STRING

MICHELLE NERY NASCIMENTO
PUC MINAS
ALGORITMOS E TÉCNICAS DE PROGRAMAÇÃO

DEFINIÇÃO

- Uma string é uma série de caracteres tratados como uma única unidade.
- Esses caracteres podem ser letras maiúsculas e minúsculas, dígitos e vários caracteres especiais como +, -, *, /, $, e outros. Eles são codificados conforme padrão UNICODE.
- Uma string no Java é um objeto da classe `String`. Os literais de `String` são frequentemente referidos como objetos e escritos entre aspas duplas em um programa.
- Objetos `String` são imutáveis — depois de criado, o conteúdo dos caracteres não pode ser alterado.

Ex.: "José P. Souza"

"Rua Principal, 9999"

"Belo Horizonte – MG"

CONSTRUTORES DE STRINGS

- A classe String fornece diversos construtores para inicializar strings de várias maneiras.

- Em geral tem-se:

```
String str = new String();
```

```
class ConstroiString
{
    public static void Main (string[] args)
     {
        String stringOriginal, string1, string2, string3,
                string4, string5, string6;

        char [ ] VetorDeCaracter = { 'F' , 'e' ,  'l' ,  'i' ,  'z' ,  ‘ , ' , 'd' , 'i' , 'a' };
        stringOriginal = "Bem vindo";
        string1 = stringOriginal;
        string2 = new string (VetorDeCaracter);
        string3 = new string (VetorDeCaracter, 6, 3);
        string4 = new string ( 'C', 5);
        string5 = "Linha 1\nLinha 2";
        System.out.println(“String1: ” + string1+ “String2: ” + string2 + “String3: ”
+ string3+ “String4: ” + string4 + “String5: ” + string5);
      }
}
```

Saída:

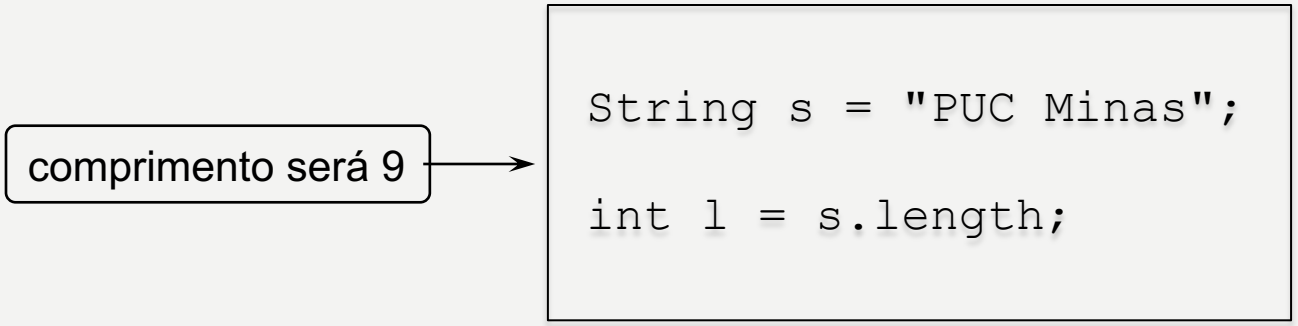
```
String1: Bem vindo
String2: Feliz,dia
String3: dia
String4: CCCCC
String5: Linha 1
Linha 2
```

SEQUENCIAS DE ESCAPE

- Precedidos pelo sinal de \ (“barra invertida)
- Exemplos:
  - `Nova linha:` `string a = "hello\n";`
  - `Aspas duplas:` `string b = "\"";`
  - `Aspa simples:` `string c = "\'";`
  - `Barra invertida:` `string b = "\\";`

COMPRIMENTO DE UMA STRING

- Método `length`
  - Retorna o comprimento da String.



INDEXAÇÃO DE STRINGS

- Facilita a recuperação de qualquer caractere em uma localização específica em uma String .
- Índice começa de 0 (zero), como em Arrays.

recupera `'M'` ⟶

```
string s = "PUC Minas";

char c = s.CharAt[4];
```

MÉTODO: GETCHARS

- O método getChars recupera um conjunto de caracteres de uma String como um array char.
- Recebe quatro argumentos:
  - o primeiro é o valor de índice inicial a partir do qual caracteres serão copiados.
  - O segundo é o valor de índice que está depois do último caractere copiado da string.
  - O array que receberá os caracteres é o terceiro argumento.
  - Finalmente, o último argumento é o valor de índice (posição) no array a partir de onde os caracteres serão armazenados.

GETCHARS: EXEMPLO

```
public static void main(String[] args)
  {
     String s1 = "hello there";
     char[] charArray = new char[5];
     s1.getChars(0, 5, charArray, 0);

     System.out.printf("%nO vetor de caracter é:
  ");
     for (char caracter: charArray)
        System.out.print(caracter);

     System.out.println();
  }
```

Será impresso:

hello

COMPARANDO STRINGS: EQUALS

- O método **equals** testa a igualdade de qualquer dois objetos (verifica se os objetos possuem conteúdos idênticos).
- O método retorna **true** se os objetos são iguais e **false**, caso contrário .
- **equals** usa uma *comparação lexicográfica* – os valores Unicode inteiros que representam cada caracter são comparados. Note que isto significa que ele diferencia maiúsculas de minúsculas.

Sintaxe: `objeto1.equals(objeto)`

`objeto1`: string original.

objeto: string a ser comparada com a original.

Exemplo

```
public static void main (string[] args)
{
    String string1 = "Ola";
    System.out.println ("String 1:“ + string1);
    //testa a igualdade usando metodo Equals
     if (string1.equals ("Ola"))
        System.out.println (" A string 1” + string1+ “é igual a Ola");
     else
         System.out.println ("A string 1” + string1+ “não é igual a Ola");
}
```

EXEMPLO:

o valor inteiro Unicode de uma letra minúscula é diferente do valor para letra maiúscula.

```
public static void main (string[] args)
{
   String string1 = "BOM DIA";
   String string2 = "bom dia";
   System.out.println ("String 1:"+ string1+ "- String 2: "+ string2);
    //testa a igualdade usando método equals
   if (string1.equals(string2))
     System.out.println("A string 1:"+ string1+ "é igual a string 2: "+ string2);
   else
     System.out.println(" A string 1:"+string1 "não eh igual a string 2:"+ string2);
}
```

Saída:

```
String 1: BOM DIA - String 2: bom dia
A string 1: BOM DIA nao eh igual a string2: bom dia

Press any key to continue...
```

COMPARANDO STRINGS: EQUALSIGNORECASE

- O método **equalsIgnoreCase** testa a igualdade de qualquer dois objetos, mas ignora se as letras estão em maiúsculas ou minúsculas.
- O método retorna **true** se os objetos são iguais e **false**, caso contrário .

Sintaxe: `objeto1.equalsIgnoreCase(objeto)`

`objeto1`: string original.

objeto: string a ser comparada com a original.

EXEMPLO

```
public static void main (string[] args)
{
   String string1 = “Feliz";
   String string2 = “feliz";
   System.out.println (“String 1:”+ string1+ “- String 2: ”+ string2);
    //testa a igualdade usando método equalsIgnoreCase
   if (string1.equalsIgnoreCase(string2))
     System.out.println("A string 1:”+ string1+ “é igual a string 2: ”+ string2);
   else
     System.out.println(" A string 1:”+string1 “não eh igual a string 2:”+ string2);
}
```

```
String 1:Feliz- String 2: feliz
A string 1: Felizé igual a string 2: feliz
(base) MacBook-Pro-de-Usuario:LabJava usuario$
```

COMPARANDO STRINGS: COMPARETO

- O método compareTo retorna:

  a) 0 se as Strings forem iguais

  b) Um número negativo se a String que invoca o método compareTo for menor que a String passada como argumento

  c) Um número positivo se a String que invoca o método compareTo for maior que a String passada como argumento

Sintaxe: `objeto1.compareTo(objeto)`

`objeto1`: string chamador (que invoca).

objeto: string passada como argumento.

Exemplo

```
public static void main(String[] args) {
    String str1 =  "Hoje o dia está maravilhoso!";
    String str2 = "felicidade";
    int result;
    result = str1.compareTo(str2);
    if(result == 0)
        System.out.println("str1 e str2 são iguais");
    else if(result < 0)
        System.out.println("str1 é menor que str2");
    else
        System.out.println("str1 é maior que str2")
}
```

Saída: str1 é maior que str2

COMPARANDO STRINGS: REGIONMATCHES – 4 ARGUMENTOS

- O método regionMatches é utilizado para comparar a igualdade entre partes de duas Strings. O primeiro argumento é o índice inicial na String que chama o método. O segundo argumento é uma comparação de String. O terceiro argumento é o índice inicial na comparação de String. O último argumento é o número de caracteres a comparar entre as duas Strings.
- O método retorna true apenas se o número especificado de caracteres for lexicograficamente igual.

```
public static void main(String[] args)
  {
    String s3 = "Happy Birthday";
    String s4 = "happy birthday";

    // teste regionMatches (case sensitive)
    if (s3.regionMatches(0, s4, 0, 5))
      System.out.println("Os 5 primeiros caracteres de s3 e s4 casam");
    else
      System.out.println(" Os 5 primeiros caracteres de s3 e s4 não   casam ");
  }
```

Saída: Os 5 primeiros caracteres de s3 e s4 não casam

COMPARANDO STRINGS: REGIONMATCHES – 5 ARGUMENTOS

- O método regionMatches com 5 argumentos também é utilizado para comparar a igualdade de partes de duas Strings.
- Diferença:
  - Quando o primeiro argumento é true, o método ignora maiúsculas e minúsculas dos caracteres sendo comparados.
  - Os argumentos restantes são idênticos àqueles descritos para o método de 4 argumentos anteriormente.

```
public static void main(String[] args)
  {
    String s3 = "Happy Birthday";
    String s4 = "happy birthday";

    // teste regionMatches (ignora maiúsculas/minúsculas)
    if (s3.regionMatches(true, 0, s4, 0, 5))
      System.out.println("Os 5 primeiros caracteres de s3 e s4 casam com
                                  case ignorado");
    else
      System.out.println(" Os 5 primeiros caracteres de s3 e s4 não casam ");
  }
```

Saída: Os 5 primeiros caracteres de s3 e s4 casam com case ignorado

COMPARANDO STRINGS

- O método ***startsWith*** determina se uma instância de uma string começa com um texto passado como argumento.

- E o método ***endsWith*** determina se uma instância de uma string termina com um texto passado como argumento.

- Em ambos, retorna **true** se essa instância começar/terminar com o texto; caso contrário, **false**.

```
public static void main (string[] args)
{
    String [ ] strings = { "inicio", "inicializar", "meio", "fim"}  ;
     for (int i =0; i < strings.length; i++)
    {
      if (strings[i].startsWith ("ini"))
          System.out.println("A string"+strings[i]+"começa com ini",);
     }
    for (int i =0; i < strings.Length; i++)
     {
      if (strings[i].endsWith ("eio"))
          System.out.println("A string"+strings[i]+"termina com eio”);
     }
}
```

Saída:

```
A string inicio comeca com ini
A string inicializar comeca com ini
A string meio termina com eio

Press any key to continue...
```

LOCALIZANDO CARACTERES E SUBSTRINGS

- O método ***IndexOf*** localiza a primeira ocorrência de um caracter ou substring em uma string.
  - Se o método encontra um caracter, ele retorna o índice do caracter especificado. Caso contrário, o método retorna -1.

- O método ***LastIndexOf*** localiza a última ocorrência de um caracter em uma string. A pesquisa é feita do final para o início.
  - Se o caracter é encontrado, o seu índice é retornado. Caso contrário o método retorna -1.

```
public static void main (string[] args)
{
    String letras = "abcdefghijklmnopqrstuvwxyzabcdefghijklmenop" ;
    int posicao;
    posicao = letras.indexOf('c');
    System.out.println("c está na posição : " + posicao);

    posicao = letras.indexOf('$');
    System.out.println("$ está na posição: " + posicao);

    posicao = letras.lastIndexOf('c');
    System.out.println("A última ocorrência de c esta na posição : " + posicao);

    posicao = letras.lastIndexOf('$');
    System.out.println("A última ocorrência de $ esta na posição : " + posicao);
}
```

Saída:

```
c esta na posicao: 2
$ esta na posicao: -1
A ultima ocorrencia de c esta na posicao: 28
A ultima ocorrencia de $ esta na posicao: -1
```

```
public static void main (string[] args)
{
    String letras = "abcdefghijklmnopqrstuvwxyzabcdefghijklmenop" ;
    int posicao;
    posicao = letras.indexOf('a', 1);  // índice que vai iniciar a busca
    System.out.println("a está na posição : " + posicao);
    posicao = letras.indexOf("def");
    System.out.println("def está na posição: " + posicao);

    posicao = letras.lastIndexOf('a', 27); // índice que vai iniciar a busca (de trás
                                                          para a frente)
    System.out.println("A última ocorrência de a esta na posição : " + posicao);
    posicao = letras.lastIndexOf("def");
    System.out.println("A última ocorrência de $ esta na posição : " + posicao);
}
```

Saída:

```
a está na posição: 26
def está na posicao: 3
A última ocorrência de a está na posição: 26
A última ocorrência de def está na posição: 29
```

EXTRAINDO SUBSTRINGS DE STRINGS

- Opção1: O argumento no método substring especifica o índice a partir do qual o método copia caracteres da string original até o fim da mesma. Inicia-se com o índice 0.
  - Obs.: Especificar um índice fora dos limites da String causa uma StringIndexOutOfBoundsException.

Exemplo:

…

```
String letras = "abcdefghijklmnopqrstuvwxyzabcdefghijklmenop" ;
String resultado;
resultado = letras.substring(20);
System.out.println(resultado);
```

…

Saída: uvwxyzabcdefghijklmenop

EXTRAINDO SUBSTRINGS DE STRINGS

- Opção2: O primeiro argumento especifica o índice a partir do qual o método copia caracteres da string original e o segundo argumento especifica o comprimento da substring a ser copiada.

Exemplo:

```
…
        String letras = "abcdefghijklmnopqrstuvwxyzabcdefghijklmenop" ;
        String resultado;
        resultado = letras.substring(0, 6);
        System.out.println(resultado);
…
```

Saída: abcdef

CONCATENANDO STRINGS

- Podemos concatenar Strings usando o operador (+) e também o método concat.

```
public static void main (string[] args)
{
    String string1 = "Feliz ";
    String string2 = "Aniversario";
    System.out.println("Concatenando com +: " +string1 + string2);
    System.out.println("Concatenando com Concat:" +string1.concat(string2));
}
```

Saída:

Concatenando com +: Feliz Aniversário
Concatenando com Concat: Feliz Aniversário

MÉTODOS DA CLASSE STRING

- toLowerCase( )
  - Converte o String chamador em letras minúsculas.

- toUpperCase( )
  - Converte o String chamador em letras maiúsculas.

- Obs.: Em ambos, a String original permanece inalterada. Se não houver caracteres para converter, ele retorna a String original.

EXEMPLO

```
public static void main(String[] args)
  {
     String s1 = "hello";
     String s2 = "GOODBYE";
      System.out.printf("s1.toUpperCase() = "+ s1.toUpperCase());
     System.out.printf("s2.toLowerCase() = "+ s2.toLowerCase());
     System.out.println();
  }
```

Saída:

```
s1.toUpperCase() = HELLO
s2.toUpperCase() = goodbye
```

MÉTODOS DA CLASSE STRING

- ToCharArray( )
  - Copia os caracteres do string chamador para um array de caracteres.
  - Se o string chamador for vazio, o array retornado será vazio.

EXEMPLO

```
public static void main(String[] args)
  {
     String s1 = "hello";
     char[] charArray = s1.toCharArray();
     System.out.print("s1 como um array de caracteres = ");

     for (char caracter :  charArray)
        System.out.print(caracter);
      System.out.println();
  }
```

Saída: s1 como um array de caracterres = hello

MÉTODOS DA CLASSE STRING

- replace( )
  - Retorna uma nova cadeia de caracteres na qual todas as ocorrências de uma cadeia de caracteres especificada na instância atual são substituídas por outra cadeia de caracteres especificada.
  - Se não for encontrado na instância atual, o método retornará a instância atual inalterada.
  - O método replace deixa a String original inalterada.

EXEMPLO

```
public static void main(String[] args)
  {
    String s1 = "hello";
    System.out.print("Substitua 'l' with 'L' in s1: ", s1.replace('l', 'L'));
    System.out.println();
  }
```

Saída: Substitua 'l' with 'L' in s1: heLLo

MÉTODOS DA CLASSE STRING

- trim( )
  - Utilizado para gerar um novo objeto String que remove todos os caracteres de espaço em branco que aparecem no início ou no final da String que trim opera.
  - Retorna um novo objeto contendo a String sem espaços em branco iniciais ou finais.
  - Se não houver caracteres de espaços em branco, no início e/ou fim, o método retorna a String original.
  - O método deixa a String original inalterada.

EXEMPLO

```
public static void main(String[] args)
   {
      String s3 = "     spaces     ";


      System.out.print("s3 após trim = \" + s3.trim()+ \" ",);
      System.out.println();
}
```

Saída: s3 após trim = “spaces”

MÉTODOS DA CLASSE STRING

- valueOf( )
  - método que aceita um argumento de qualquer tipo e o converte em um objeto String.

String.valueOf(argumento)

EXEMPLO

Saída:

```
char array = abcdef
parte de um array de char = def
boolean = true
char = Z
int = 7
double = 33.333
```

```
public static void main(String[] args)
  {
    char[] charArray = {'a', 'b', 'c', 'd', 'e', 'f'};
    boolean booleanValue = true;
    char characterValue = 'Z';
    int integerValue = 7;
    double doubleValue = 33.333;
    System.out.println("char array = "+  String.valueOf(charArray));
    System.out.println("parte de um array de char= "+ String.valueOf(charArray, 3, 3));
    System.out.println ("boolean = "+ String.valueOf(booleanValue));
    System.out.println ("char = "+ String.valueOf(characterValue));
    System.out.println ("int =" +  String.valueOf(integerValue));
    System.out.println ("double =" + String.valueOf(doubleValue));
  }
```

MÉTODOS DA CLASSE STRING

- Split( )

  -Divide uma String em subcadeias de caracteres baseadas nos caracteres de um array.

  -O método retorna um array de strings, contendo as respectivas partes, definidas de acordo com a string passada como parâmetro.

EXEMPLO

```
public static void main(String[] args)
  {
        String valor = "DEVMEDIA - Java";
        String[] valorComSplit = valor.split("-");

        for(String s : valorComSplit)
        {
            System.out.println(s);
        }
  }
```

Saída:

DEVMEDIA

Java

MÉTODOS DA CLASSE STRING

- contains( )

  O método não tem como retorno uma nova String, mas sim um boolean. Ele avalia se a String original contém a String passada como parâmetro para o método.

  É também *Case Sensitive*.

EXEMPLO

```
public static void main(String[] args)
  {
     String s = "001MARCOS PAULO     M19803112";
     System.out.println(s.contains("MARCOS"));
   }
```

Saída:

```
true
```

EXEMPLO

- Faça um programa que receba uma frase, calcule e mostre a quantidade de vogais da frase digitada. O programa deverá contar vogais maiúsculas e minúsculas.

```
public static void main(String args[])
  {
      String frase;
      int tam, i, qtde= 0;
      Scanner entrada = new Scanner(System.in);
      System.out.println("Digite uma frase");
      frase = entrada.nextLine();
      tam = frase.length();

      for (i=0; i < tam; i++)
      {
          if (frase.charAt(i) == 'a' || frase.charAt(i) == 'A'
              || frase.charAt(i) == 'e' || frase.charAt(i) == 'E'
              || frase.charAt(i) == 'i' || frase.charAt(i) == 'I'
              || frase.charAt(i) == 'o' || frase.charAt(i) == 'O'
              || frase.charAt(i) == 'u' || frase.charAt(i) == 'U')
          {
              qtde = qtde + 1;
          }
      }
      System.out.println("Quantidade de vogais = "+qtde);
  }
```

EXERCÍCIOS

1) Construir um programa que conte e imprima o número de ocorrências de uma letra, fornecida pelo usuário, em uma dada string, também digitada pelo usuário.

2) Construa um programa que:

i) Peça para o usuário digitar uma string *s*.

ii) Peça para o usuário digitar um caractere *ch1*.

iii) Peça para o usuário digitar um caractere *ch2*.

O programa deve substituir todas as ocorrências do caractere *ch1* em s pelo caractere *ch2*.

3) Construir um programa que seja capaz de concatenar uma string *s1* e uma outra string *s2* em uma string *s3*. Por exemplo:

Digite a stirng s1: Quem canta os males espanta.

Digite a string s2: Há males que vêm para o bem.

Nova string s3: Quem canta os males espanta. Há males que vêm para o bem.

AGRADECIMENTO

- À professora Soraia Lúcia pela cessão do material deste guia de aula