

Programação 1

Aula 7

Valeri Skliarov, Prof. Catedrático

Email: skl@ua.pt

URL: <http://sweet.ua.pt/skl/>

Departamento de Eletrónica, Telecomunicações e Informática
Universidade de Aveiro

<http://elearning.ua.pt/>

Revisão da aula anterior

No final da aula

Classe Character

Classe String

Declaração e definição, objetos, operações,
funções (métodos), entrada e saída

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

Strings (sequências de caracteres)

- 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

Diagram illustrating the ASCII code mapping across three columns: Código binário, Código octal, Código decimal, and Código hexadecimal.

Arrows indicate the mapping from the binary code to the octal, decimal, and hexadecimal codes.

Código binário	Código octal	Código decimal	Código hexadecimal
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
011 0001	061	49	31
011 0010	062	50	32
011 0011	063	51	33
011 0100	064	52	34
011 0101	065	53	35
011 0110	066	54	36
011 0111	067	55	37
011 1000	070	56	38
011 1001	071	57	39
011 1010	072	58	3A
100 0001	101	65	41
100 0010	102	66	42
100 0011	103	67	43
100 0100	104	68	44
100 0101	105	69	45
100 0110	106	70	46
100 0111	107	71	47
100 1000	110	72	48
100 1001	111	73	49
100 1010	112	74	4A
100 1011	113	75	4B
100 1100	114	76	4C
100 1101	115	77	4D
100 1110	116	78	4E
100 1111	117	79	4F
101 0000	120	80	50
101 0001	121	81	51
101 0010	122	82	52
101 0011	123	83	53
101 0100	124	84	54
101 0101	125	85	55
101 0110	126	86	56
101 0111	127	87	57
101 1000	130	88	58
101 1001	131	89	59
101 1010	132	90	5A
110 0001	141	97	61
110 0010	142	98	62
110 0011	143	99	63
110 0100	144	100	64
110 0101	145	101	65
110 0110	146	102	66
110 0111	147	103	67
110 1000	150	104	68
110 1001	151	105	69
110 1010	152	106	6A
110 1011	153	107	6B
110 1100	154	108	6C
110 1101	155	109	6D
110 1110	156	110	6E
110 1111	157	111	6F
111 0000	160	112	70
111 0001	161	113	71
111 0010	162	114	72
111 0011	163	115	73
111 0100	164	116	74
111 0101	165	117	75
111 0110	166	118	76
111 0111	167	119	77
111 1000	170	120	78
111 1001	171	121	79
111 1010	172	122	7A

Exemplo:

```
public class String1 {
    public static void main (String args[]) {
        String s = "Aveiro";
        for(int i=0; i < s.length(); i++)
            System.out.printf("%c - %d\n", s.charAt(i), (int)s.charAt(i));
    }
}
```

A - 65

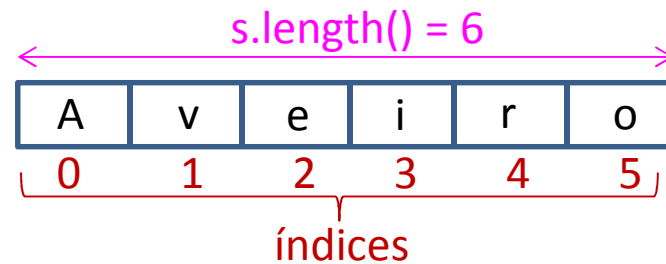
v - 118

e - 101

i - 105

r - 114

o - 111



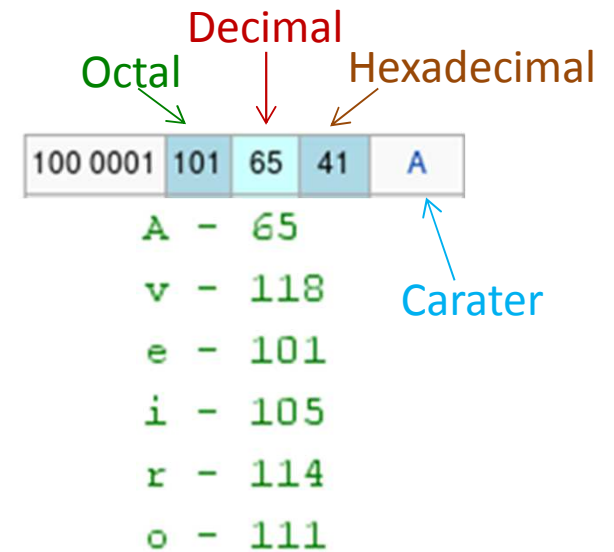
010 1011	053	43	2B	+
010 1100	054	44	2C	,
010 1101	055	45	2D	-
010 1110	056	46	2E	.
010 1111	057	47	2F	/
011 0000	060	48	30	0
011 0001	061	49	31	1
011 0010	062	50	32	2
011 0011	063	51	33	3
011 0100	064	52	34	4
011 0101	065	53	35	5
011 0110	066	54	36	6
011 0111	067	55	37	7
011 1000	070	56	38	8
011 1001	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

Exemplo:

```
public class String1 {  
    public static void main (String args[])    {  
        String s = "Aveiro";  
        for(int i=0; i < s.length(); i++)  
            System.out.printf("%c - %d\n", s.charAt(i), (int)s.charAt(i));  
    }  
}
```



```
public class String1 {  
    public static void main (String args[])    {  
        String s = "Aveiro";  
        System.out.printf("Character: %c; Octal: %o; Decimal: %d; Hexadecimal: %h\n",  
                           s.charAt(0), (int)s.charAt(0), (int)s.charAt(0), (int)s.charAt(0));  
    } }  
}
```

Character: A; Octal: 101; Decimal: 65; Hexadecimal: 41

Exemplo:

```
public class String1 {  
    public static void main (String args[])    {  
        String s = "Aveiro";  
        for(int i=0; i < s.length(); i++)  
            System.out.printf("%c - %d\n", s.charAt(i), (int)s.charAt(i));  
    }  
}
```

```
public class StringBinario {  
    public static void main (String args[])    {  
        String s = "Aveiro";  
        System.out.printf("Character: %c; Binário: %s\n", s.charAt(0),  
                           Integer.toString((int)s.charAt(0),2));  
    }  
}
```

Binário	Carater
100 0001	A

A	-	65
v	-	118
e	-	101
i	-	105
r	-	114
o	-	111

Character: A; Binário: 1000001

Integer.toString((int)s.charAt(0),2)

s.charAt(0) = 'A'

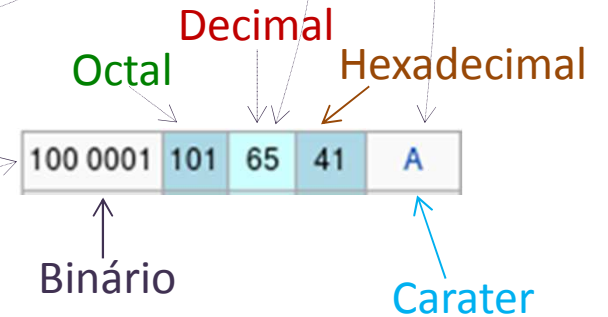
(int)s.charAt(0) = (int)'A' = 65

Integer.toString((int)s.charAt(0),2) =
Integer.toString(65,2) = 1000001

Radix – significa
converter para
binário

A função toString faz
parte da classe Integer

Integer.toString



Classe Character

- A classe `Character` contém um conjunto de funções (**métodos**) para processamento de caracteres (**operações sobre caracteres**).
- As funções (**os métodos**) disponibilizadas(**os**) 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`, **`isWhitespace`**, `isLowerCase`, `isUpperCase`, ...
 - funções de conversão que devolvem outro carater:
 - `toLowerCase`, `toUpperCase`, ...
- Estas funções utilizam-se tal como as da classe `Math`:

`Character.nomeDaFuncao(...)`

Classe String

- 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 carater numa determinada posição
int `length()` – devolve a dimensão de uma `String`
int `indexOf(char)` - pesquisa a primeira ocorrência do carater
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()`.
- <https://docs.oracle.com/javase/8/docs/api/java/lang/String.html>
- <https://docs.oracle.com/javase/8/docs/api/java/lang/Character.html>

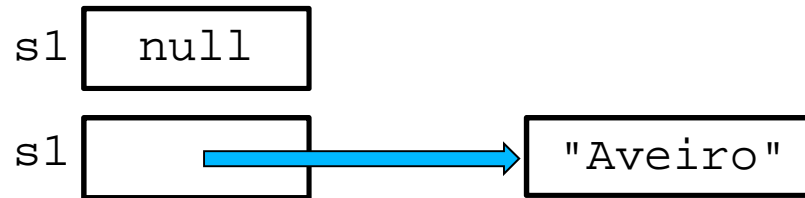
Declaração de variáveis do tipo String

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
```

Exemplo:

Letters: 14; Digits: 4; LettersOrDigits: 18; LowerCases: 5; UpperCases 9

```
public class StringTeste {  
    public static void main (String args[])    {  
        String s = "Aveiro (PORTUGAL) 2014";  
        int Letters=0,Digits=0,LettersOrDigits=0,LowerCases=0,UpperCases=0;  
        for(int i=0; i<s.length();i++) {  
            if(Character.isLetter(s.charAt(i)))    Letters++;  
            if(Character.isDigit(s.charAt(i)))    Digits++;  
            if(Character.isLetterOrDigit(s.charAt(i))) LettersOrDigits++;  
            if(Character.isLowerCase(s.charAt(i))) LowerCases++;  
            if(Character.isUpperCase(s.charAt(i))) UpperCases++;  
        }  
        System.out.printf("Letters: %d; Digits: %d; LettersOrDigits: %d; LowerCases: %d; UpperCases %d\n",  
                           Letters,Digits,LettersOrDigits,LowerCases,UpperCases);  
    }  
}
```

Pode escrever em
várias linhas

```
System.out.printf("Letters: %d; Digits: %d; "+  
                  "LettersOrDigits: %d; LowerCases: %d;"+  
                  " UpperCases %d\n",  
                  Letters,Digits,LettersOrDigits,  
                  LowerCases,UpperCases);
```

```

public class StringTeste {
public static void main (String args[])    {
    String s = " Aveiro (PORTUGAL) 2014";
    int Letters=0,Digits=0, Spaces=0, LettersOrDigits=0,LowerCases=0,UpperCases=0;
    for(int i=0; i<s.length();i++) {
        if(Character.isLetter(s.charAt(i)))        Letters++;
        if(Character.isDigit(s.charAt(i)))          Digits++;
        if(Character.isWhitespace(s.charAt(i)))      Spaces++;
        if(Character.isLetterOrDigit(s.charAt(i)))   LettersOrDigits++;
        if(Character.isLowerCase(s.charAt(i)))       LowerCases++;
        if(Character.isUpperCase(s.charAt(i)))       UpperCases++;
    }
    System.out.printf("Letters: %d; Digits: %d; Spaces: %d;" +
        " LettersOrDigits: %d; LowerCases: %d;" +
        " UpperCases %d\n",
        Letters,Digits, Spaces, LettersOrDigits,
        LowerCases,UpperCases);
    }
}

```

```

Letters: 14; Digits: 4; Spaces: 3; LettersOrDigits: 18; LowerCases: 5; UpperCases 9

```

```
if(Character.isWhitespace(s.charAt(i)))    Spaces++;
```

Pode utilizar a função *isSpace* em vez de *isWhitespace* mas vai aparecer a mensagem seguinte

1 warning found:

File: H:\MostFrequentlyNeeded__2014\Education2011\2014_2015\FirstSemester\Programacao\AulasTeoricas\8_11_11\Exemplos\StringTeste.java [line: 8]

Warning: The method isSpace(char) from the type java.lang.Character is deprecated

```
javac "StringTeste.java" (in directory: /media/aveiro/My Passport/)
```

```
Note: StringTeste.java uses or overrides a deprecated API.
```

```
Note: Recompile with -Xlint:deprecation for details.
```

```
Compilation finished successfully.
```

É melhor utilizar a função *isWhitespace* (ver <https://docs.oracle.com/javase/7/docs/api/java/lang/Character.html> para detalhes adicionais)

Leitura e escrita de Strings

- 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(...)`, `System.out.println(...)` e `System.out.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);
```

Exemplo:

```
// 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 dígito");
    else
        System.out.println("Não inseriu uma letra ou dígito");
} while(c != '.');
```

Exemplo:

```
import java.util.*;

public class Format_s {
    public static void main(String[] args) throws Exception {
        String str = "Hello world!";
        System.out.println("println  '+'\t'+'\t'+'\t'+str);
        System.out.print("print  '+'\t'+'\t'+'\t'+str+'\n');
        System.out.printf("printf          %s\n",str);
        System.out.printf("printf  %%30s  %%30s  %30s%30s\n",str,str);
        System.out.printf("printf  %%-30s  %%-30s  %-30s%-30s\n",str,str);
    }
}
```

```
println          Hello world!
print            Hello world!
printf           Hello world!
printf  %30s  %30s          Hello world!          Hello world!
printf  %-30s  %-30s  Hello world!          Hello world!
```

Exemplo

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

Código completo:

```
import java.util.*;
public class Slide12 {
    static final Scanner sc = new Scanner(System.in);
    public static void main (String args[])    {
        String frase = new String();
        char letra;
        System.out.print("Escreva uma frase: ");
        frase = sc.nextLine();
        System.out.printf("A frase tem as letras:\n");
        for(int i = 0 ; i < frase.length() ; i++)
            System.out.println(frase.charAt(i));
        }
    }
```

Escreva uma frase:

A frase tem as letras:

A
v
e
i
r
o

U
A

Diferença entre as funções *next* e *nextLine* da classe *Scanner*

Exemplo:

```
import java.util.*;
public class next_nextLine {
    static final Scanner read = new Scanner(System.in);
    public static void main (String args[]) {
        String s;
        System.out.print("Linha ? ");
        s = read.nextLine(); ←
        System.out.println("s = "+s);
    }
}
```

Linha ? Universidade de Aveiro
s = Universidade de Aveiro

Ok

```
import java.util.*;
public class next_nextLine {
    static final Scanner read = new Scanner(System.in);
    public static void main (String args[]) {
        String s;
        System.out.print("Linha ? ");
        s = read.next(); ←
        System.out.println("s = "+s);
    }
}
```

Linha ? Universidade de Aveiro
s = Universidade

Ok

mas os resultados são diferentes

Exemplo:

```
import java.util.*;
public class next_nextLine {
    static final Scanner read = new Scanner(System.in);
    public static void main (String args[])    {
        String s;
        System.out.print("Linha ? ");
        s = read.next();
        System.out.println("s = "+s);
        s = read.next();
        System.out.println("s = "+s);
        s = read.next();
        System.out.println("s = "+s);
    }
}
```

```
Linha ?  Universidade de Aveiro
s = Universidade
s = de
s = Aveiro
```


Problema:

```
import java.util.*;
public class next_nextLine {
    static final Scanner read = new Scanner(System.in);
    public static void main (String args[]) {
        String s;
        int a;
        System.out.print("Inteiro ? ");
        a = read.nextInt();
        System.out.println("a = "+a);
        read.skip("\n");
        System.out.print("Linha ? ");
        s = read.nextLine();
        System.out.println("s = "+s);
    }
}
```

```
Inteiro ? 123
a = 123
Linha ? s =
>
```

Solução

```
Inteiro ? 123
a = 123
Linha ? Universidade de Aveiro
s = Universidade de Aveiro
```

Problema:

```
import java.util.*;
public class next_nextLine_charAt {
    static final Scanner read = new Scanner(System.in);
    public static void main (String args[])
    {
        char c;
        int a;
        System.out.print("char ? ");
        c = read.nextLine().charAt(0);
        System.out.println("c = "+c);
        System.out.print("int ? ");
        a = read.nextInt();
        System.out.println("a = "+a);
        read.skip("\n");
        System.out.print("char ? ");
        c = read.nextLine().charAt(0);
        System.out.println("c = "+c);
    }
}
```


```
char ? R
c = R
int ? 123
a = 123
char ? java.lang.StringIndexOutOfBoundsException: String index out of range: 0
    at java.lang.String.charAt(Unknown Source)
    at next_nextLine_charAt.main(next_nextLine_charAt.java:21)
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at sun.reflect.NativeMethodAccessorImpl.invoke(Unknown Source)
    at sun.reflect.DelegatingMethodAccessorImpl.invoke(Unknown Source)
    at java.lang.reflect.Method.invoke(Unknown Source)
    at edu.rice.cs.drjava.model.compiler.JavacCompiler.runCommand(JavacCompiler.java:271)
```

Solução

```
char ? R
c = R
int ? 123
a = 123
char ? v
c = v
```

Para função **next** pode utilizar separadores. Exemplo:

```
import java.util.*;
public class Deliminer {
public static void main (String args[]) {
    String cidades = "cidade Lisboa cidade Porto "+
        "cidade Coimbra cidade Aveiro cidade Braga cidade Faro";
    Scanner read = new Scanner(cidades).useDelimiter("\\s*cidade\\s*");
    while (read.hasNext()) System.out.printf("%s\n",read.next());
    read.close();
    }
}
```



```
Lisboa
Porto
Coimbra
Aveiro
Braga
Faro
```

Operações com caracteres

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

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

26 caracteres

Código completo:

```
import java.util.*;
public class Deslocamento {
    static final Scanner read = new Scanner(System.in);
    public static void main (String args[]) {
        char letra,novaLetra = '0';
        int pos, novaPos;
        System.out.print("Letra ? ");
        letra = read.nextLine().charAt(0);
        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);
        }
        System.out.println("Nova letra = "+novaLetra);
    }
}
```

Letra ? Y

Nova letra = B

Letra ? A

Nova letra = D

+3

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

+3

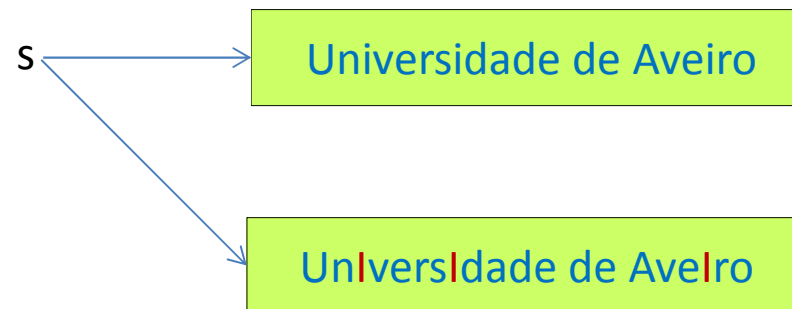
Propriedades das Strings

- 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 fato 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...).

Exemplo:

```
import java.util.*;
public class ChangedString {
    static final Scanner read = new Scanner(System.in);
    public static void main (String args[]) {
        String s = "Universidade de Aveiro";
        System.out.println(s);
        s = s.replace('i', 'I');
        System.out.println(s);
    }
}
```

Universidade de Aveiro
UnIversIdade de AveIro



Passagem de Strings a funções

- 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);  
}
```


Código completo:

```
import java.util.*;
public class Slide13 {
    static final Scanner sc = new Scanner(System.in);
    public static void main (String args[])    {
        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){
        System.out.printf("%s\n", s);           //imprime "Aveiro"
        s = "ola";                             // s passa a referenciar algo diferente...
        System.out.printf("%s\n", s);
    }
}
```

Aveiro
ola
Aveiro

Exemplo: Algumas funções da classe *String*

```
public class Operadores_mais_e_menos {  
    public static void main (String args[])    {  
        String s1 = "Aveiro";  
        String s2 = "PORTUGAL";  
        String soma,tmp;  
        soma = s1 + " " + s2;  
        System.out.println(soma);                // Aveiro  PORTUGAL  
        System.out.println(tmp=soma.replaceFirst("PORTUGAL", " ")); // Aveiro  
        System.out.println(s1.concat(s2));        // AveiroPortugal  
        System.out.println(tmp.toLowerCase() );    // aveiro  
        System.out.println(tmp.toUpperCase() );     // AVEIRO  
    }  
}
```

```
Aveiro  PORTUGAL  
Aveiro  
AveiroPORTUGAL  
aveiro  
AVEIRO
```

Exemplo: Algumas funções da classe *String*

```
public class String_Digits {  
    public static void main (String args[])    {  
        String inteiro = String.valueOf(987);  
        String s = "Universidade de Aveiro (Portugal)";  
        System.out.printf("String = %s\n",String.valueOf(123.4567)); // String = 123.4567  
        System.out.printf("String (inteiro) = %s\n", inteiro);        // String (inteiro) = 987  
        System.out.println(s.substring(16,22));                      // Aveiro  
        System.out.println(s.replaceFirst("Aveiro","Porto"));        // Universidade de Porto (Portugal)  
        System.out.println(s.split(" ")[0]);                        // Universidade  
        System.out.println(s.split(" ")[1]);                        // de  
        System.out.println(s.split(" ")[2]);                        // Aveiro  
        System.out.println(s.split(" ")[3]);                        // (Portugal)  
        System.out.println(s.split("Aveiro")[0]);                  // Universidade de  
        System.out.println(s.split("Aveiro")[1]);                  // (Portugal)  
    }  
}
```

```
String = 123.4567  
String (inteiro) = 987  
Aveiro  
Universidade de Porto (Portugal)  
Universidade  
de  
Aveiro  
(Portugal)  
Universidade de  
(Portugal)
```

Revisão da aula anterior

Exemplo: Declarar uma classe "Retângulo" do seguinte tipo:

```
class Retângulo
{
    double ladoA;
    double ladoB;
    double diagonal;
}
```

Tarefas:

- 1) Criar uma função *G* que permite gerar lados A e B aleatoriamente no intervalo 2.0-10.0.
- 2) Criar uma função *Diag* que retorna diagonal de retângulo.
- 3) Criar uma função *Dif* que retorna a diferença entre os lados A e B.

```
static Random rand = new Random();
```

```
public static void G(Retangulo R)
{
    R.ladoA = (double)(rand.nextInt(8) + 2);
    R.ladoB = (double)(rand.nextInt(8) + 2);
}
```

Exemplo:

Tarefas:

- 1) Criar uma função *G* que permite gerar lados A e B aleatoriamente no intervalo 2.0-10.0.

```
import java.util.*;  
public class rect  
{ static Random rand = new Random();  
  public static void main(String[] args)  
  { Rectangulo rect = new Rectangulo();  
    G(rect);  
    System.out.println("lado A = "+rect.ladoA);  
    System.out.println("lado B = "+rect.ladoB);  
  }  
  public static void G(Rectangulo R)    {  
    R.ladoA = (double)(rand.nextInt(8) + 2);  
    R.ladoB = (double)(rand.nextInt(8) + 2);  }  
}  
class Rectangulo  
{  
    double ladoA;  
    double ladoB;  
    double diagonal;  
}
```

1

rand.nextDouble(); // não tem argumentos

lado A = 8.0

lado B = 3.0

Exemplo:

Tarefas:

2) Criar uma função *Diag* que retorna diagonal de retângulo.

```
public static double Diag(Rectangulo R)    {  
    return Math.sqrt(Math.pow(R.ladoA,2)+Math.pow(R.ladoB,2))    }
```



$$\textit{return} \sqrt{(ladoA^2 + ladoB^2)}$$

Exemplo:

Tarefas:

2) Criar uma função *Diag* que retorna diagonal de retângulo.

```
import java.util.*;
public class rect
{
    public static void main(String[] args)
    {
        Rectangulo rect = new Rectangulo();
        rect.ladoA = 4.0;
        rect.ladoB = 3.0;
        System.out.println("diagonal = "+Diag(rect));
    }
    public static double Diag(Rectangulo R)
    {
        return Math.sqrt(Math.pow(R.ladoA,2)+Math.pow(R.ladoB,2));
    }
}
class Rectangulo
{
    double ladoA;
    double ladoB;
    double diagonal;
}
```

2

diagonal = 5.0

Exemplo:

Tarefas:

3) Criar uma função *Dif* que retorna a diferença entre os lados A e B.

```
public static double Dif(Rectangulo R)
{   return Math.abs(R.ladoA - R.ladoB);   }
```



Exemplo:

Tarefas:

3) Criar uma função *Dif* que retorna a diferença entre os lados A e B.

```
public class rect
{
    public static void main(String[] args)
    {   Rectangulo rect = new Rectangulo();
        rect.ladoA = 7.0;
        rect.ladoB = 10.0;
        System.out.println("diferenca = "+Dif(rect));
    }

    public static double Dif(Rectangulo R)
    {   return Math.abs(R.ladoA - R.ladoB);   }
}

class Rectangulo
{
    double ladoA;
    double ladoB;
    double diagonal;
}
```

3

diferenca = 3.0

Código completo:

```
import java.util.*;
public class rect
{ static Random rand = new Random();
  public static void main(String[] args)
  { Rectangulo rect = new Rectangulo();
    G(rect);
    System.out.println("lado A = "+rect.ladoA);
    System.out.println("lado B = "+rect.ladoB);
    System.out.println("diagonal = "+Diag(rect));
    System.out.println("diferenca = "+Dif(rect));
  }
  public static void G(Rectangulo R)      {
    R.ladoA = (double)(rand.nextInt(8) + 2);
    R.ladoB = (double)(rand.nextInt(8) + 2);  }
  public static double Diag(Rectangulo R)
  { return Math.sqrt(Math.pow(R.ladoA,2)+Math.pow(R.ladoB,2)); }
  public static double Dif(Rectangulo R)
  { return Math.abs(R.ladoA - R.ladoB); }
}
class Rectangulo
{
  double ladoA;
  double ladoB;
  double diagonal;
}
```

```
lado A = 7.0
lado B = 3.0
diagonal = 7.615773105863909
diferenca = 4.0
```

O que é uma referência

```
dados d = new dados();  
//.....  
class dados  
{ int a = 10, int b = 20; }
```

1. Para os tipos novos (para registos) memória deve ser reservada, por exemplo, `new dados();`
2. Na linha `dados d = new dados();` `d` é uma referência que significa **onde fica na memória o objeto novo que foi criado**

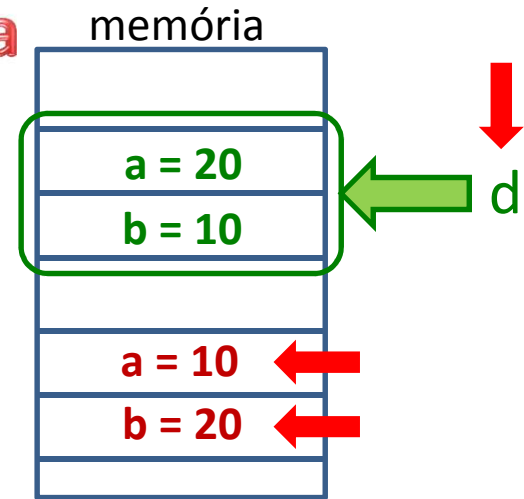
3. De notar que memória foi reservada fora de funções
4. Vamos chamar uma função `f` e passar `d` como argumento `f(d);`
5. Agora o argumento `d` dentro da função `f` pode utilizar dados na memória fora da função `f`
6. Quando a função `f` terminar a memória da função vai ser destruída mas a memória fora da função não vai ser destruída. Por isso todas as alterações do objeto feitas pela função `f` são válidas depois de terminação da função.
7. Para aceder um campo do objeto é necessário utilizar: *referencia.nome_do_campo*, por exemplo: `d.a`

O que é uma referência

```
import java.util.*;
public class trocar
{
    public static void trocar(dados d, int a, int b)    {
        int tmp;
        tmp = d.a; d.a = d.b; d.b = tmp;
        tmp = a; a = b; b = tmp;
    }

    public static void main(String[] args)
    {
        dados d = new dados();
        int a=10, b=20;
        System.out.printf("d.a = %d; d.b= %d\n", d.a, d.b);
        System.out.printf("a = %d; b= %d\n", a, b);
        trocar(d,a,b);
        System.out.printf("d.a = %d; d.b= %d\n", d.a, d.b);
        System.out.printf("a = %d; b= %d\n", a, b);
    }
}

class dados
{
    int a = 10;
    int b = 20;
}
```



Troca de valores foi feita
através de referências e não
foi feita através de valores

```
public static void trocar(dados d, int a, int b)    {
    int tmp;
    tmp = d.a; d.a = d.b; d.b = tmp;
    tmp = a; a = b; b = tmp;
}
```

