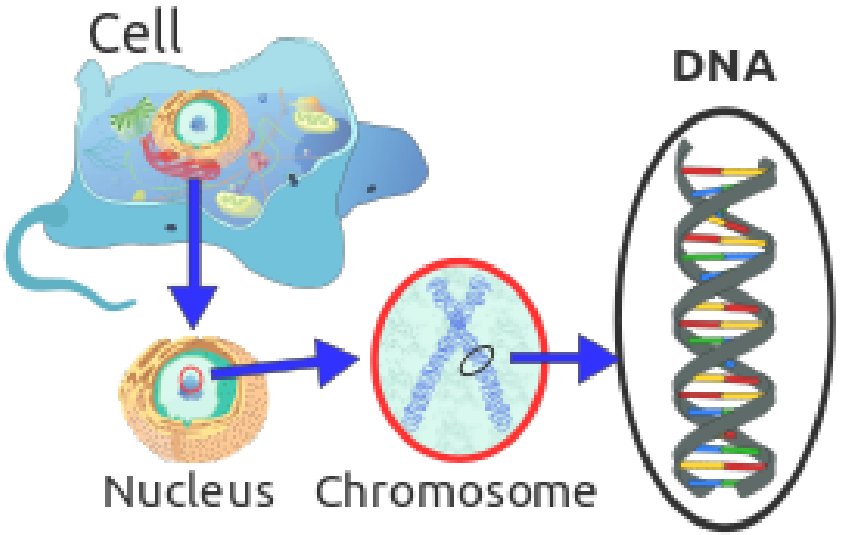
Processamento da Informação

Caracteres e Strings

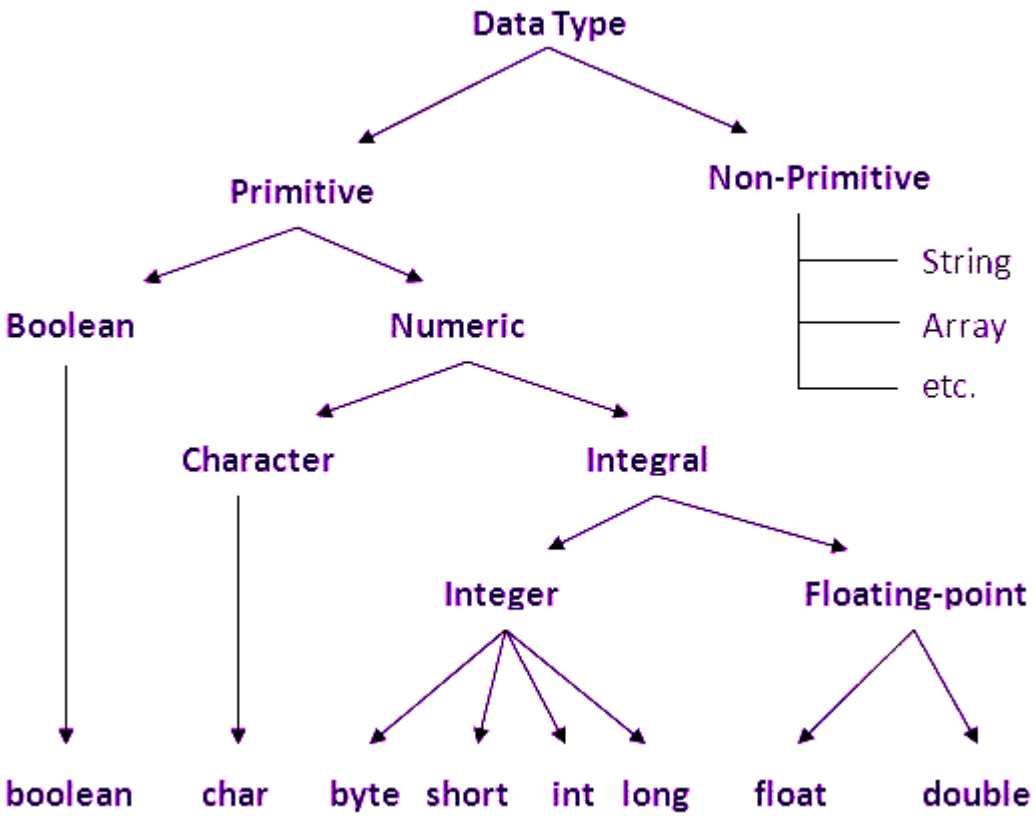
Prof. Jesús P. Mena-Chalco
CMCC/UFABC

Q2/2018


Cell
DNA
Nucleus
Chromosome


melgaco
ramos
oliveira
igor
luiz
antonio
nathalia
andre
samuel
renata
eduardo
tadeu
nozaki
ferreira
cristina
teixeira
marques
almeida
massaiuqui
pinto
cabello
caroline
formiga
ramalho
rodrigo
cavallari
martins
nascimento
batista
jorge
carvalho
daniel
brito
csapo
kalila
theo
mariana
morgado
akira
aline
felipe
victor
silva
lima
victoria
ribeiro
ana
mioshi
joao
fernando
gustavo
giovanna
saldanha
maria
lara
rodrigues
alves
pires
mie
goncalles
andrades
luisa
tognella
alino
kenji
juliana
rafael
allegretti
giovanni
wendel
petagna
tenore
alyne
miguel
follador
mendes
goncalves
paulo
gomes
gluz
christian
rudah
cunha
araujo
henrique
reckziegel
marfil
reis
otinger
monteiro
mirelli
gabriel
godoy
pereira
pavan
sa
caio
pontes
dimitri
beatriz
heleno
couri
vitor
bueno
sampaio
borghi
vinicius
pedro
bruno
carolina
alecsander
guimaraes
meneses
nayara
lucas
alex
amanda
soares
guilherme
peres
ayumi
augusto
ramon
matheus
fassina
camacho
barbieri
neres
souza
neto
isabella
natalia
episcopo
gabriela
sousa
fernanda
inacio
leonardo
lopes
rocha
vieira
santos
renan
nathan
carlos
melotto
izabella
desiderio
hasegava
cabral

Tipos de dados em Java



(*) Fonte: http://www.javatpoint.com/variable-datatype

Tipos de dados em Java

Data Type	Default Value	Default size
boolean	false	1 bit
char	'\u0000'	2 byte
byte	0	1 byte
short	0	2 byte
int	0	4 byte
long	0L	8 byte
float	0.0f	4 byte
double	0.0d	8 byte

(*) Fonte: http://www.javatpoint.com/variable-datatype

Tipos de dados em Java

Data Type	Bits	Range
byte	8	-2^7 to 2^7-1 (-128 to 127)
short	16	-2^{15} to $2^{15}-1$ (-32,768 to 32,767)
int	32	-2^{31} to $2^{31}-1$ (-2,147,483,648 to 2,147,483,647)
long	64	-2^{63} to $2^{63}-1$ (-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807)
float	32	1.40129846432481707e-45 to 3.40282346638528860e+38 (positive or negative) (single-precision 32-bit IEEE 754 floating point)
double	64	4.94065645841246544e-324d to 1.79769313486231570e+308d (positive or negative) (double-precision 64-bit IEEE 754 floating point)
char	16 (unsigned)	0 to 65,535

(*) Fonte: http://www.javatpoint.com/variable-datatype

Tipos de dados em Java

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Caracteres e strings

Caracteres

Um **char** é um tipo de dado que pode ser utilizado para armazenar apenas UM caractere, por exemplo:

- Letra
- Número
- Sinal de pontuação
- Símbolo

Os caracteres, em Java, são representados com aspas simples:

- `char var1 = 'a';`
- `char var2 = 'b';`

Erros comuns

```
char var1 = 'aaaaa';
char var2 = 'bbbbb';
```

← **Erro**

← **Erro**

```
                  error: unclosed character literal
        char var2 = 'bbbbb';
                     ^
                  error: unclosed character literal
        char var2 = 'bbbbb';
```

Erros comuns

```
char var1 = 'a';
```
← OK

```
char var2 = "b";
```
← **Erro**

```
                    error: incompatible types
        char var2 = "b";
                    ^
  required: char
  found:    String
1 error
```

Caracteres e Strings

```
char var1 = 'a';        // aspas simples para char

String var2 = "abcd";   // aspas duplas para string
```

char

→ É um e somente 1 caractere
→ É um tipo de dados primitivo

String

→ É zero (0) o mais caracteres
→ É uma **classe**.

Caracteres e Strings

Uma String **não** é um vetor de caracteres.

```
String var2 = "ufabc";

System.out.println(var2[2]);    ← erro
```

Vetor de caracteres

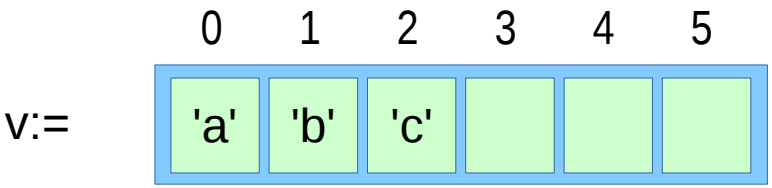
Vetor de caracteres

```
char v[] = new char[6];

v[0] = 'a';
v[1] = 'b';
v[2] = 'c';

for (int i=0; i<v.length; i++)
    System.out.println(">"+v[i]+"<");
```

```
>a<
>b<
>c<
><
><
><
```

Vetor de caracteres

`v.length → 6`



Vetor de caracteres

`v.length → 6`

v:=

0	1	2	3	4	5
'a'	'b'	'c'			

v:=

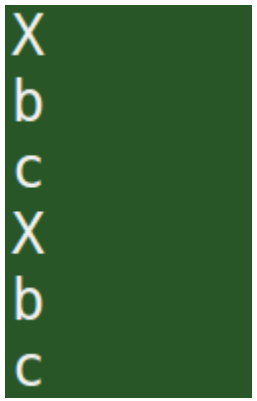
0	1	2	3	4	5
97	98	99	0	0	0

Exemplo: vetores de caracteres

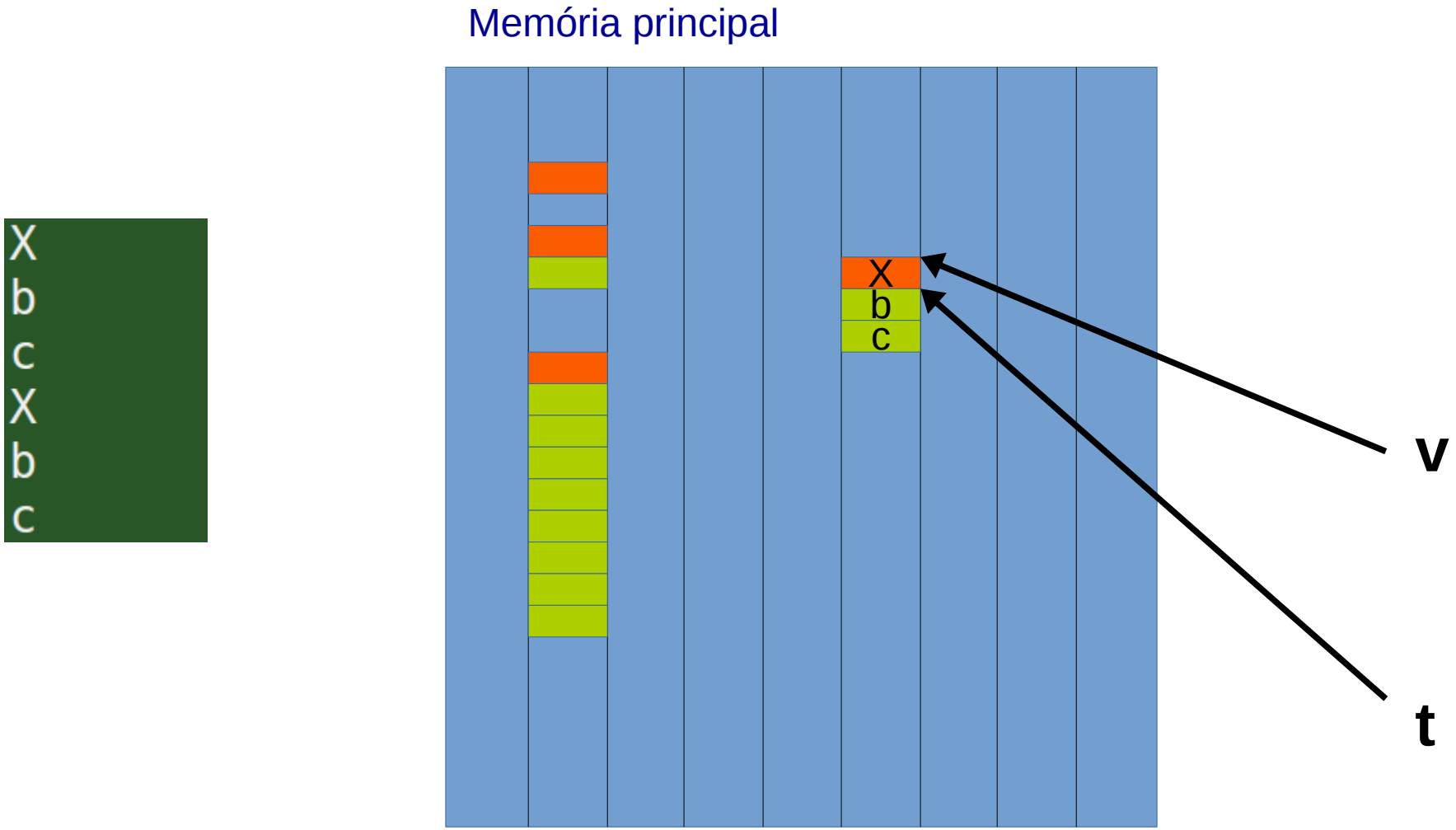
Qual seria o resultado da execução das seguintes instruções?

```
char v[] = {'a', 'b', 'c'};
char t[];

t = v;

t[0] = 'X';

for (int i=0; i<v.length; i++)
    System.out.println( v[i] );

for (int i=0; i<t.length; i++)
    System.out.println( t[i] );
```

Exemplo: vetores de caracteres



Exemplo: vetores de caracteres

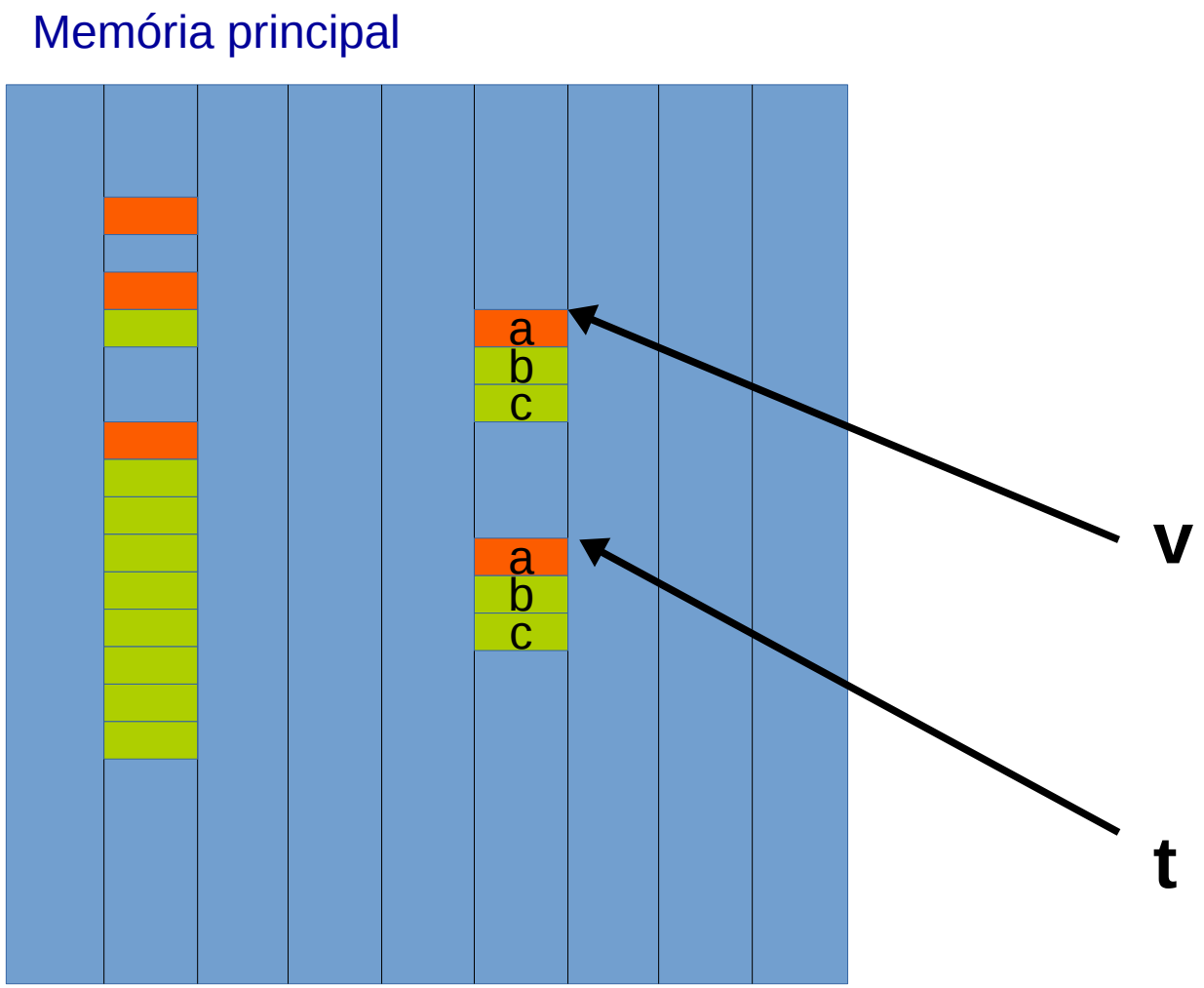


Exemplo: Comparando vetores

Qual seria o resultado da execução das seguintes instruções?

```
char v[] = {'a', 'b', 'c'};
char t[] = {'a', 'b', 'c'};

if (v==t)
    System.out.println( "vetores iguais" );
else
    System.out.println( "vetores diferentes" );
```

Exemplo: Comparando vetores




```
vetores diferentes
```

Exemplo: Comparando vetores

```
int i, cont=0;

char v[] = {'a', 'b', 'c'};
char t[] = {'a', 'b', 'c'};

for (i=0; i<v.length; i++)
    if (v[i]==t[i])
        cont = cont+1;

if (cont==v.length)
    System.out.println( "vetores iguais" );
else
    System.out.println( "vetores diferentes" );
```

```
vetores iguais
```

Vetor de strings

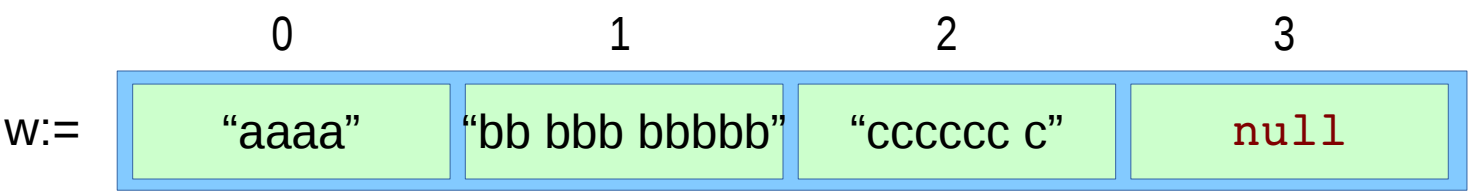
Vetor de strings

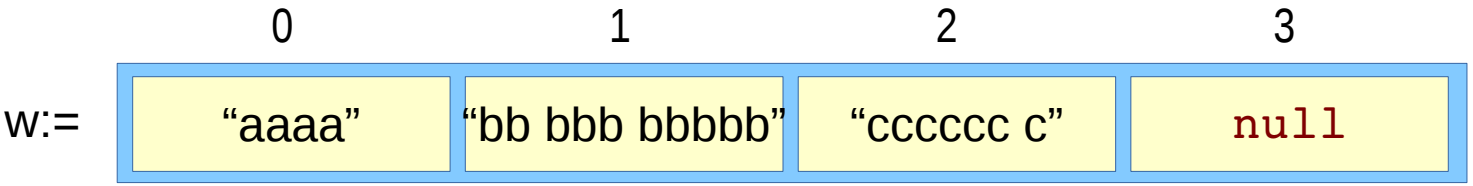
```
String [] w = new String[6];

w[0] = "aaaa";
w[1] = "bb bbb bbbbb";
w[2] = "cccccc c";

for (int i=0; i<w.length; i++)
    System.out.println( w[i] );
```

```
aaaa
bb bbb bbbbb
cccccc c
null
null
null
```

Vetor de strings

```
w.length → 4
```

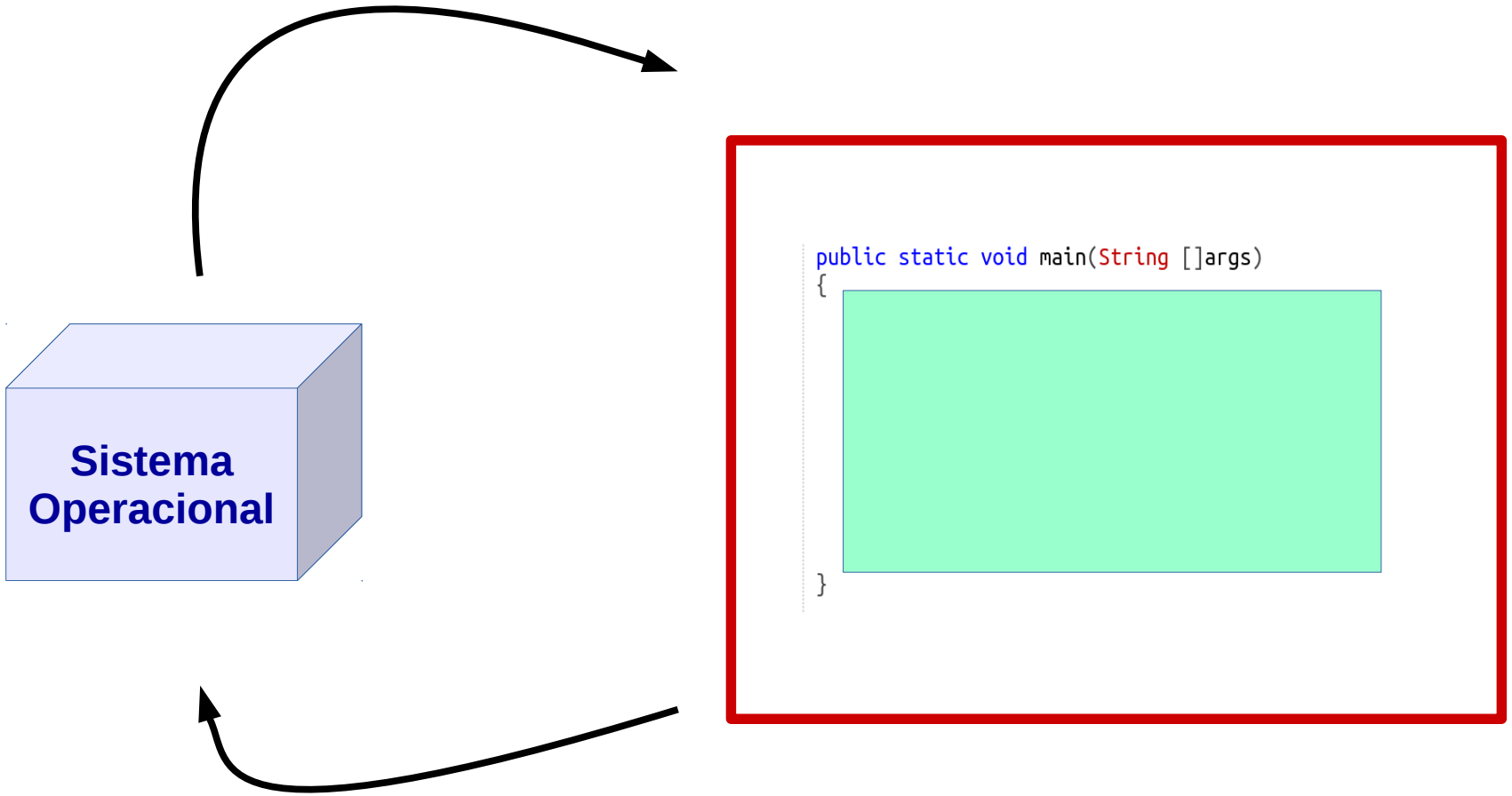




Sobre o método/função main

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

Sobre o método/função main



Sobre o método/função main

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

Sobre o método/função main

```
sh-4.3# javac Teste.java
sh-4.3# java Teste parametro1 parametro2 parametro3    444  555
parametro1
parametro2
parametro3
444
555
```

Exercícios

Exercício 01: Interseção de vetores

Crie um método que permita **verificar** se existe interseção de elementos em dois vetores de caracteres.

Assinatura:

```
static boolean temIntersecao( char u[], char v[] )
```

Exemplos:

```
u = {'m','n','o'}
v = {'s','o','r','o'}
```
Resposta = **true**

```
u = {'m','n','o'}
v = {'s','p','p','r'}
```
Resposta = **false**

Exercício 01: Interseção de vetores

```
static boolean temIntersecao( char u[], char v[] ) {
    int i, j;

    for (i=0; i<u.length; i=i+1) {
        for (j=0; j<v.length; j=j+1) {
            if (u[i]==v[j]) {
                return true;
            }
        }
    }
    return false;
}
```

Exercício 01: Interseção de vetores

```
static boolean temIntersecao2( char u[], char v[] ) {
    int i, j;
    boolean resposta = false;

    for (i=0; i<u.length && !resposta; i=i+1) {
        for (j=0; j<v.length && !resposta; j=j+1) {
            if (u[i]==v[j]) {
                resposta= true;
            }
        }
    }
    return resposta;
}
```

Exercício 02: União de vetores

Crie um método que permita **unir/concatenar** dois vetores de caracteres.

Assinatura:

```
static char[] uniao( char u[], char v[] )
```

Exemplos:

```
u = {'m','n','o'}
v = {'s','o','r','o'}
```
Resposta = **{'m','n','o','s','o','r','o'}**

```
u = {'m','n','o'}
v = {'s','p','p','r'}
```
Resposta = **{'m','n','o','s','p','p','r'}**

Exercício 02: União de vetores

```
static char[] uniao2( char u[], char v[] ) {
    int i;
    char w[] = new char[u.length + v.length];

    for (i=0; i<w.length; i=i+1) {
        if (i<u.length) {
            w[i] = u[i];
        }
        else {
            w[i] = v[i-u.length];
        }
    }
    return w;
}
```

Exercício 02: União de vetores

```
static char[] uniao( char u[], char v[] ) {
    int i;
    char w[] = new char[u.length + v.length];

    for (i=0; i<u.length; i=i+1) {
        w[i] = u[i];
    }

    for (i=0; i<v.length; i=i+1) {
        w[i+u.length] = v[i];
    }

    return w;
}
```