



# -Introdução a Linguagem JAVA

Disciplina: Linguagem de Programação  
Profa. Me Juliana Pasquini  
11-8-2025

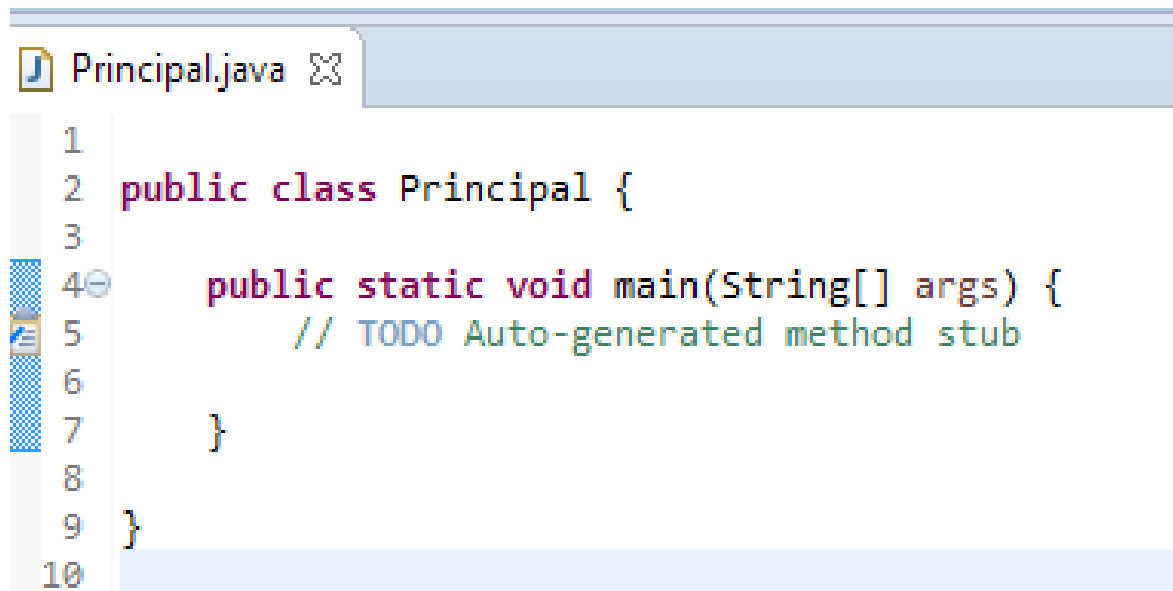
# Roteiro

2

- ❑ Meu primeiro programa;
- ❑ Tipos Primitivos;
- ❑ Declaração, atribuição de valores, casting e comparação de variáveis;
- ❑ Controle de Fluxo ( if e else);
- ❑ SWITCH;
- ❑ Operadores lógicos;
- ❑ Instruções de laço (for/while/ do while);
- ❑ Desvio com break e continue

# Meu primeiro programa

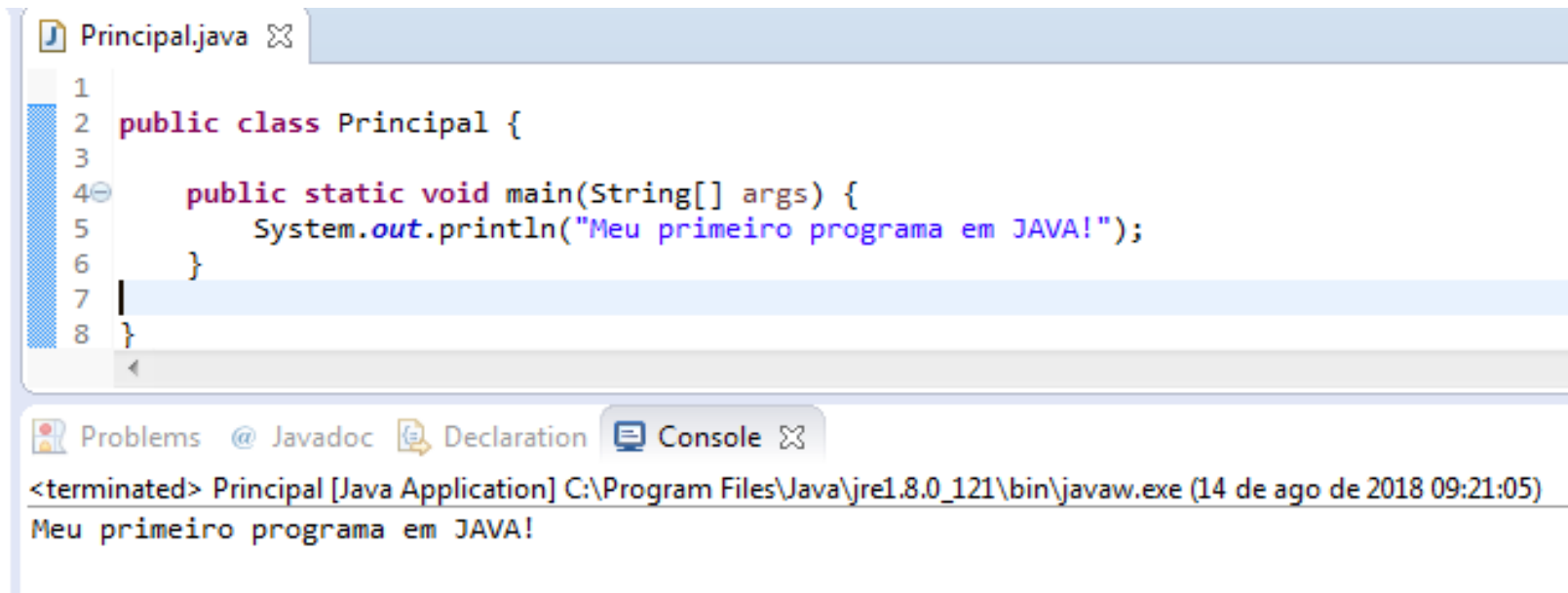
- ✓ Todas as instruções devem estar dentro de algum método, todo método deve estar dentro de alguma classe e toda classe deve estar em um arquivo com o mesmo nome da classe. Por exemplo, o método `main` esta dentro da classe `Principal`, que por sua vez, está no arquivo `Principal.java`



```
Principal.java ✖
1
2 public class Principal {
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6
7     }
8
9 }
10
```

# Meu primeiro programa

O primeiro método executado em uma aplicação JAVA é o método **main**. Ele é obrigatório dentro de uma classe da aplicação e deve ser modificado para o tipo static.



The screenshot shows an IDE with a file named 'Principal.java' open. The code is as follows:

```
1
2 public class Principal {
3
4     public static void main(String[] args) {
5         System.out.println("Meu primeiro programa em JAVA!");
6     }
7
8 }
```

Below the code editor, the 'Console' tab is active, displaying the output of the program:

```
<terminated> Principal [Java Application] C:\Program Files\Java\jre1.8.0_121\bin\javaw.exe (14 de ago de 2018 09:21:05)
Meu primeiro programa em JAVA!
```

System é uma API (*Application Programming Interface*) - Interface de Programação de Aplicação.

APIs são as bibliotecas do java. Existe uma API padrão chamada *lang*.

# Meu primeiro programa

5

```
public class Teste {  
    /**  
     * @param args  
     */  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        System.out.println("Meu primeiro programa");  
    }  
}
```

# Tipos Primitivos

6

TIPO	TAMANHO
<u>boolean</u>	<u>1</u> bit
<u>byte</u>	<u>1</u> byte
<u>short</u>	<u>2</u> bytes
<u>char</u>	<u>2</u> bytes
<u>int</u>	<u>4</u> bytes
<u>float</u>	<u>4</u> bytes
<u>long</u>	<u>8</u> bytes
<u>double</u>	<u>8</u> bytes

# Variáveis

7

## □ Declaração:

sintaxe tipo nomevariável;

Exemplo:   int idade;  
              int idade=20;  
              int idade=idade+3;  
              int resto=10%3;  
              double pi=3.14;  
              boolean verdade = true;  
              char letra = 'a';

# Exemplo - Variáveis

8

```
public class Teste {
```

```
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        int idade = 20;  
        System.out.print("Idade:"+idade+"\n");  
        boolean menorDeIdade = idade < 18;  
        System.out.print(menorDeIdade);  
        System.out.print("\n");  
        double pi=3.14;  
        char letra='a';  
        System.out.print(pi);  
        System.out.print("\n");  
        System.out.print(letra);  
    }  
}
```



# Exemplo – Variáveis -Casting

9

```
public class Teste2 {  
  
    /**  
     * @param args  
     */  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        double d3 = 3.14;  
        int i = (int) d3;  
        System.out.printf("O número float é: %.2f\n",d3);  
        System.out.printf("O número inteiro é : %d",i);  
  
    }  
}
```

# Erros - Variáveis

10

```
double d = 3.1415;  
int i = d; // não compila  
int i = 3.14; // não compila
```

```
double d = 5;  
int i = d; // não compila
```

```
long x = 10000;  
int i = x; // não compila
```

# Normas para nomear variáveis

- ✓ É convencional (mas não obrigatório) usar:
  - ✓ letra minúscula para a primeira letra do nome de uma variável
  - ✓ se o nome tiver mais de uma palavra usa-se retirar os espaços e colocar a primeira letra da próxima palavra em maiúsculo. Por exemplo, `código do cliente` se torna `codigoCliente`
  - ✓ Não usar palavras reservadas.

# Comentários

- ✓ Um comentário curto relativo a uma linha de código pode ser incluído no fim da linha, precedido de //

`int x = 10; //declaração e inicialização da variável x`

- ✓ Um comentário precedido de /\* e seguido de \*/ pode conter várias linhas

`/* no trecho de código a seguir  
serão inicializadas as variáveis e  
calculada a expressão */`

`int x = 10;`

`x = (x * 15) - 5;`

# Operadores Aritméticos

Os operadores aritméticos disponíveis em Java são:

Operador	Significado
+	adição
-	subtração
*	multiplicação
/	divisão
%	resto da divisão (módulo)

`x = 8 % 3; // x irá receber o valor 2`

Java não possui um operador específico para potência e raiz quadrada

**Precedência:** \*, /, %, +, -.

# Operadores de Atribuição

- ✓ O símbolo de igual `=` é o operador de atribuição em Java
- ✓ Este operador quando combinado com um operador aritmético (`x += y`), representa uma notação compacta de uma operação aritmética é equivalente a `x = x + y`

# Operadores de Atribuição

Operador	Exemplo	Expressão equivalente
<code>+=</code>	<code>x += y</code>	<code>x = x + y</code>
<code>-=</code>	<code>x -= y</code>	<code>x = x - y</code>
<code>*=</code>	<code>x *= y</code>	<code>x = x * y</code>
<code>/=</code>	<code>x /= y</code>	<code>x = x / y</code>
<code>%=</code>	<code>x %= y</code>	<code>x = x % y</code>

# Operadores de incremento e decremento

São operadores que atuam sobre uma única variável numérica, aumentando ou diminuindo o seu valor de uma unidade.

Operador	Exemplo	Significado
++	++a	adicionar 1 à variável <b>a</b> e depois calcular a expressão na qual <b>a</b> reside
	a++	calcular a expressão na qual <b>a</b> reside e depois adicionar 1 à variável <b>a</b>
--	--a	subtrair 1 da variável <b>a</b> e depois calcular a expressão na qual <b>a</b> reside
	a--	calcular a expressão na qual <b>a</b> reside e depois subtrair 1 da variável <b>a</b>



# Incremento/Decremento

A atribuição  $x = x + 1$  pode ser escrita das seguintes formas:

```
x++; // pós-incremento  
++x; // pré-incremento  
x+=1;
```

Da mesma forma,  $x = x - 1$  pode ser:

```
x--; // pós-decremento  
--x; // pré-decremento  
x-=1;
```

# Incremento/Decremento

Cuidado :  
pré-incremento != pós-incremento

Exemplos:

```
x=9;  
y=++x;
```

teremos, no final, y=10 e x=10

```
x=9;  
y=x++;
```

teremos, no final, y=9 e x=10

# Operadores de comparação

O resultado de uma operação de comparação é um valor booleano, true (verdadeiro) ou false (falso).

Operador	Significado
==	igual a
!=	diferente de
<	menor que
>	maior que
<=	menor ou igual a
>=	maior ou igual a

# Operadores lógicos

Operador	Significado	Exemplo	Explicação
&&	E ("logical AND")	a && b	retorna <b>true</b> se <b>a</b> e <b>b</b> forem ambos <b>true</b> . Se não retorna <b>false</b> . Se <b>a</b> for <b>false</b> , <b>b</b> não é avaliada
&	E ("boolean logical AND")	a & b	retorna <b>true</b> se <b>a</b> e <b>b</b> forem ambos <b>true</b> . Se não retorna <b>false</b> . Ambas expressões <b>a</b> e <b>b</b> são sempre avaliadas
	OU ("logical OR")	a    b	retorna <b>true</b> se <b>a</b> ou <b>b</b> for <b>true</b> . Se não retorna <b>false</b> . Se <b>a</b> for <b>true</b> , <b>b</b> não é avaliada

# Operadores lógicos

Operador	Significado	Exemplo	Explicação
	OU ("boolean logical inclusive OR")	$a \mid b$	retorna <b>true</b> se <b>a</b> ou <b>b</b> for <b>true</b> . Se não retorna <b>false</b> . Ambas expressões <b>a</b> e <b>b</b> são sempre avaliadas
^	OU EXCLUSIVO ("boolean logical exclusive OR")	$a \wedge b$	retorna <b>true</b> se <b>a</b> for <b>true</b> e <b>b</b> for <b>false</b> ou vice-versa. Se não retorna <b>false</b>
!	NÃO ("logical NOT")	$!a$	retorna <b>true</b> se <b>a</b> for <b>false</b> . Se não retorna <b>false</b>

# Controle de Fluxo - if

22

## Sintaxe:

```
if (condição_booleana) {  
    código;  
}
```

```
1  
2 public class Principal {  
3  
4     public static void main(String[] args) {  
5         int idade = 15;  
6         if (idade >= 18)  
7             System.out.println("Pode entrar");  
8  
9     }  
10  
11 }
```

Podemos omitir as chaves nos blocos das estruturas `if` quando existir apenas uma instrução no bloco.

# Controle de Fluxo – if/else

23

## Sintaxe:

```
if (condição) declaração_1;  
else declaração_2;
```

```
2 public class Principal {  
3  
4- public static void main(String[] args) {  
5     int idade = 15;  
6  
7     if (idade >= 18)  
8         System.out.println("Pode entrar");  
9     else  
10        System.out.println("Não pode entrar");  
11 }  
12 }
```

Podemos omitir as chaves nos blocos das estruturas **if** e **else** quando existir apenas uma instrução no bloco.

# ○ comando if-else-if

- A estrutura `if-else-if` é apenas uma extensão da estrutura `if-else`.
- Sua forma geral pode ser escrita como sendo:

```
if (condição_1) declaração_1;  
else if (condição_2) declaração_2;  
else if (condição_3) declaração_3;  
. . .  
else if (condição_n) declaração_n;  
else declaração_default;
```



# Controle de Fluxo – if/else if

25

```
1
2 public class Principal {
3
4     public static void main(String[] args) {
5         int num=10;
6         if (num>10)
7             System.out.println("\n\n0 numero e maior que 10");
8         else if (num==10){
9             System.out.println("\n\nVocê acertou!\n");
10            System.out.println("0 numero e igual a 10.");
11        }
12        else
13            System.out.println ("\n\n0 numero e menor que 10");
14
15    }
16
17 }
```

# SWITCH

26

```
2 public class Principal {  
3  
4 public static void main(String[] args) {  
5     char ch='1';  
6     switch (ch){  
7         case '1': System.out.printf("Opção 1");  
8         break;  
9         case '2': System.out.printf ("Opção 2");  
10        break;  
11        case '3': System.out.printf("Opção 3");  
12        break;  
13    }  
14  
15 }  
16  
17 }  
18
```

# Operadores lógicos - Exemplos

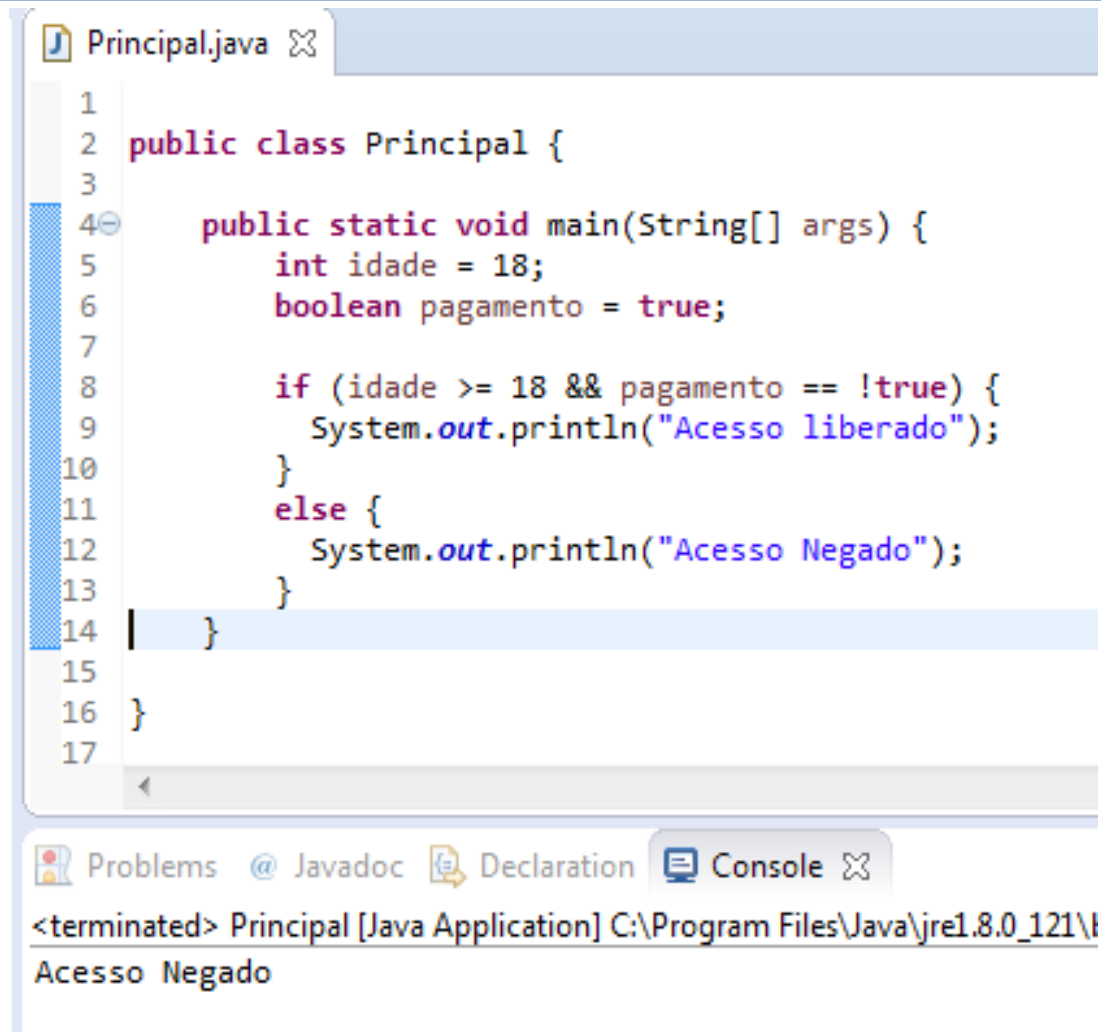
27

Principal.java

```
1
2 public class Principal {
3
4     public static void main(String[] args) {
5         int idade = 18;
6         boolean pagamento = true;
7
8         if (idade >= 18 && pagamento == true)
9             System.out.println("Acesso liberado");
10        else
11            System.out.println("Acesso Negado");
12    }
13
14 }
15
```

# Operadores lógicos - Exemplos

28



The screenshot shows an IDE window titled 'Principal.java'. The code defines a class 'Principal' with a 'main' method. Inside 'main', it sets 'idade' to 18 and 'pagamento' to true. It then uses an if-else statement with the condition '(idade >= 18 && pagamento == !true)'. Since 'pagamento' is true, the condition is false, and it prints 'Acesso Negado'. The bottom of the window shows a console output: '<terminated> Principal [Java Application] C:\Program Files\Java\jre1.8.0\_121\l Acesso Negado'.

```
1
2 public class Principal {
3
4     public static void main(String[] args) {
5         int idade = 18;
6         boolean pagamento = true;
7
8         if (idade >= 18 && pagamento == !true) {
9             System.out.println("Acesso liberado");
10        }
11        else {
12            System.out.println("Acesso Negado");
13        }
14    }
15
16 }
17
```

Problems @ Javadoc Declaration Console

<terminated> Principal [Java Application] C:\Program Files\Java\jre1.8.0\_121\l  
Acesso Negado

# Controle de Fluxo - While

29

```
1
2 public class Principal {
3
4     public static void main(String[] args) {
5         int i = 0;
6         while (i < 10) {
7             System.out.println(i);
8             i = i + 1;
9         }
10    }
11 }
```

Problems @ Javadoc Declaration Console

<terminated> Principal [Java Application] C:\Program Files\Java\jre

0  
1  
2  
3  
4  
5  
6  
7  
8  
9

# Controle de Fluxo – do while

30

```
1
2 public class Principal {
3
4 public static void main(String[] args) {
5     int i=0;
6     do{
7         System.out.println(i);
8         i=i+1;
9     } while (i < 10);
10
11
```

# Controle de Fluxo - for

31

## ❑ Sintaxe:

```
for (inicializacao; condicao; incremento) {  
    codigo;  
}
```

## ❑ Exemplo:

```
for (int i = 0; i < 10; i = i + 1) {  
    System.out.println(i);  
}
```

# Comandos de Desvio

- break;
- continue.



# BREAK

- O comando break pode ser utilizado:
  - Terminar um case em um comando switch;
  - Forçar uma terminação imediata de um laço.

# BREAK – Exemplo 1

```
public static void main(String[] args) {  
    int i;  
    for (i = 0; i < 10; i++) {  
        if (i == 5) {  
            System.out.printf("encerre o for e vá para a próxima instrução!\n ");  
            break;  
        }  
        System.out.printf("loop:%d\n", i);  
    }  
    System.out.printf("executando a próxima instrução");  
}
```

Problems Javadoc Declaration Console

<terminated> teste\_do [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (02/04/2013 20:53:41)

```
loop:0  
loop:1  
loop:2  
loop:3  
loop:4  
encerre o for e vá para a próxima instrução!  
executando a próxima instrução
```

# Continue

- ❑ O *continue* serve apenas para interromper uma determinada iteração do loop onde se encontra, fazendo um salto para a próxima iteração.
- ❑ O *continue* é utilizado somente em estruturas de repetição.

# EXEMPLO CONTINUE

```
public static void main(String[] args) {  
    int i ;  
    for (i = 0; i<10; i++) {  
        if(i == 5 ){  
            System.out.printf ("Continue e pule para o próximo loop \n");  
            continue;  
        }  
        System.out.printf("loop:  %d\n",i);  
    }  
}
```

Problems @ Javadoc Declaration Console X

<terminated> PRIMEIRA [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (04/04/2013 23:57:28)

```
loop:  0  
loop:  1  
loop:  2  
loop:  3  
loop:  4  
Continue e pule para o próximo loop  
loop:  6  
loop:  7  
loop:  8  
loop:  9
```

# Escopo de variáveis

37

```
*/  
public static void main(String[] args) {  
    // aqui a variável i não existe  
    int i = 5;  
    // a partir daqui ela existe  
    while (i > 5) {  
        // o i ainda vale aqui  
        int j = 7;  
        // o j passa a existir  
    }  
    // aqui o j não existe mais, mas o i continua dentro do escopo  
}  
}
```

# Questão 1

As estruturas de decisão são usadas para fazer com que somente parte do código seja executada mediante alguma condição. Marque a alternativa que contém um valor para a variável a que faça o texto D ser impresso na tela.

```
if( a < 20 ) {  
    System.out.println("A");  
} else if (a < 30 ){  
    System.out.println("B");  
} else if (a < 40 ){  
    System.out.println("C");  
} else if (a < 50 ){  
    System.out.println("D");  
} else{  
    System.out.println("E");  
}
```

- a) 55
- b) 25
- c) 45
- d) 15
- e) 35

## Questão 2

As estruturas de repetição são usadas para repetir um bloco de código enquanto uma condição é verdadeira. Marque a alternativa que contém o valor a ser impresso na tela.

```
public static void main(String[] args) {  
    int a = 15, s = 0;  
    while( a < 20 ) {  
        if( a % 2 == 0 ) {  
            s++;  
        }  
        a++;  
    }  
    System.out.println(s);  
}
```

- a) 0
- b) 2
- c) 10
- d) 20
- e) 30

## Questão 3

Marque a alternativa que contém o valor de x e y ser impresso na tela.

```
public class Principal {  
    public static void main(String[] args) {  
        int x = 0;  
        int y = 0;  
        for(int z = 0; z < 5; z++)  
            if ( (++x > 2) || (++y > 2) )  
                x++;  
        System.out.println(x);  
        System.out.print(y);  
    }  
}
```

- a) 5 1
- b) 5 2
- c) 5 3
- d) 8 1
- e) 8 2
- f) 8 3



## Questão 4

Marque a alternativa que contém o valor de x e y a ser impresso na tela.

```
1
2 public class Principal {
3
4     public static void main(String[] args) {
5         int x = 0;
6         int y = 0;
7         for(int z = 0; z < 5; z++)
8             if ( (++x > 2) && (y++ > 2))
9                 x++;
10        System.out.println(x);
11        System.out.println(y) ;
12    }
13 }
```

- a) 5 1
- b) 5 2
- c) 5 3
- d) 8 1
- e) 8 2
- f) 8 3

## Questão 5

Qual será a saída quando tentar compilar e executar o seguinte código?

```
public static void main(String[] args) {  
    int i=0, j=2;  
    do {  
        i=++i;  
        j--;  
    } while(j>0);  
    System.out.print(i);  
}
```

- a) 0
- b) 1
- c) 2
- d) O programa não compila por causa da linha 4

## Questão 6

Considere o seguinte código fonte a seguir, escrito na linguagem de programação Java.

```
public static void main(String[] args) {  
    int num1=20,resultado=0;  
    resultado=(20+2)/2;  
    if (resultado > 13)  
        System.out.print(num1);  
    else System.out.print (resultado+=3);  
}
```

Considerando o código fonte apresentado, marque a alternativa que descreve o que será impresso na tela.

- (A) 11
- (B) 20
- (C) 14
- (D) 13
- (E) 9

# Questão 7

Qual será a saída quando tentarmos compilar e executar o seguinte código?

```
public static void main(String[] args) {  
    int i=1;  
    switch (i) {  
        case 0:  
            System.out.print("zero");  
            break;  
        case 1:  
            System.out.print("one");  
        case 2:  
            System.out.print("two");  
        default:  
            System.out.print("default");  
    }  
}
```

- (A) one
- (B) one, default
- (C) one, two, default
- (D) default
- (E) zero