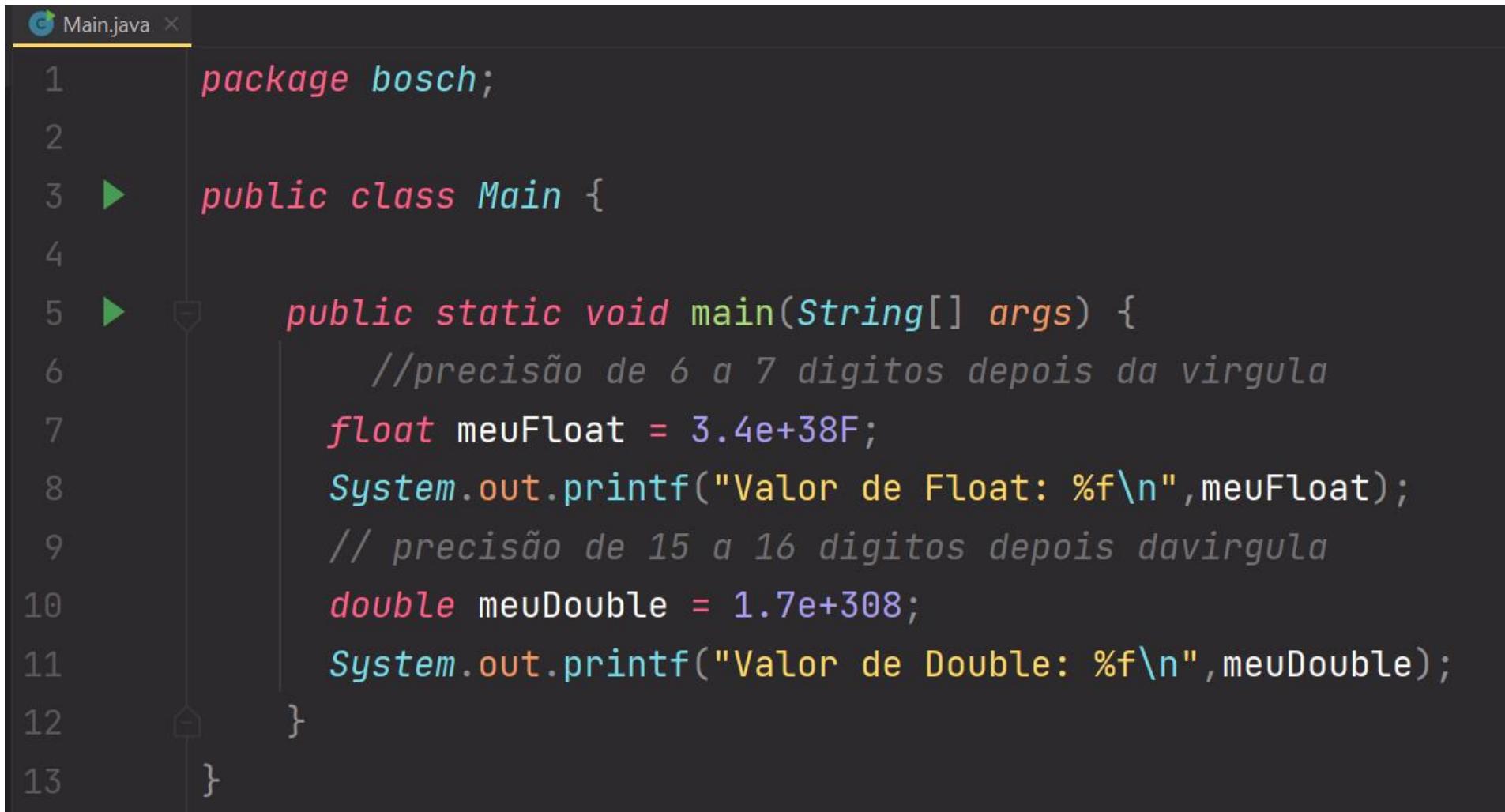
# PROGRAMAÇÃO ORIENTADA A OBJETOS

## 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

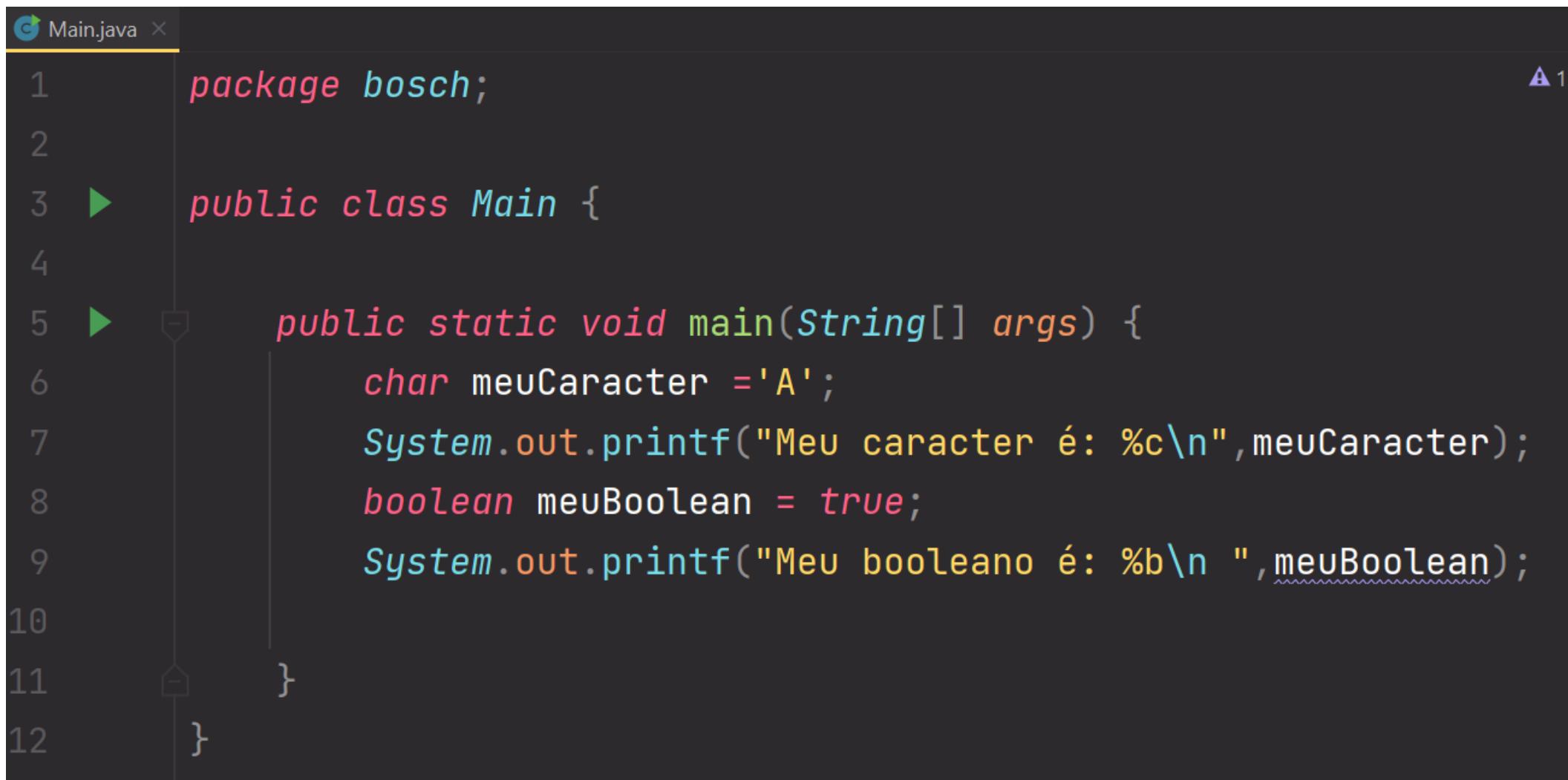
## Java – Tipos de Dados Primitivos



```
>Main.java ×
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 file named "Main.java". The code demonstrates the use of primitive data types: character ('char') and boolean ('boolean'). The code is as follows:

```
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 }
```

# PROGRAMAÇÃO ORIENTADA A OBJETOS

## Java – String

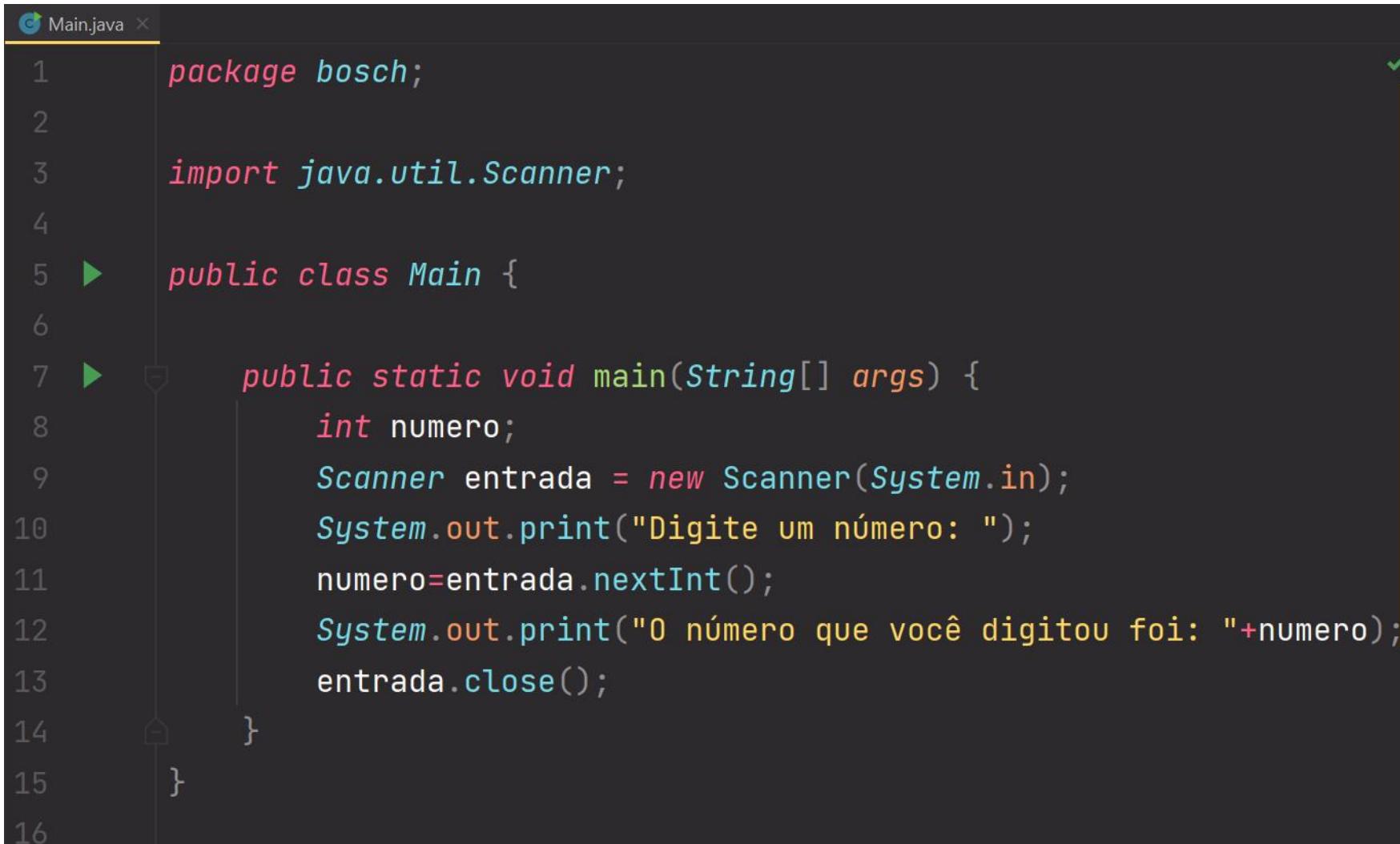
The screenshot shows a Java code editor window with the file 'Main.java' open. The code is as follows:

```
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 }
```

The code is color-coded: 'package' and 'String' are in blue, 'bosch' and 'System' are in green, and the class name 'Main' and variable name 'minhaString' are in red. The code editor has a dark theme with light-colored text. There are some small icons on the left margin, likely indicating code completion or refactoring suggestions.

# PROGRAMAÇÃO ORIENTADA A OBJETOS

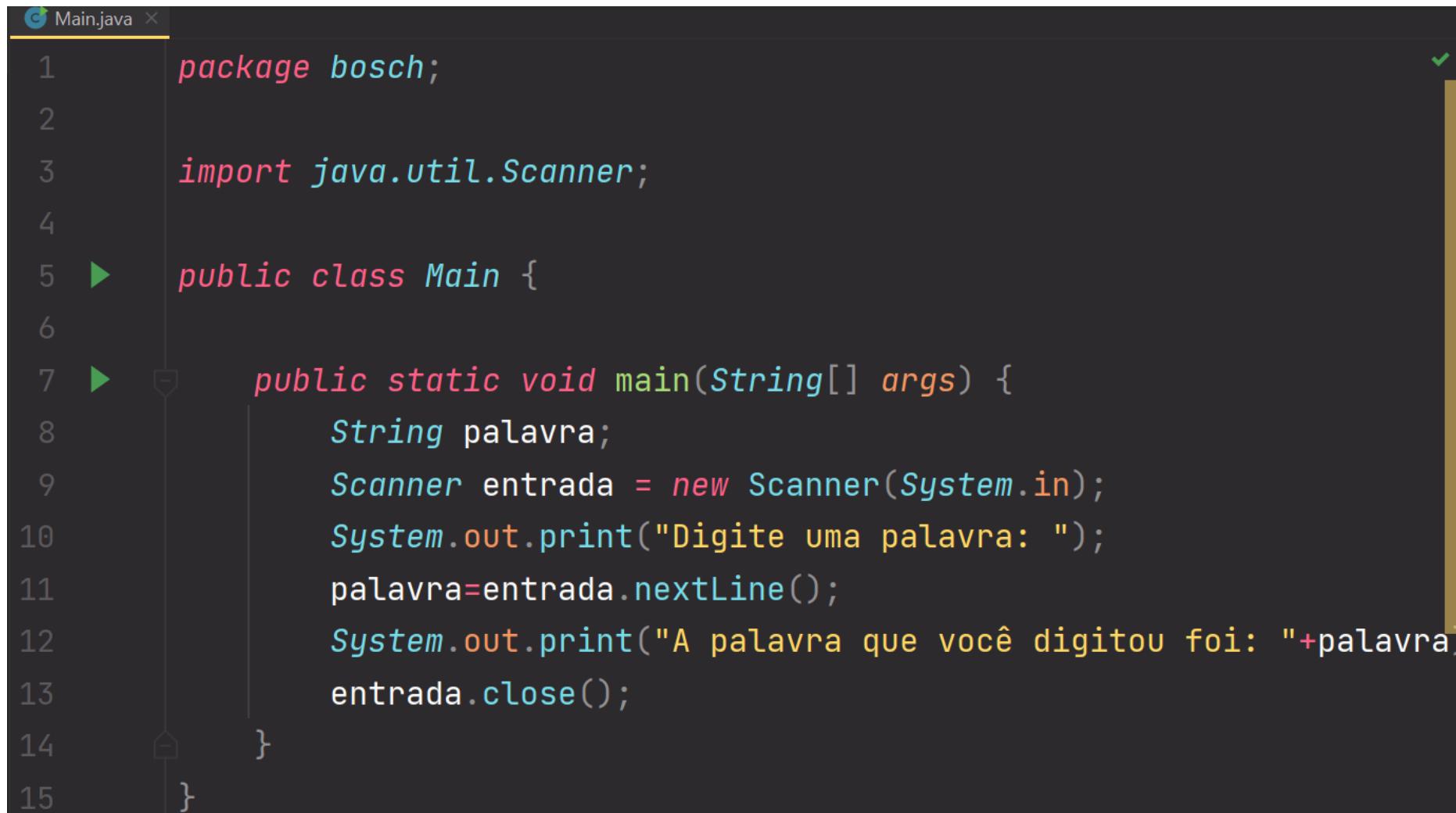
## Java – Tipos de Dados Primitivos



```
>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         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 }
```

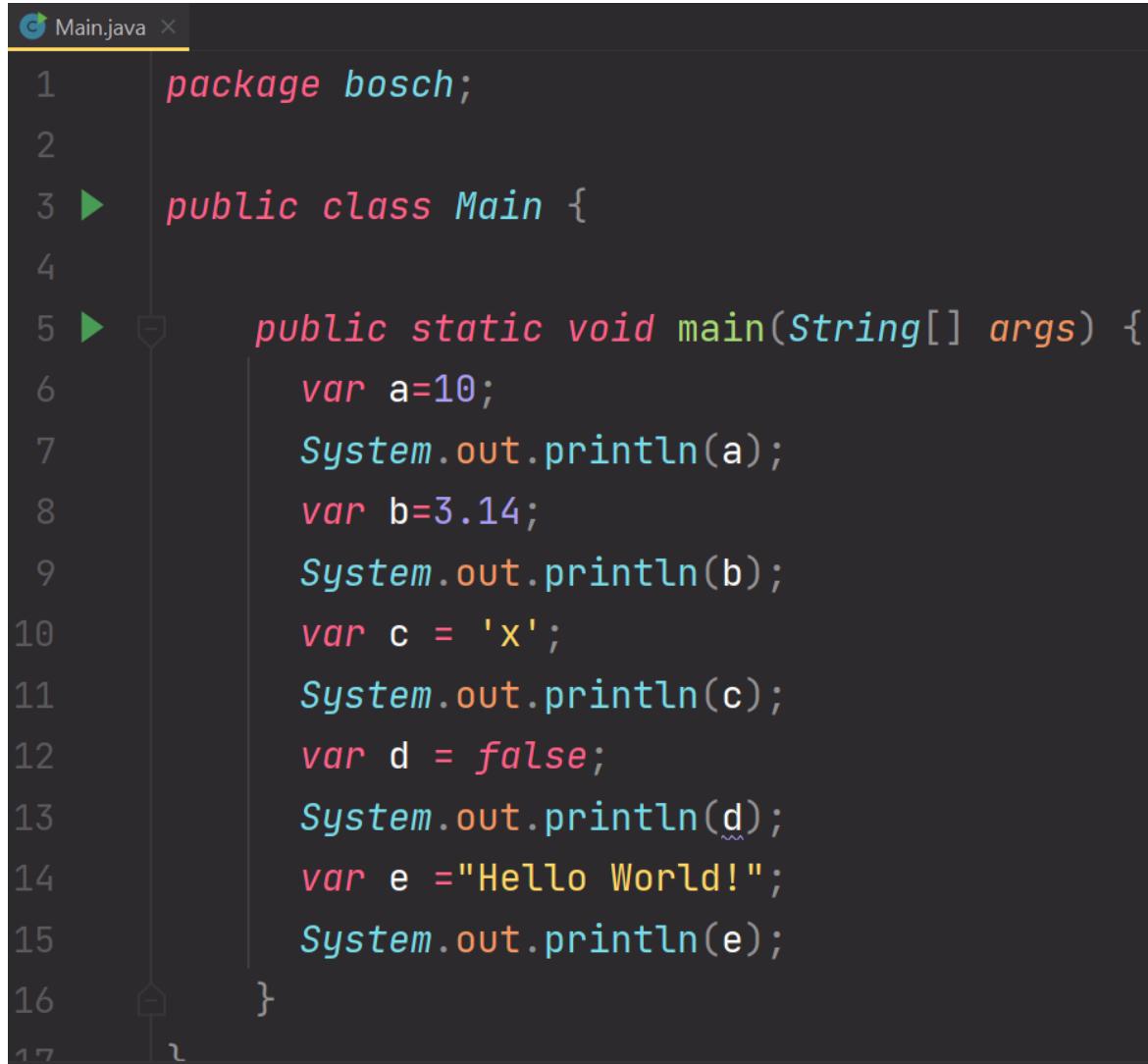
# PROGRAMAÇÃO ORIENTADA A OBJETOS

## 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 demonstrates Java's type inference feature using the 'var' keyword. 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         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}
```

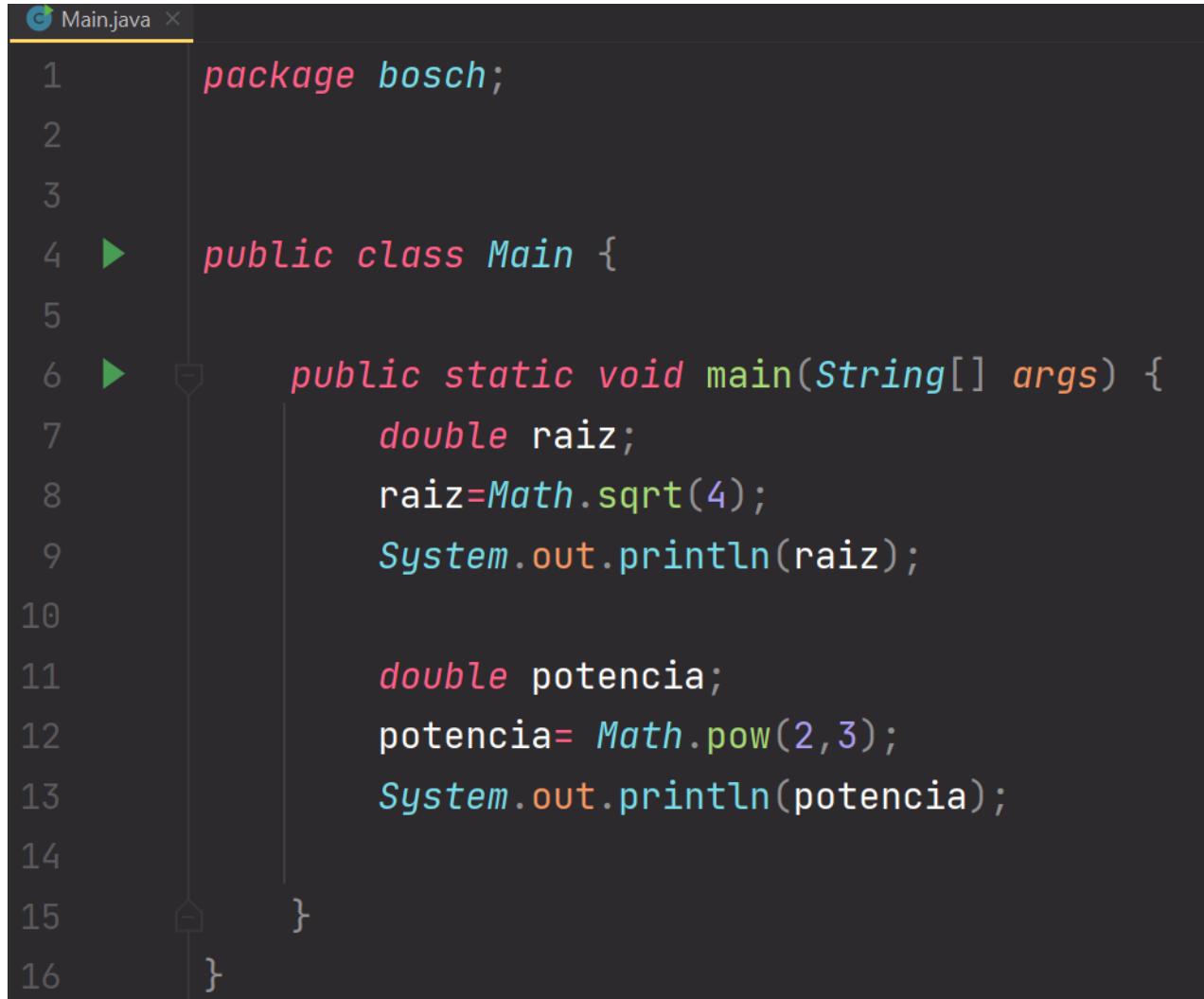
# PROGRAMAÇÃO ORIENTADA A OBJETOS

## Java – Convertendo para Objeto e Verificando o Tipo

```
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}
```

# PROGRAMAÇÃO ORIENTADA A OBJETOS

## Java – Classe Math

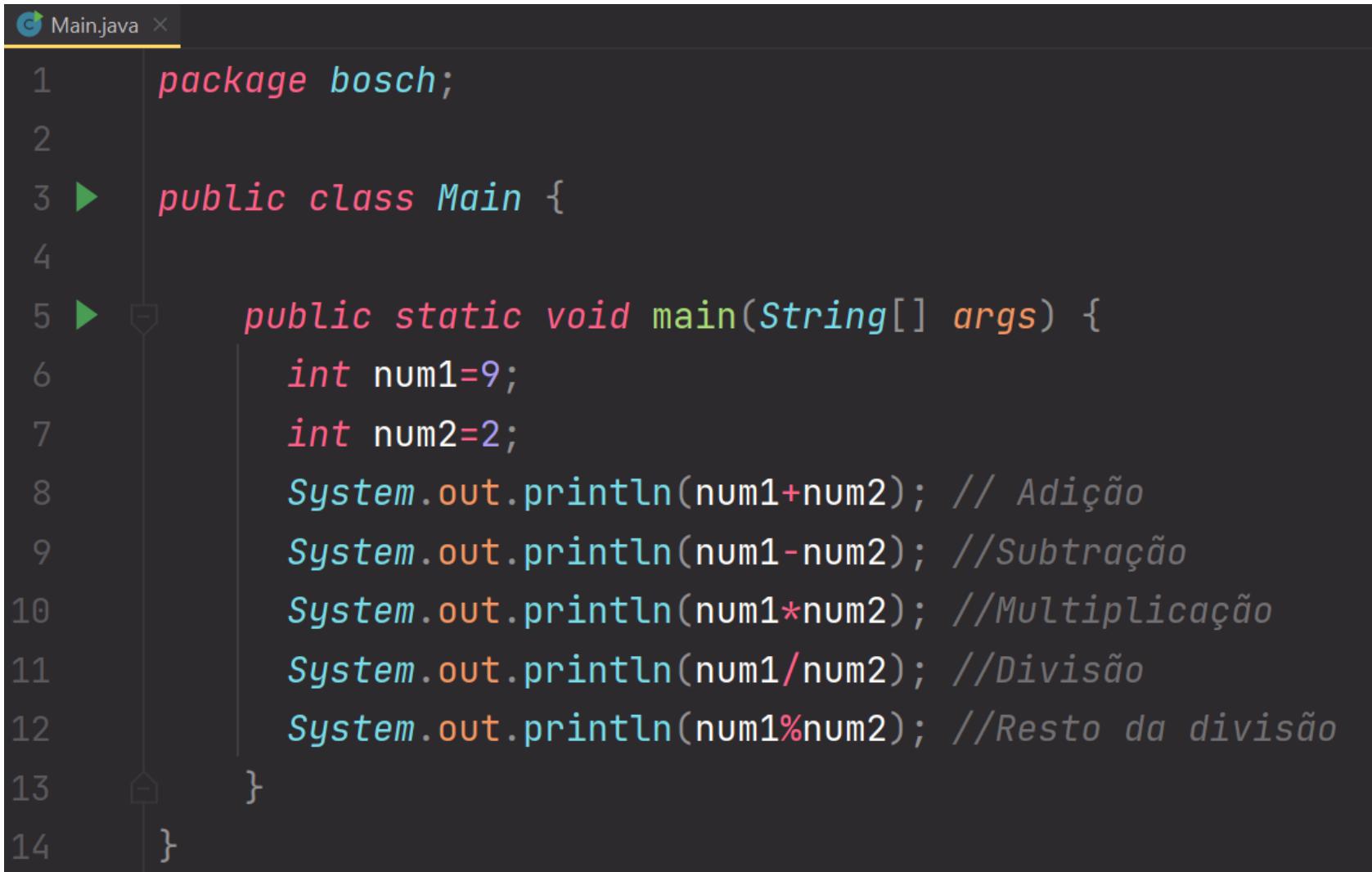


The screenshot shows a Java code editor with a dark theme. The file is named 'Main.java'. The code demonstrates the use of the Math class to calculate square root and power.

```
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 }
```

# PROGRAMAÇÃO ORIENTADA A OBJETOS

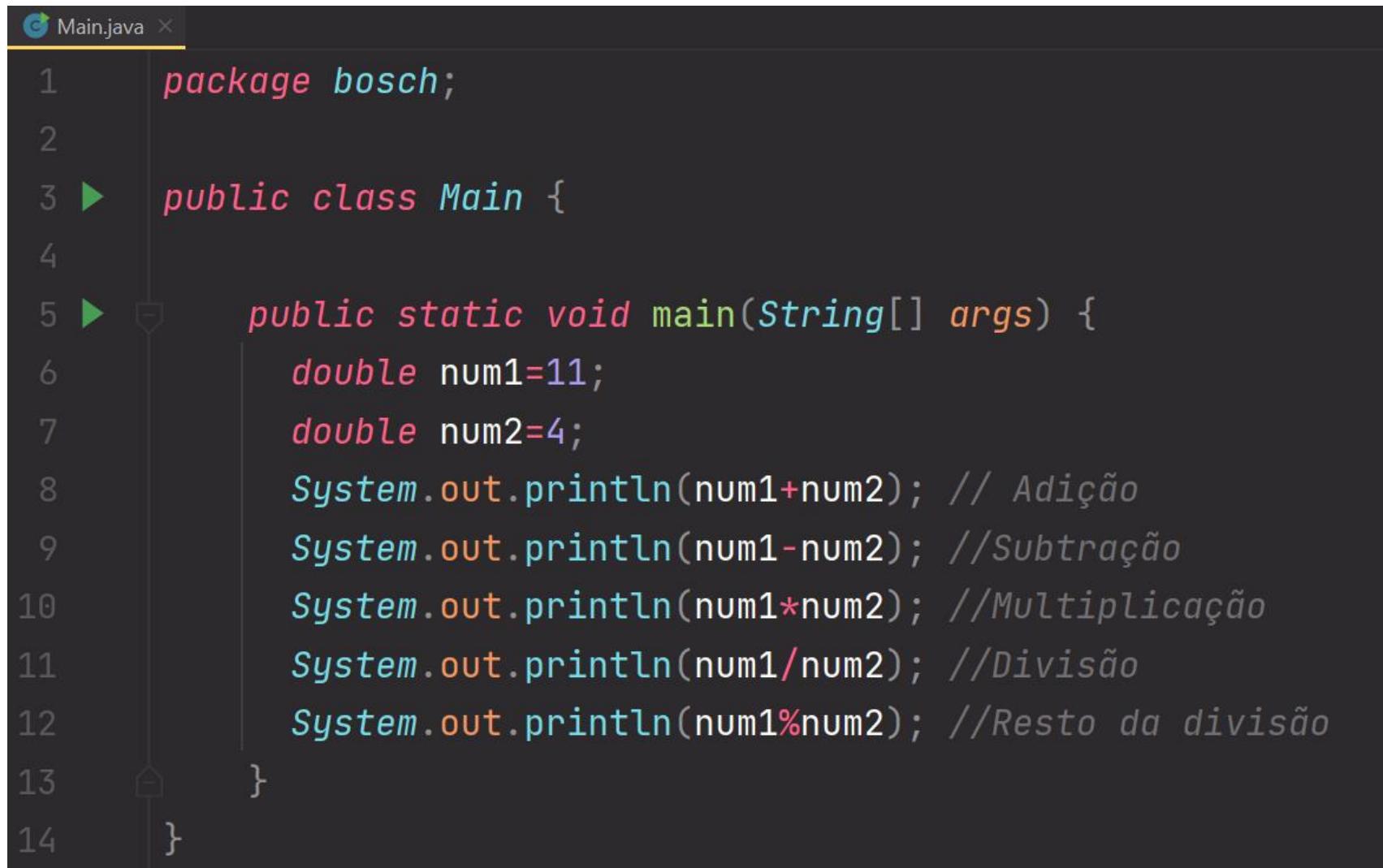
## Java – Operadores Aritméticos



```
>Main.java x
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

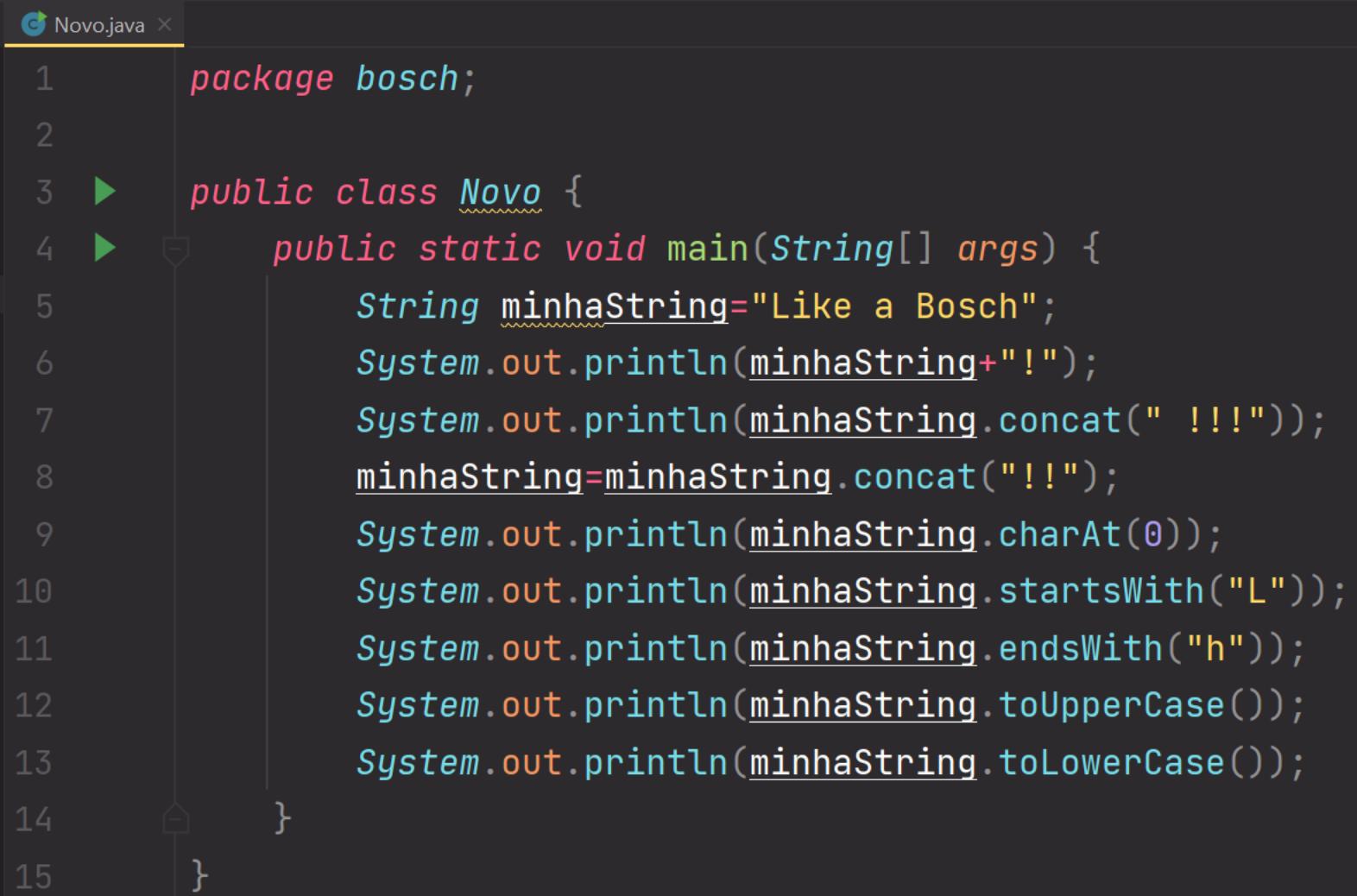
# PROGRAMAÇÃO ORIENTADA A OBJETOS

## 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

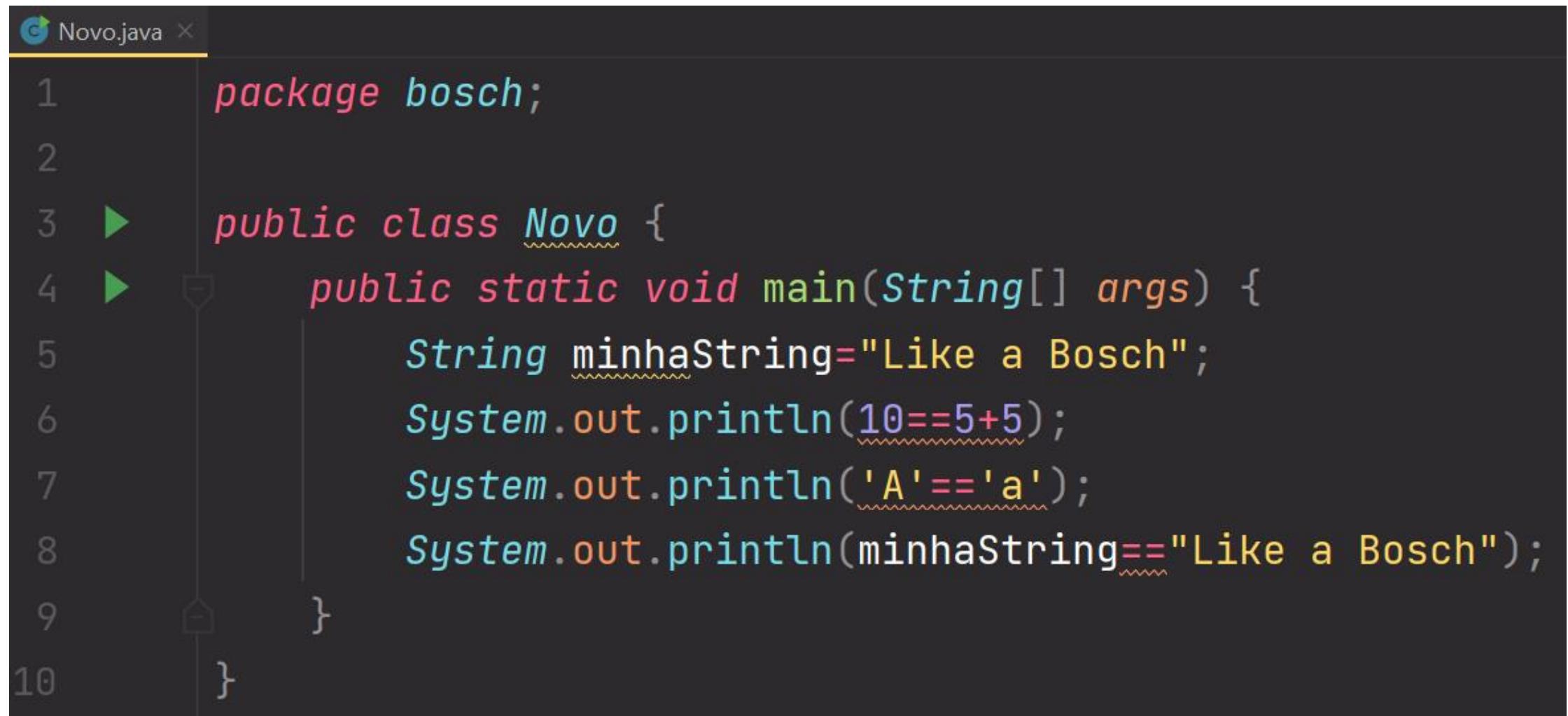


The screenshot shows a Java code editor with a dark theme. The file is named 'Novo.java'. The code defines a class 'Novo' with a main method. It creates a string 'minhaString' with the value "Like a Bosch", then prints it and its concatenation with three exclamation marks. It then concatenates two exclamation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

```
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(minhaString+"!");
7         System.out.println(minhaString.concat(" !!!"));
8         minhaString=minhaString.concat("!!!");
9         System.out.println(minhaString.charAt(0));
10        System.out.println(minhaString.startsWith("L"));
11        System.out.println(minhaString.endsWith("h"));
12        System.out.println(minhaString.toUpperCase());
13        System.out.println(minhaString.toLowerCase());
14    }
15 }
```

# PROGRAMAÇÃO ORIENTADA A OBJETOS

## Java – Classe String

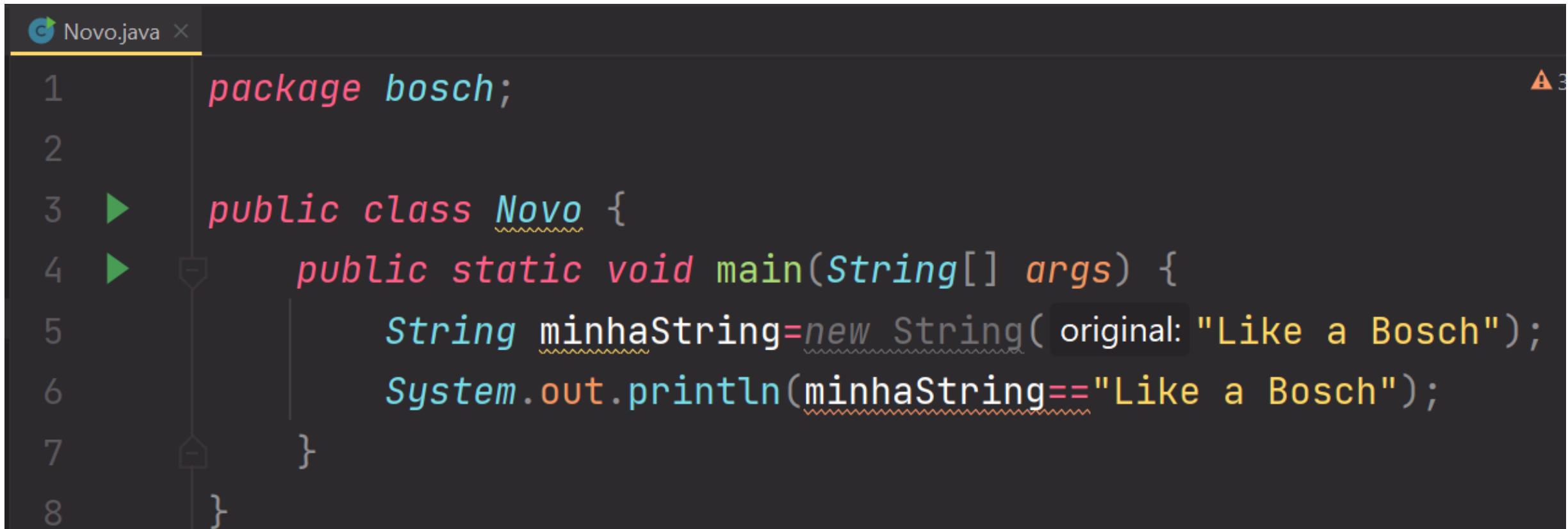


The screenshot shows a Java code editor with a dark theme. The file is named "Novo.java". The code is as follows:

```
1 package bosch;
2
3 public class Novo {
4     public static void main(String[] args) {
5         String minhaString="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 }
```

# PROGRAMAÇÃO ORIENTADA A OBJETOS

## Java – Classe String



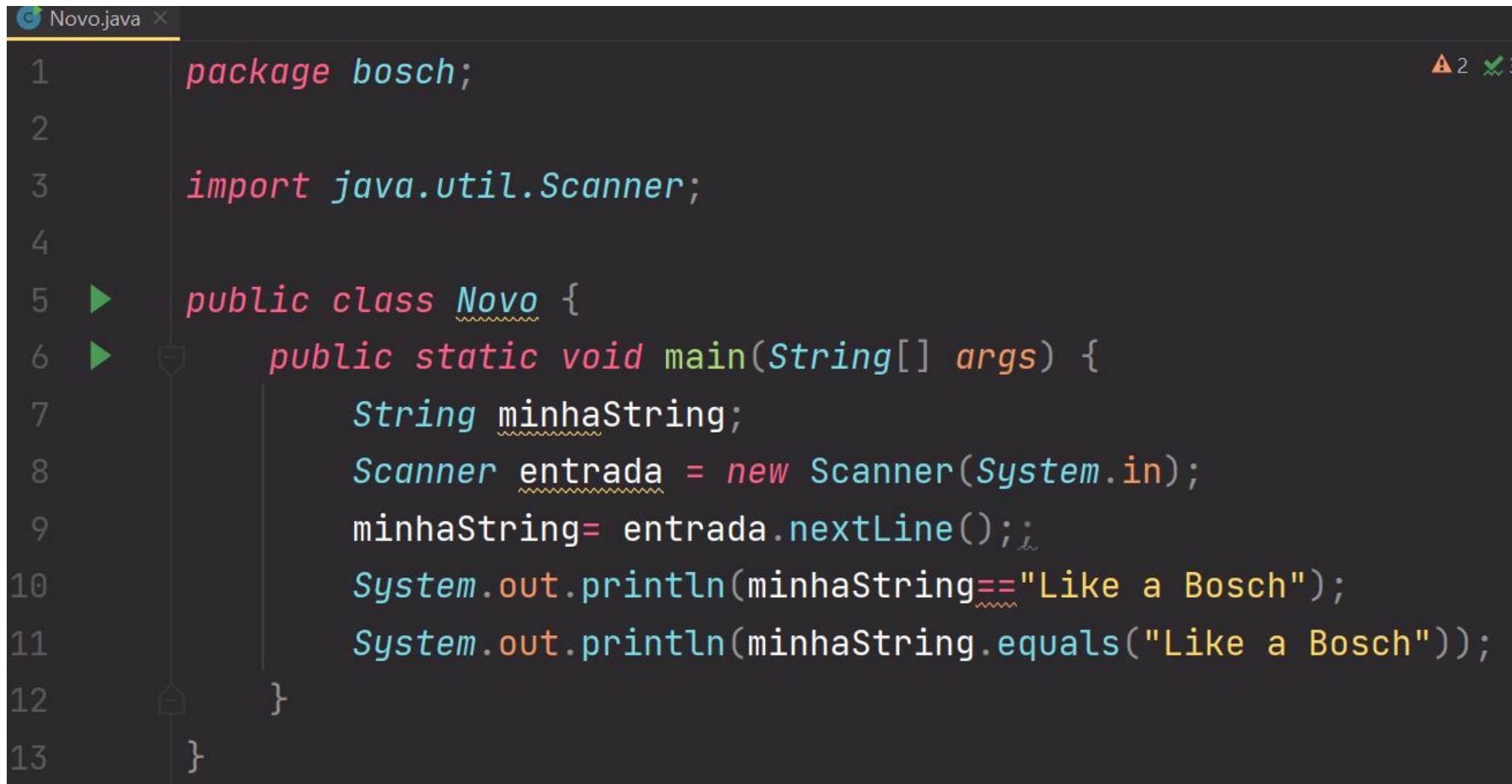
The screenshot shows a Java code editor with a dark theme. The file is named 'Novo.java'. The code is as follows:

```
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' and 'String' in red, while 'minhaString' and 'original' are underlined in red. A tooltip for 'original' shows the value 'Like a Bosch'. The 'main' method signature has a yellow warning icon with 'A 3' above it. The code editor interface includes tabs for 'Novo.java' and 'X', and a status bar at the bottom.

# PROGRAMAÇÃO ORIENTADA A OBJETOS

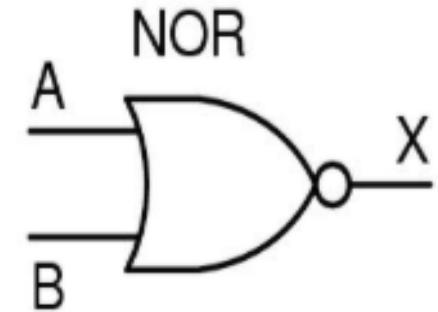
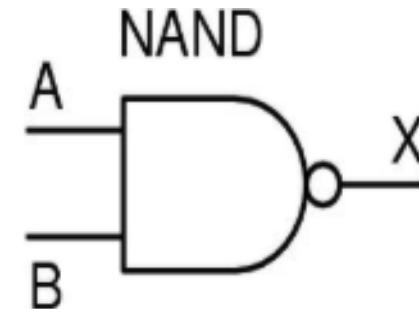
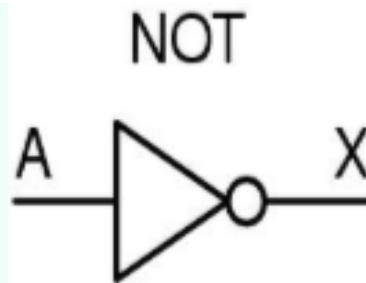
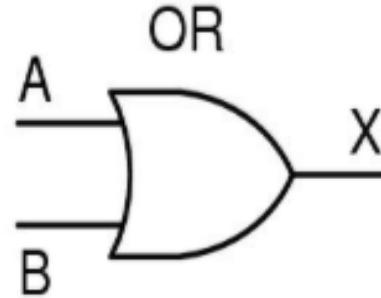
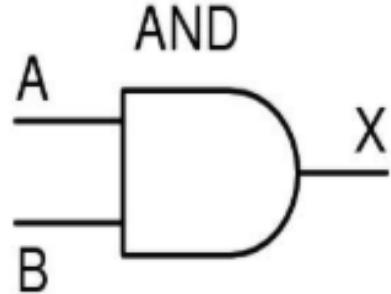
## Java – Classe String



```
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

## Tabela Verdade



A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

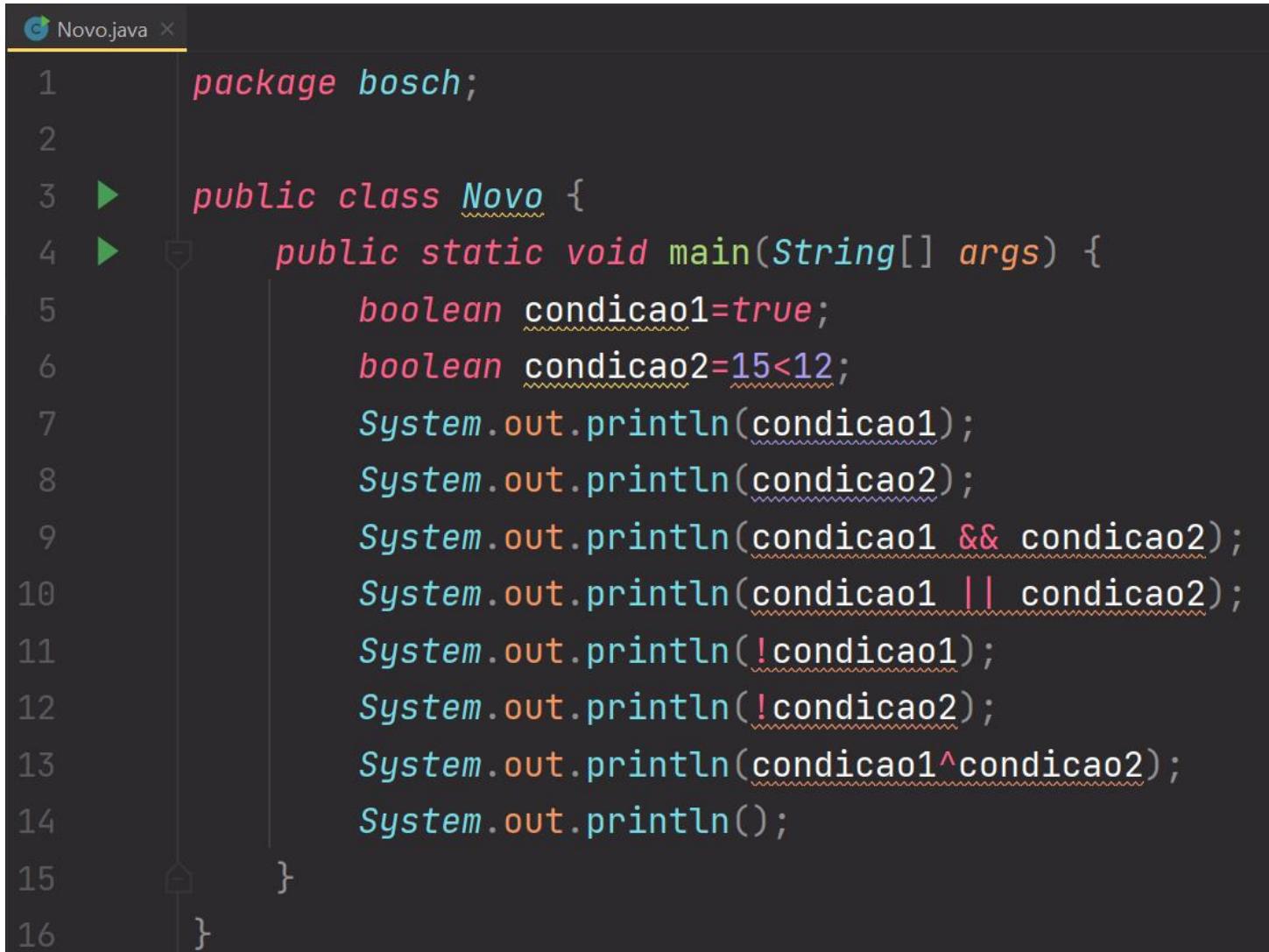
A	X
0	1
1	0

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

# PROGRAMAÇÃO ORIENTADA A OBJETOS

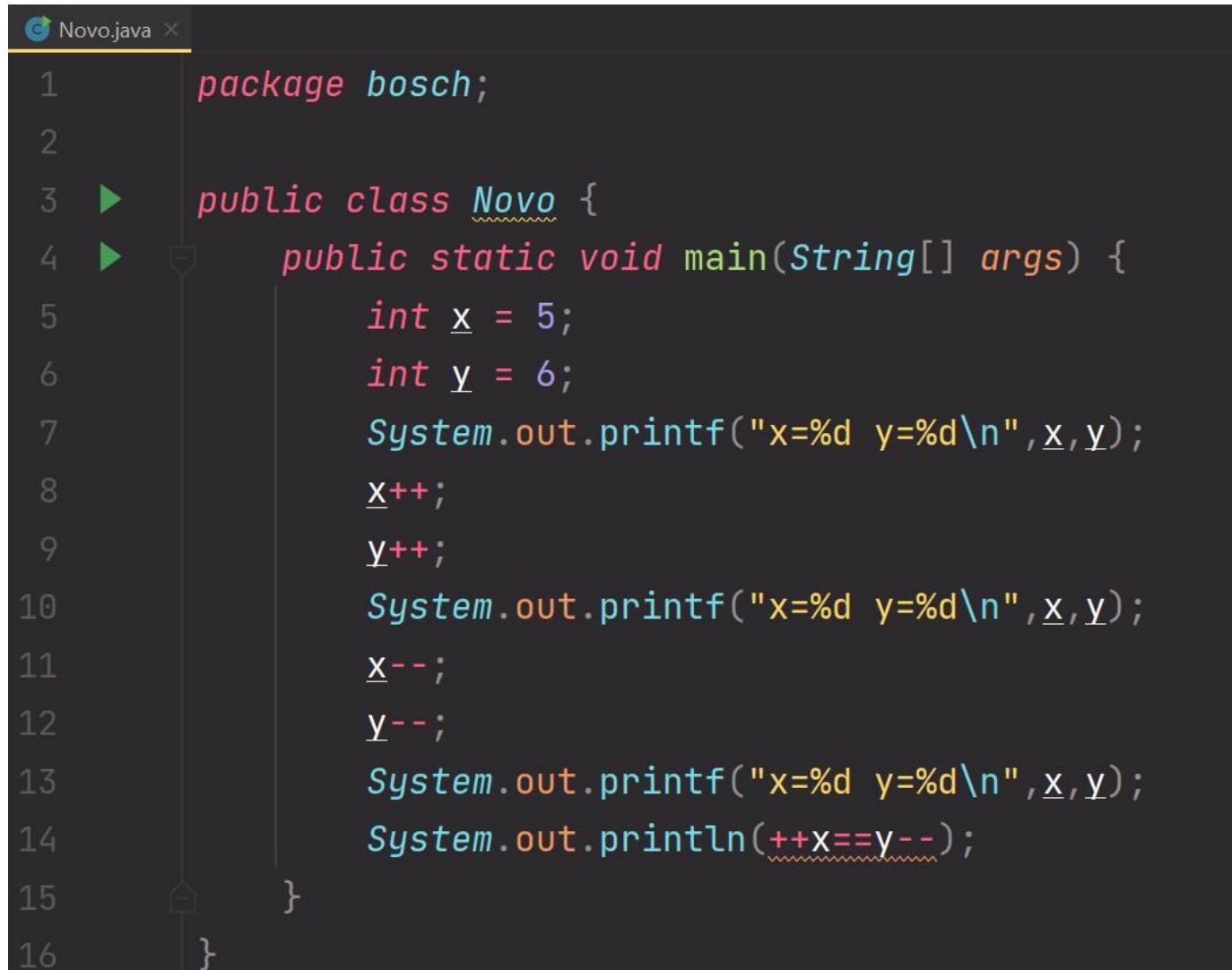
## Java – Operadores Lógicos



```
Novo.java
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



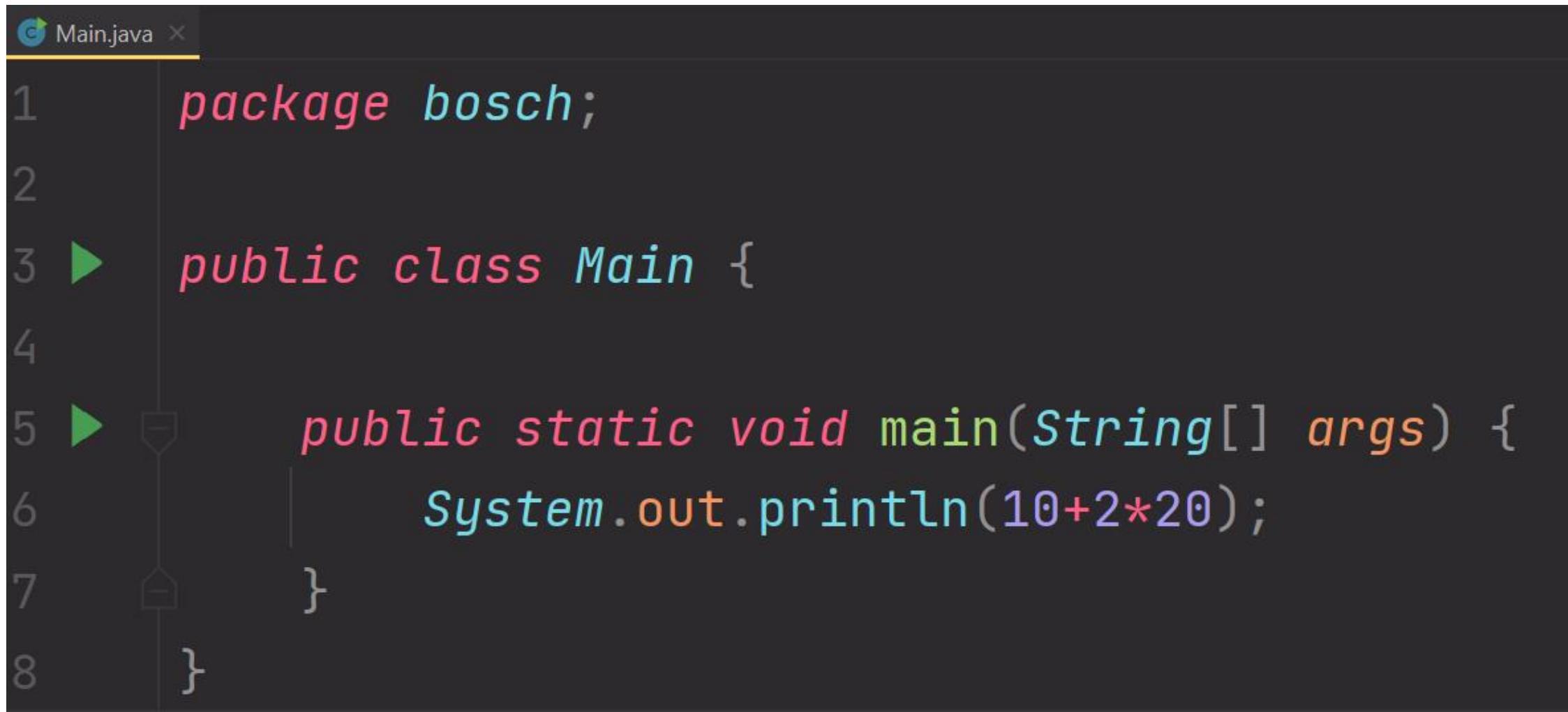
```
Novo.java x
1 package bosch;
2
3 ► public class Novo {
4 ►     public static void main(String[] args) {
5         int x = 5;
6         int y = 6;
7         System.out.printf("x=%d y=%d\n", x, y);
8         x++;
9         y--;
10        System.out.printf("x=%d y=%d\n", x, y);
11        x--;
12   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

# PROGRAMAÇÃO ORIENTADA A OBJETOS

## Tabela de Precedência

Nível	Operador(es)
9	<code>++ (sufixo) -- (sufixo)</code>
8	<code>* / %</code>
7	<code>+ -</code>
6	<code>&gt; &gt;= &lt; &lt;=</code>
5	<code>== !=</code>
4	<code>&amp;&amp;</code>
3	<code>  </code>
2	<code>?:</code>
1	<code>= &lt;operador&gt;= (sinal de atribuição composto)</code>

# PROGRAMAÇÃO ORIENTADA A OBJETOS

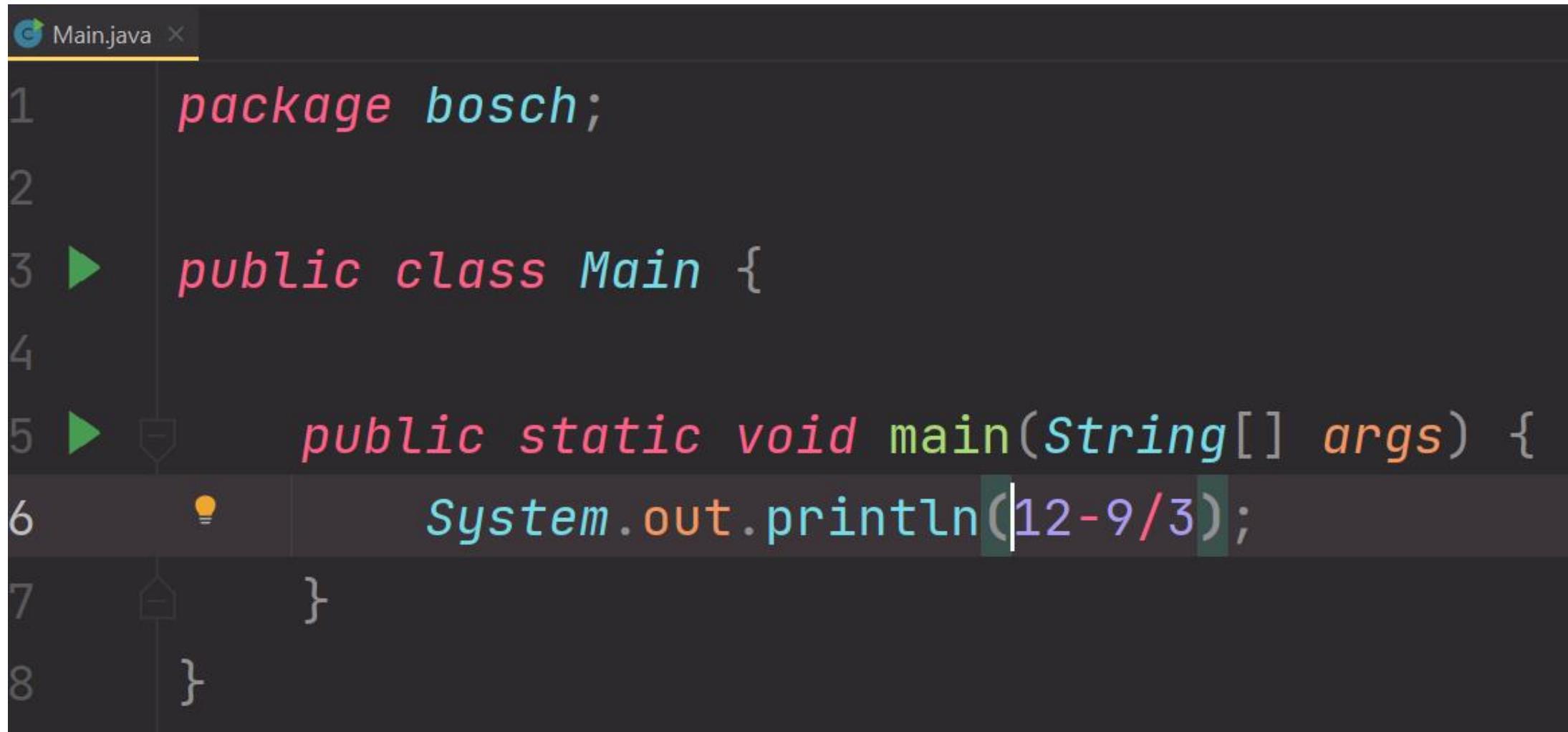


The screenshot shows a code editor window with a dark theme. The file tab at the top left says "Main.java". The code itself is a simple Java program:

```
1 package bosch;
2
3 ► public class Main {
4
5 ►     public static void main(String[] args) {
6         System.out.println(10+2*20);
7     }
8 }
```

The code uses color-coded syntax highlighting: package, class, and static keywords are in blue; variable names and method names are in green; strings are in red; and comments are in purple. Line numbers are visible on the left side of the code area.

# PROGRAMAÇÃO ORIENTADA A OBJETOS

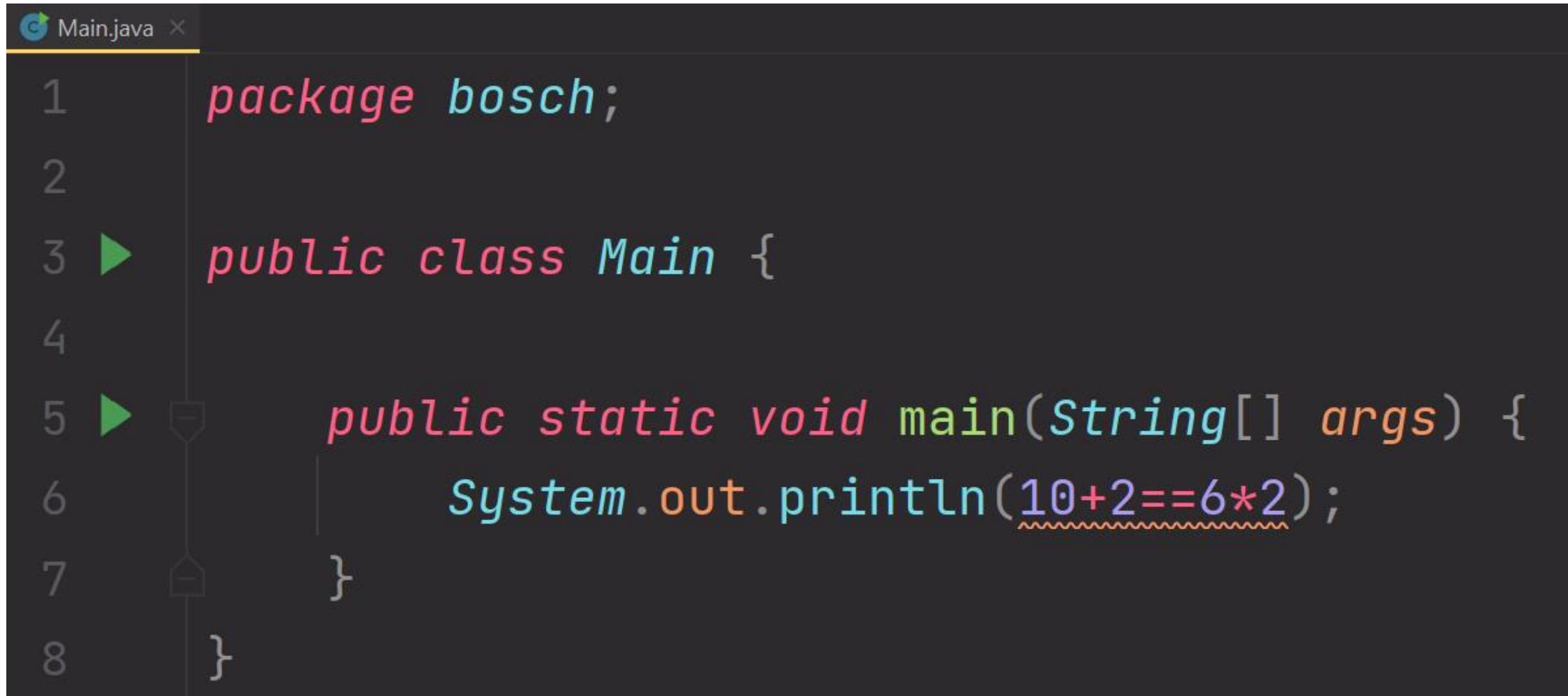


The screenshot shows a Java code editor with a dark theme. The file is named "Main.java". The code is as follows:

```
1 package bosch;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         System.out.println(12-9/3);
7     }
8 }
```

The code prints the value 3 to the console. The line "System.out.println(12-9/3);" is highlighted in green, indicating it is the current line of execution.

# PROGRAMAÇÃO ORIENTADA A OBJETOS



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

# PROGRAMAÇÃO ORIENTADA A OBJETOS

The screenshot shows a code editor window 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         System.out.println(16>8+8);
7     }
8 }
```

The code uses color-coded syntax highlighting: package, class, and static keywords are in blue; void and main are in red; strings are in green; and numbers are in orange. The editor also features code folding, indicated by small green arrows and brackets on the left side of the code.

# PROGRAMAÇÃO ORIENTADA A OBJETOS

```
Main.java X
1 package bosch;
2
3 ► public class Main {
4
5     ►     public static void main(String[] args) {
6         int x=3, y=4, z=5;
7
8             System.out.println(x<y && z>y);
9
10    }
```

# PROGRAMAÇÃO ORIENTADA A OBJETOS

The screenshot shows a code editor window 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         int x=3, y=4, z=5;
7
8         System.out.println(x>z || x<z);
9     }
10}
```

The code uses color-coded syntax highlighting: package, class, and static keywords are in blue; variable names are in red; and strings and comments are in green. Line numbers are displayed on the left side. Braces and other punctuation are shown in white or light gray.

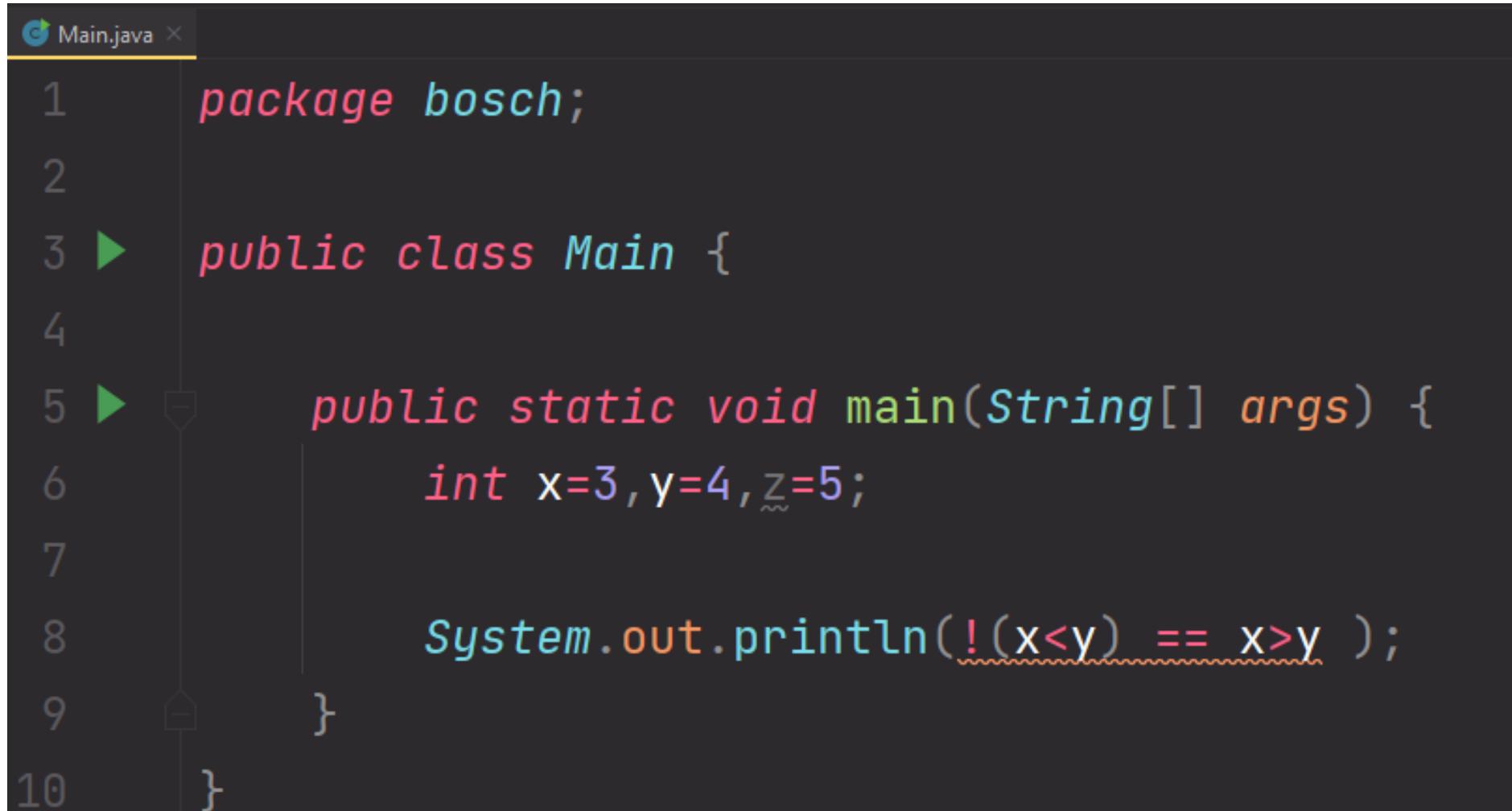
# PROGRAMAÇÃO ORIENTADA A OBJETOS

The screenshot shows a code editor window with a dark theme. The title bar says "Main.java". The code is as follows:

```
1 package bosch;
2
3 ► public class Main {
4
5 ►     ► public static void main(String[] args) {
6         int x=3, y=4, z=5;
7
8         System.out.println(y<x!=y>x);
9     }
10 }
```

The code uses color-coded syntax highlighting: package, class, and static keywords are in blue; identifiers like Main, x, y, z, and args are in purple; strings and comments are in green; and operators like =, !=, <, and > are in red. Line numbers are on the left.

# PROGRAMAÇÃO ORIENTADA A OBJETOS

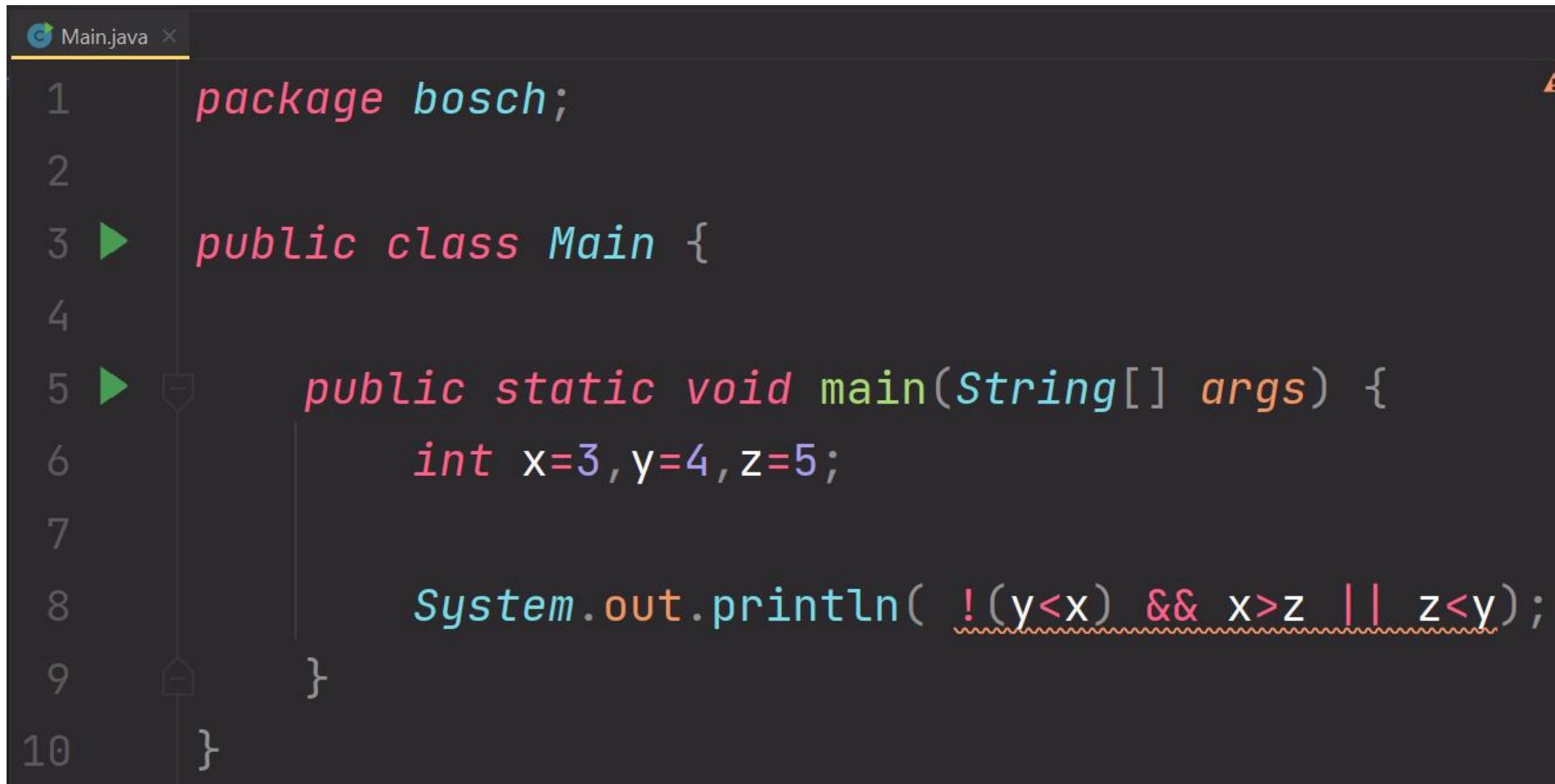


The screenshot shows a code editor window with a dark theme. The file is named "Main.java". The code is as follows:

```
1 package bosch;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         int x=3, y=4, z=5;
7
8         System.out.println(!(x<y) == x>y );
9     }
10 }
```

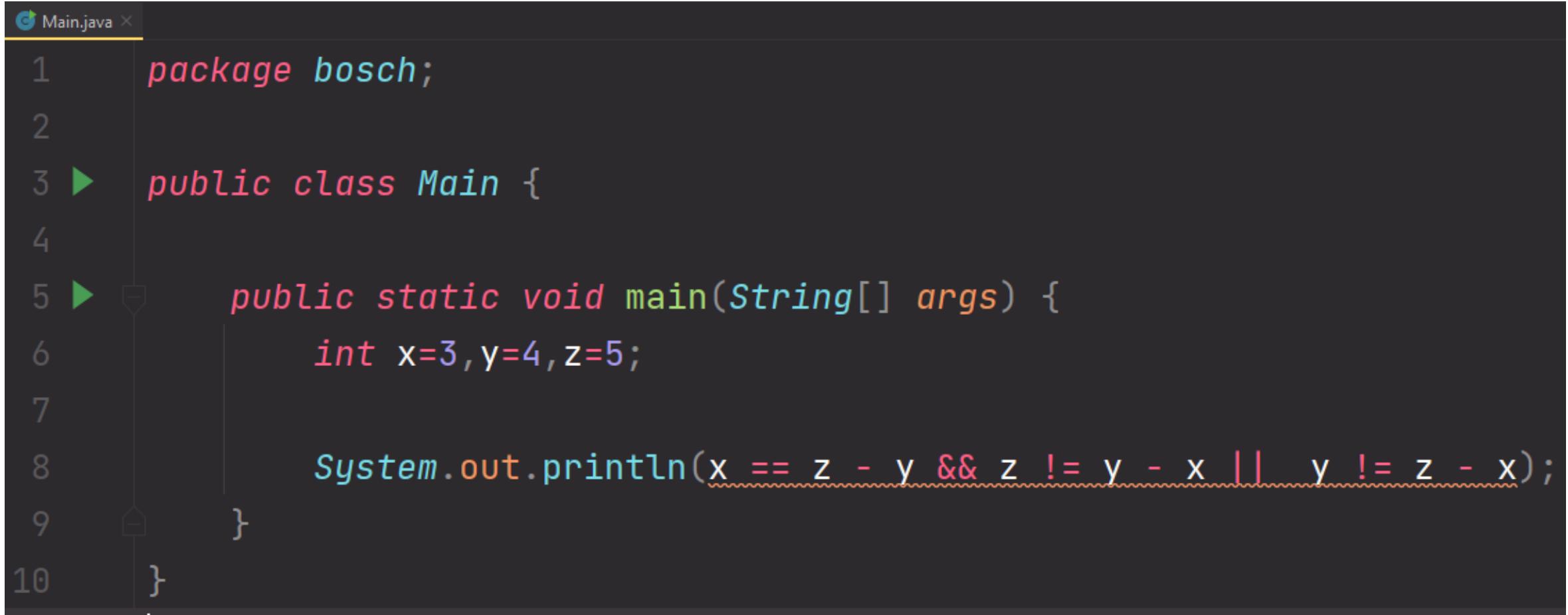
The code uses color-coded syntax highlighting: package and class names are blue, method names are green, and variable names are purple. The IDE interface includes a tab bar at the top with "Main.java" and a close button.

# PROGRAMAÇÃO ORIENTADA A OBJETOS



```
1 package bosch;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         int x=3, y=4, z=5;
7
8         System.out.println( !(y<x) && x>z || z<y );
9     }
10 }
```

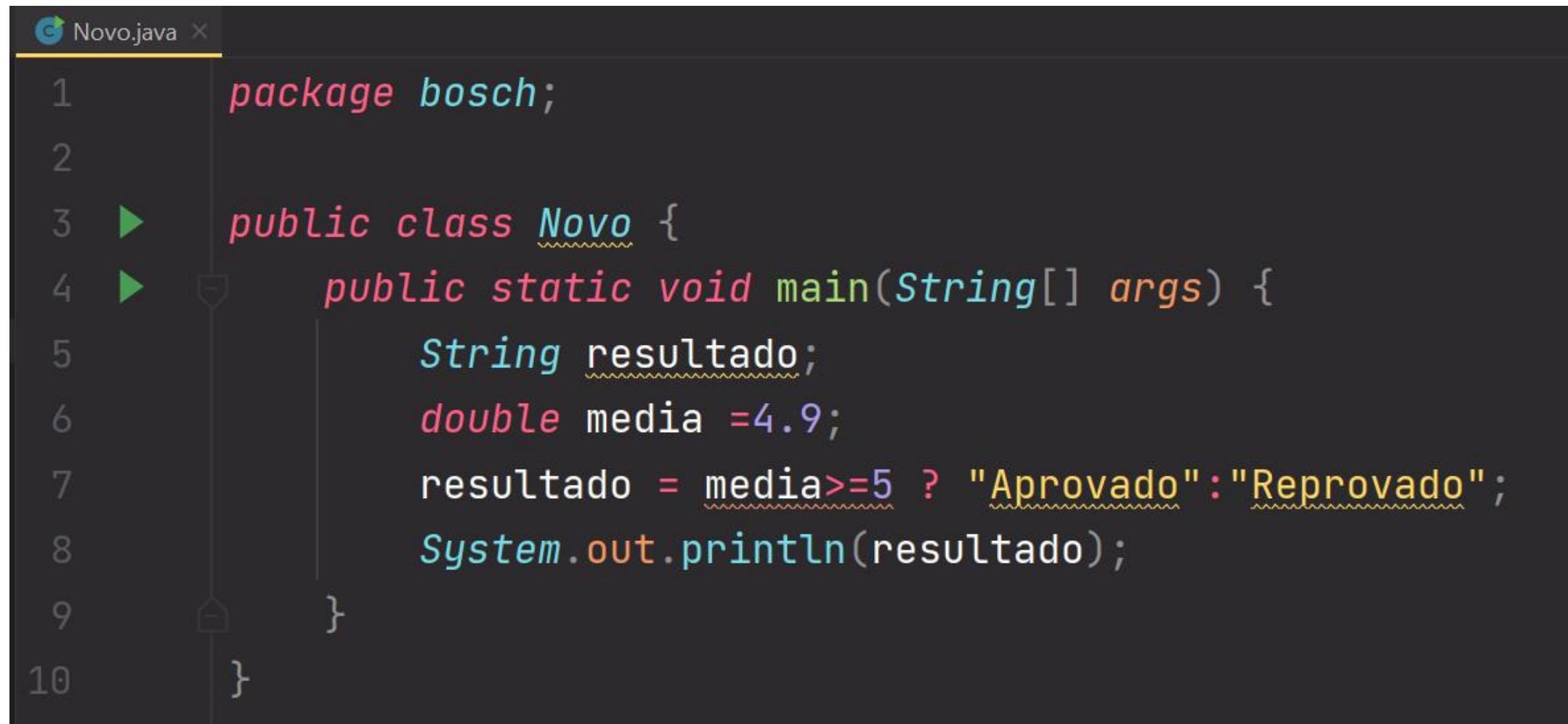
# PROGRAMAÇÃO ORIENTADA A OBJETOS



```
1 package bosch;
2
3 ► public class Main {
4
5 ►     public static void main(String[] args) {
6         int x=3,y=4,z=5;
7
8         System.out.println(x == z - y && z != y - x || y != z - x);
9     }
10 }
```

# 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 }
```

# PROGRAMAÇÃO ORIENTADA A OBJETOS

## Par ou Impar

- Faça um programa com que ao digitar um número no terminal, ele verifica se o numero é par ou impar.

# PROGRAMAÇÃO ORIENTADA A OBJETOS

## FizzBuzz

- ▶ Faça um programa que ao digitar um número no terminal.
- ▶ Printe Fizz se o número for múltiplo de 2
- ▶ Printe Buzz se o número for múltiplo de 5
- ▶ Printe FizzBuzz se o número for múltiplo de 2 e de 5
- ▶ Printe ERRO! Caso nenhuma das condições acima seja satisfeita.

# PROGRAMAÇÃO ORIENTADA A OBJETOS

## Wrappers

```
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    }
```

# PROGRAMAÇÃO ORIENTADA A OBJETOS

## Wrappers

```
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 }
```

# PROGRAMAÇÃO ORIENTADA A OBJETOS

## Conversão de tipos

```
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 }
```

# PROGRAMAÇÃO ORIENTADA A OBJETOS

## Conversão de Tipos

```
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 }
```

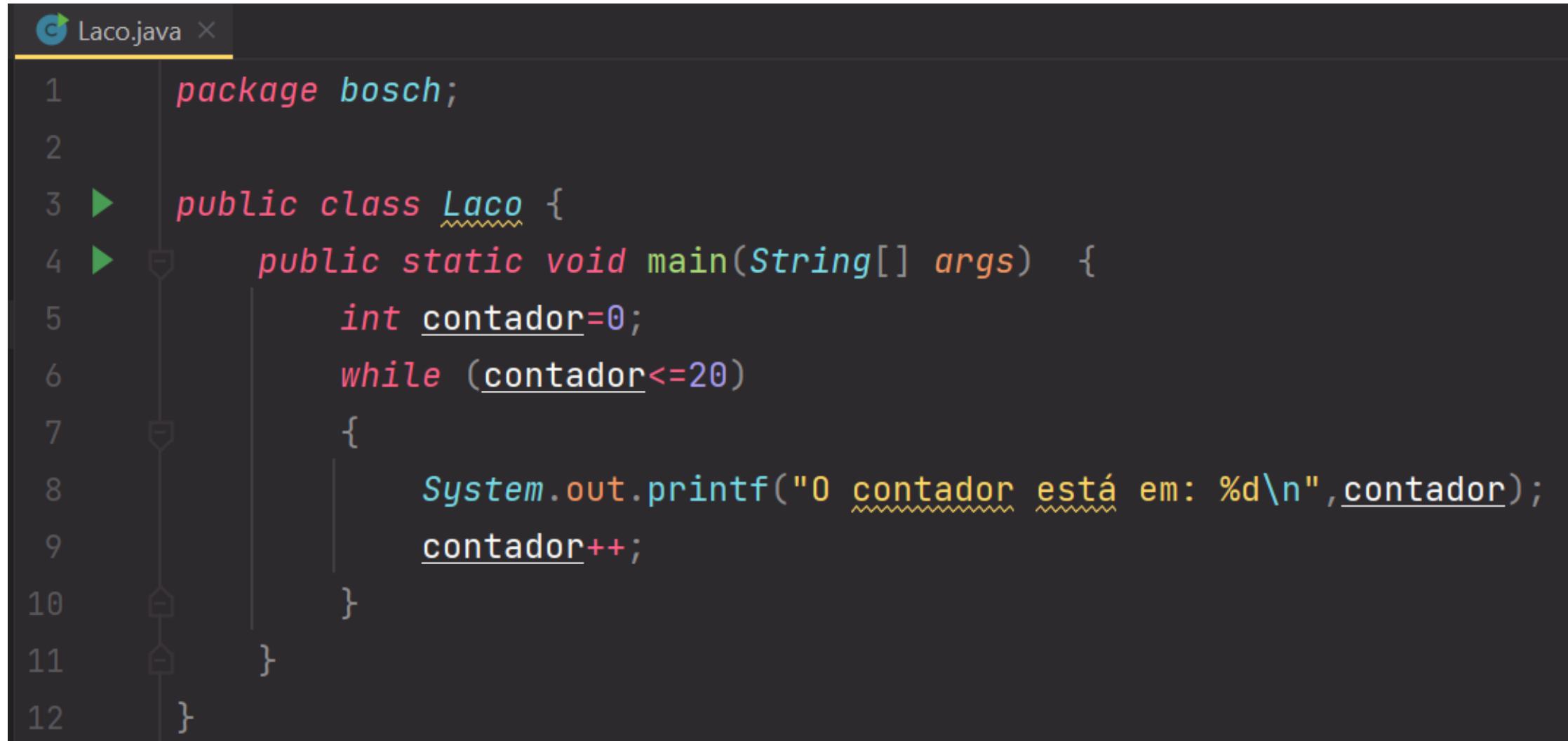
# PROGRAMAÇÃO ORIENTADA A OBJETOS

## Conversão de Tipos

```
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 contains a simple while loop that prints the value of the variable "contador" from 0 to 20. The code is color-coded, with keywords like "package", "public", "class", "void", "int", "while", and "System.out.printf" in various colors, and variable names in purple.

```
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 }
```

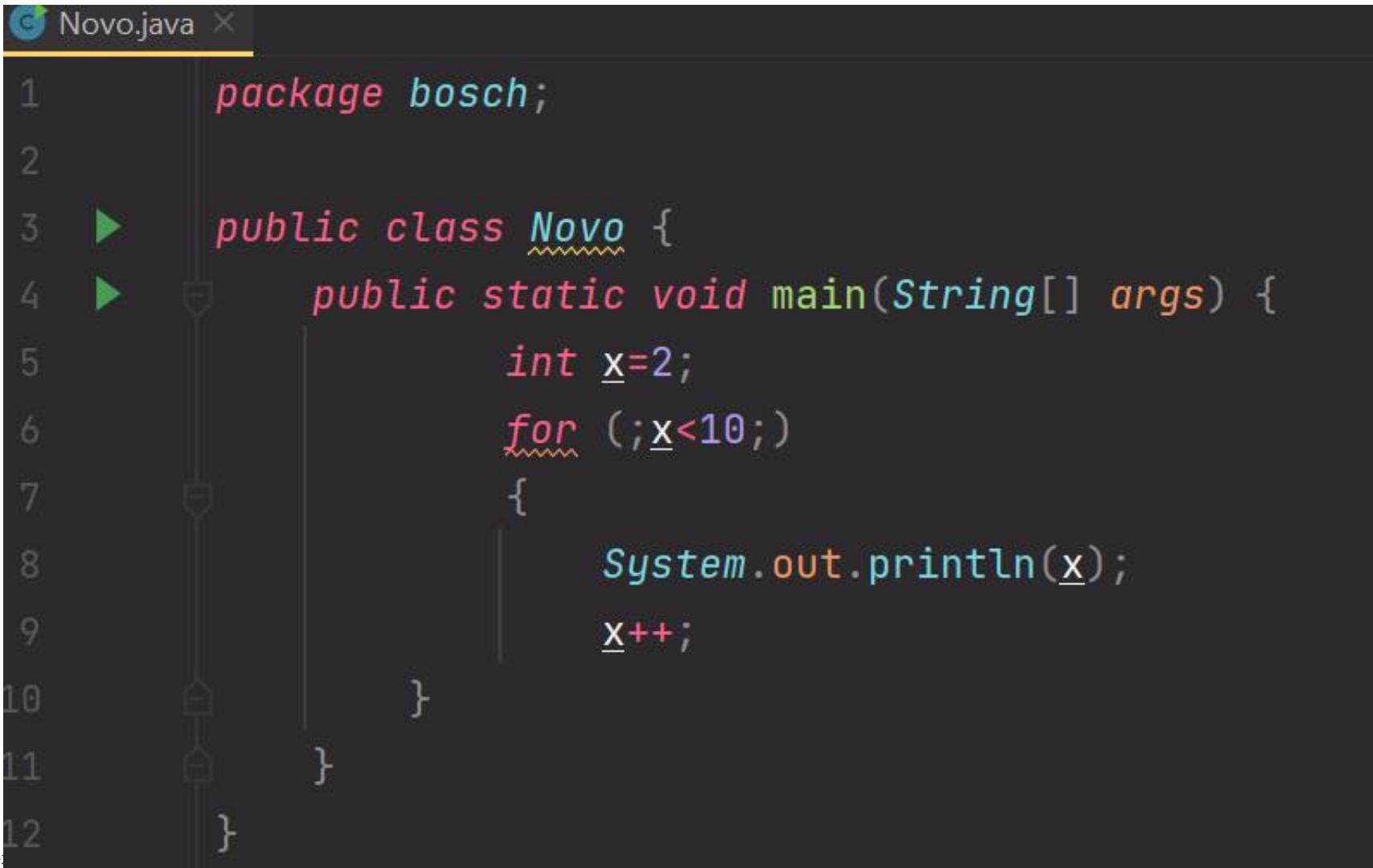
# PROGRAMAÇÃO ORIENTADA A OBJETOS

## Java – Laços de Repetição

```
Novo.java x
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

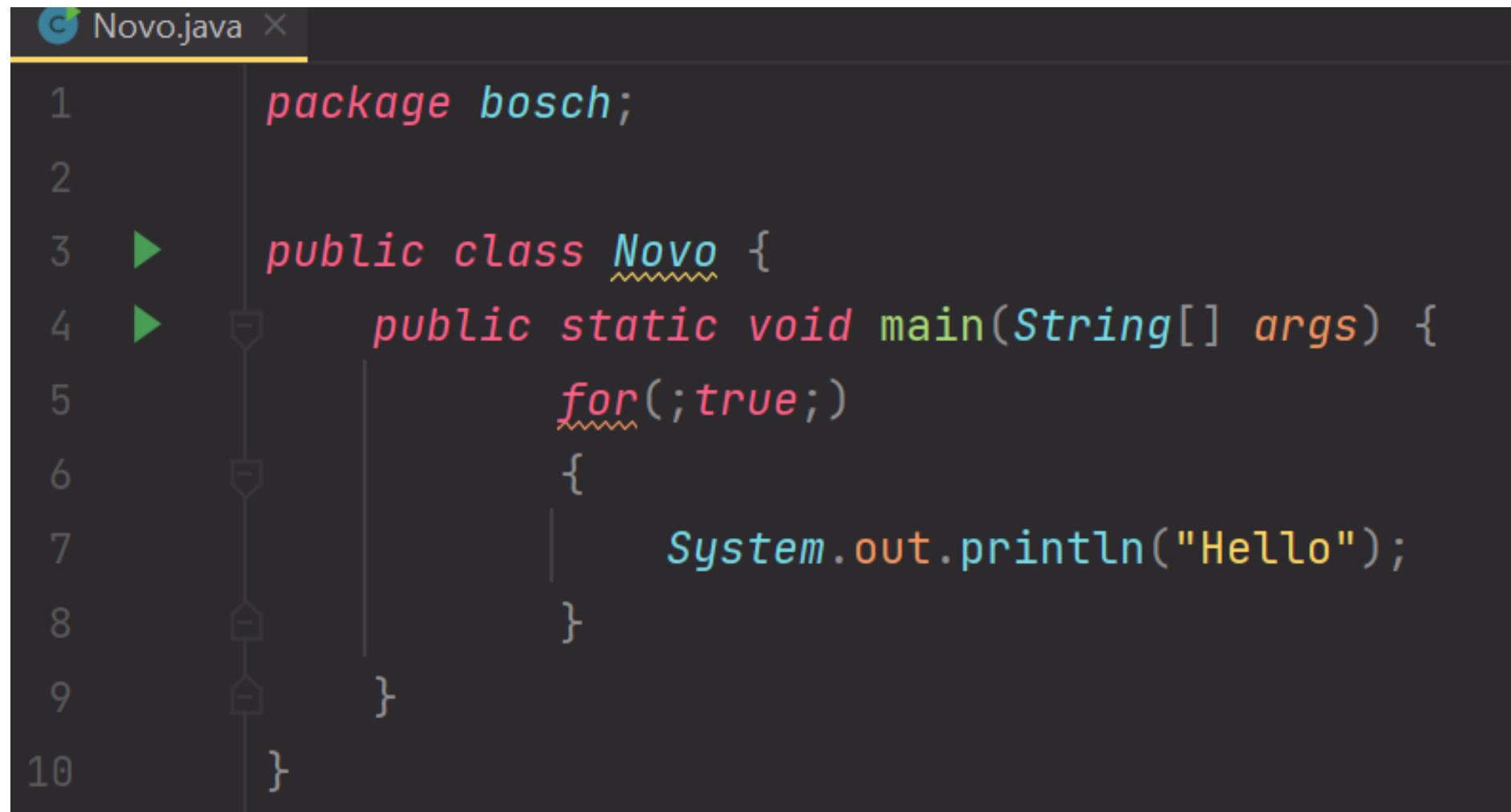


The screenshot shows a Java code editor with a dark theme. The file is named 'Novo.java'. The code contains a for loop that prints integers from 2 to 10 to the console.

```
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

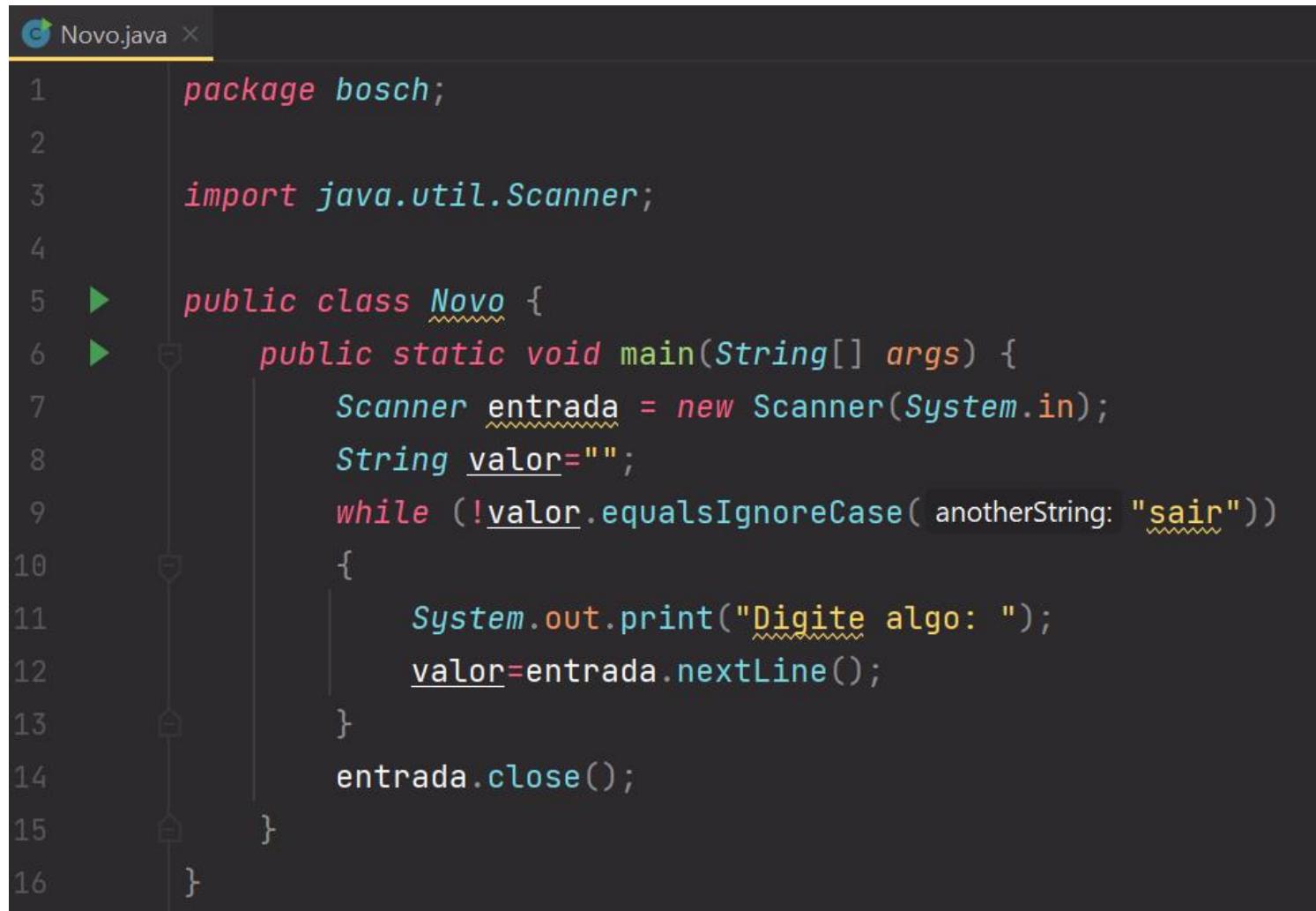


The image shows a screenshot of a Java code editor with a dark theme. The file is named "Novo.java". The code contains a package declaration, a class definition named "Novo" with a main method, and an infinite loop using a for loop with a true condition. The code is color-coded: package, class, and main are in blue; for, System.out.println, and Hello are in orange; and the rest of the text is in white or light gray.

```
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

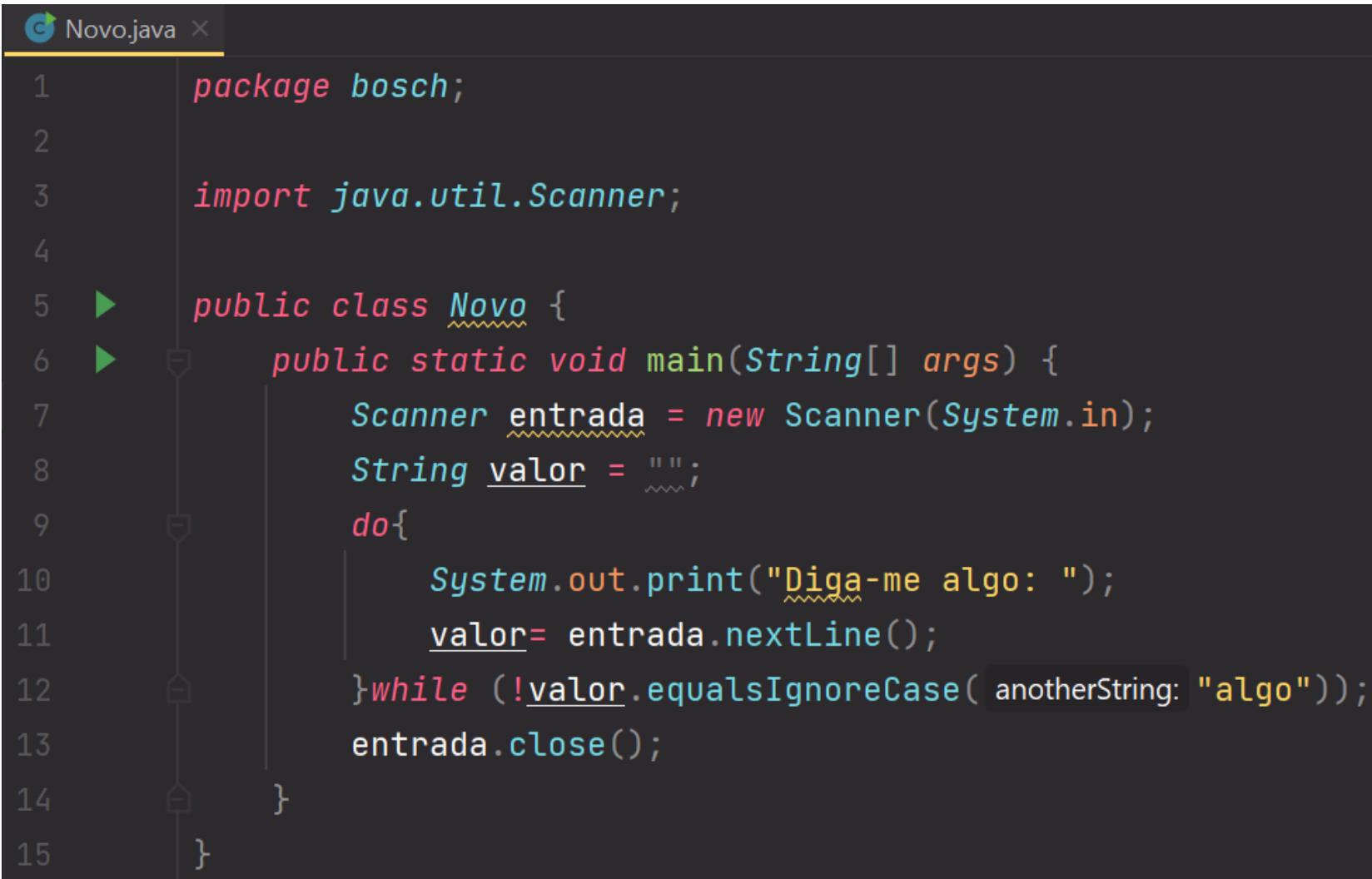


The screenshot shows a Java code editor with a dark theme. The file is named "Novo.java". The code implements a simple command-line application using the Scanner class to read input from the user. It prompts the user to "Digite algo:" and reads their input into the variable "valor". The program then checks if "valor" equals "sair" (case-insensitively). If it does not, the loop continues. Inside the loop, the user's input is printed back to the console. Finally, after the loop exits, the scanner is closed.

```
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         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

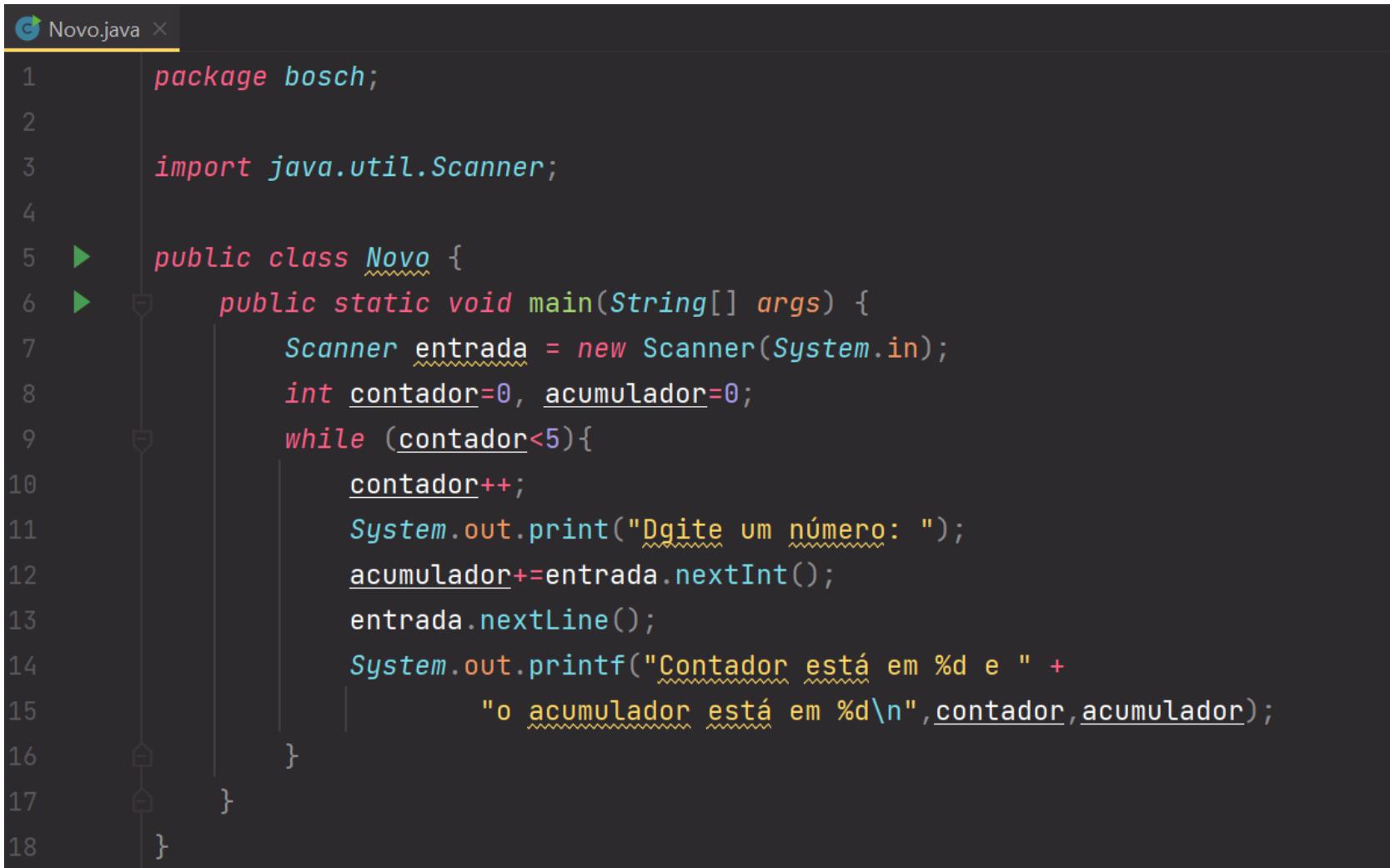
## Java – Laços de Repetição



```
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         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 }
```

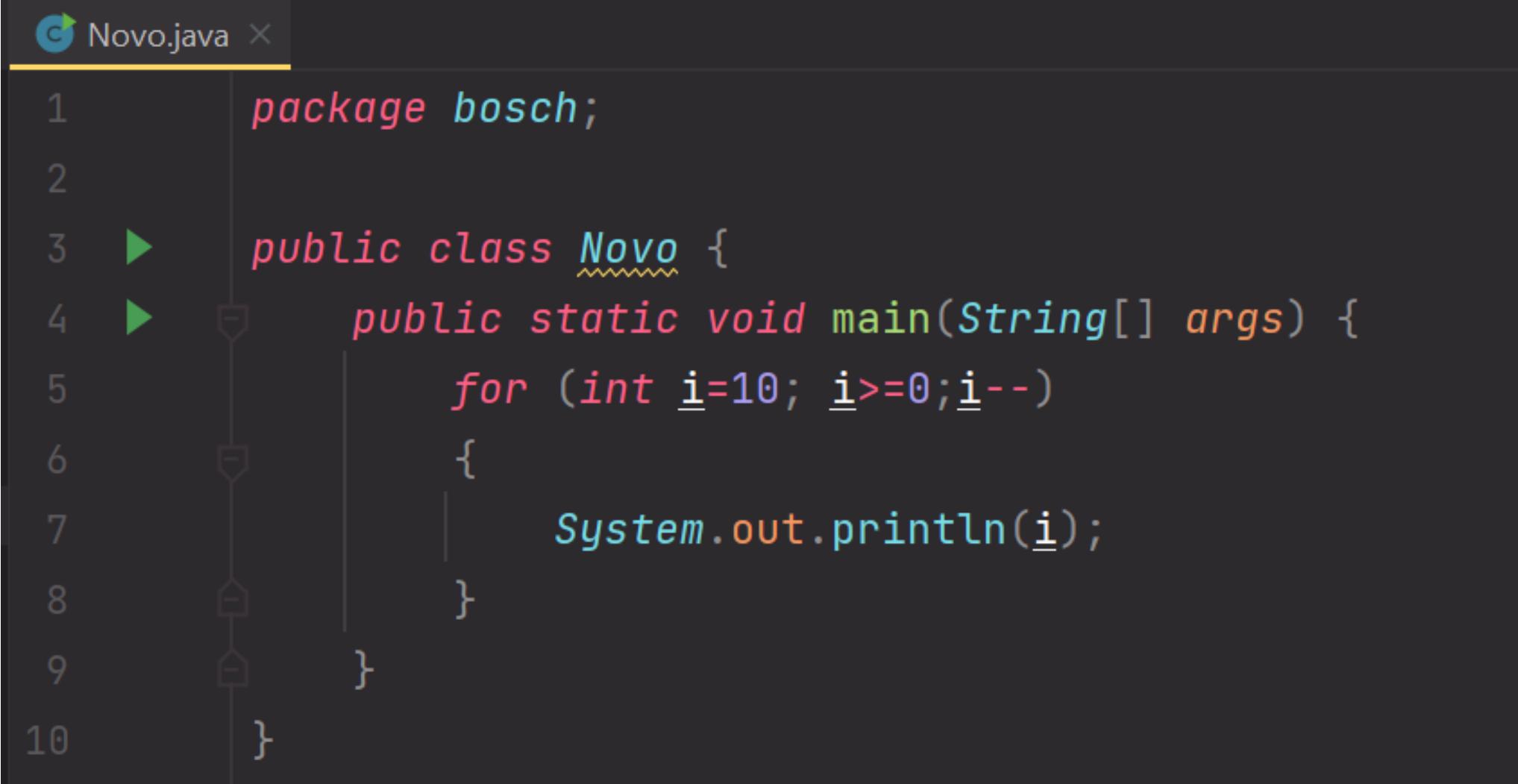
# PROGRAMAÇÃO ORIENTADA A OBJETOS

## Java – Laços de Repetição

```
Novo.java ×  
1 package bosch;  
2  
3 ► public class Novo {  
4 ►     public static void main(String[] args) throws InterruptedException {  
5         int contador=0, acumulador=0;  
6         for(int i=0; i<5; i++){  
7             contador=i;  
8             acumulador+=i*i;  
9             System.out.printf("Contador está em %d e " +  
10                         "o acumulador está em %d\n",contador,acumulador);  
11             Thread.sleep( millis: 2000);  
12         }  
13     }  
14 }
```

# PROGRAMAÇÃO ORIENTADA A OBJETOS

## Java – Laços de Repetição

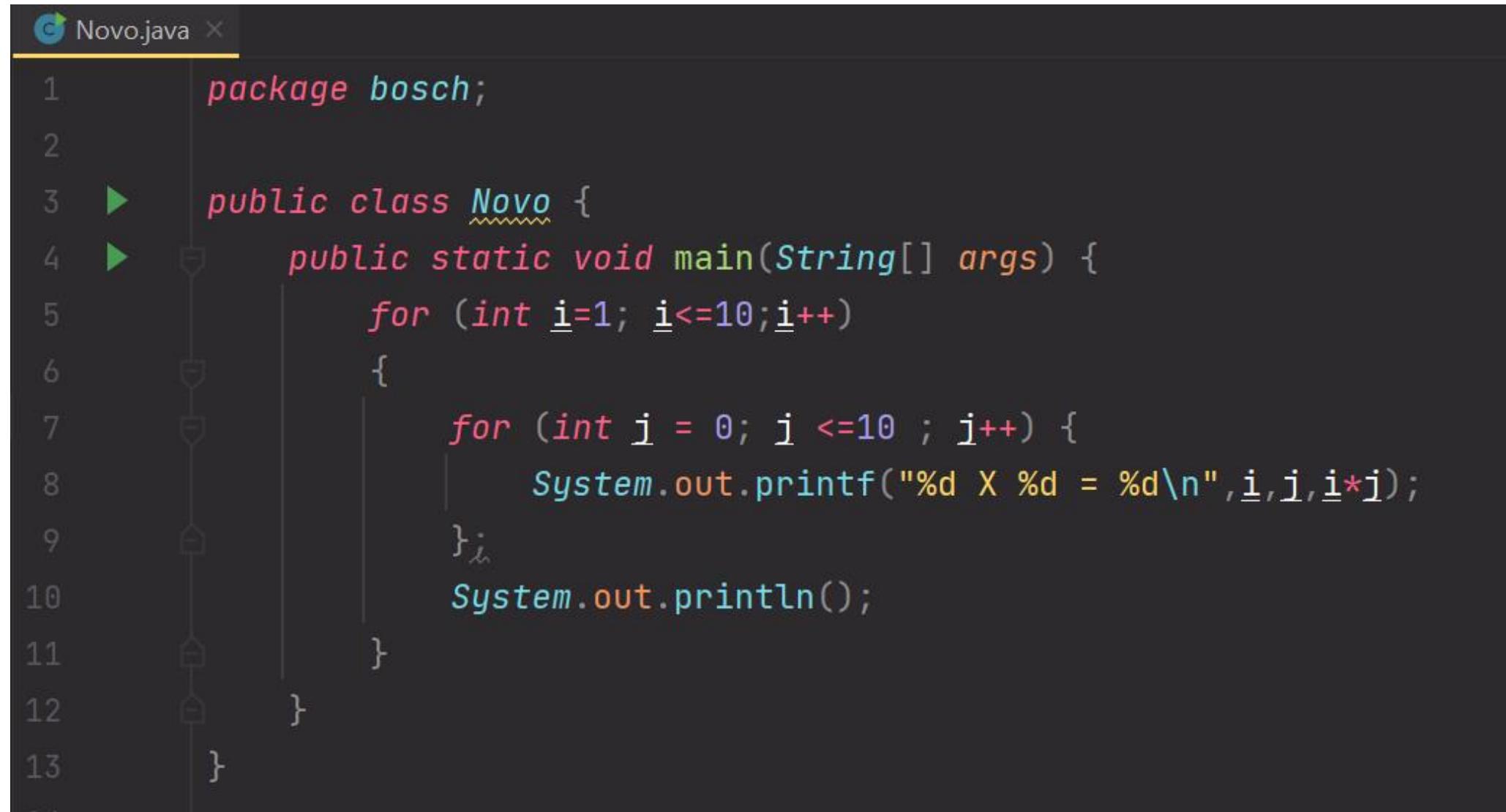


The screenshot shows a Java code editor with a dark theme. The file is named "Novo.java". The code contains a for loop that prints numbers from 10 down to 0. The code is color-coded: package, class, and main are in blue; for, int, and System.out.println are in orange; and i is in purple. Brackets and operators are in white. The code is as follows:

```
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

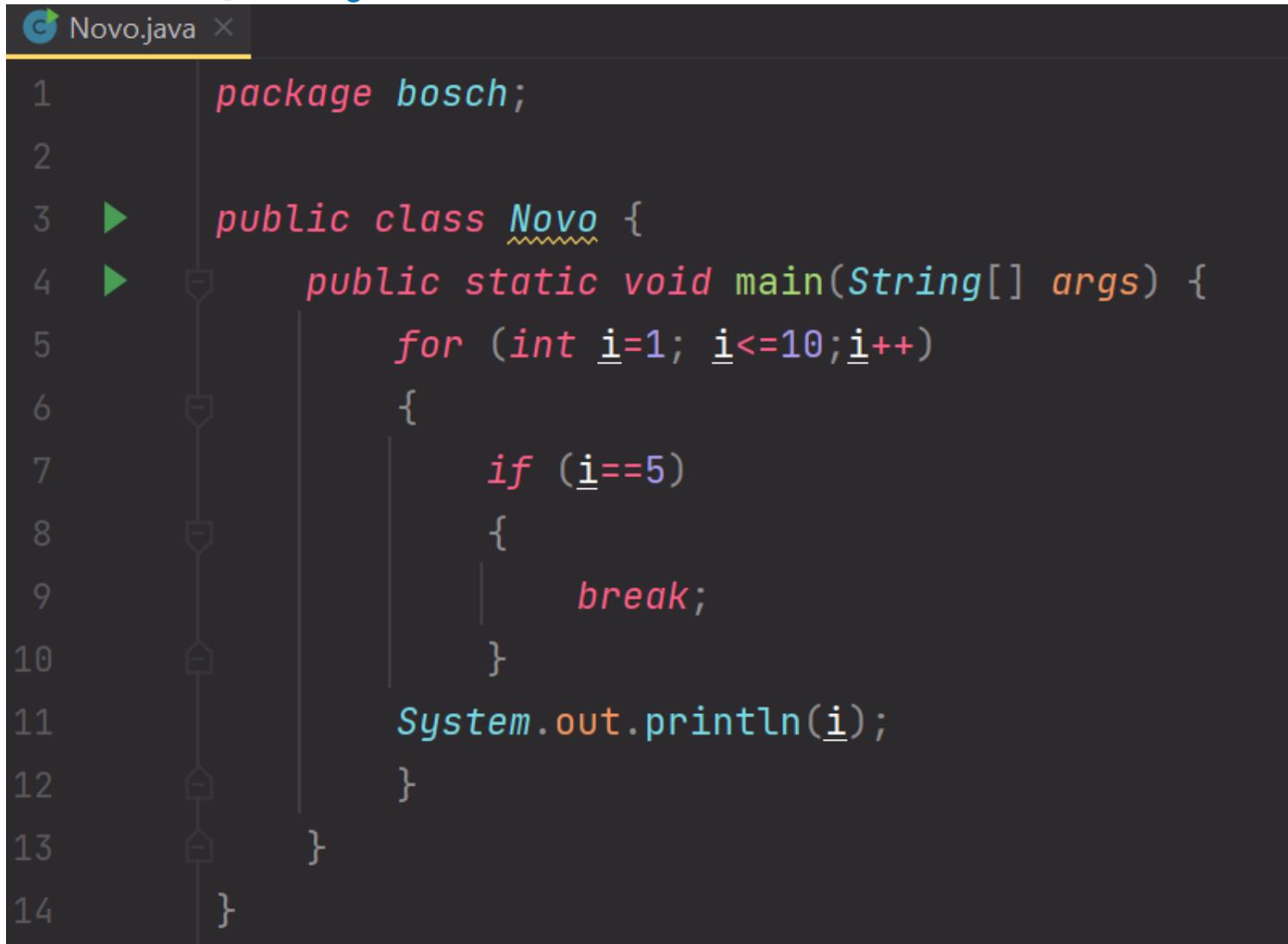
## Java – Laços de Repetição



```
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             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 }
```

# PROGRAMAÇÃO ORIENTADA A OBJETOS

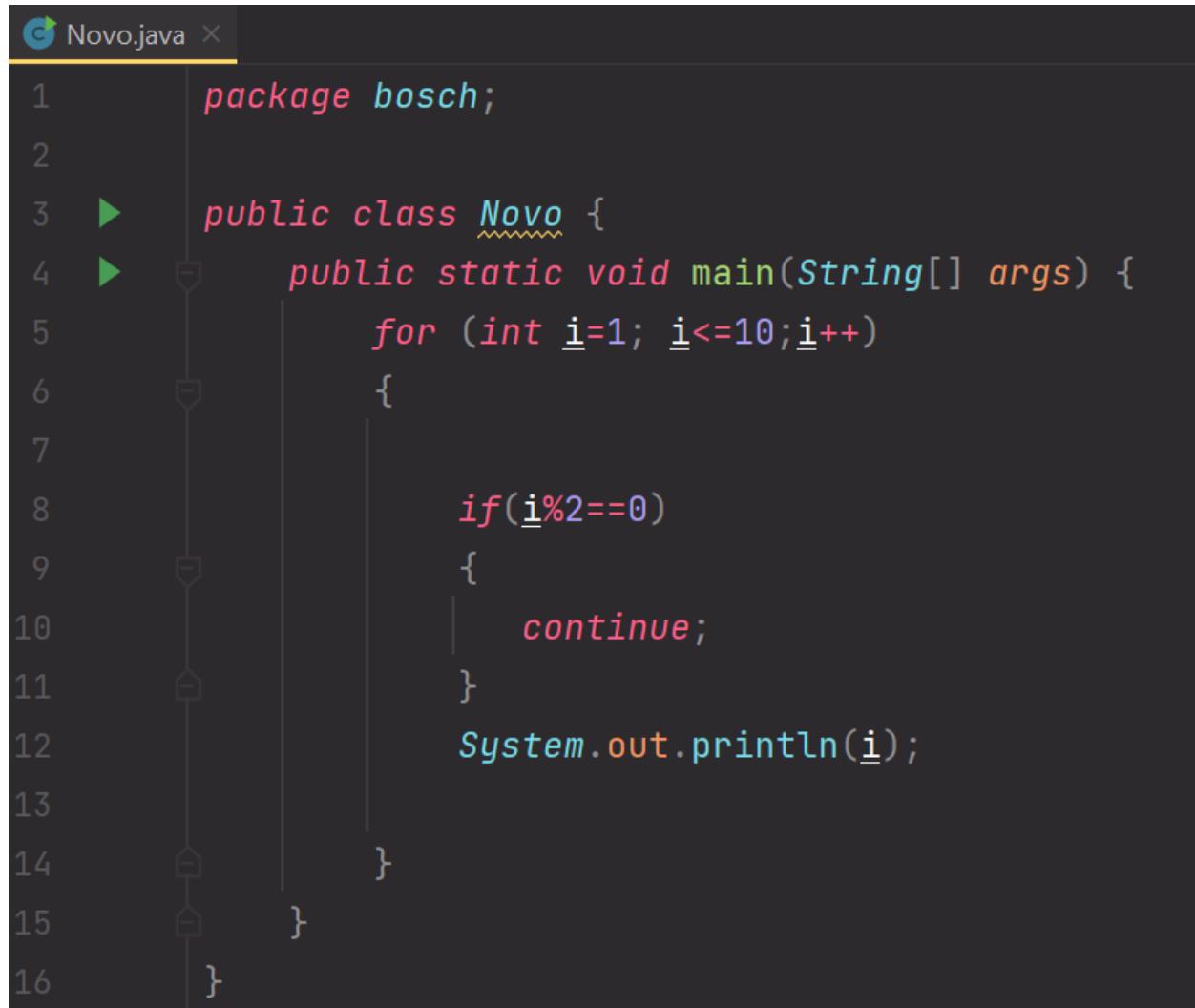
## Java – Laços de Repetição



```
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

## Exercício Tabuada - For

- Faça a tabuada do 1 até a tabuada do 10, utilizando a seguinte formatação.
- $1 \times 1 = 1$
- $1 \times 2 = 2$
- $2 \times 1 = 2$
- $2 \times 2 = 4$
- Faça utilizando o laço de repetição **for**

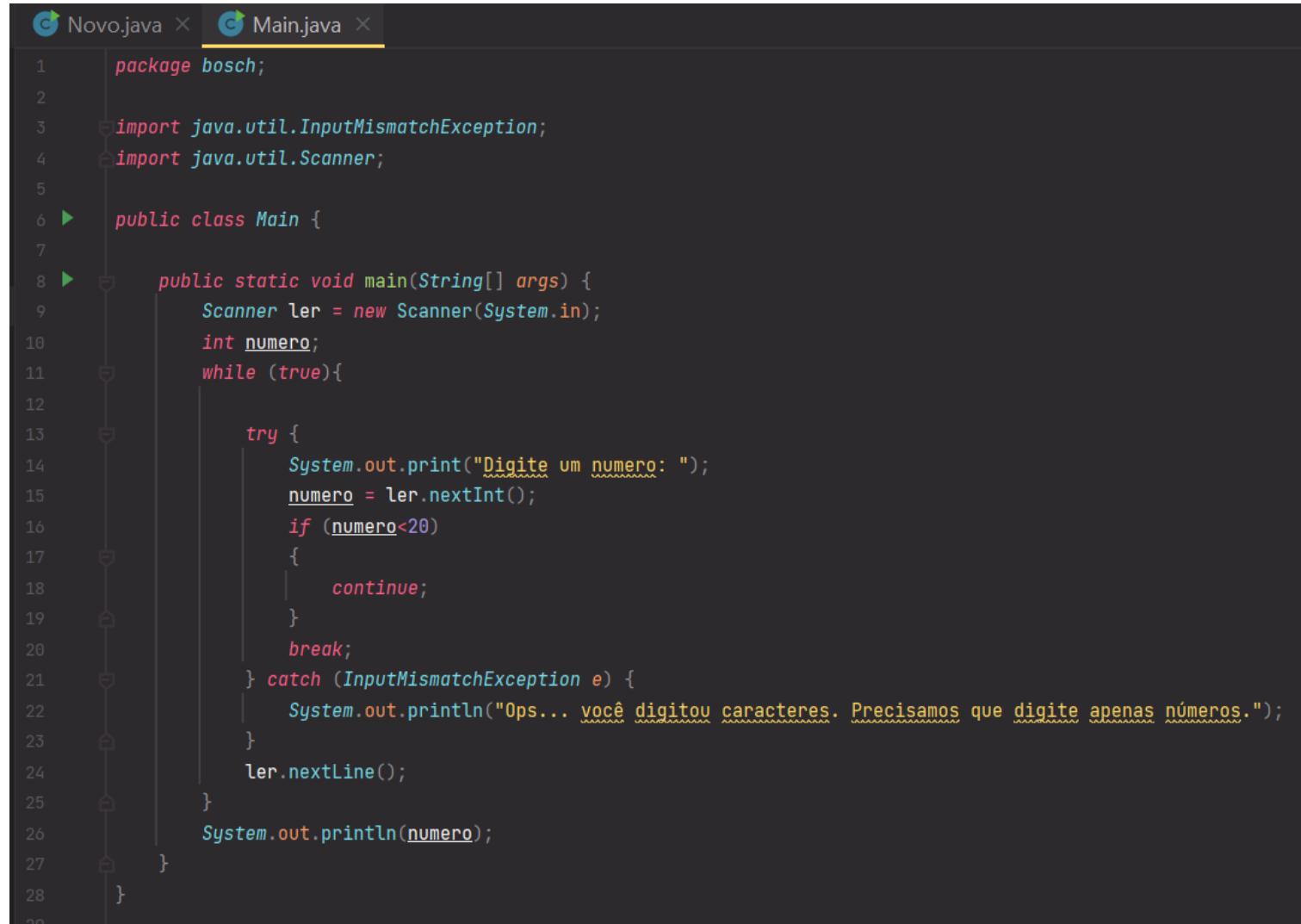
# PROGRAMAÇÃO ORIENTADA A OBJETOS

## Exercício Tabuada - while

- Faça a tabuada do 1 até a tabuada do 10, utilizando a seguinte formatação.
  - $1 \times 1 = 1$
  - $1 \times 2 = 2$
  - $2 \times 1 = 2$
  - $2 \times 2 = 2$
- Faça utilizando o laço de repetição **while**

# PROGRAMAÇÃO ORIENTADA A OBJETOS

## Java – Exceções

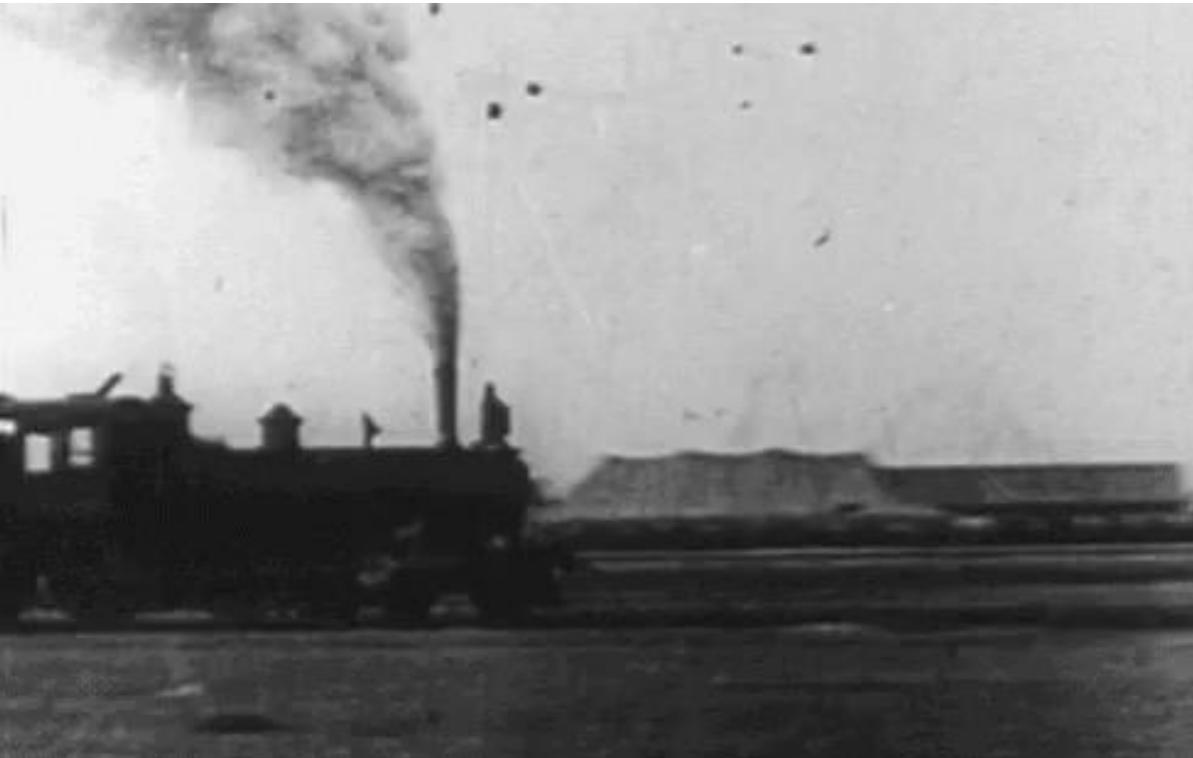


```
Novo.java x Main.java x
1 package bosch;
2
3 import java.util.InputMismatchException;
4 import java.util.Scanner;
5
6 public class Main {
7
8     public static void main(String[] args) {
9         Scanner ler = new Scanner(System.in);
10        int numero;
11        while (true){
12
13            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}
```

# PYTHON

## LAB 02 – Colisão de Trems

Suponha que dois trens partam ao mesmo tempo de cidades diferentes, de maneira que em algum momento eles irão colidir. Você não pode fazer nada para impedir a tragédia, a única coisa que você pode fazer é determinar o instante e o local que colidem.



# PYTHON

## LAB 02 – Colisão de Trems



- Considere que as extremidades da ferrovia vão do KM 0 até o KM 10.000
- Considere que a velocidade do trem A sempre será positiva e a velocidade do trem B sempre será negativa.
- Considere que o módulo da velocidade do trem será de no máximo 300 km/h
- Seu programa terá 4 variáveis de entrada: posição do trem A, posição do trem B, velocidade do trem A e velocidade do trem B.
- Seu programa deve exibir após quantos segundos ocorreu a colisão e em que KM ocorreu a colisão.

# PYTHON

## LAB 02 – Colisão de Treins – Requisitos Básicos



- Seu programa deve exibir o print no seguinte formato:

```
print(f'A colisão dos trens acontecerá no KM {posicao_final:.0f} e  
ocorrerá após {(tempo)*3600:.0f} segundos
```

- Faça com que nas entradas sejam aceito somente números, se necessário crie uma função para isso.
- Ao encerra o programa faça com que apareça “FIM DO PROGRAMA” na tela

# PYTHON

## LAB 02 – Colisão de Treins – Requisitos de Desafio



- Faça com que seu programa limite as posição entre o KM 0 e KM 10.000 e exiba uma mensagem de erro caso seja uma posição invalida
- Faça com que seu programa limite o módulo da velocidade dos trens a 300 km/h e exiba uma mensagem de erro caso valor seja invalido
- Faça com que a velocidade do trem A seja sempre positiva e a velocidade do trem B seja sempre negativa.
- Faça com que seu programa pergunte ao usuário se deseja executar novamente.
- Existe uma situação específica em que os trens não irão colidir, determine qual é esta situação e faça com que seu programa mostre uma mensagem falando que os trens não irão colidir.

# PYTHON

## LAB 02 – Colisão de Trems - Fórmulas

$$S(t) = S_0 + V \cdot t$$

Equação horaria do espaço

$$Sa(t) = S0a + Va \cdot t$$

Equação do trem A

$$Sb(t) = S0b + Vb \cdot t$$

Equação do trem B

$$t = \frac{S0a - S0b}{Vb - Va}$$

Equação do tempo

- S = Posição final
- S<sub>0</sub> = Posição inicial
- V = Velocidade
- t = Tempo
- a = Refere-se ao trem A
- b = Refere-se ao trem B

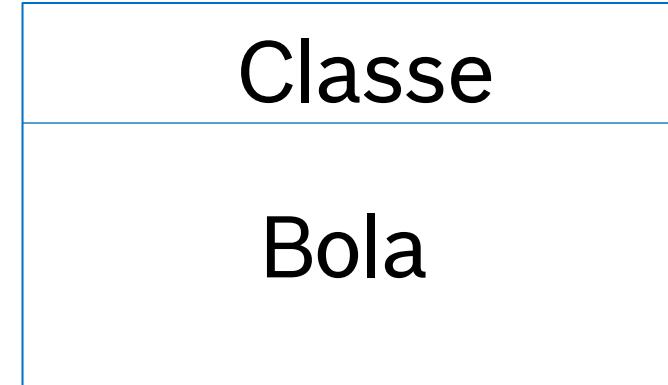
# PYTHON

## LAB 02 – Colisão de Treins – Entradas e Saídas

ENTRADAS	SAÍDAS
S0a=0, S0b=200, Va=20, Vb=-30	KM 80, 14400 segundos, 21:00:00
S0a=20, S0b=100, Va=100, Vb=-100	KM 60, 1440 segundos, 17:24:00
S0a=300, S0b=400, Va=20, Vb=0	KM 400, 18000 segundos, 22:00:00
S0a=800, S0b=3500, Va=200, Vb=-250	KM 2000, 21600, 23:00:00
S0a=H	Você digitou um caractere inválido. Por favor, digite novamente:
S0a=15000	O número deve ser maior ou igual a zero e menor que 10000.
S0a=20, S0b=60, Va=-240	A velocidade do trem A sempre será positiva. Por favor, digite novamente:
S0a=400, S0b=600, Va=280, Vb=200	A velocidade do trem B sempre será negativa. Por favor, digite novamente:
S0a=40, S0b=400, Va=200, Vb=c8	Você digitou um caractere inválido. Por favor, digite novamente:

# 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

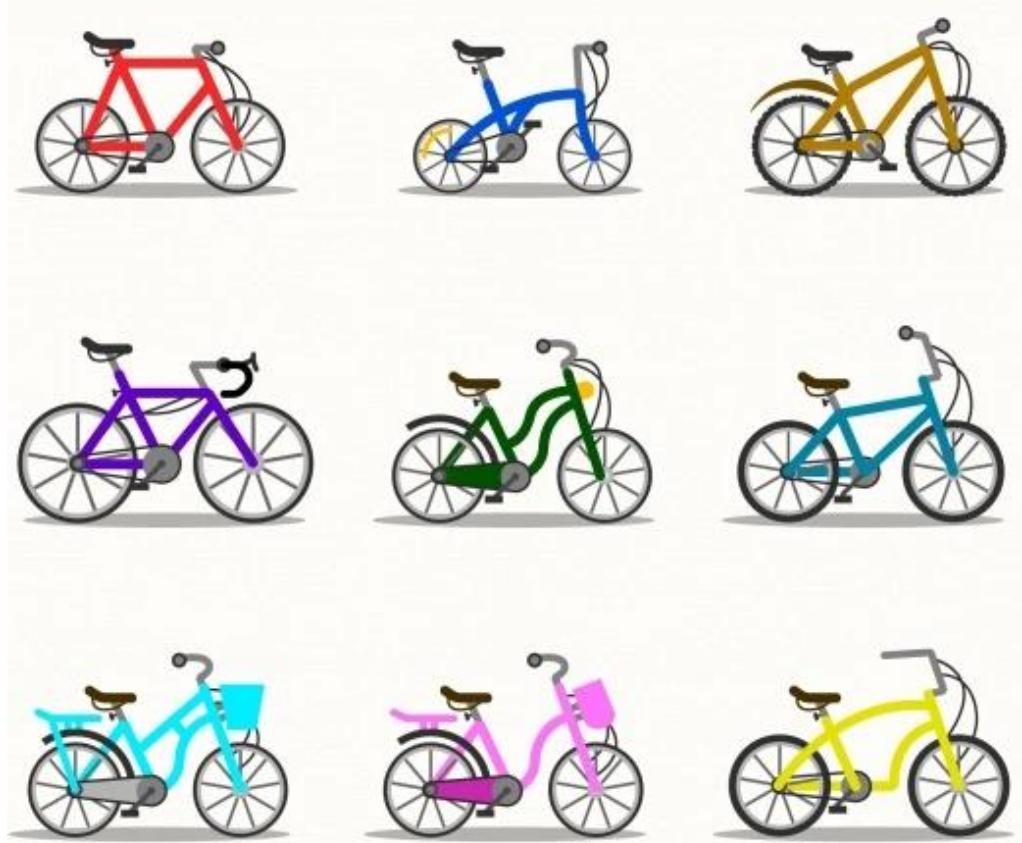
Uma classe pode ser considerada um molde através qual objetos de um certo tipo são criados.

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

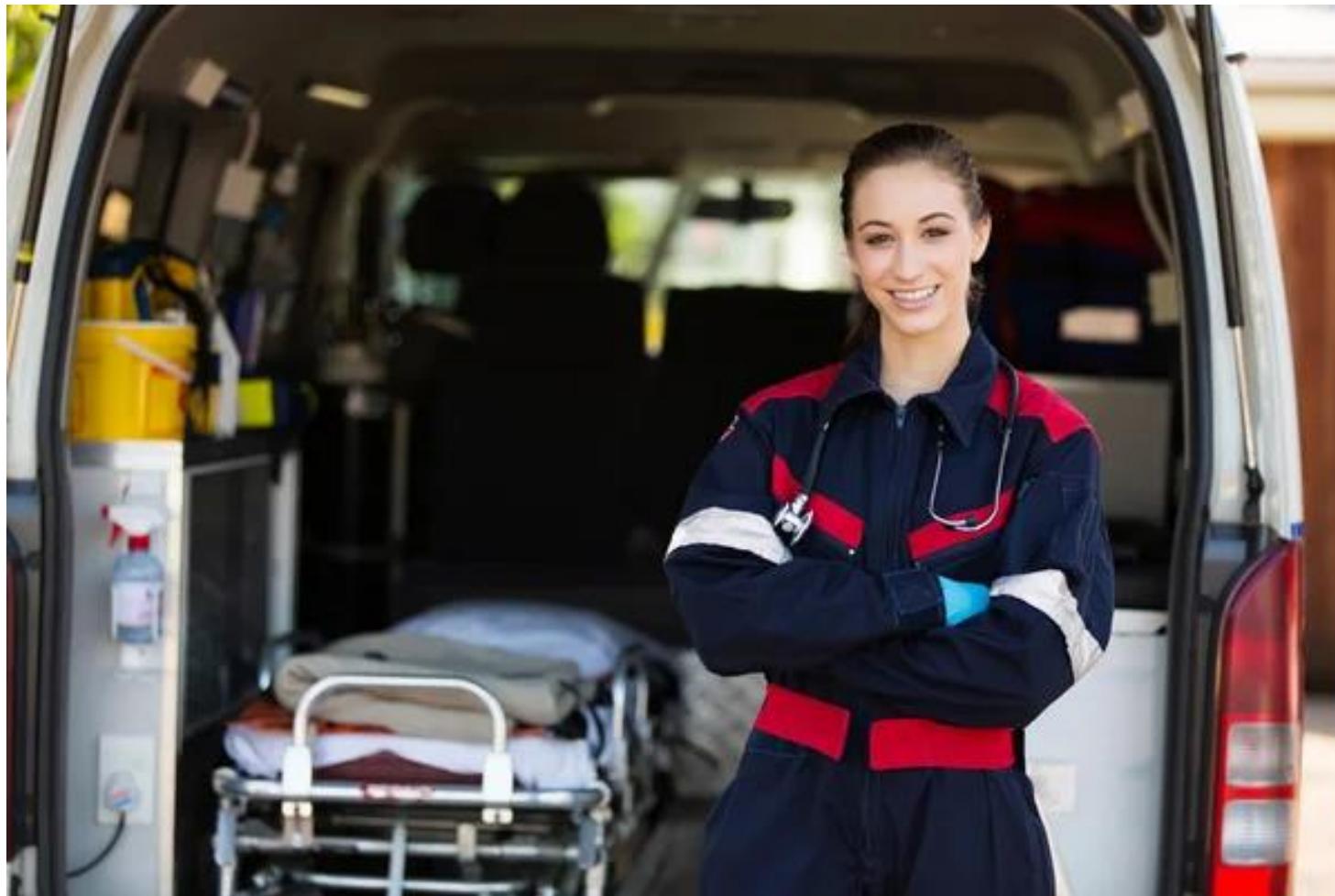


**Classe: Avião**  
**Instância: Avião da viagem**

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	

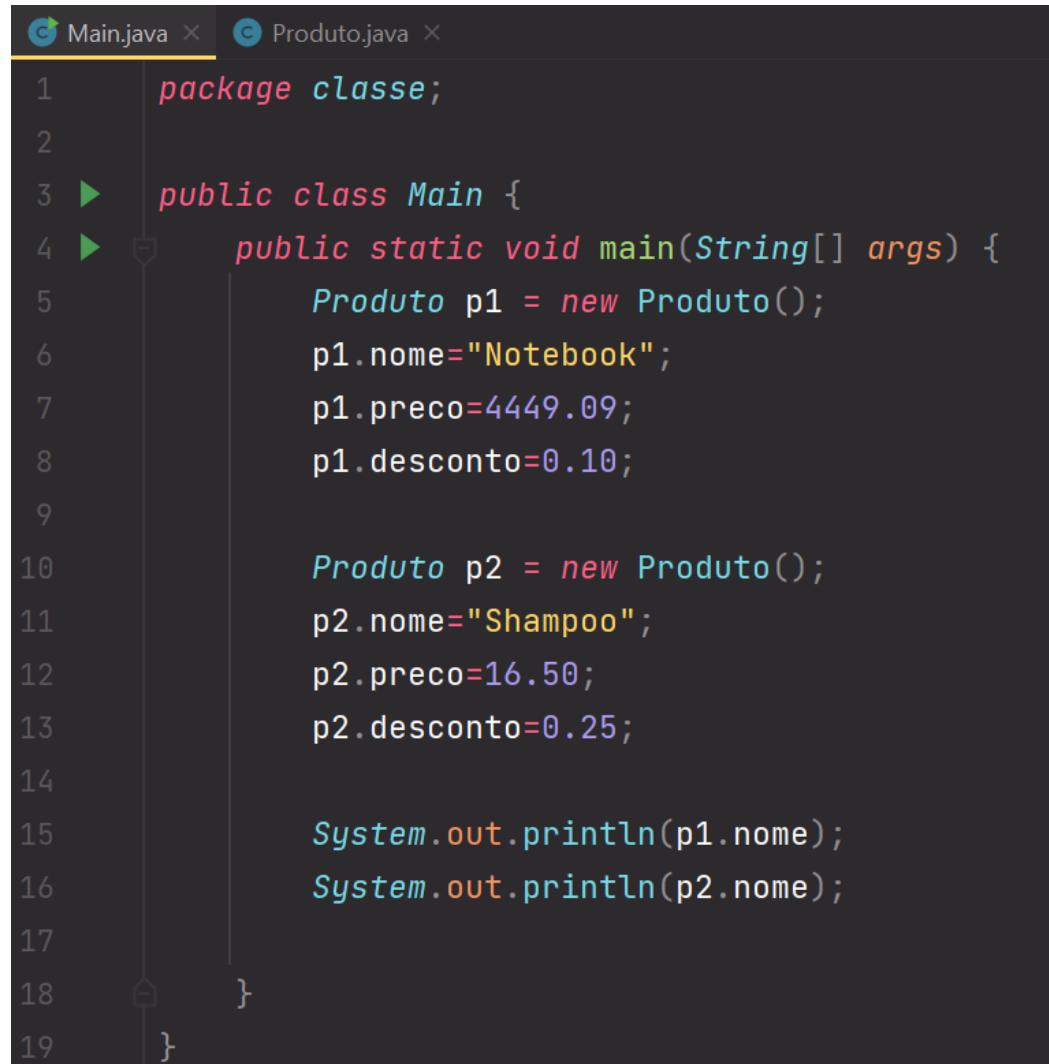
# PROGRAMAÇÃO ORIENTADA A OBJETOS

## Classe

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

# PROGRAMAÇÃO ORIENTADA A OBJETOS

## Classe



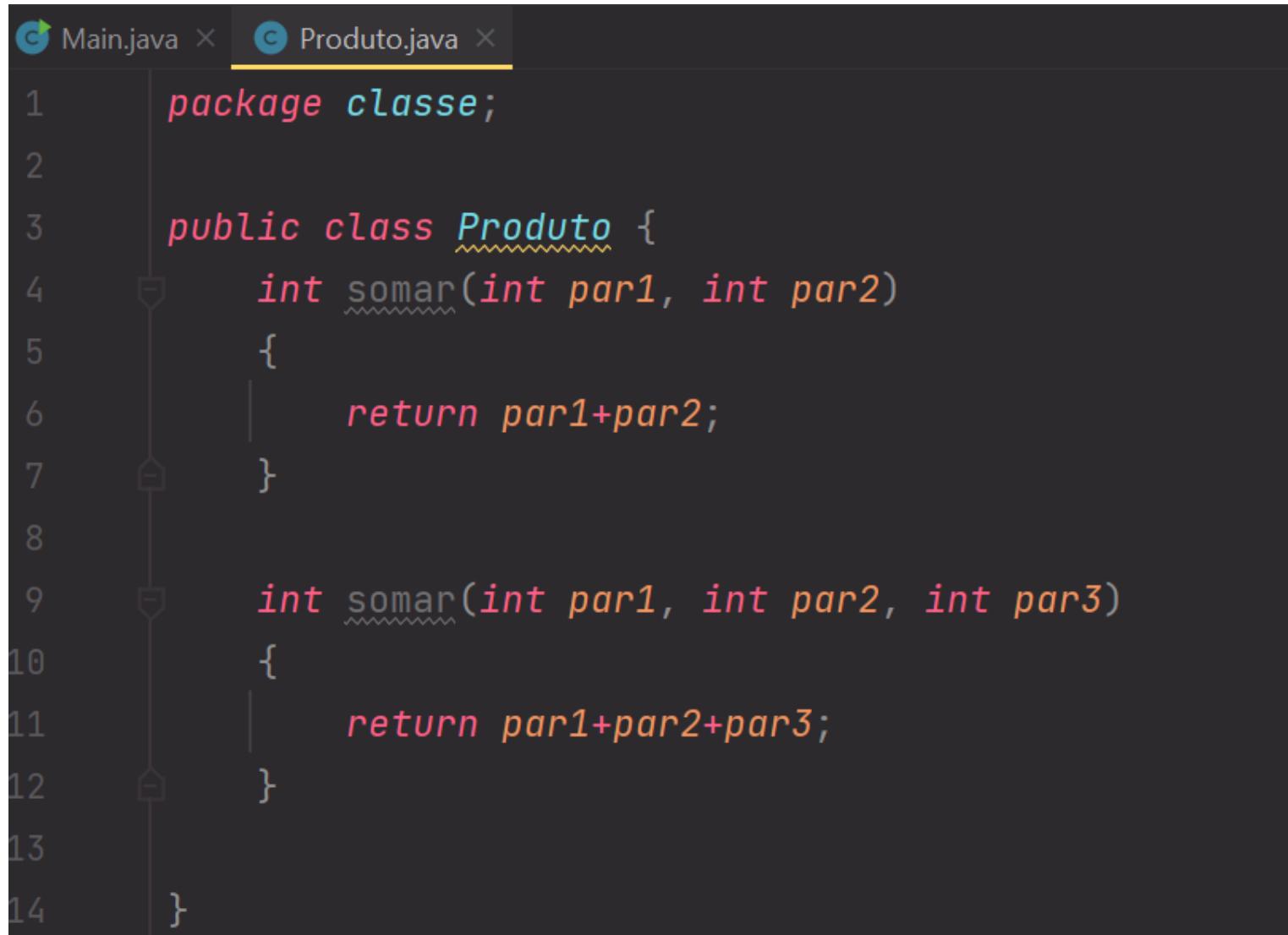
The screenshot shows a Java code editor with two tabs: 'Main.java' and 'Produto.java'. The 'Main.java' tab is active, displaying the following code:

```
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.nome);
16        System.out.println(p2.nome);
17    }
18 }
19 }
```

The 'Produto.java' tab is visible in the background, indicating it is another file in the project.

# PROGRAMAÇÃO ORIENTADA A OBJETOS

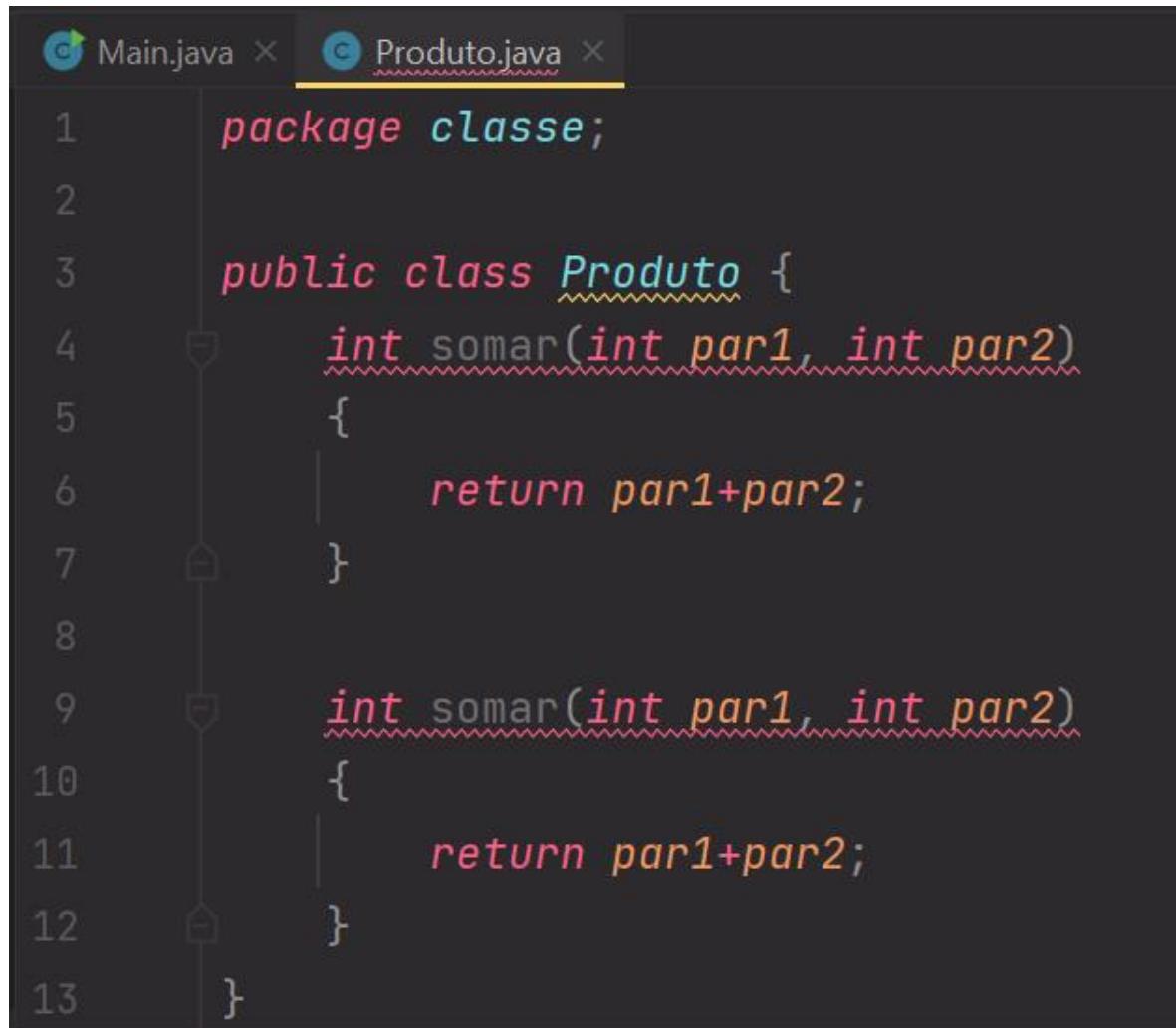
## Métodos



```
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, int par3)
10    {
11        return par1+par2+par3;
12    }
13
14 }
```

# PROGRAMAÇÃO ORIENTADA A OBJETOS

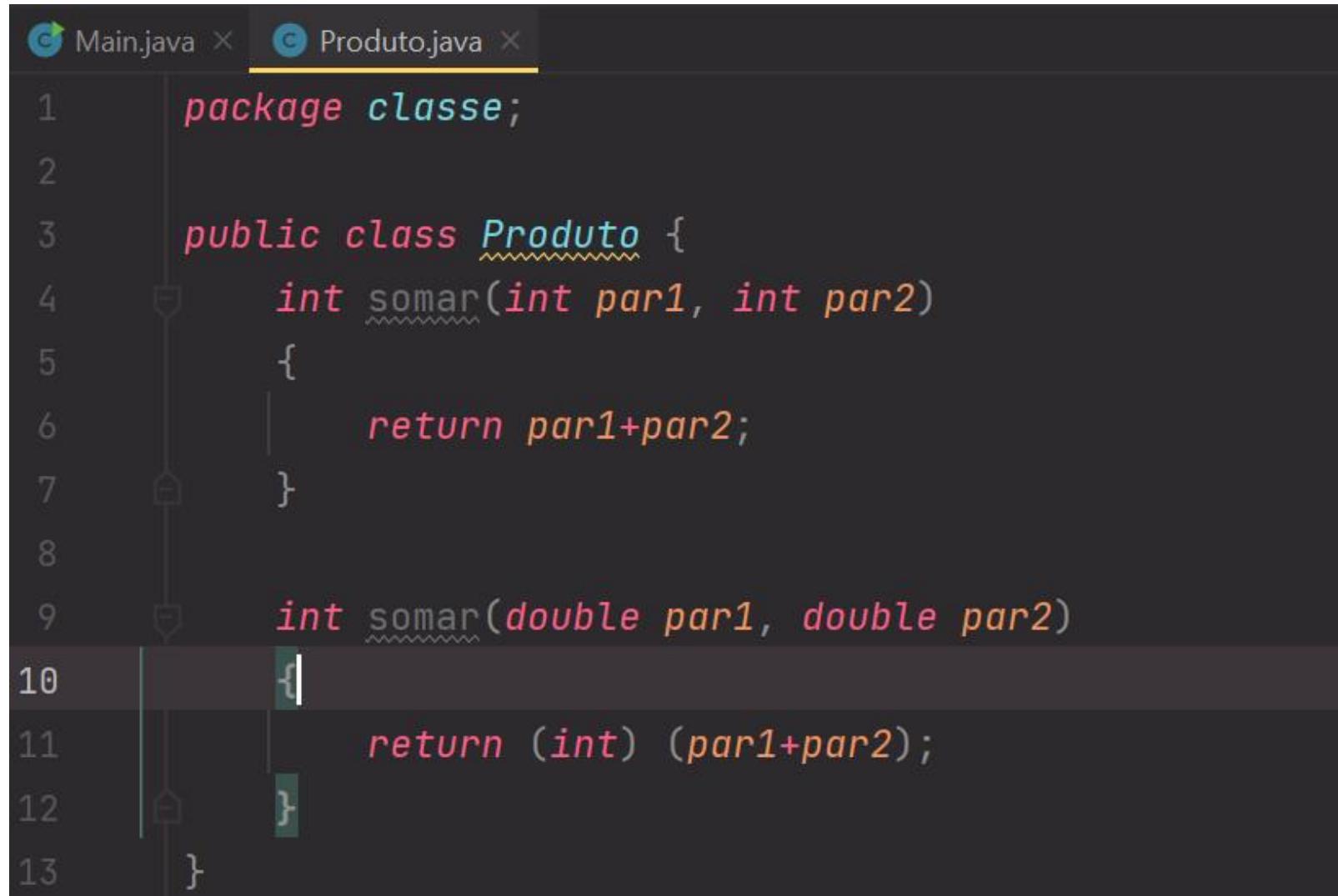
## Métodos



```
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 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(double par1, double par2)
10    {
11        return (int) (par1+par2);
12    }
13 }
```

# PROGRAMAÇÃO ORIENTADA A OBJETOS

## Construtor



```
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
10    void Produto(){}
11 }
```

# PROGRAMAÇÃO ORIENTADA A OBJETOS

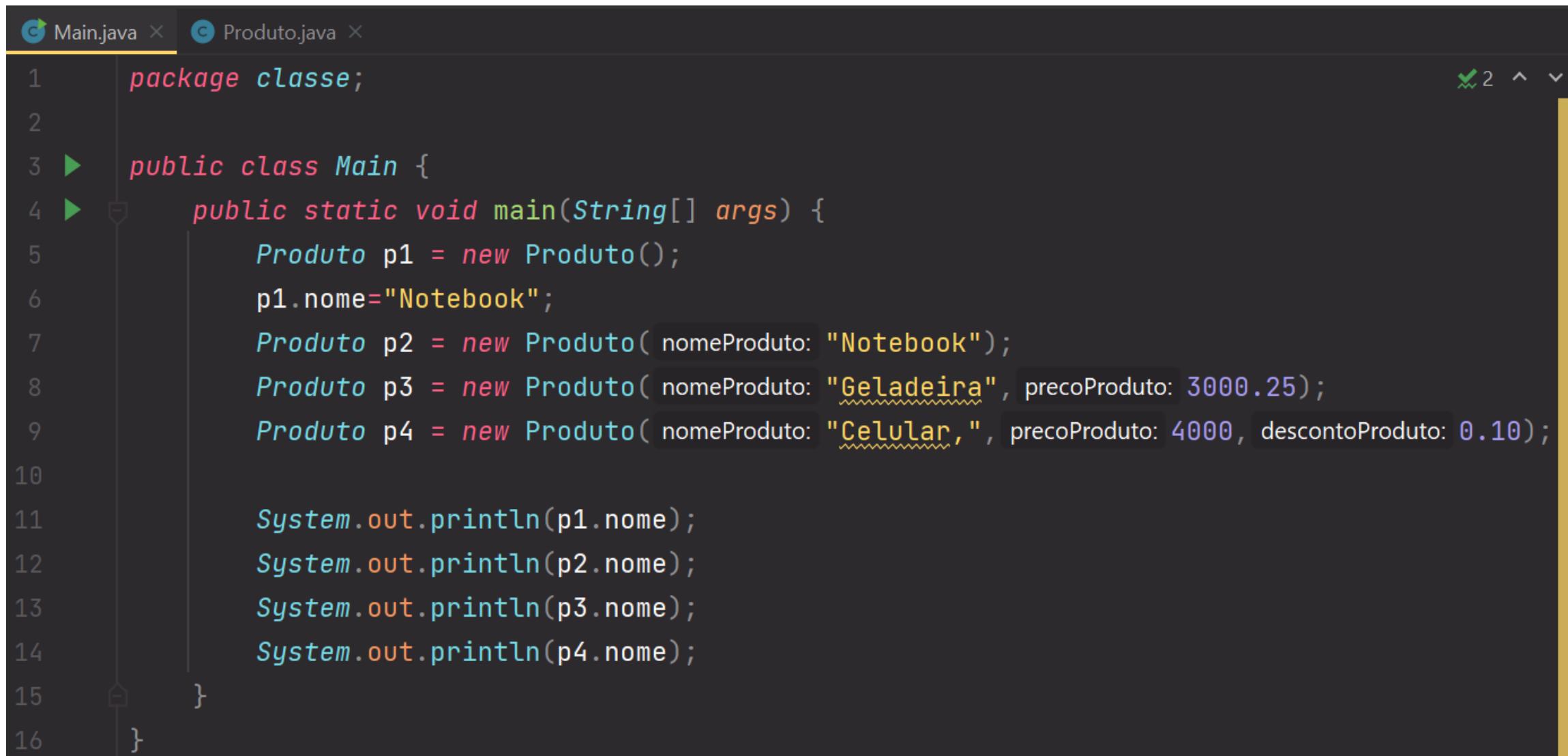
## Métodos



```
Main.java x Produto.java x
Hide Shift+Escape 1 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

## Construtor



```
Main.java x Produto.java x
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 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:

```
package classe;

public class Main {
    public static void main(String[] args) {
        Produto p1 = new Produto();
        p1.nome="Notebook";
        Produto p2 = new Produto( nomeProduto: "Notebook");
        Produto p3 = new Produto( nomeProduto: "Geladeira", precoProduto: 3000.25);
        Produto p4 = new Produto( nomeProduto: "Celular,", precoProduto: 4000, descontoProduto: 0.10);

        System.out.println(p1.nome);
        System.out.println(p2.nome);
        System.out.println(p3.nome);
        System.out.println(p4.nome);
    }
}
```

The 'Produto.java' tab is visible but its content is not shown. The code editor has a dark theme with syntax highlighting. The BOSCH logo is visible in the bottom right corner.

# 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

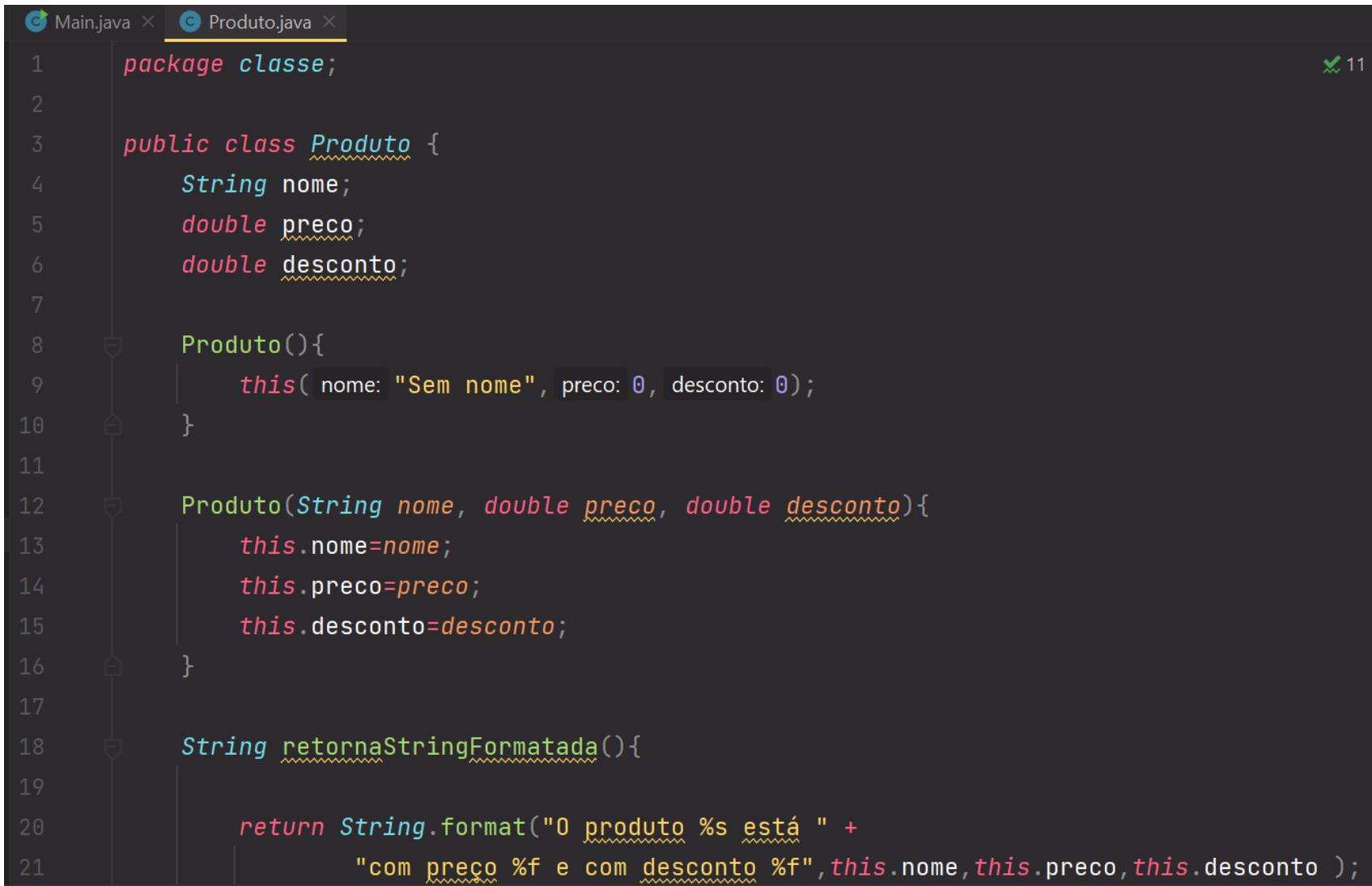
### ► **Construtores podem ser sobrecarregados**

### ► **Pode existir mais de um**

- **Sobrecarga do construtor é permitida**
- Construtores são chamados com a palavra-chave **new**

# PROGRAMAÇÃO ORIENTADA A OBJETOS

## This



The screenshot shows 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 );
22    }
}
```

The code defines a class `Produto` with three fields: `nome`, `preco`, and `desconto`. It contains two constructors: a default constructor that initializes the product name to "Sem nome" and sets both price and discount to 0; and a parameterized constructor that takes the product name, price, and discount as arguments and assigns them to their respective fields. The class also has a method `retornaStringFormatada` that returns a formatted string describing the product.

# 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}
```

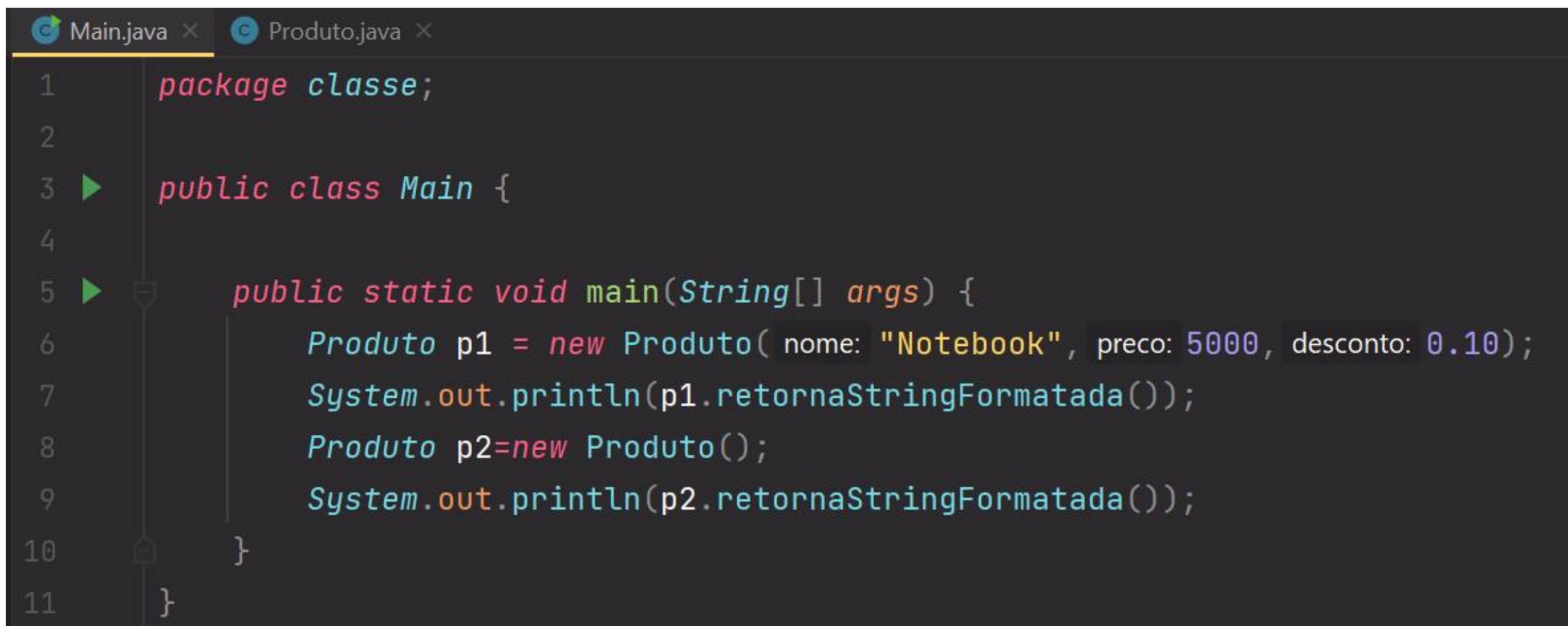
# PROGRAMAÇÃO ORIENTADA A OBJETOS

## 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

