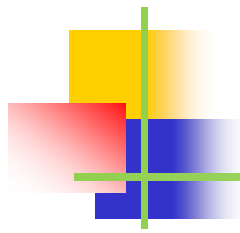


- Strings (sequências de caracteres)
- Código ASCII
- Classe `Character`
- Operações com caracteres
- Propriedades das Strings
- Leitura e escrita
- Classe `String`
- Strings como argumentos de funções
- Arrays de Strings



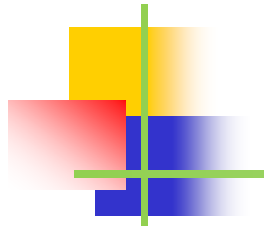
Strings (sequências de caracteres)



deti

universidade de aveiro
departamento de electrónica,
telecomunicações e informática

- Existem aplicações informáticas que, para além de necessitarem de processar dados numéricos, também necessitam de processar texto.
- Uma sequência de caracteres não é simplesmente uma sequência capaz de armazenar caracteres pois estes têm particularidades e necessitam de um conjunto de operações específicas para a sua manipulação.
- Em JAVA existe o tipo de dados referência `String` para a manipulação de texto.
- Este tipo de dados é promovido pela classe `String` que disponibiliza um vasto conjunto de funções para a sua manipulação.
- A classe `Character` tem também um papel importante...



Código ASCII

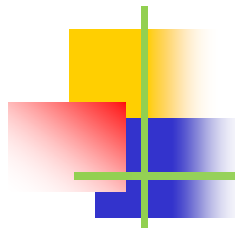
010 0001	041	33	21	!
010 0010	042	34	22	"
010 0011	043	35	23	#
010 0100	044	36	24	\$
010 0101	045	37	25	%
010 0110	046	38	26	&
010 0111	047	39	27	'
010 1000	050	40	28	(
010 1001	051	41	29)
010 1010	052	42	2A	*
010 1011	053	43	2B	+
010 1100	054	44	2C	,
010 1101	055	45	2D	-
010 1110	056	46	2E	.
010 1111	057	47	2F	/
011 0000	060	48	30	0
011 0001	061	49	31	1
011 0010	062	50	32	2
011 0011	063	51	33	3
011 0100	064	52	34	4
011 0101	065	53	35	5
011 0110	066	54	36	6
011 0111	067	55	37	7
011 1000	070	56	38	8
011 1001	071	57	39	9
011 1010	072	58	3A	:

100 0001	101	65	41	A
100 0010	102	66	42	B
100 0011	103	67	43	C
100 0100	104	68	44	D
100 0101	105	69	45	E
100 0110	106	70	46	F
100 0111	107	71	47	G
100 1000	110	72	48	H
100 1001	111	73	49	I
100 1010	112	74	4A	J
100 1011	113	75	4B	K
100 1100	114	76	4C	L
100 1101	115	77	4D	M
100 1110	116	78	4E	N
100 1111	117	79	4F	O
101 0000	120	80	50	P
101 0001	121	81	51	Q
101 0010	122	82	52	R
101 0011	123	83	53	S
101 0100	124	84	54	T
101 0101	125	85	55	U
101 0110	126	86	56	V
101 0111	127	87	57	W
101 1000	130	88	58	X
101 1001	131	89	59	Y
101 1010	132	90	5A	Z

110 0001	141	97	61	a
110 0010	142	98	62	b
110 0011	143	99	63	c
110 0100	144	100	64	d
110 0101	145	101	65	e
110 0110	146	102	66	f
110 0111	147	103	67	g
110 1000	150	104	68	h
110 1001	151	105	69	i
110 1010	152	106	6A	j
110 1011	153	107	6B	k
110 1100	154	108	6C	l
110 1101	155	109	6D	m
110 1110	156	110	6E	n
110 1111	157	111	6F	o
111 0000	160	112	70	p
111 0001	161	113	71	q
111 0010	162	114	72	r
111 0011	163	115	73	s
111 0100	164	116	74	t
111 0101	165	117	75	u
111 0110	166	118	76	v
111 0111	167	119	77	w
111 1000	170	120	78	x
111 1001	171	121	79	y
111 1010	172	122	7A	z



universidade de aveiro
departamento de electrónica,
telecomunicações e informática



Classe Character

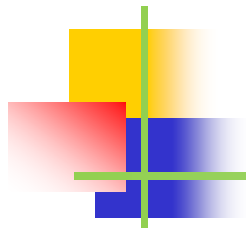


deti

universidade de aveiro
departamento de electrónica,
telecomunicações e informática

- A classe `Character` contém um conjunto de funções para processamento de caracteres.
- As funções disponibilizadas dividem-se, funcionalmente, em dois grupos:
 - funções de teste de caracteres que devolvem um valor booleano se o argumento pertence ao “grupo” associado:
 - `isLetter`, `isDigit`, `isLetterOrDigit`,
`isSpace`, `isLowerCase`, `isUpperCase`, ...
 - funções de conversão que devolvem outro caracter:
 - `toLowerCase`, `toUpperCase`, ...
- Estas funções utilizam-se tais como as da classe `Math`:

```
Character.nomeDaFuncao( ... )
```



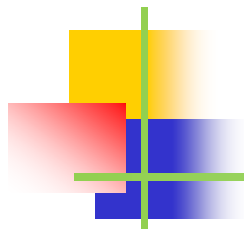
Exemplo



deti

universidade de aveiro
departamento de electrónica,
telecomunicações e informática

```
// Leitura de caracteres até aparecer o '.'  
char c;  
do{  
    System.out.print("Insira uma letra: ");  
    c = sc.nextLine.charAt(0); // leitura de um char  
    if(Character.isLetter(c))  
        System.out.println("Inseriu uma letra");  
    else if(Character.isDigit(c))  
        System.out.println("Inseriu um digito");  
    else  
        System.out.println("Não inseriu uma letra ou digito");  
}while(c != '.');
```



Operações com caracteres

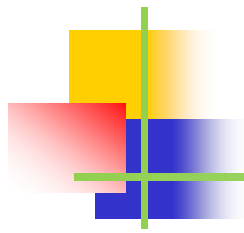


deti

universidade de aveiro
departamento de electrónica,
telecomunicações e informática

- Para transformar um caracter noutro caracter temos que recorrer ao código ASCII.
- Exemplo do deslocamento de caracteres 3 posições para a frente:

```
if(Character.isLowerCase(letra)) {  
    pos = (int)(letra - 'a'); // posição relativa de letra  
    novaPos = (pos + 3) % 26; // deslocamento circular  
    novaLetra = (char)('a' + novaPos); // nova letra...  
}  
  
else if(Character.isUpperCase(letra)) {  
    pos = (int)(letra - 'A');  
    novaPos = (pos + 3) % 26;  
    novaLetra = (char)('A' + novaPos);  
} ...
```



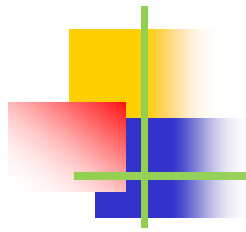
Propriedades das Strings



deti

universidade de aveiro
departamento de electrónica,
telecomunicações e informática

- Em JAVA a sequência de caracteres é um tipo de dados referência com propriedades limitadas ao nível da alteração do seu conteúdo.
- O maior problema na gestão das sequências de caracteres tem a ver com o facto de cada uma ter um número diferente de caracteres.
- A dimensão e conteúdo de uma sequências de caracteres fica definida quando esta é criada, não sendo possível mais tarde modificar o seu conteúdo (é imutável).
- Na passagem como argumento a funções, apesar de ser um tipo de referência, o seu conteúdo não pode ser modificado (veremos mais à frente...).



Declaração de variáveis String



deti

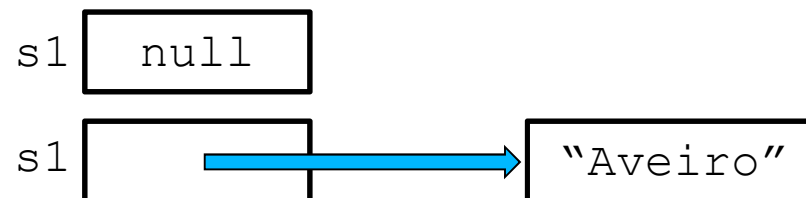
universidade de aveiro
departamento de electrónica,
telecomunicações e informática

- A declaração de variáveis do tipo `String` obedece às mesmas regras de declaração de tipos referência.

- Exemplos:

```
String s1;
```

```
s1 = new String("Aveiro"); //String com texto Aveiro
```



```
String s2;
```

```
s2 = new String(); // String nula
```

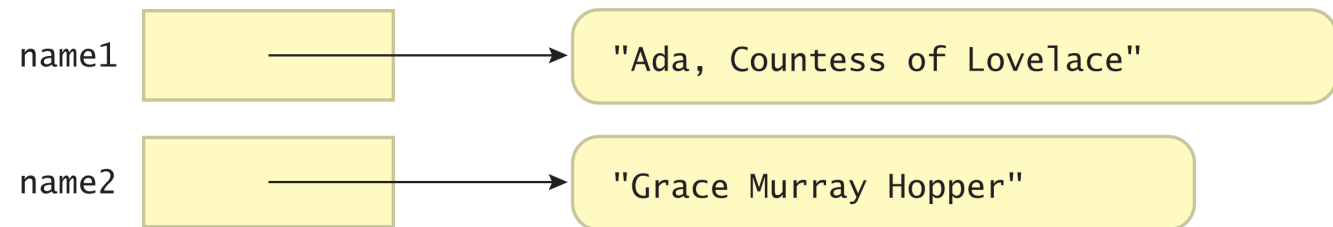
- O operador de atribuição `'='` também é capaz de reservar o espaço em memória e atualizar a referência:

```
String s3 = "Aveiro"; // Declaração simplificada
```

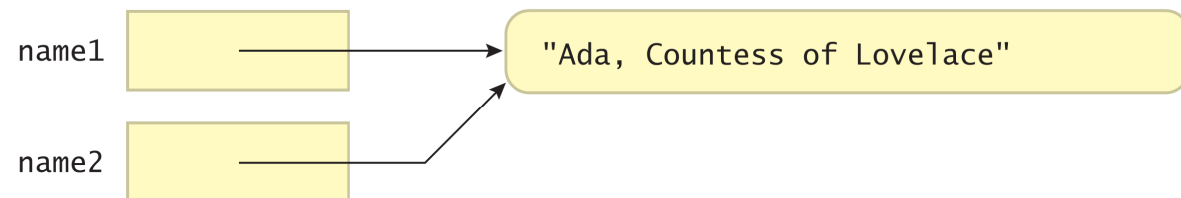

String, tipo referência

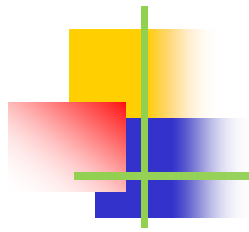
```
String name1 = "Ada, Countess of Lovelace";
```

```
String name2 = "Grace Murray Hopper";
```



```
name2 = name1;
```





Leitura e escrita de Strings



deti

universidade de aveiro
departamento de electrónica,
telecomunicações e informática

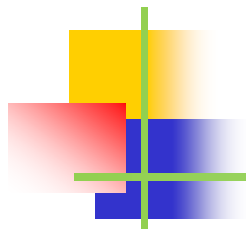
- Uma String pode ser lida do teclado através da função `nextLine()` do `Scanner`. Esta função lê todos os caracteres introduzidos pelo utilizador até encontrar o `'\n'`.
- Para imprimir no terminal o conteúdo de uma String, podemos utilizar qualquer uma das funções `System.out.print()`, `println()` e `printf()`.
- No `printf` utiliza-se o especificador de conversão `%s` para escrever uma String. Este pode ser precedido de um número com o qual se controla o formato (`%10s` `%-10s`).

```
String s = new String();
```

```
s = sc.nextLine();
```

```
System.out.printf("O texto lido foi %s\n", s);
```

```
System.out.println("O texto lido foi " + s);
```



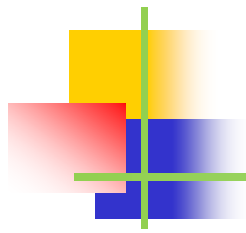
Classe String



deti

universidade de aveiro
departamento de electrónica,
telecomunicações e informática

- A classe `String` disponibiliza um vasto conjunto de funções que podemos separar em dois tipos:
 - funções que se aplicam sobre variáveis do tipo `String`:
`variavel.nomeDaFuncao()`;
`char charAt(int)` – devolve o caracter numa determinada posição
`int length()` – devolve a dimensão de uma `String`
`int indexOf(char)` - pesquisa a primeira ocorrência do caracter
`boolean equals(String)` – verifica se duas `Strings` são iguais
`boolean compareTo(String)` – compara duas `Strings`
 - funções que se aplicam sem a necessidade de ter uma variável do tipo `String`: `String.nomeDaFuncao()`.
- <http://download.oracle.com/javase/6/docs/api/java/lang/String.html>



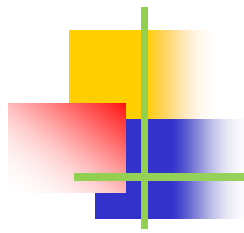
Exemplo



deti

universidade de aveiro
departamento de electrónica,
telecomunicações e informática

```
// Escrita dos caracteres de uma String
String frase = new String();
char letra;
int i;
System.out.print("Escreva uma frase: ");
frase = sc.nextLine();
System.out.printf("A frase tem as letras:\n");
for(i = 0 ; i < frase.length() ; i++)
{
    letra = frase.charAt(i);
    System.out.println(letra);
}
```



Passagem de Strings a funções



deti

universidade de aveiro
departamento de electrónica,
telecomunicações e informática

- Na passagem de Strings como argumento de funções, apesar de ser um tipo de referência o seu conteúdo não pode ser modificado, dado que são objetos imutáveis.
- Isto quer dizer que, quando atribuímos um novo valor a uma `String`, o seu endereço na memória do computador muda.

```
String frase = new String("Aveiro");  
f(frase); // argumento da função passa a referenciar frase  
System.out.printf("%s\n", frase); //imprime "Aveiro"  
}  
  
public static void f(String s){  
    s = "ola"; // s passa a referenciar algo diferente...  
    System.out.printf("%s\n", s);  
}
```

Arrays de Strings



deti

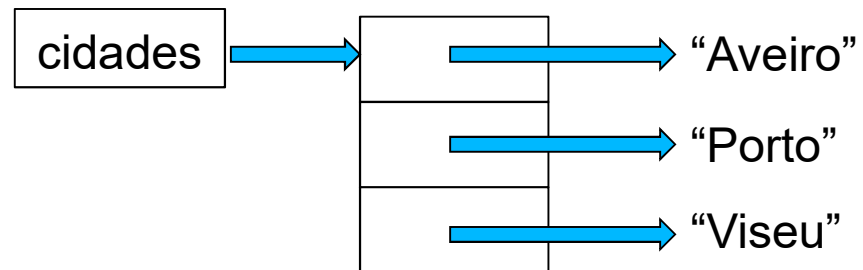
universidade de aveiro
departamento de electrónica,
telecomunicações e informática

- É então possível criar uma sequência de Strings, ou seja, uma estrutura bidimensional de caracteres.
- A declaração de uma sequência de Strings cria um array de referências nulas para String que depois serão preenchidas por instruções de atribuição...

```
String cidades[];  
cidades = new String[3];  
cidades[0] = "Aveiro";  
cidades[1] = "Porto";  
cidades[2] = "Viseu";
```

// ou

```
String cidades[] = {"Aveiro", "Porto", "Viseu"};
```



Exemplo

```
// ler frases até aparecer a palavra fim
public static int lerFrases(String frases[]){
    String s = new String(); int n = 0;
    do{
        System.out.print("Frase: "); s = sc.nextLine();
        if(!s.equalsIgnoreCase("fim")){
            frases[n] = s; n++;
        }
    }while(!s.equalsIgnoreCase("fim") && n < frases.length);
    return n;
}

public static void imprimirFrases(String frases[], int n){
    for(int i = 0 ; i < n ; i++){
        System.out.printf("[%d] -> %s\n", i, frases[i]);
    }
}
```

Strings – funções (1)



deti

universidade de aveiro
departamento de electrónica,
telecomunicações e informática

```
int x;  
String s = new String;  
String[] t;  
System.out.printf("ler inteiro: ");  
x = ler.nextInt();  
System.out.printf("Ler frase: ");  
do {  
    s = ler.nextLine();  
} while (s.length() == 0);  
System.out.printf("%d %s\n", x, s);  
System.out.printf("converte '231' para inteiro = %d\n",  
    Integer.parseInt("231"));  
System.out.printf("primeira posicao de 'av': %d\n", s.indexOf("av"));  
System.out.printf("última posicao de 'av': %d\n", s.lastIndexOf("av"))
```

`nextInt()` só lê dígitos, assim avança até início da próxima linha!

ler inteiro: 34
Ler frase: Ria de aveiro e OVAR
34 Ria de aveiro e OVAR
converte '231' para inteiro = 231
primeira posicao de 'av': 7
última posicao de 'av': 7

Strings – funções (2)



deti

universidade de aveiro
departamento de electrónica,
telecomunicações e informática

```
System.out.printf("substitui: %s\n", s.replace("av", "AVEIRO"));
System.out.printf("sub string (3,5): %s\n", s.substring(3, 5));
System.out.printf("sub string:(3, até fim) %s\n", s.substring(3));
System.out.printf("string começa com 'av'- a partir da posição 4: %B\n",
    s.startsWith("av", 4));
System.out.printf("string começa com 'av': %B\n", s.startsWith("av"));
```

Ler frase: Ria de aveiro e OVAR

...

substitui: Ria de AVEIROeiro e OVAR

sub string (3,5): d

sub string:(3, até fim): de aveiro e OVAR

string começa com 'av'- a partir da posição 4: FALSE

string começa com 'av': FALSE

Strings – funções (3)



deti

universidade de aveiro
departamento de electrónica,
telecomunicações e informática

```
t=s.split(" ");  
for (String a : t)System.out.printf("sub string: %s\n", a);  
System.out.printf("PARA MAIUSCULAS: %s\n", s.toUpperCase());  
System.out.printf("para minúsculas: %s\n", s.toLowerCase());
```

Ler frase: Ria de aveiro e OVAR

...

sub string: Ria

sub string: de

sub string: aveiro

sub string: e

sub string: OVAR

PARA MAIUSCULAS: RIA DE AVEIRO E OVAR

para minúsculas: ria de aveiro e ovar