PROGRAMAÇÃO ORIENTADA A OBJETOS

AULA 7 – OPERADORES DE REPETIÇÃO

WALTER TRAVASSOS SARINHO

@PROF.WALTERTRAVASSOS

WALTER.TRAVASSOS@SECTRAS.EDU.BR

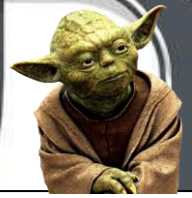
Dúvidas sobre a aula passada

Mapa de Estudos

AULA 01
Paradigmas de Programação e história do Java.
IDEs e Hello Universe

AULA 03
Visibilidade de variáveis, classe String e casting.

AULA 02
Manipulação de datas.
Sintaxe da Linguagem Java e variáveis primitivas e de referência.



AULA 04
Classes Wrapper.
Operadores e métodos de conversão.

AULA 05
Formatar a saída de dados numérica.
Operador condicional if.
Métodos para comparar Strings

Avaliação 1

Java

Mapa de Estudos

AULA 06
Manipulação de Strings. Operador condicional switch.

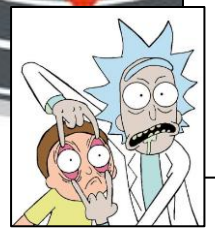
AULA 08
Vetores (for each) e matrizes. Classe ArrayList

AULA 10
Construindo métodos e Métodos construtores

Avaliação 2

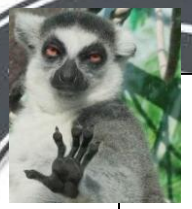
AULA 09
O paradigma Orientado a Objetos

AULA 07
Operadores de atribuição compostos. Estruturas de repetição: while, do while e for.



Java

Mapa de Estudos



AULA 12
Classes Abstratas e Interfaces

AULA 11
Pilares da POO: Abstração, Encapsulamento, Herança e Polimorfismo

Avaliação 3

Java

Aula de Hoje

- Operadores de atribuição compostos.
- Estruturas de Repetição: for, while e do while.
- Exercícios

Operadores de atribuição compostos

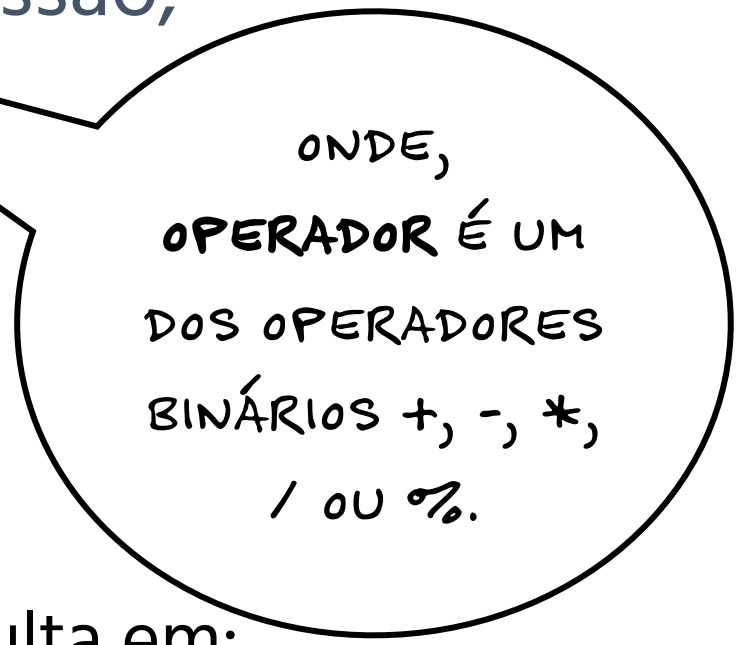
Operadores de Atribuição Compostos

- Abreviam expressões de atribuição. As instruções como:

variável = variável **operador** expressão;

- Podem ser escritas como:

variável **operando** = expressão;



- Por exemplo:

c = c + 3;

- Com o operador composto de atribuição, **+=**, resulta em:

c +=3;

Exemplos

Suponha: int c = 3; d = 5; e = 4; f = 6; g = 12;

Operador de Atribuição	Expressão de Exemplo	Explicação	Atribuição
+=	c += 7	c = c + 7	10 para c
-=	d -= 4	d = d – 4	1 para d
*=	e *= 5	e = e * 5	20 para e
/=	f /= 3	f = f / 3	2 para f
%=	g %= 9	g = g % 9	3 para g

Operadores de Incremento e Decremento

- Java fornece dois **operadores unários** para adicionar 1 e subtrair 1 do valor de uma variável.
- Operador de incremento **(++)** e decremento (**--**).
- Em vez de utilizar **i+=1** podemos utilizar **i++** ou utilizar **i--** ao invés de **i-=1.**

Operadores de Incremento e Decremento

- Se o operador for colocado antes da variável, por exemplo, **++i**, é chamado operador de **pré-incremento**.
- Se for colocado após a variável, por exemplo, i++, é chamado de operador de **pós-incremento**.

- E como isso vai se comportar no programa?

Operadores de Incremento e Decremento

Operador	Nome do Operador	Exemplo	Explicação
++	Pré-incremento	++a	Incrementa **a** por 1, então utiliza o novo valor de **a** na expressão em que **a** reside.
++	Pós-incremento	a++	Usa o valor atual de **a** na expressão em que **a** reside, então incrementa **a** por 1
--	Pré-decremento	--b	Decrementa **b** por 1, então utiliza o novo valor de **b** na expressão em que **b** reside.
--	Pós-decremento	b--	Usa o valor atual de **b** na expressão em que **b** reside, então decrementa **b** por 1.

Principal.java

```java

public class Principal {

    public static void main(String[] args) {
        int a=0;
        int b = 10;
        System.out.println(a++);
        System.out.println(a);
        System.out.println(--b);
        System.out.println(b++);
        System.out.println(b);

    }

}
```

Principal.java

```
1
2  public class Principal {
3
4      public static void main(String[] args) {
5          int a=0;
6          int b = 10;
7          System.out.println(a++);
8          System.out.println(a);
9          System.out.println(--b);
10         System.out.println(b++);
11         System.out.println(b);
12
13     }
14
15 }
```

0

1

9

9

10

O QUE VAI APARECER NO CONSOLE?

```java

public class Principal {

    public static void main(String[] args) {
        int a=0;
        int b = 10;
        System.out.println(a++);
        System.out.println(a);
        System.out.println(--b);
        System.out.println(b++);
        System.out.println(b);

    }

}
```

O QUE VAI APARECER NO CONSOLE?

```java

public class Principal {

    public static void main(String[] args) {
        int a=0;
        int b = 10;
        System.out.println(a++);
        System.out.println(a);
        System.out.println(--b);
        System.out.println(b++);
        System.out.println(b);

    }

}
```

```
0
1
9
9
10
```

```java
public class Principal {

    public static void main(String[] args) {

        int a = 1;
        go(a++, a);
        System.out.println("Valor final de a: "+a);
    }

    private static void go(int b, int c) {
        System.out.println("1° argumento (a++): " + b);
        System.out.println("2° argumento (a): " + c);
    }
}
```

```java
public class Principal {

    public static void main(String[] args) {

        int a = 1;
        go(a++, a);
        System.out.println("Valor final de a: "+a);
    }

    private static void go(int b, int c) {
        System.out.println("1° argumento (a++): " + b);
        System.out.println("2° argumento (a): " + c);
    }
}
```

Problems @ Javadoc Declaration Console

<terminated> Principal (41) [Java Application] C:\Users\walte\.p2\

```
1° argumento (a++): 1
2° argumento (a): 2
Valor final de a: 2
```

```
13          int c = 0;
14          int d = c++;
15          System.out.println(d);
16
17          int e = 0;
18          int f = ++e;
19          System.out.println(f);
```

```
13          int c = 0;
14          int d = c++;
15          System.out.println(d);
16
17          int e = 0;
18          int f = ++e;
19          System.out.println(f);
```

0

1

Estruturas de Repetição

Em Java, existem três tipos de estruturas de repetição:

- Comando for: Equivalente ao comando PARA em algoritmos.
- Comando while: Equivalente ao comando ENQUANTO em algoritmos.
- Comando do...while: Equivalente ao comando REPITA ENQUANTO em algoritmos.

Estruturas de Repetição for

DEITEL, P.; DEITEL, H. Java: como programar. 10. ed. São Paulo: Pearson, 2017. p. 121-128.

for

- A estrutura de repetição for permite que uma **lista de comandos** seja **executada várias vezes**.
- A estrutura adota uma **variável de controle**, que possui um **valor inicial**. A cada iteração do laço, o valor da variável de controle é incrementado ou decrementado, e o laço se repete.
- O laço se repetirá até que a variável de controle atinja a **condição de continuação no loop** (por meio de uma expressão relacional), ou seja, até que a condição não seja mais satisfeita(o valor seja falso).

for

Sintaxe:

```
for (inicialização; condição; iteração) {
        comando_1;
        comando_2;
        …
}
```

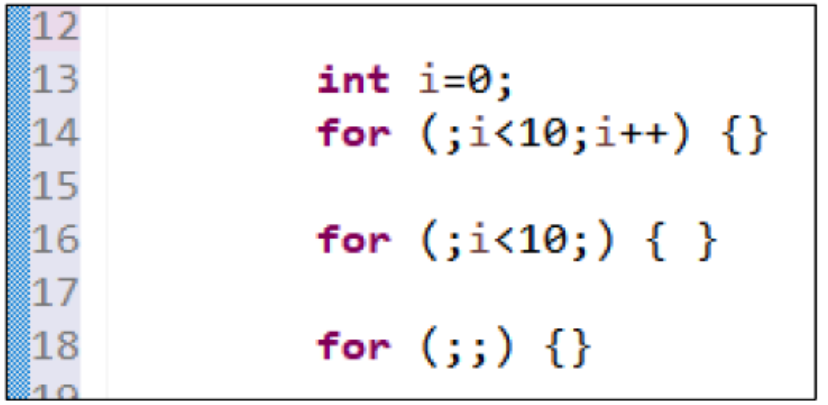
Erro comum de programação

- Utilizar uma condição de continuação no loop que nunca será satisfeita ou um valor final incorreto, pode causar um **erro fora-por-um**.
- Utilize o valor <= ou >= para evitar o erro.
- Cuidado com o estouro da variável de loop!
- Não coloque o ponto e vírgula imediatamente após os parênteses do for. Vai transformar o corpo do for numa instrução vazia.

As expressões no cabeçalho de uma instrução for são opcionais!

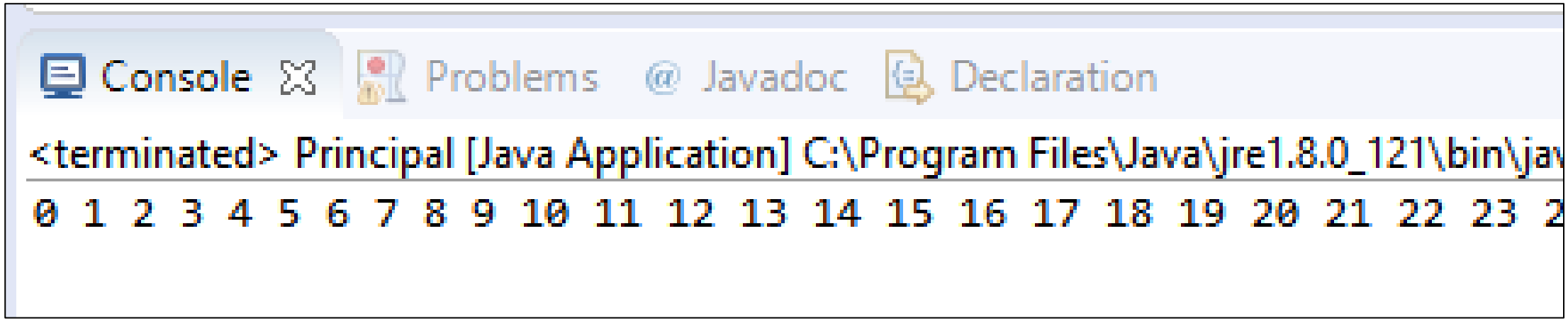
Todas as três expressões do cabeçalho são opcionais (linha 18).

- Pode omitir a **expressão de inicialização** se o programa inicializar a variável antes do loop (linhas 13 e 14).
- Se a **condição de continuação** for omitida, o Java assume que é um loop infinito (linha 18).
- Pode omitir o **incremento** se ele acontecer dentro do loop (linha 18).



Exercício 1

Faça um programa que escreve lado a lado os números entre 0 e 300.



Resposta – Exercício 1

*Principal.java

```
1  public class Principal {
2      public static void main (String[] args) {
3          int i;
4
5              for(i=0; i<=300; i++)                {
6                  System.out.print(i + " ");
7               }
8
9      }
10 }
```

Console | Problems | Javadoc | Declaration

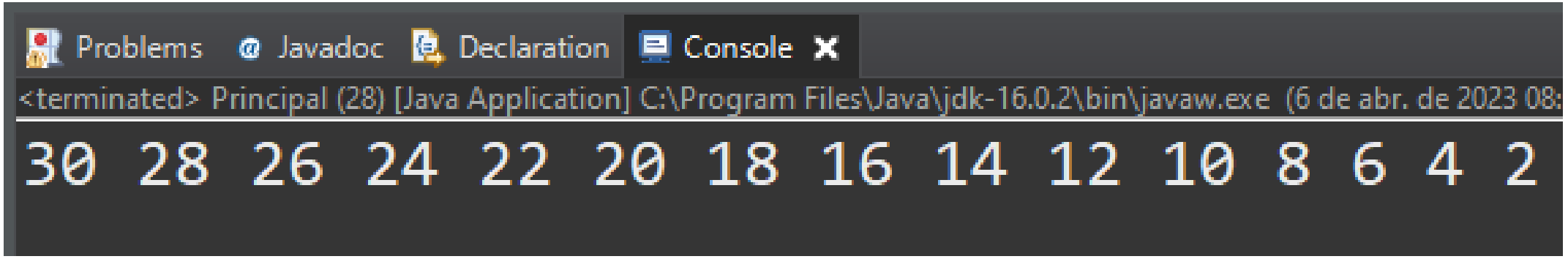
<terminated> Principal [Java Application] C:\Program Files\Java\jre1.8.0_121\bin\jav

```
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 2
```

Exercício 2

Faça um programa em Java que gera e exibe os números inteiros **pares** de 30 até 1.

Sem usar um if. Use apenas a estrutura de repetição for.



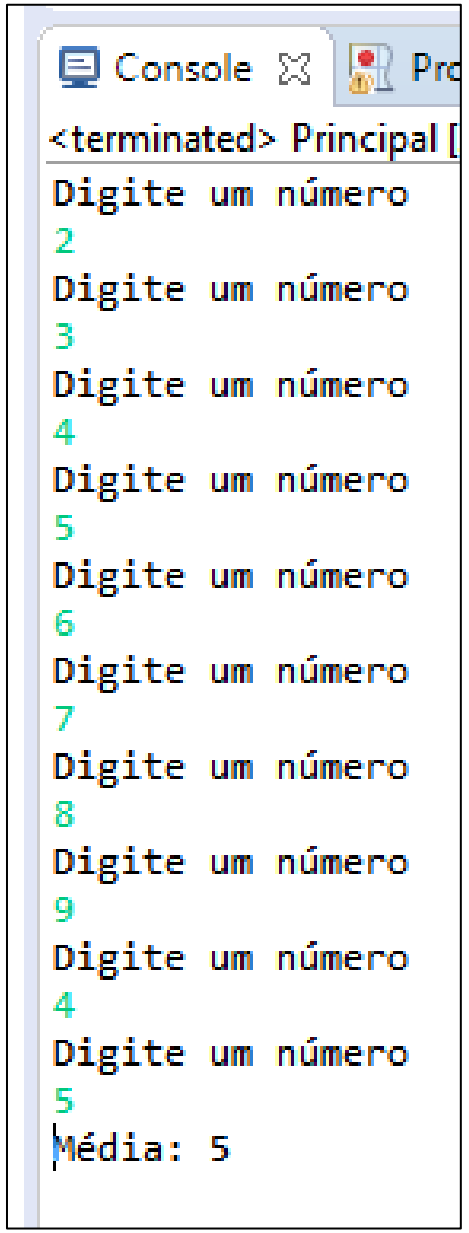
Resposta – Exercício 2

```
 3 public class Principal {
 4
 5     public static void main(String[] args) {
 6         for (int i=30; i>=1; i-=2) {
 7             System.out.print(i+" ");
 8         }
 9     }
10 }
```

Exercício 3

Escreva um programa em Java que recebe 10 números do usuário e em seguida calcula e exibe a média dos números.

OBS: Será que é preciso declarar 10 variáveis para armazenar cada número digitado pelo usuário?



Resposta – Exercício 3

Principal.java

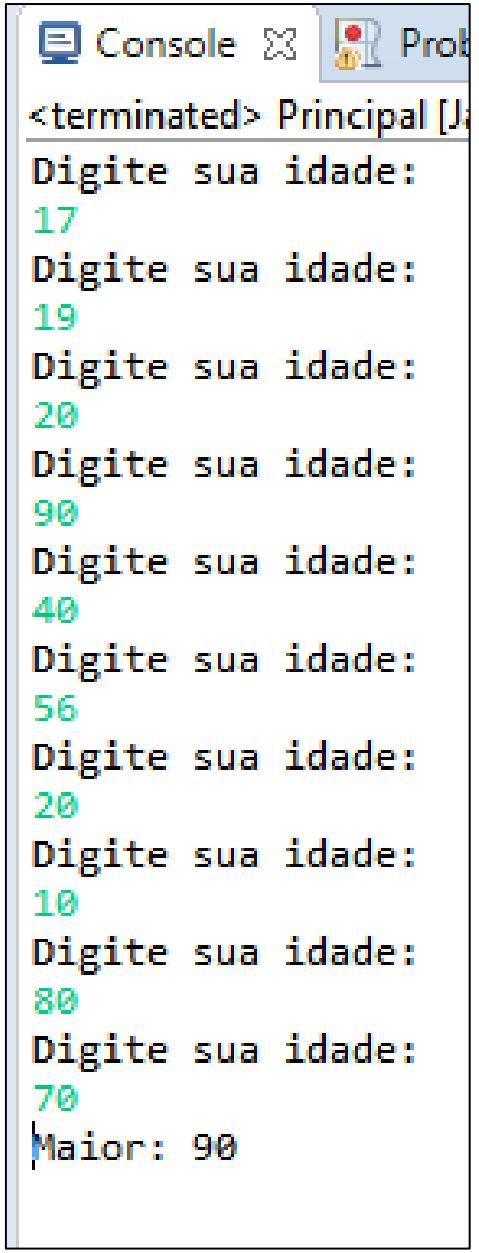
```java
import java.util.Scanner;

public class Principal {
    public static void main (String[] args) {
        Scanner entrada = new Scanner(System.in);
        int i, num, soma = 0;

        for(i=1; i <= 10; i++) {
            System.out.println("Digite um número");
            num = entrada.nextInt();
            soma = soma + num;
        }
        System.out.println("Média: " + (soma/10));
    }
}

```

Console

```
<terminated> Principal [
Digite um número
2
Digite um número
3
Digite um número
4
Digite um número
5
Digite um número
6
Digite um número
7
Digite um número
8
Digite um número
9
Digite um número
4
Digite um número
5
Média: 5
```

Exercício 4

Modifique o exercício anterior para que o programa receba a idade de 10 pessoas e informe a maior idade.



Resposta – Exercício 4

Principal.java

```java
import java.util.Scanner;

public class Principal {
    public static void main (String[] args) {
        Scanner entrada = new Scanner(System.in);
        int i, idade, maior = 0;
        for(i=0; i < 10; i++) {
            System.out.println("Digite sua idade:");
            idade = entrada.nextInt();

            if(idade > maior)   {
                maior = idade;
            }
        }
        System.out.println("Maior: " + maior);
    }
}
```

Console

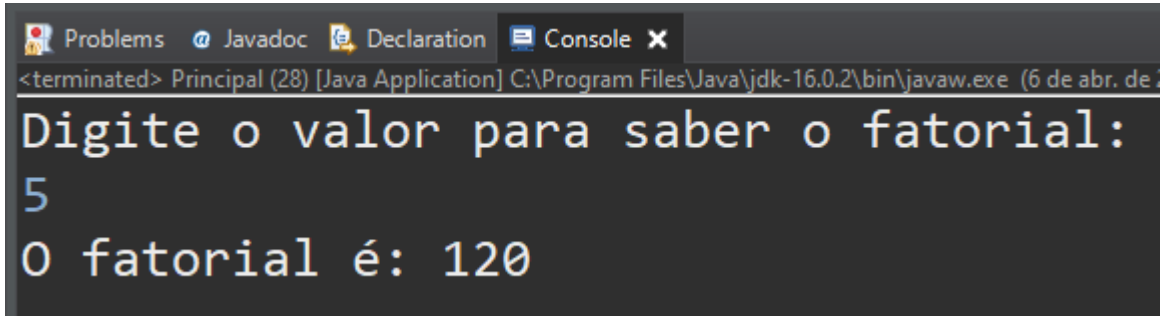
```
<terminated> Principal [J
Digite sua idade:
17
Digite sua idade:
19
Digite sua idade:
20
Digite sua idade:
90
Digite sua idade:
40
Digite sua idade:
56
Digite sua idade:
20
Digite sua idade:
10
Digite sua idade:
80
Digite sua idade:
70
Maior: 90
```

Exercício 5

Escreva um programa em java para ler um valor inicial A e imprimir a sequência de valores do cálculo de A! e o seu resultado.

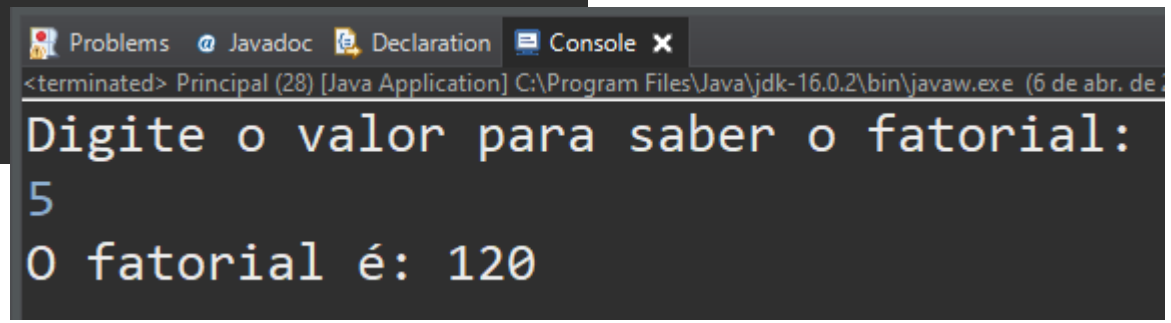
Por exemplo:

5! = 5 * 4 * 3 * 2 * 1 = 120



Exercício 5 - Resposta

```java
public class Principal {
    public static void main(String[] args) {
        Scanner teclado = new Scanner(System.in);
        System.out.println("Digite o valor para saber o fatorial:");
        int a = teclado.nextInt();
        int resultado = fatorial(a);
        System.out.println("O fatorial é: "+resultado);
    }

    public static int fatorial(int a) {
        int resposta = 1;
        for (int i = a; i>0; i--) {
            resposta *= i;
        }
        return resposta;
    }
}
```



Intervalo: 15 minutos

Estrutura de repetição while

DEITEL, P.; DEITEL, H. Java: como programar. 10. ed. São Paulo: Pearson, 2017. p. 89-97.

Instrução de Repetição

- Permite especificar que um programa deve repetir uma ação enquanto alguma condição permanece verdadeira.

Pseudocódigo:

- **Enquanto houver mais itens em minha lista de compras, Comprar o próximo item e riscá-lo da minha lista.**

- A ação de comprar próximo item e riscá-lo da lista será repetida enquanto houverem itens na lista de compras.

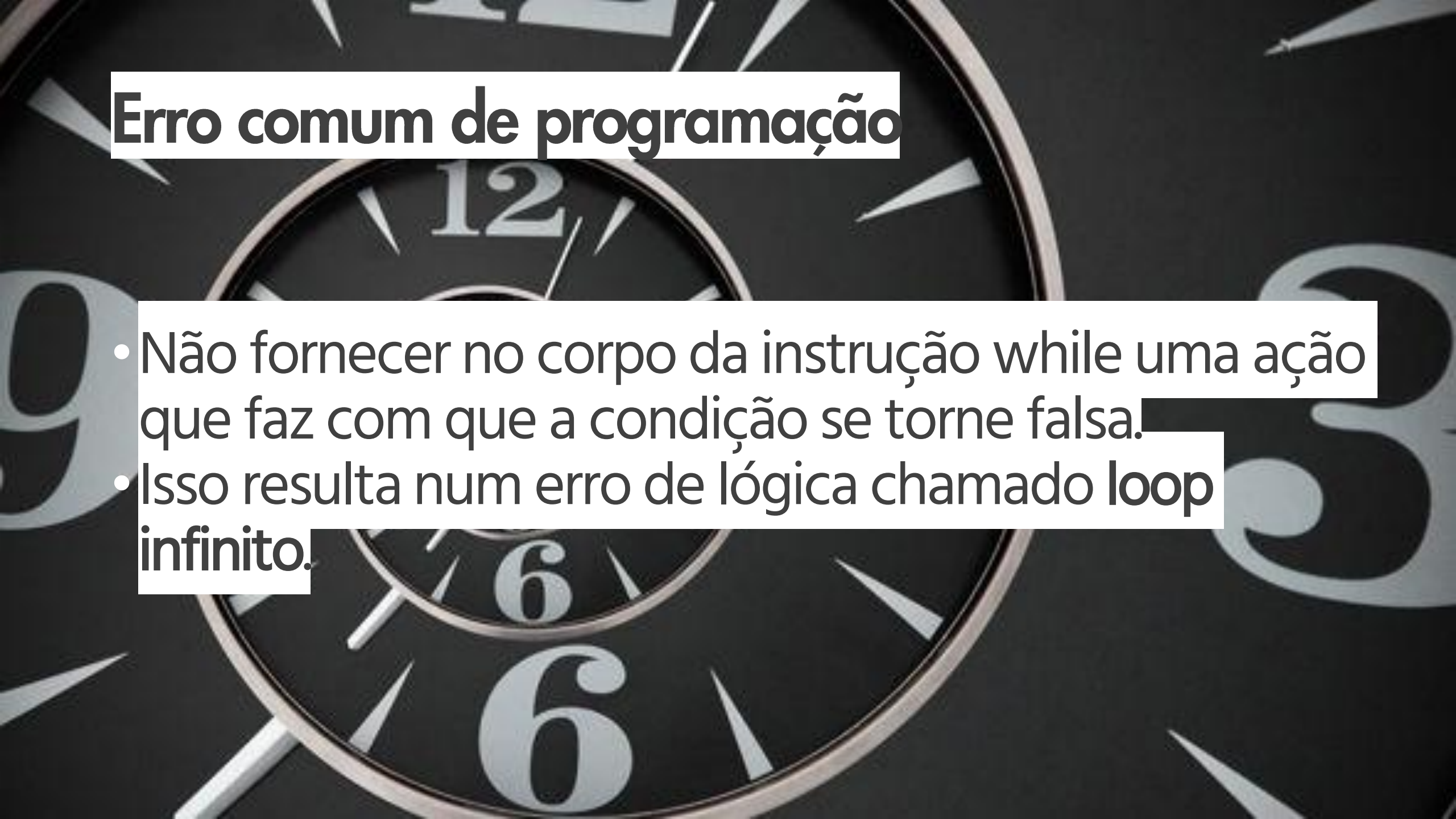
while

- A estrutura de repetição while permite que um ou mais comandos sejam executados várias vezes, **enquanto uma condição de controle for verdadeira**.
- Enquanto essa condição for verdadeira, a sequência de comandos é executada. Quando a condição se tornar falsa, a sequência será ignorada.
- Naturalmente, pode ocorrer que a sequência não seja executada nenhuma vez (caso a condição seja falsa na primeira avaliação).

while

- Funcionamento similar ao comando **enquanto...faca** em algoritmos.
- Sintaxe:

```
while (condição)  {
        comando_1;
        comando_2;
        ...
}
```



Erro comum de programação

- Não fornecer no corpo da instrução while uma ação que faz com que a condição se torne falsa.
- Isso resulta num erro de lógica chamado **loop infinito**.

Outro erro comum de programação

- Declarar (criar) a variável **contador** dentro do escopo da estrutura while. A cada iteração a variável é criada novamente. Isso irá zerar sua contagem.
- Utilizar uma variável local antes de ser inicializada resulta em erro de compilação. Todas as variáveis locais devem ser inicializadas antes de seus valores serem utilizados nas expressões.

Controlando a Repetição

Variável de Controle

- Utilizamos repetição controlada por **contador** (uma variável de controle).
- O contador é incrementado ao final da estrutura while.
- É possível prever a quantidade de repetições, por este motivo, também é chamado de **repetição definida**.

Problema

- Desenvolva um programa para tirar a média da classe que processe as notas de acordo com um número arbitrário de alunos toda vez que é executado.

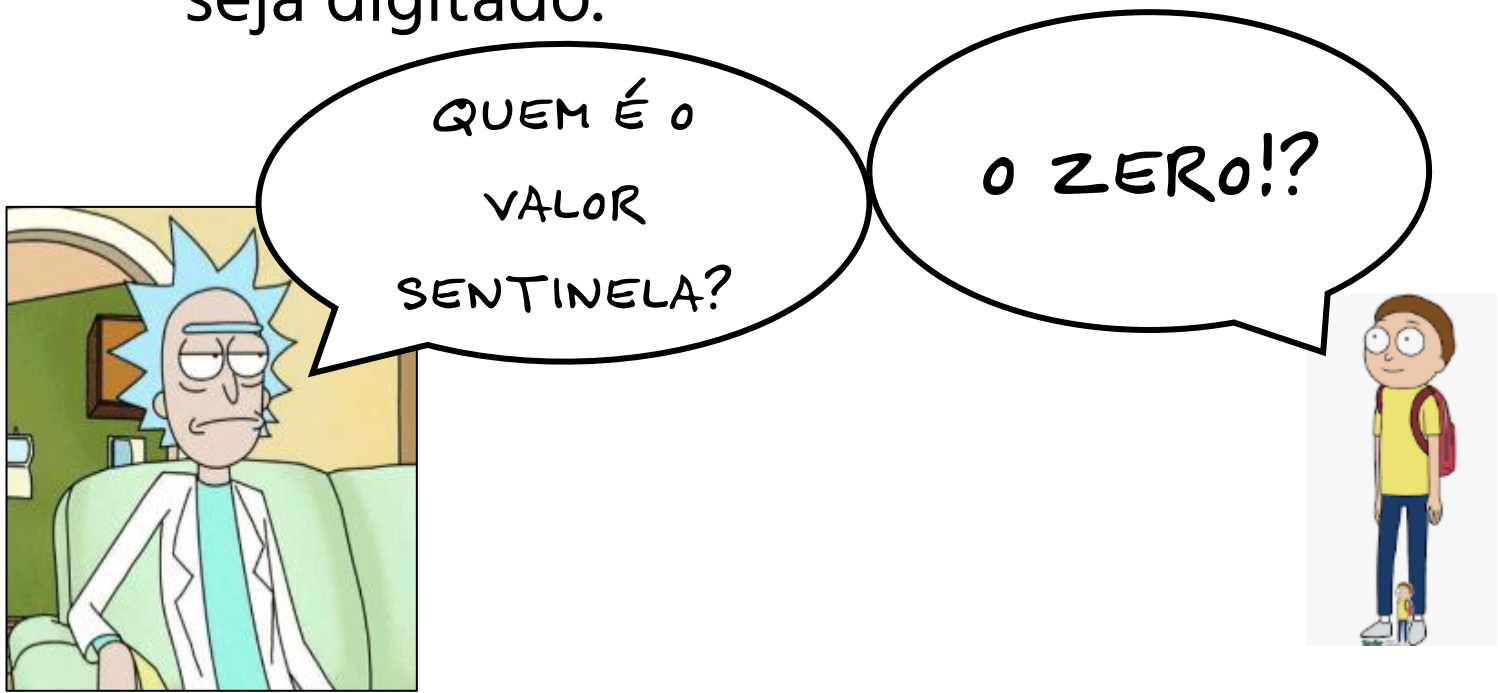
- Como podemos determinar quando parar de inserir as notas? Como saber quando calcular e imprimir a média da classe?
- R.:
- Utilizando um valor especial chamado **valor de sentinela**!

Valor de Sentinela

- Também chamado de **valor de sinal**, **valor fictício** ou **valor de flag.**
- Indica o **final** da entrada de dados.
- É frequentemente chamada **repetição indefinida**, uma vez que o número de repetições não é conhecido antes do loop iniciar a execução.

Exercício 6

Faça um programa para receber do usuário, repetidamente, vários números inteiros. Assim que recebe um número, ele será escrito no console. Este processo deve se repetir até que o número zero seja digitado.




```
Console   Problems
<terminated> Principal [Java A
Digite um número:
3
Número digitado: 3
Digite um número:
4
Número digitado: 4
Digite um número:
0
Número digitado: 0
```

Resposta – Exercício 6

Principal.java

```
1  import java.util.Scanner;
2
3  public class Principal {
4      public static void main (String[] args) {
5          Scanner entrada = new Scanner(System.in);
6          int numero = 1;
7          while(numero != 0) {
8              System.out.println("Digite um número:");
9              numero = entrada.nextInt();
10             System.out.println("Número digitado: "+numero);
11             }
12     }
13 }
14
```

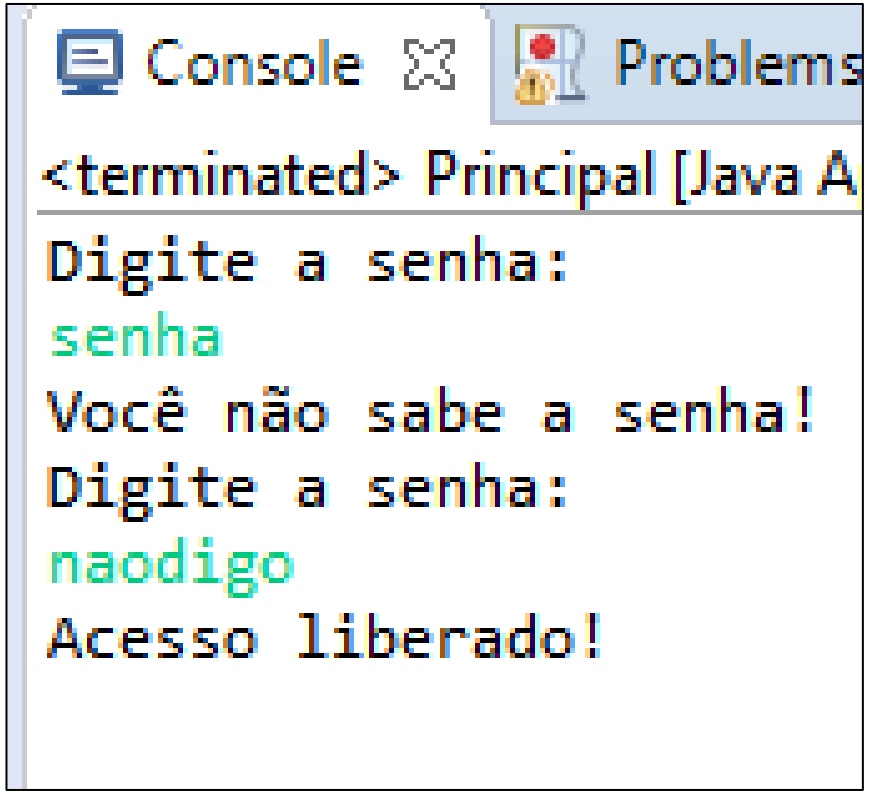
Console | Problems

<terminated> Principal [Java A

```
Digite um número:
3
Número digitado: 3
Digite um número:
4
Número digitado: 4
Digite um número:
0
Número digitado: 0
```

Exercício 7

Faça um programa que sempre repetirá a frase '**Você não sabe a senha! =P**' enquanto o usuário não digitar a senha 'naodigo'.



Resposta – Exercício 7

Principal.java

```java
import java.util.Scanner;

public class Principal {
    public static void main (String[] args) {
        Scanner entrada = new Scanner(System.in);
        String senha;

        System.out.println("Digite a senha:");
        senha = entrada.next();

        while(!senha.equals("naodigo")) {
            System.out.println("Você não sabe a senha!");
            System.out.println("Digite a senha:");
            senha = entrada.next();
        }
        System.out.println("Acesso liberado!");
    }
}
```

Console | Problems

```
<terminated> Principal [Java A
Digite a senha:
senha
Você não sabe a senha!
Digite a senha:
naodigo
Acesso liberado!
```

Estrutura de Repetição do...while

DEITEL, P.; DEITEL, H. Java: como programar. 10. ed. São Paulo: Pearson, 2017. p. 128-129.

do...while

- A estrutura de repetição do...while executa repetidamente uma sequência de instruções até que uma dada condição seja verdadeira.
- As instruções do laço serão executadas pelo menos uma vez, ao contrário do while, que pode ser executada zero ou mais vezes.

do...while

Sintaxe:

```
do {
    comando_1;
    ...
    comando_n;
} while (condição);
```

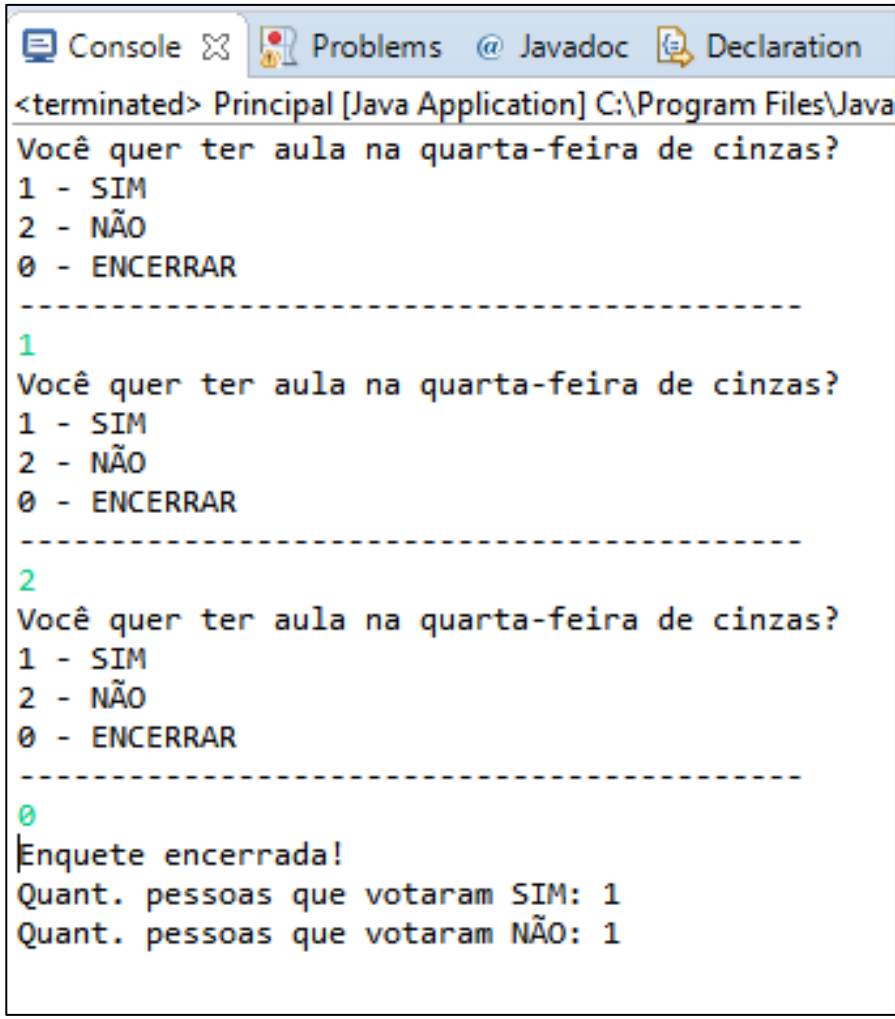
Exercício 8

Faça um programa que receba e conte votos para a seguinte enquete:

"Você quer ter aula na quarta-feira de cinzas?".

Caso o eleitor digite 1, seu voto será SIM; Caso digite 2, será NÃO. O programa deverá repetir a operação acima (através do laço de repetição do...while) até que o eleitor digite 0 em seu voto.

Ao final, exiba o total dos votos.



Resposta – Exercício 8

Principal.java

```java
import java.util.Scanner;

public class Principal {
    public static void main (String[] args) {
        Scanner entrada = new Scanner(System.in);
        int voto, quant_sim=0, quant_nao=0;
        do {
            System.out.println("Você quer ter aula na quarta-feira de cinzas?");
            System.out.println("1 - SIM");
            System.out.println("2 - NÃO");
            System.out.println("0 - ENCERRAR");
            System.out.println("----------------------------------------");
            voto = entrada.nextInt();
            switch(voto) {
                    case 1: quant_sim++;
                            break;
                    case 2: quant_nao++;
                            break;
                    case 0: System.out.println("Enquete encerrada!");
                            break;
                    default: System.out.println("Opção inválida");
            }
        } while(voto != 0);
        System.out.println("Quant. pessoas que votaram SIM: " + quant_sim);
        System.out.println("Quant. pessoas que votaram NÃO: " + quant_nao);
    }
}
```

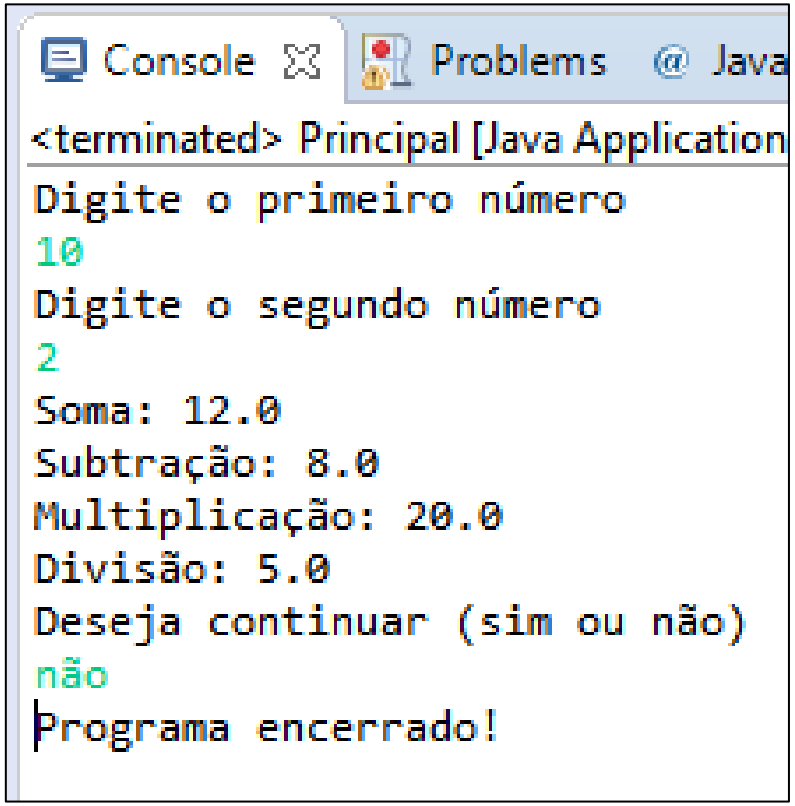
Console | Problems | Javadoc | Declaration

```
<terminated> Principal [Java Application] C:\Program Files\Java
Você quer ter aula na quarta-feira de cinzas?
1 - SIM
2 - NÃO
0 - ENCERRAR
----------------------------------------
1
Você quer ter aula na quarta-feira de cinzas?
1 - SIM
2 - NÃO
0 - ENCERRAR
----------------------------------------
2
Você quer ter aula na quarta-feira de cinzas?
1 - SIM
2 - NÃO
0 - ENCERRAR
----------------------------------------
0
Enquete encerrada!
Quant. pessoas que votaram SIM: 1
Quant. pessoas que votaram NÃO: 1
```

Exercício 9

Faça um programa que repita as instruções abaixo (utilizando o do...while):

- Receber dois números do usuário;
- Calcular e exibir a soma, subtração, multiplicação e divisão entre os números lidos;
- Perguntar ao usuário se deseja continuar a executar o programa;
- Caso o usuário digite não, o programa deverá encerrar sua execução.



Resposta – Exercício 9

Principal.java

```java
import java.util.Scanner;

public class Principal {
    public static void main (String[] args) {
        Scanner entrada = new Scanner(System.in);
        double num1, num2;
        String continua;
        do {
            System.out.println("Digite o primeiro número");
            num1 = entrada.nextDouble();
            System.out.println("Digite o segundo número");
            num2 = entrada.nextDouble();

            System.out.println("Soma: " + (num1+num2));
            System.out.println("Subtração: " + (num1-num2));
            System.out.println("Multiplicação: " + (num1*num2));
            System.out.println("Divisão: " + (num1/num2));

            System.out.println("Deseja continuar (sim ou não)");
            continua = entrada.next();
        } while(!continua.equals("não"));
        System.out.println("Programa encerrado!");
    }
}
```

Console | Problems | Java

```
<terminated> Principal [Java Application
Digite o primeiro número
10
Digite o segundo número
2
Soma: 12.0
Subtração: 8.0
Multiplicação: 20.0
Divisão: 5.0
Deseja continuar (sim ou não)
não
Programa encerrado!
```

Lista de Exercícios

Cavaleiro Jedi

PROGRAMAÇÃO ORIENTADA A OBJETOS

AULA 7 – OPERADORES DE REPETIÇÃO

WALTER TRAVASSOS SARINHO

@PROF.WALTERTRAVASSOS

WALTER.TRAVASSOS@SECTRAS.EDU.BR