Aula 11 – Arranjos II

Norton Trevisan Roman

19 de abril de 2018

Copiando Arranjos

- Como fazemos para copiar um arranjo em outro?

Copiando Arranjos

- Como fazemos para copiar um arranjo em outro?
- Primeira tentativa:

```
public static void main(String[]
                                args) {
  int[] a1 = {0,1,2,3};
  int[] a2 = new int[4];

  a2 = a1;

  for (int val : a1)
    System.out.print(val+", ");
  System.out.println();

  for (int val : a2)
    System.out.print(val+", ");
  System.out.println();
}
```

Copiando Arranjos

- Como fazemos para copiar um arranjo em outro?
- Primeira tentativa:
- Aparentemente funciona:

```
$ java AreaCasa
0, 1, 2, 3,
0, 1, 2, 3,
```

```
public static void main(String[]
                                args) {
  int[] a1 = {0,1,2,3};
  int[] a2 = new int[4];

  a2 = a1;

  for (int val : a1)
    System.out.print(val+", ");
  System.out.println();

  for (int val : a2)
    System.out.print(val+", ");
  System.out.println();
}
```

Copiando Arranjos

- E se fizermos:

```
public static void main(String[]
                                args) {
  int[] a1 = {0,1,2,3};
  int[] a2 = new int[4];

  a2 = a1;
  a1[3] = 9;

  for (int val : a1)
    System.out.print(val+", ");
  System.out.println();

  for (int val : a2)
    System.out.print(val+", ");
  System.out.println();
}
```

Copiando Arranjos

- E se fizermos:
- Teremos

```
$ java AreaCasa
0, 1, 2, 9,
0, 1, 2, 9,
```

```
public static void main(String[]
                                args) {
  int[] a1 = {0,1,2,3};
  int[] a2 = new int[4];

  a2 = a1;
  a1[3] = 9;

  for (int val : a1)
    System.out.print(val+", ");
  System.out.println();

  for (int val : a2)
    System.out.print(val+", ");
  System.out.println();
}
```

Copiando Arranjos

- E se fizermos:
- Teremos

```
$ java AreaCasa
0, 1, 2, 9,
0, 1, 2, 9,
```

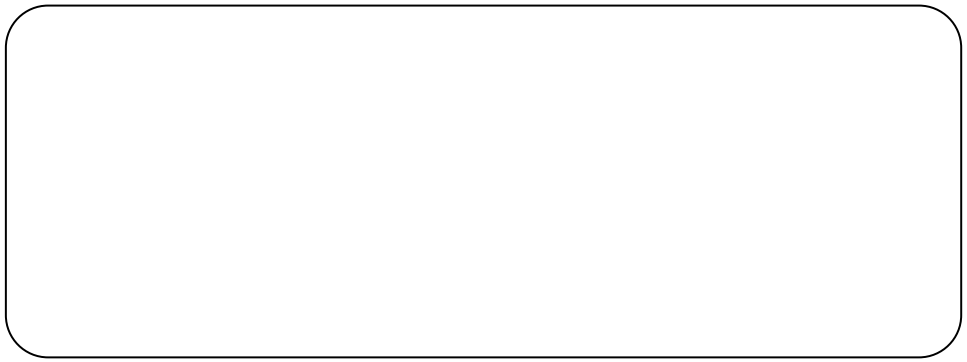
- O que houve?
  Mudamos também a2

```
public static void main(String[]
                                args) {
  int[] a1 = {0,1,2,3};
  int[] a2 = new int[4];

  a2 = a1;
  a1[3] = 9;

  for (int val : a1)
    System.out.print(val+", ");
  System.out.println();

  for (int val : a2)
    System.out.print(val+", ");
  System.out.println();
}
```

Copiando Arranjos

- Voltemos à memória

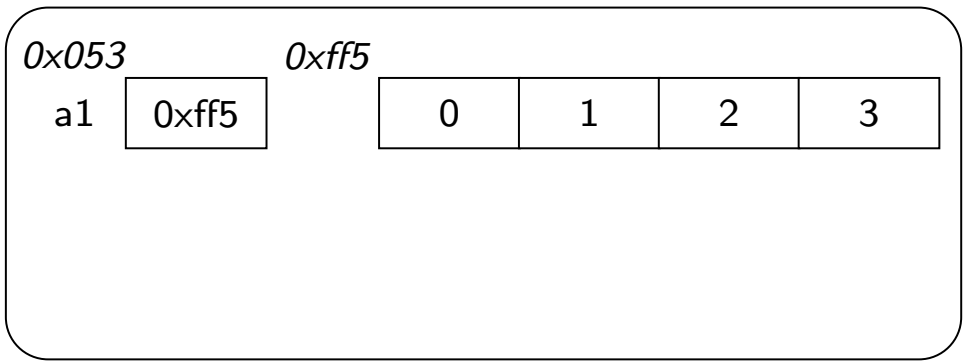
```
public static void main(String[]
                                args) {
  int[] a1 = {0,1,2,3};
  int[] a2 = new int[4];

  a2 = a1;
  a1[3] = 9;
  ...
}
```



Copiando Arranjos

- Voltemos à memória
- Quando a1 é declarado, espaço é alocado e seus valores inicializados

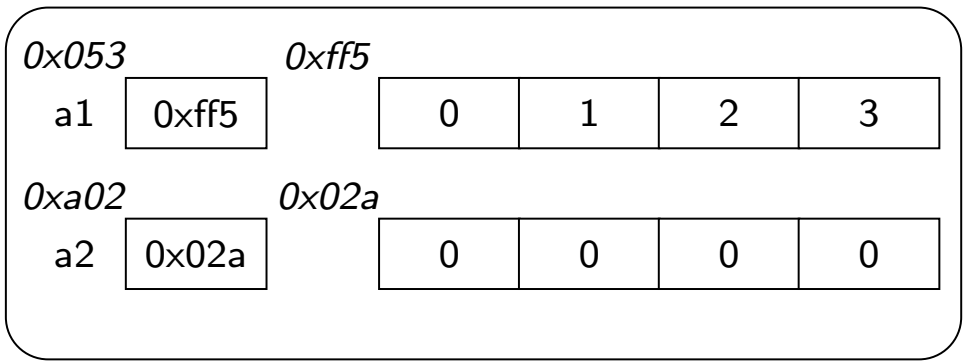
```
public static void main(String[]
                                  args) {
  int[] a1 = {0,1,2,3};
  int[] a2 = new int[4];

  a2 = a1;
  a1[3] = 9;
  ...
}
```



Copiando Arranjos

- Voltemos à memória
- Quando a1 é declarado, espaço é alocado e seus valores inicializados
- De forma semelhante, quando *a2* é declarado, espaço é alocado e seus valores inicializados com zero

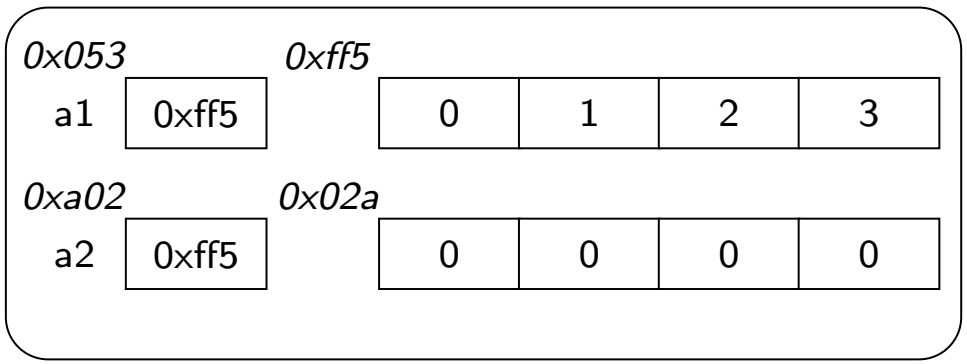
```
public static void main(String[]
                                args) {
  int[] a1 = {0,1,2,3};
  int[] a2 = new int[4];

  a2 = a1;
  a1[3] = 9;
  ...
}
```



Copiando Arranjos

- Ao fazermos a2 = a1, copiamos o conteúdo de a1 para dentro de a2

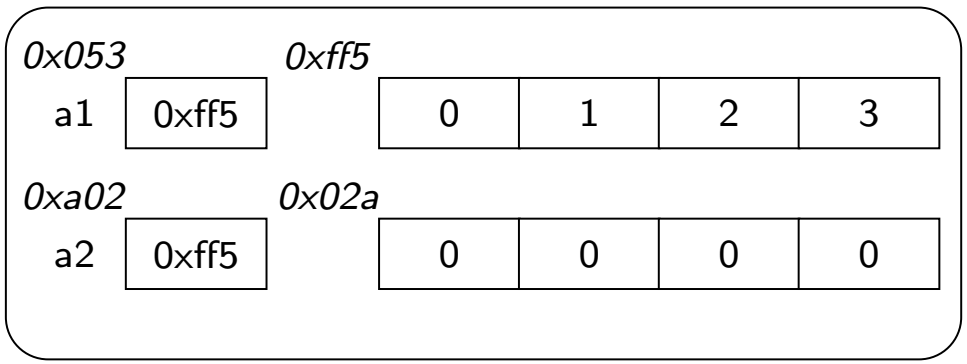
```
public static void main(String[]
                                args) {
  int[] a1 = {0,1,2,3};
  int[] a2 = new int[4];

  a2 = a1;
  a1[3] = 9;
  ...
}
```



Copiando Arranjos

- Ao fazermos a2 = a1, copiamos o conteúdo de a1 para dentro de a2
- Copiamos o endereço (referência) do arranjo

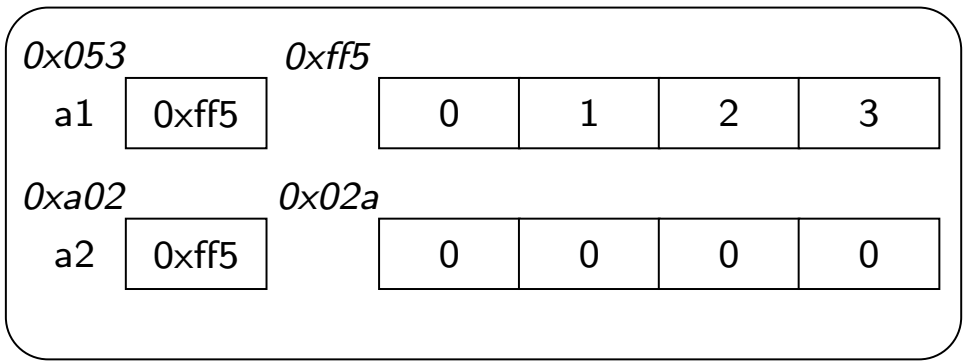
```
public static void main(String[]
                                args) {
  int[] a1 = {0,1,2,3};
  int[] a2 = new int[4];

  a2 = a1;
  a1[3] = 9;
  ...
}
```



Copiando Arranjos

- Ao fazermos a2 = a1, copiamos o conteúdo de a1 para dentro de a2
- Copiamos o endereço (referência) do arranjo
- Perdemos a referência ao arranjo 0x02a

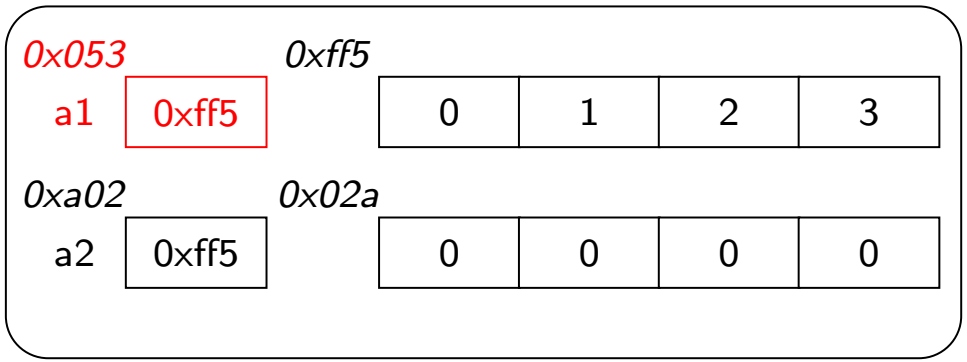
```
public static void main(String[]
                                args) {
  int[] a1 = {0,1,2,3};
  int[] a2 = new int[4];

  a2 = a1;
  a1[3] = 9;
  ...
}
```



Copiando Arranjos

- Ao fazermos `a1[3] = 9`, vamos ao endereço de memória de a1 e lemos seu conteúdo – endereço do arranjo

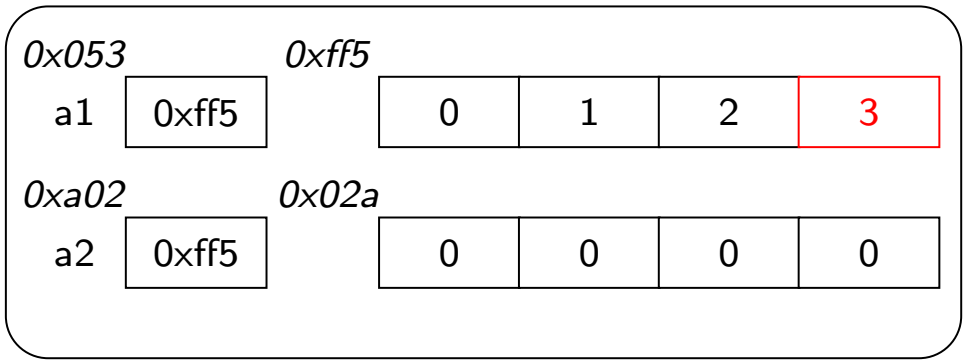
```
public static void main(String[]
                                  args) {
  int[] a1 = {0,1,2,3};
  int[] a2 = new int[4];

  a2 = a1;
  a1[3] = 9;
  ...
}
```



Copiando Arranjos

- Ao fazermos `a1[3] = 9`, vamos ao endereço de memória de `a1` e lemos seu conteúdo – endereço do arranjo
- Vamos ao endereço correspondente a $\texttt{0xff5} + 3 \times 4$
  - Quarto elemento do arranjo (lembre que o *int* é 4B)

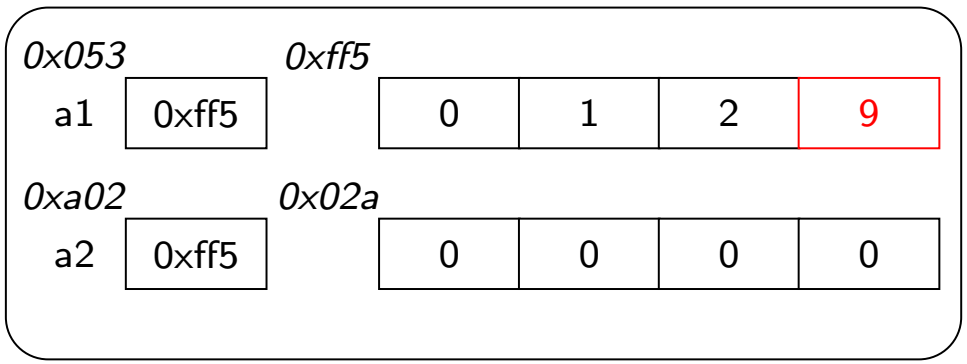
```
public static void main(String[]
                                 args) {
  int[] a1 = {0,1,2,3};
  int[] a2 = new int[4];

  a2 = a1;
  a1[3] = 9;
  ...
}
```



Copiando Arranjos

- Modificamos o valor que lá estava

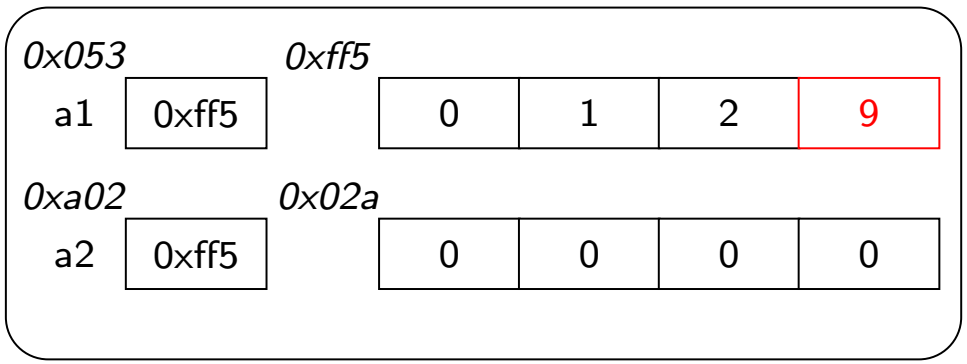
```
public static void main(String[]
                                args) {
  int[] a1 = {0,1,2,3};
  int[] a2 = new int[4];

  a2 = a1;
  a1[3] = 9;
  ...
}
```



Copiando Arranjos

- Modificamos o valor que lá estava
- Como a2 também referencia esse mesmo arranjo, parece que o mudamos também

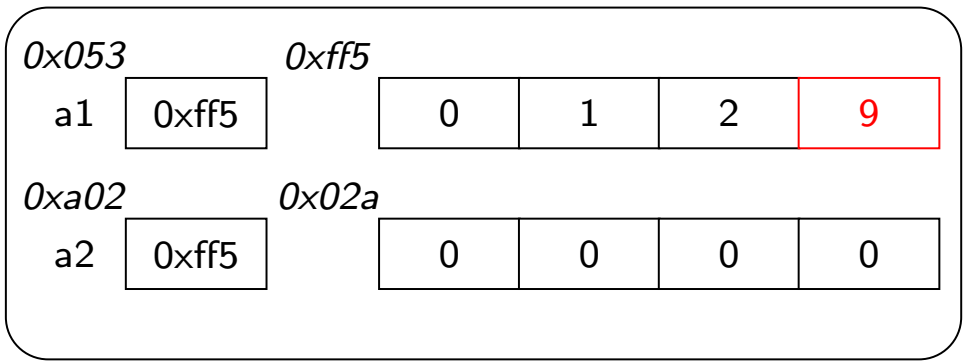
```
public static void main(String[]
                                args) {
  int[] a1 = {0,1,2,3};
  int[] a2 = new int[4];

  a2 = a1;
  a1[3] = 9;
  ...
}
```



Copiando Arranjos

- Na verdade, fizemos tanto a1 quanto a2 referenciarem o mesmo arranjo na memória

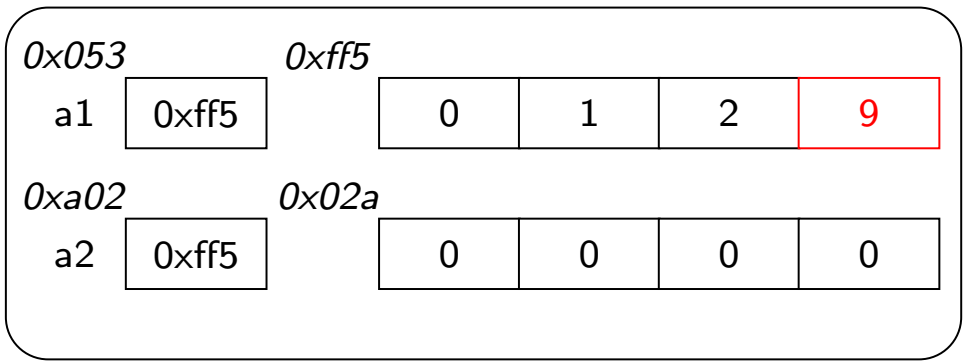
```
public static void main(String[]
                                args) {
  int[] a1 = {0,1,2,3};
  int[] a2 = new int[4];

  a2 = a1;
  a1[3] = 9;
  ...
}
```



Copiando Arranjos

- Na verdade, fizemos tanto a1 quanto a2 referenciarem o mesmo arranjo na memória
- Perdendo o originalmente referenciado por a2

```
public static void main(String[]
                                args) {
  int[] a1 = {0,1,2,3};
  int[] a2 = new int[4];

  a2 = a1;
  a1[3] = 9;
  ...
}
```



Copiando Arranjos

- Fazer `a2 = a1` não dá muito certo
- Que fazer?

Copiando Arranjos

- Fazer a2 = a1 não dá muito certo
- Que fazer? Copiar termo a termo os valores do arranjo correspondente a a1 para o referenciado por a2

```
public static void main(String[]
                                  args) {
  int[] a1 = {0,1,2,3};
  int[] a2 = new int[4];

  for (int i=0; i<a1.length; i++)
    a2[i] = a1[i];
  a1[3] = 9;

  for (int val : a1)
    System.out.print(val+", ");
  System.out.println();

  for (int val : a2)
    System.out.print(val+", ");
  System.out.println();
}
```

Copiando Arranjos

- Note que tivemos que correr o arranjo via seu índice

```
public static void main(String[]
                                args) {
  int[] a1 = {0,1,2,3};
  int[] a2 = new int[4];

  for (int i=0; i<a1.length; i++)
    a2[i] = a1[i];
  a1[3] = 9;

  for (int val : a1)
    System.out.print(val+", ");
  System.out.println();

  for (int val : a2)
    System.out.print(val+", ");
  System.out.println();
}
```

Copiando Arranjos

- Note que tivemos que correr o arranjo via seu índice
- E a saída será:

```
$ java AreaCasa
0, 1, 2, 9,
0, 1, 2, 3,
```

```
public static void main(String[]
                                args) {
  int[] a1 = {0,1,2,3};
  int[] a2 = new int[4];

  for (int i=0; i<a1.length; i++)
    a2[i] = a1[i];
  a1[3] = 9;

  for (int val : a1)
    System.out.print(val+", ");
  System.out.println();

  for (int val : a2)
    System.out.print(val+", ");
  System.out.println();
}
```

Arranjos

- Considere agora nosso código para calcular o preço médio dos materiais da piscina

```
public static void main(
                    String[] args) {
  double media = 0;

  for (double valor : precos) {
    media += valor;
  }
  media = media/precos.length;
}
```

Arranjos

- Considere agora nosso código para calcular o preço médio dos materiais da piscina
- Como podemos generalizá-lo?

```
public static void main(
                    String[] args) {
  double media = 0;

  for (double valor : precos) {
    media += valor;
  }
  media = media/precos.length;
}
```

Arranjos

- Considere agora nosso código para calcular o preço médio dos materiais da piscina
- Como podemos generalizá-lo?

```
public static void main(
                    String[] args) {
  double media = 0;

  for (double valor : precos) {
    media += valor;
  }
  media = media/precos.length;
}
```

- Criando um método que calcule a média dos elementos de um arranjo genérico

Arranjos

- Como?

Arranjos

- Como?

```
static double media(double[]
                          arranjo) {
  double resp = 0;

  for (double valor : arranjo) {
    resp += valor;
  }
  return(resp/arranjo.length);
}
```

Arranjos

- Como?
- Arranjos podem ser passados como parâmetro também

```
static double media(double[]
                          arranjo) {
  double resp = 0;

  for (double valor : arranjo) {
    resp += valor;
  }
  return(resp/arranjo.length);
}
```

Arranjos

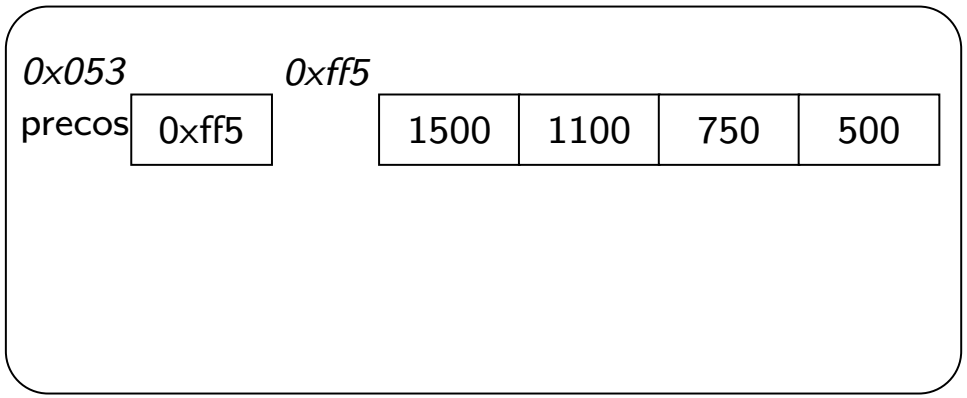
- O que acontece quando passamos um arranjo como parâmetro?

```
static double media(double[]
                          arranjo) {
  double resp = 0;
  ...
}
public static void main(String[]
                              args) {
  System.out.println(
                    media(precos));
}
```

Arranjos

- O que acontece quando passamos um arranjo como parâmetro?
- Lembre que o arranjo passado já está na memória

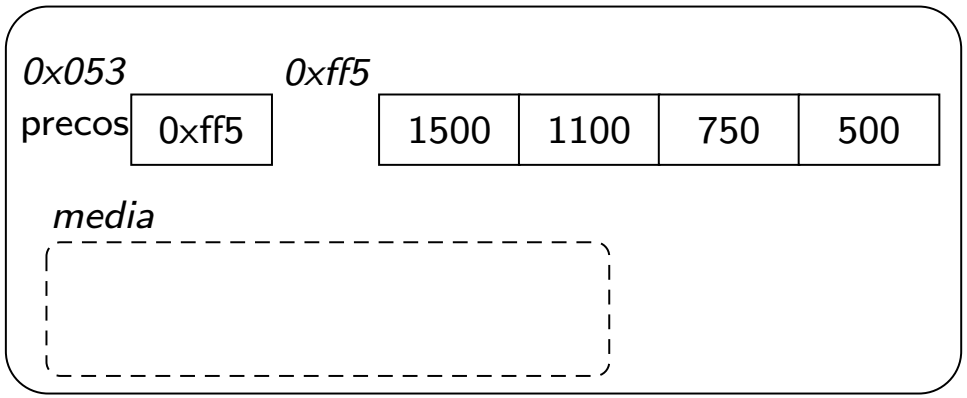
```
static double media(double[]
                          arranjo) {
  double resp = 0;
  ...
}
public static void main(String[]
                            args) {
  System.out.println(
                   media(precos));
}
```



Arranjos

- O que acontece quando passamos um arranjo como parâmetro?
- Lembre que o arranjo passado já está na memória
- O computador separa espaço para o método invocado

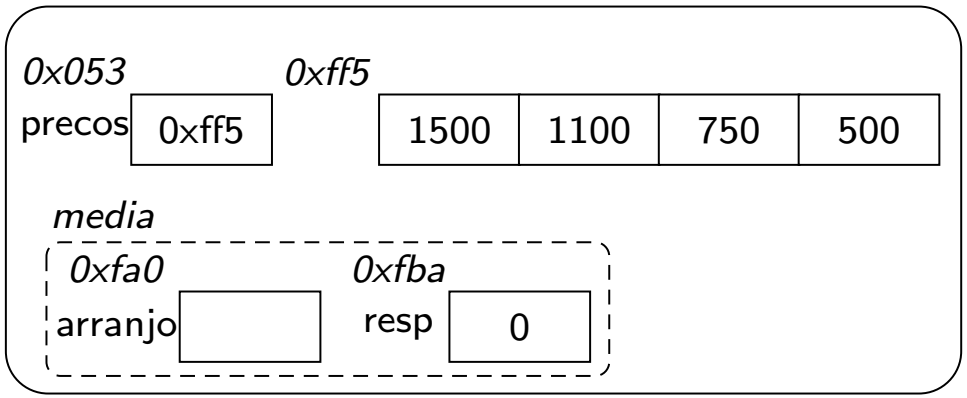
```
static double media(double[]
                          arranjo) {
  double resp = 0;
  ...
}
public static void main(String[]
                            args) {
  System.out.println(
                   media(precos));
}
```



Arranjos

- O que acontece quando passamos um arranjo como parâmetro?
- Lembre que o arranjo passado já está na memória
- O computador separa espaço para o método invocado
  - Para seu parâmetro e variável local

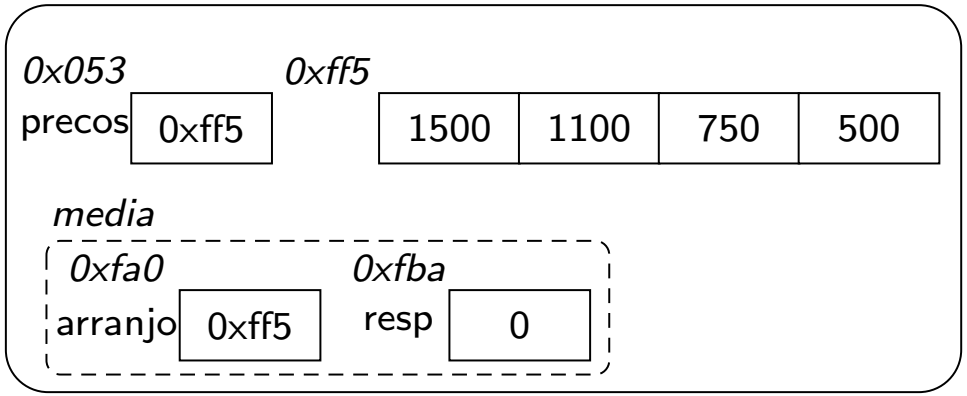
```
static double media(double[]
                          arranjo) {
  double resp = 0;
  ...
}
public static void main(String[]
                            args) {
  System.out.println(
                   media(precos));
}
```



Arranjos

- Copiando para seu parâmetro o conteúdo de precos

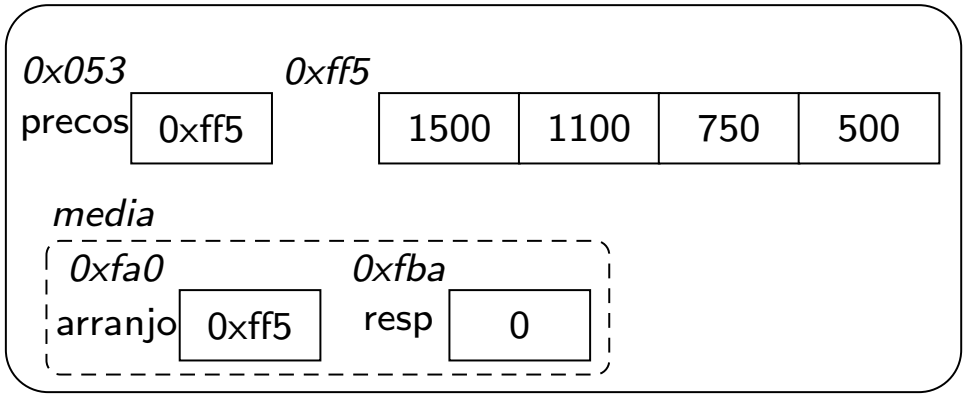
```
static double media(double[]
                          arranjo) {
  double resp = 0;
  ...
}
public static void main(String[]
                              args) {
  System.out.println(
                    media(precos));
}
```



Arranjos

- Copiando para seu parâmetro o conteúdo de `precos`
  - Ou seja, o endereço do arranjo referenciado por `precos`

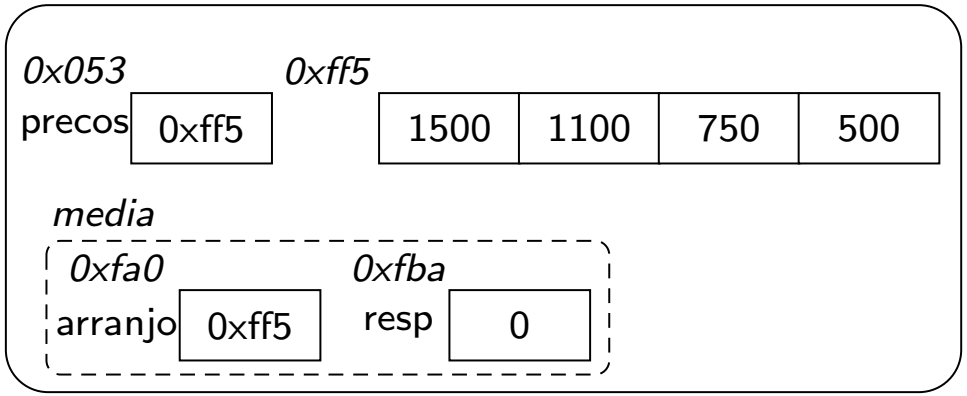
```
static double media(double[]
                          arranjo) {
  double resp = 0;
  ...
}
public static void main(String[]
                              args) {
  System.out.println(
                    media(precos));
}
```



Arranjos

- Copiando para seu parâmetro o conteúdo de `precos`
  - Ou seja, o endereço do arranjo referenciado por `precos`
- Com isso, ao modificarmos qualquer valor em `arranjo`, dentro de `media`, mudaremos `precos` também

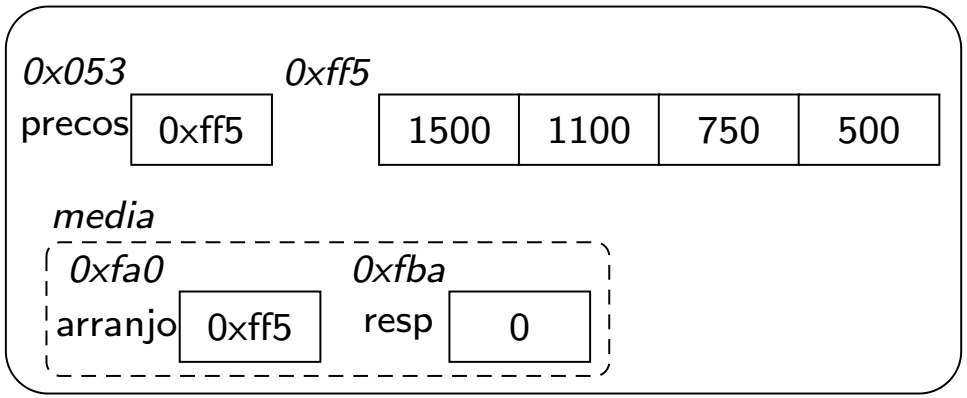
```
static double media(double[]
                          arranjo) {
  double resp = 0;
  ...
}
public static void main(String[]
                              args) {
  System.out.println(
                    media(precos));
}
```



Arranjos

- Pois tanto `arranjo` quanto `precos` referenciam a mesma região de memória

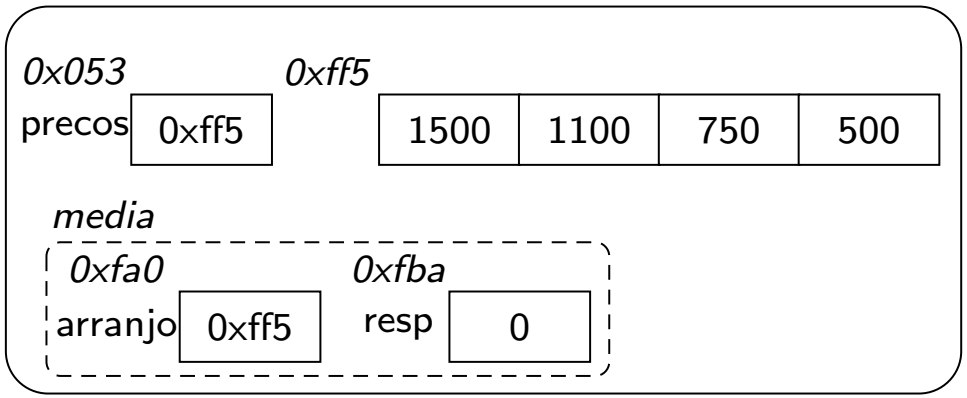
```
static double media(double[]
                          arranjo) {
  double resp = 0;
  ...
}
public static void main(String[]
                              args) {
  System.out.println(
                    media(precos));
}
```



Arranjos

- Pois tanto `arranjo` quanto `precos` referenciam a mesma região de memória
- Passagem de parâmetro por referência

```
static double media(double[]
                          arranjo) {
  double resp = 0;
  ...
}
public static void main(String[]
                              args) {
  System.out.println(
                    media(precos));
}
```



Passagem de Parâmetros

Por valor:

- O conteúdo de uma determinada região da memória é copiado para outra

Por referência:

- O conteúdo de uma determinada região da memória é copiado para outra

Passagem de Parâmetros

Por valor:

- O conteúdo de uma determinada região da memória é copiado para outra
- Esse conteúdo representa o valor para alguma variável

Por referência:

- O conteúdo de uma determinada região da memória é copiado para outra
- Esse conteúdo representa um endereço de memória $\rightarrow$ é uma referência a outra região da memória

Passagem de Parâmetros

Por valor:

- Modificações em uma das regiões não afetam a outra

Por referência:

- Modificações na região referenciada são sentidas por todas as referências àquela região

Caracteres

- Imagine que agora, em vez de guardarmos somente o tipo do material, queremos também o nome e descrição

Caracteres

- Imagine que agora, em vez de guardarmos somente o tipo do material, queremos também o nome e descrição
- Precisaríamos de frases $\rightarrow$ precisaríamos de caracteres

Caracteres

- Imagine que agora, em vez de guardarmos somente o tipo do material, queremos também o nome e descrição
- Precisaríamos de frases $\rightarrow$ precisaríamos de caracteres
- E como representamos um caractere em java?

Caracteres

- Imagine que agora, em vez de guardarmos somente o tipo do material, queremos também o nome e descrição
- Precisaríamos de frases $\rightarrow$ precisaríamos de caracteres
- E como representamos um caractere em java?
  - `char meu_caractere = 'a';`

Caracteres

- Assim como há tipos numéricos e lógicos, Java possui um tipo especial para caracteres

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

  c = 'a';

  if (c == 'a')
    System.out.println(c);
}
```

Caracteres

- Assim como há tipos numéricos e lógicos, Java possui um tipo especial para caracteres
- Valores dados a esse tipo devem estar entre aspas simples

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

  c = 'a';

  if (c == 'a')
    System.out.println(c);
}
```

Caracteres

- Assim como há tipos numéricos e lógicos, Java possui um tipo especial para caracteres
- Valores dados a esse tipo devem estar entre aspas simples
- São usados como qualquer outra variável

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

  c = 'a';

  if (c == 'a')
    System.out.println(c);
}
```

Caracteres

Valores do tipo `char` armazenam:

- Símbolos (letras, algarismos, pontuação etc.)

Caracteres

Valores do tipo `char` armazenam:

- Símbolos (letras, algarismos, pontuação etc.)
  - Cuidado! Caracteres não são números

Caracteres

Valores do tipo `char` armazenam:

- Símbolos (letras, algarismos, pontuação etc.)
  - Cuidado! Caracteres não são números
  - '2' é diferente de 2 (veremos mais adiante)

Caracteres

Valores do tipo `char` armazenam:

- Símbolos (letras, algarismos, pontuação etc.)
  - Cuidado! Caracteres não são números
  - '2' é diferente de 2 (veremos mais adiante)
- Sinais de controle (tabulação, fim de linha, fim de arquivo, etc)

Caracteres

Valores do tipo `char` armazenam:

- Símbolos (letras, algarismos, pontuação etc.)
  - Cuidado! Caracteres não são números
  - '2' é diferente de 2 (veremos mais adiante)
- Sinais de controle (tabulação, fim de linha, fim de arquivo, etc)
  - Normalmente representados por um caractere precedido de \

Caracteres

Valores do tipo `char` armazenam:

- Símbolos (letras, algarismos, pontuação etc.)
  - Cuidado! Caracteres não são números
  - '2' é diferente de 2 (veremos mais adiante)
- Sinais de controle (tabulação, fim de linha, fim de arquivo, etc)
  - Normalmente representados por um caractere precedido de \
  - Ex: `\n` `\t` `\'` e `\\`

Caracteres

Tipos primitivos do java

Tipo	*Tamanho*	*Tipo*	*Tamanho*
byte	8 bits	short	16 bits
int	32 bits	long	64 bits
float	32 bits	double	64 bits
boolean	não definido	char	16 bits

Caracteres

- O computador trabalha apenas com binário $\rightarrow$ números
- Como então consegue trabalhar com caracteres?

Caracteres

- O computador trabalha apenas com binário $\rightarrow$ números
- Como então consegue trabalhar com caracteres?
- Transformando em números, por meio de uma tabela que associe cada caractere a um número

Caracteres

- O computador trabalha apenas com binário $\rightarrow$ números
- Como então consegue trabalhar com caracteres?
- Transformando em números, por meio de uma tabela que associe cada caractere a um número
  - ASCII
  - Unicode

ASCII

- *American Standard Code for Information Interchange*
- Padrão com 128 caracteres, ou estendido com 256 caracteres
  - Cada caractere ocupa 8 bits
  - A parte estendida obedece a vários padrões
  - No Brasil, usamos a ISO-8859-1, ou Latin-1
- Bastante usada até por volta do final dos anos 80
- Formato limitado, principalmente no suporte a outros idiomas

ASCII e ISO-8859-1

Regular ASCII Chart (character codes 0 – 127)

000d	00h		(nul)	016d	10h	►	(dle)	032d	20h	␣	048d	30h	0	064d	40h	@	080d	50h	P	096d	60h	‘	112d	70h	p
001d	01h	☺	(soh)	017d	11h	◄	(dc1)	033d	21h	!	049d	31h	1	065d	41h	A	081d	51h	Q	097d	61h	a	113d	71h	q
002d	02h	☻	(stx)	018d	12h	↕	(dc2)	034d	22h	"	050d	32h	2	066d	42h	B	082d	52h	R	098d	62h	b	114d	72h	r
003d	03h	♥	(etx)	019d	13h	‼	(dc3)	035d	23h	#	051d	33h	3	067d	43h	C	083d	53h	S	099d	63h	c	115d	73h	s
004d	04h	♦	(eot)	020d	14h	¶	(dc4)	036d	24h	$	052d	34h	4	068d	44h	D	084d	54h	T	100d	64h	d	116d	74h	t
005d	05h	♣	(enq)	021d	15h	§	(nak)	037d	25h	%	053d	35h	5	069d	45h	E	085d	55h	U	101d	65h	e	117d	75h	u
006d	06h	♠	(ack)	022d	16h	▬	(syn)	038d	26h	&	054d	36h	6	070d	46h	F	086d	56h	V	102d	66h	f	118d	76h	v
007d	07h	•	(bel)	023d	17h	↨	(etb)	039d	27h	'	055d	37h	7	071d	47h	G	087d	57h	W	103d	67h	g	119d	77h	w
008d	08h	◘	(bs)	024d	18h	↑	(can)	040d	28h	(	056d	38h	8	072d	48h	H	088d	58h	X	104d	68h	h	120d	78h	x
009d	09h		(tab)	025d	19h	↓	(em)	041d	29h	)	057d	39h	9	073d	49h	I	089d	59h	Y	105d	69h	i	121d	79h	y
010d	0Ah	◙	(lf)	026d	1Ah		(eof)	042d	2Ah	*	058d	3Ah	:	074d	4Ah	J	090d	5Ah	Z	106d	6Ah	j	122d	7Ah	z
011d	0Bh	♂	(vt)	027d	1Bh	←	(esc)	043d	2Bh	+	059d	3Bh	;	075d	4Bh	K	091d	5Bh	[	107d	6Bh	k	123d	7Bh	{
012d	0Ch		(np)	028d	1Ch	└	(fs)	044d	2Ch	’	060d	3Ch	<	076d	4Ch	L	092d	5Ch	\	108d	6Ch	l	124d	7Ch	|
013d	0Dh	♪	(cr)	029d	1Dh	↔	(gs)	045d	2Dh	-	061d	3Dh	=	077d	4Dh	M	093d	5Dh	]	109d	6Dh	m	125d	7Dh	}
014d	0Eh	♬	(so)	030d	1Eh	▲	(rs)	046d	2Eh	.	062d	3Eh	>	078d	4Eh	N	094d	5Eh	^	110d	6Eh	n	126d	7Eh	~
015d	0Fh	☼	(si)	031d	1Fh	▼	(us)	047d	2Fh	/	063d	3Fh	?	079d	4Fh	O	095d	5Fh	_	111d	6Fh	o	127d	7Fh	⌂

Extended ASCII Chart (character codes 128 – 255) Latin1/CP1252

128d	80h	€	144d	90h		160d	A0h		176d	B0h	°	192d	C0h	À	208d	D0h	Đ	224d	E0h	à	240d	F0h	ð
129d	81h		145d	91h	‘	161d	A1h	¡	177d	B1h	±	193d	C1h	Á	209d	D1h	Ñ	225d	E1h	á	241d	F1h	ñ
130d	82h	‚	146d	92h	’	162d	A2h	¢	178d	B2h	²	194d	C2h	Â	210d	D2h	Ò	226d	E2h	â	242d	F2h	ò
131d	83h	*f*	147d	93h	“	163d	A3h	£	179d	B3h	³	195d	C3h	Ã	211d	D3h	Ó	227d	E3h	ã	243d	F3h	ó
132d	84h	„	148d	94h	”	164d	A4h	¤	180d	B4h	´	196d	C4h	Ä	212d	D4h	Ô	228d	E4h	ä	244d	F4h	ô
133d	85h	...	149d	95h	•	165d	A5h	¥	181d	B5h	µ	197d	C5h	Å	213d	D5h	Õ	229d	E5h	å	245d	F5h	õ
134d	86h	†	150d	96h	-	166d	A6h	¦	182d	B6h	¶	198d	C6h	Æ	214d	D6h	Ö	230d	E6h	æ	246d	F6h	ö
135d	87h	‡	151d	97h	--	167d	A7h	§	183d	B7h	·	199d	C7h	Ç	215d	D7h	×	231d	E7h	ç	247d	F7h	÷
136d	88h	ˆ	152d	98h	˜	168d	A8h	¨	184d	B8h	¸	200d	C8h	È	216d	D8h	Ø	232d	E8h	è	248d	F8h	ø
137d	89h	‰	153d	99h	™	169d	A9h	©	185d	B9h	¹	201d	C9h	É	217d	D9h	Ù	233d	E9h	é	249d	F9h	ù
138d	8Ah	Š	154d	9Ah	š	170d	AAh	ª	186d	BAh	º	202d	CAh	Ê	218d	DAh	Ú	234d	EAh	ê	250d	FAh	ú
139d	8Bh	<	155d	9Bh	>	171d	ABh	«	187d	BBh	»	203d	CBh	Ë	219d	DBh	Û	235d	EBh	ë	251d	FBh	û
140d	8Ch	Œ	156d	9Ch	œ	172d	ACh	¬	188d	BCh	¼	204d	CCh	Ì	220d	DCh	Ü	236d	ECh	ì	252d	FCh	ü
141d	8Dh		157d	9Dh		173d	ADh		189d	BDh	½	205d	CDh	Í	221d	DDh	Ý	237d	EDh	í	253d	FDh	ý
142d	8Eh	Ž	158d	9Eh	ž	174d	AEh	®	190d	BEh	¾	206d	CEh	Î	222d	DEh	Þ	238d	EEh	î	254d	FEh	þ
143d	8Fh		159d	9Fh	Ÿ	175d	AFh	¯	191d	BFh	¿	207d	CFh	Ï	223d	DFh	ß	239d	EFh	ï	255d	FFh	ÿ

Hexadecimal to Binary

0	0000	4	0100	8	1000	C	1100
1	0001	5	0101	9	1001	D	1101
2	0010	6	0110	A	1010	E	1110
3	0011	7	0111	B	1011	F	1111

Groups of ASCII-Code in Binary

Bit 6	Bit 5	Group
0	0	Control Characters
0	1	Digits and Punctuation
1	0	Upper Case and Special
1	1	Lower Case and Special

Unicode

- Movimento iniciado em 1986, discutindo-se a criação de um padrão internacional
- Consórcio Unicode fundado em 1991
- O consórcio mapeou cada caractere a um número único (*code point*), normalmente em hexadecimal, independente de plataforma, programa ou língua

Unicode

- A primeira versão do Unicode (1991 a 1995) era uma codificação de 16 bits
- A partir da Unicode 2.0, os códigos estão em um espaço de 21 bits
- Valores de U+0000 a U+007F equivalem ao ASCII
- Valores de U+00A0 a U+00FF equivalem ao ISO-8859-1

Unicode

- Existem diferentes formas para representar um unicode

Unicode

- Existem diferentes formas para representar um unicode
  - UTFs – Unicode Transformation Format

Unicode

- Existem diferentes formas para representar um unicode
  - UTFs – Unicode Transformation Format
- UTF é um mapeamento de cada ponto Unicode para uma sequência única de bytes
  - UTF-8 usa de 1 a 4 bytes
  - UTF-16 de 1 a 2 unidades de 16 bits, e
  - UTF-32 ocupa 32 bits

Caracteres

- Em java, caracteres são codificados usando UTF-16
- E como escrevemos um caractere em java?

Caracteres

- Em java, caracteres são codificados usando UTF-16
- E como escrevemos um caractere em java?

```
public static void main(
                  String[] args) {
  char c = 'ö';
  char x = '\u00F6';
  int y = 246;

  System.out.println(c);
  System.out.println(x);
  System.out.println((char)y);
}
```

Caracteres

- Podemos abastecer a variável diretamente

```
public static void main(
                    String[] args) {
    char c = 'ö';
    char x = '\u00F6';
    int y = 246;

    System.out.println(c);
    System.out.println(x);
    System.out.println((char)y);
}
```

Caracteres

- Podemos abastecer a variável diretamente
- Fornecer seu código unicode (em hexa), como caractere

```
public static void main(
                    String[] args) {
   char c = 'ö';
   char x = '\u00F6';
   int y = 246;

   System.out.println(c);
   System.out.println(x);
   System.out.println((char)y);
}
```

Caracteres

- Podemos abastecer a variável diretamente
- Fornecer seu código unicode (em hexa), como caractere
- Fornecer seu código decimal (com *cast*)

```
public static void main(
                    String[] args) {
    char c = 'ö';
    char x = '\u00F6';
    int y = 246;

    System.out.println(c);
    System.out.println(x);
    System.out.println((char)y);
}
```

Type Casting

- Vimos que trata-se da mudança de um tipo de dado em outro:
  - `int x = (int)3.23;`

    (nesse caso, `x` recebe a parte inteira de `3.23`)

Type Casting

- Vimos que trata-se da mudança de um tipo de dado em outro:
  - `int x = (int)3.23;`

    (nesse caso, `x` recebe a parte inteira de `3.23`)
- O que ocorre no caso de caracteres?

Type Casting

- Vimos que trata-se da mudança de um tipo de dado em outro:
  - `int x = (int)3.23;`

    (nesse caso, `x` recebe a parte inteira de `3.23`)
- O que ocorre no caso de caracteres?
  - Ex:

```
int y=3;
char c = (char)y;
```

Type Casting

- Uma variável `char` nada mais é que um inteiro que corresponde a um caractere unicode

Type Casting

- Uma variável `char` nada mais é que um inteiro que corresponde a um caractere unicode
- Valor padrão: '\u0000' ou '\0'

Type Casting

- Uma variável `char` nada mais é que um inteiro que corresponde a um caractere unicode
- Valor padrão: '\u0000' ou '\0'
- NÃO o caractere '0' !

Type Casting

- Uma variável `char` nada mais é que um inteiro que corresponde a um caractere unicode
- Valor padrão: '\u0000' ou '\0'
- NÃO o caractere '0'!
- Por isso '2' é diferente de 2

Type Casting

- Podemos, por exemplo, inspecionar toda a tabela ascii

```
public static void main(String[] args) {
  for (int i = 32; i <= 126; i++) {
    System.out.println(i+" : "+(char)i);
  }
}
```

Type Casting

- Podemos, por exemplo, inspecionar toda a tabela ascii

```
public static void main(String[] args) {
  for (int i = 32; i <= 126; i++) {
    System.out.println(i+" : "+(char)i);
  }
}
```

- E o que é mais estranho...

```
public static void main(String[] args) {
  for (char i = 32; i <= 126; i++) {
    System.out.println((int)i+" : "+i);
  }
}
```

Caracteres

- Sabendo da tabela, como fazer para saber se uma variável contém uma letra minúscula?

Caracteres

- Sabendo da tabela, como fazer para saber se uma variável contém uma letra minúscula?
- Note que entre 'a' e 'z' estão todas as minúsculas na tabela

Caracteres

- Sabendo da tabela, como fazer para saber se uma variável contém uma letra minúscula?
- Note que entre 'a' e 'z' estão todas as minúsculas na tabela

Então...

```
/*
  Retorna true se c for
  minúscula, false se não
*/
static boolean minuscula(
                          char c) {
  return(c >= 'a' && c <= 'z');
}
```

Caracteres

- E como traduzir de maiúscula para minúscula?

Caracteres

- E como traduzir de maiúscula para minúscula?

```
static char paraMin(char c) {
  int aux;

  if (c >= 'A' && c <= 'Z') {
    aux = c - 'A' + 'a';
    return((char)aux);
  }
  return(c);
}
```

Caracteres

- E como traduzir de maiúscula para minúscula?
- Usamos a matemática para nos poupar código

```
static char paraMin(char c) {
  int aux;

  if (c >= 'A' && c <= 'Z') {
    aux = c - 'A' + 'a';
    return((char)aux);
  }
  return(c);
}
```

Caracteres

- Nosso problema inicial, no entanto, era como representar o nome de um material

Caracteres

- Nosso problema inicial, no entanto, era como representar o nome de um material
  - Uma palavra ou frase, portanto

Caracteres

- Nosso problema inicial, no entanto, era como representar o nome de um material
  - Uma palavra ou frase, portanto
- Já sabemos como representar um caractere...

Caracteres

- Nosso problema inicial, no entanto, era como representar o nome de um material
  - Uma palavra ou frase, portanto
- Já sabemos como representar um caractere...
- Que fazer?

Caracteres

- Um arranjo de caracteres $\longrightarrow$ string

```
/* nomes dos materiais */
static char[] nAlvenaria = {'A','l',
       'v','e','n','a','r','i','a'};
static char[] nVinil = {'V','i','n',
                              'i','l'};
static char[] nFibra = {'F','i','b',
                              'r','a'};
static char[] nPlastico = {'P','l',
          'á','s','t','i','c','o'};
...
public static void main(...) {
  System.out.print("Piscina de ");
  System.out.print(nFibra);
  System.out.println(": "+
          valorPiscina(100,FIBRA));
}
```

Caracteres

- Um arranjo de caracteres $\longrightarrow$ string
- Extremamente bizarro em java

```
/* nomes dos materiais */
static char[] nAlvenaria = {'A','l',
       'v','e','n','a','r','i','a'};
static char[] nVinil = {'V','i','n',
                              'i','l'};
static char[] nFibra = {'F','i','b',
                              'r','a'};
static char[] nPlastico = {'P','l',
          'á','s','t','i','c','o'};
...
public static void main(...) {
  System.out.print("Piscina de ");
  System.out.print(nFibra);
  System.out.println(": "+
          valorPiscina(100,FIBRA));
}
```

Caracteres

- Um arranjo de caracteres $\longrightarrow$ string
- Extremamente bizarro em java
- Mais adiante veremos meios BEM melhores de representar isso

```
/* nomes dos materiais */
static char[] nAlvenaria = {'A','l',
       'v','e','n','a','r','i','a'};
static char[] nVinil = {'V','i','n',
                              'i','l'};
static char[] nFibra = {'F','i','b',
                              'r','a'};
static char[] nPlastico = {'P','l',
           'á','s','t','i','c','o'};
...
public static void main(...) {
  System.out.print("Piscina de ");
  System.out.print(nFibra);
  System.out.println(": "+
           valorPiscina(100,FIBRA));
}
```

Caracteres

- Como em arranjos, podemos acessar os caracteres individuais de um *string*:

```
public static void main(
                    String[] args) {
  System.out.println(
                          nVinil[1]);
}
```

Caracteres

- Como em arranjos, podemos acessar os caracteres individuais de um *string*:

```
public static void main(
                    String[] args) {
  System.out.println(
                         nVinil[1]);
}
```

- Ou então modificar algum dos caracteres

```
public static void main(
                    String[] args) {
  nVinil[1] = 'c';
  System.out.println(nVinil);
}
```

Referências Adicionais

- http://www.unicode.org/
- http://blog.caelum.com.br/entendendo-unicode-e-os-character-encodings/
- http://www.mobilefish.com/tutorials/java/java_quickguide_char.html

Videoaula

https://www.youtube.com/watch?v=H6kWSniXHtoe

https://www.youtube.com/watch?v=8pjm-vDrcXE