

# PARADIGMAS DE PROGRAMAÇÃO

2023/2024

P.PORTO

ESCOLA  
SUPERIOR  
DE TECNOLOGIA  
E GESTÃO

## Aula 01

1. Palavras Reservadas (*Java Keywords*)
2. Tipos primitivos
3. Literais (*Literals*)
4. Variáveis de Tipos Primitivos
5. Constantes
6. Variáveis Unidimensionais (*Arrays*)
7. Exemplos
8. Palavras Reservadas Usadas
9. Links Úteis



# Palavras Reservadas

abstract	continue	for	new	switch
assert***	default	goto*	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum****	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp**	volatile
const*	float	native	super	while

\* not used

\*\* added in 1.2

\*\*\* added in 1.4

\*\*\*\* added in 5.0



# Tipos Primitivos

- A linguagem de programação *Java* tem os seguintes tipos de dados primitivos:



Tipo Primitivo	Descrição
boolean	true/false
byte	8 bits
char	16 bits (UNICODE)
short	16 bits
int	32 bits
long	64 bits
float	32 bits IEEE 754-1985
double	64 bits IEEE 754-1985



Tipo Primitivo	Valor por defeito
boolean	FALSE
byte	0
char	'\u0000'
short	0
int	0
long	0L
float	0.0f
double	0.0d



## ❖ Literals

```
boolean result = true;  
char capitalC = 'C';  
byte b = 100;  
short s = 10000;  
int i = 100000;
```

- ❖ Em *Java*, existem tipos literais primitivos para representar valores booleanos (verdadeiro ou falso), caracteres, valores numéricos inteiros e valores numéricos em vírgula flutuante



```
int decVal = 26;    // The number 26, in decimal
int octVal = 032;   // The number 26, in octal
int hexVal = 0x1a;  // The number 26, in hexadecimal
```

```
double d1 = 123.4;
double d2 = 1.234e2; /* same value as d1, but in
scientific notation */
```

```
float f1 = 123.4f;
```



# Constantes

- ❖ Para declarar uma constante é necessário usar a palavra reservada final, exemplo:

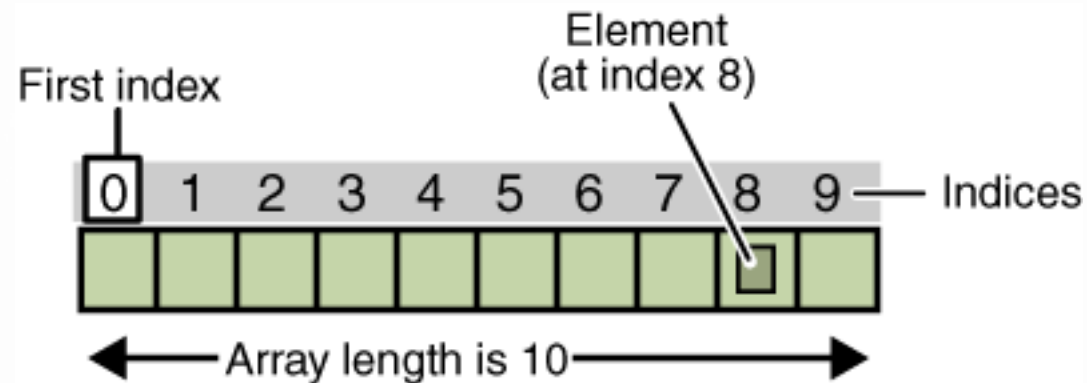
```
final int CONSTANTE = 100;
```





# Arrays

- ❖ Um *array* é um repositório que armazena um número fixo de valores de um determinado tipo
- ❖ O tamanho do *array* é definido na sua criação
- ❖ Depois de criado o seu tamanho é fixo



- ❖ Cada item de um array é denominado de elemento.
- ❖ Podemos aceder a cada um dos elementos pelo seu index numérico como podemos ver na figura.



## Declarar um Array

```
<data-type>[ ] <variable-name> =  
    new <data-type>[length];
```

## Exemplo de como declarar um array e inicializar os seus elementos

```
int[] anArray;  
  
anArray = new int[10];  
  
anArray[0] = 100;  
anArray[1] = 200;  
...  
anArray[9] = 1000;
```



- ❖ Existe também uma notação para declarar um array e inicializar todos os elementos em apenas um passo ao usar dados separados por vírgulas e entre chavetas:

Declarar e inicializar um array numa "única" instrução

```
<data-type>[ ] <variable-name> = {  
    <expression>, <expression>, ...  
};
```



## Declarar e inicializar um array numa "única" instrução

```
int x = 4;  
int[] anArray = {3, x, 9, 2};
```



## Declarar e inicializar um array numa "única" instrução

```
int x = 4;  
int[] anArray = {3, x, 9, 2};  
x= 5;  
anArray[1] = ???
```



```
/*  
Isto é um pequeno programa Java de demonstração.  
No sistema de ficheiros o nome do ficheiro que  
possui o código fonte é Exemplo.java  
*/
```

```
class Exemplo {
```

```
/* Em Java, o ponto de entrada (entry point) de  
uma aplicação de consola é o método main() */
```

```
    public static void main(String[] args) {
```

```
        System.out.println("Saudações Javanesas!!!");
```

```
    }
```

```
}
```



```
class Exemplo {
```

```
/*Um programa Java começa com uma chamada ao  
método main() */
```

```
public static void main(String args[]) {
```

```
/* Nos primeiros exemplos todo o código  
será implementado neste método */
```

```
}
```

```
}
```





```
class ArrayDemo {
    public static void main(String[] args) {
        int[] anArray;
        anArray = new int[10];

        anArray[0] = 100;
        anArray[1] = 200;
        anArray[2] = 300;
        anArray[3] = 400;
        anArray[4] = 500;
        anArray[5] = 600;
        anArray[6] = 700;
        anArray[7] = 800;
        anArray[8] = 900;
        anArray[9] = 1000;

        System.out.println("Element at index 0: " + anArray[0]);
        System.out.println("Element at index 1: " + anArray[1]);
        System.out.println("Element at index 2: " + anArray[2]);
        System.out.println("Element at index 3: " + anArray[3]);
        System.out.println("Element at index 4: " + anArray[4]);
        System.out.println("Element at index 5: " + anArray[5]);
        System.out.println("Element at index 6: " + anArray[6]);
        System.out.println("Element at index 7: " + anArray[7]);
        System.out.println("Element at index 8: " + anArray[8]);
        System.out.println("Element at index 9: " + anArray[9]);
    }
}
```



# Palavras Reservadas Usadas

abstract	continue	for	new	switch
assert**	default	goto*	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum****	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp**	volatile
const*	float	native	super	while

\* not used

\*\* added in 1.2

\*\*\* added in 1.4

\*\*\*\* added in 5.0



# Links Úteis

- ❖ <http://download.oracle.com/javase/tutorial/java/nutsandbolts/variables.html>
- ❖ <http://www.netbeans.org/kb/trails/java-se.html>