Capítulo – 11

11.1 OPERAÇÕES COM `String`

`String` é um tipo de texto que corresponde à união de um conjunto de caracteres. Em Java, uma variável do tipo `String` é uma instância da classe **`String`**, isto é, gera objetos que possuem propriedades e métodos, diferentes. A exemplo disso é o método **`equals()`** que utilizamos no capítulo 7 no item 7.7. Portanto o tipo `String` é diferente dos tipos primitivos como **`int`**, **`float`**, **`double`**, etc. Este assunto será estudado mais adiante no capítulo 15 (orientação à objetos).

Os valores referenciados nas Strings podem ser manipulados de várias formas.

Por exemplo, é possível verificar seu comprimento, retirar uma parte do valor contido na **`String`**, acessar ou mudar caracteres individuais. As **`Strings`** constituem uma cadeia de caracteres entre aspas.

Exemplo: `frase = "Linguagem Java".` Da mesma forma como no capítulo anterior aprendemos a manipulação da classe `Math` que manipula as operações matemáticas, existem diversos métodos para manipulação de `String`.

Forma de uso da classe `String`:

```
<Nome_da_String>.<nome_do_método>(<argumentos>)
```

Abaixo veremos os métodos mais comuns da classe String:

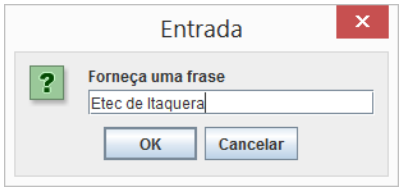
11.2 – MÉTODO length

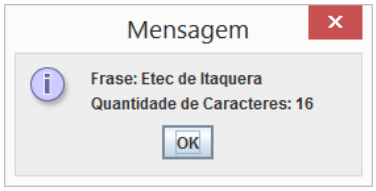
Este importante método é utilizado toda vez que desejamos obter o tamanho do conteúdo de uma `String` incluído também os valores em branco. Este método sempre retorna um valor do tipo int.

```
<Nome_da_String>.length()
```

Para podermos realizar a busca de palavras ou caracteres utilizamos o método `length` em um algoritmo. Veja o exemplo abaixo:

```
3  import javax.swing.JOptionPane;
4
5  public class StringTamanhoDaFrase {
6
7      public static void main(String[] args) {
8          String frase = JOptionPane.showInputDialog("Forneça uma frase");
9          int tamanho = frase.length();
10         JOptionPane.showMessageDialog(null, "Frase: "+ frase +
11                                       "\nQuantidade de Caracteres: "+tamanho);
12
13     }
14
15 }
```





11.3-MÉTODO `charAt`

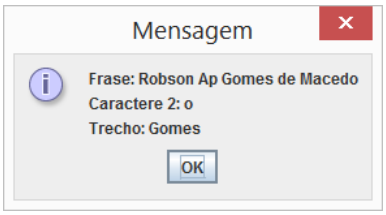
O método `charAt` é usado para retornar um caractere de uma `string` de acordo com um índice especificado entre parênteses. Esse índice refere-se à posição do caractere na `String`, sendo 0 (zero) o índice do primeiro caractere, 1 (um) o do segundo e assim por diante. O método `charAt` é útil quando for necessário verificar a existência de um caractere na `String`. Por exemplo, suponha que uma determinada `String` só possa conter números, o método `charAt` pode ser usado para verificar a existência de dígitos numéricos nessa `String`.

A forma de uso do `charAt` é a seguinte:

```
<string>.charAt(<índice>)
```

Avalie o exemplo abaixo:

```
3  import javax.swing.JOptionPane;
4
5  public class StringCharAt {
6
7      public static void main(String[] args) {
8          String frase =  JOptionPane.showInputDialog("Forneça uma frase");
9          String trecho = "";
10         for(int i = 10;i<=15;i++){
11             trecho += frase.charAt(i);
12         }
13
14         JOptionPane.showMessageDialog(null, "Frase: " + frase +
15                 "\nCaractere 2: " + frase.charAt(1) +
16                 "\nTrecho: " + trecho);
17     }
18 }
```



11.4 – BRINCANDO COM OS MÉTODOS `length` e `charAt`

Para fixar na sua memória os métodos `string.length` e `string.charAt` vamos construir um banner que exibe cada um dos caracteres de uma frase previamente inserida pelo usuário.

```
3  import javax.swing.JOptionPane;
4
5  public class Banner {
6      public static void main(String args[]) {
7          int cont = 0;
8          String palavra = JOptionPane.showInputDialog("Forneça uma palavra");
9          if (palavra != null) {  // se o usuario digitou algo
10             System.out.println("Mostra a palavra 3 vezes, letra a letra.");
11             while (cont++ < 3){ // looping 3 vezes
12                 for (int i = 0; i < palavra.length(); i++) {
13                     System.out.print(palavra.charAt(i)); // varre os caracteres
14                     for (int x = 0; x < 999999999; x++); //temporizador
15                 }
16                 System.out.println();
17             }
18             System.out.println();
19             System.out.println("Banner encerrado");
20         } else {
21             System.out.println("Entre com uma palavra qualquer.");
22         }
23     }
24 }
```

Na linha 14 foi criado um for que conta até 999.999.999, isto foi uma adaptação para atrasar em alguns milésimos de segundos para que o computador exiba cada caractere pausadamente. Você poderá alterar este valor para mais ou para menos dependendo do atraso desejado.

Execute esta classe e divirta-se com este efeito!

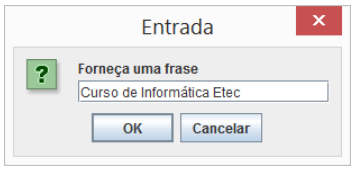
11.5 - MÉTODO `toUpperCase` e `toLowerCase`

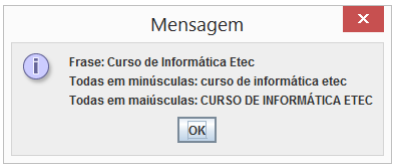
Os métodos to `UpperCase` e to `LowerCase` são utilizados para transformar todas as letras de uma determinada string em maiúsculas ou minúsculas. O método `toUpperCase` transforma todos os caracteres de uma `String` em maiúsculos. O método `toLowerCase` transforma todos os caracteres de uma `String` em minúsculos. Sua forma de uso é a seguinte:

```
<string>.toUperCase()   = deixa os caracteres em MAIÚSCULOS
<string>.toLowerCase()  = deixa os caracteres em minúsculos
```

Abaixo teremos um exemplo que será rapidamente entendido por você aluno. Porém vale ressaltar que os método `toUpperCase` e `toLowerCase` não alteram o valor original da `String`.

```
3 import javax.swing.JOptionPane;
4
5 public class ToUpperCaseToLowerCase {
6
7     public static void main(String[] args) {
8         String frase = JOptionPane.showInputDialog("Forneça uma frase");
9
10        JOptionPane.showMessageDialog(null, "Frase: " + frase +
11                                      "\nTodas em minúsculas: " + frase.toLowerCase() +
12                                      "\nTodas em maiúsculas: " + frase.toUpperCase());
13    }
14
15 }
```





Lembre-se, a alteração para maiúsculo ou minúsculo ocorre apenas para fins de impressão em tela. Se for necessário alterar o conteúdo da variável string, substituindo seu valor original pelo transformado, a própria variável deverá receber o valor desta transformação, por exemplo, `palavra1 = palavra.toLowerCase()`.

11.5 MÉTODO `substring`

O método `substring` retorna a cópia de caracteres de uma `String` a partir de índices inteiros especificados.

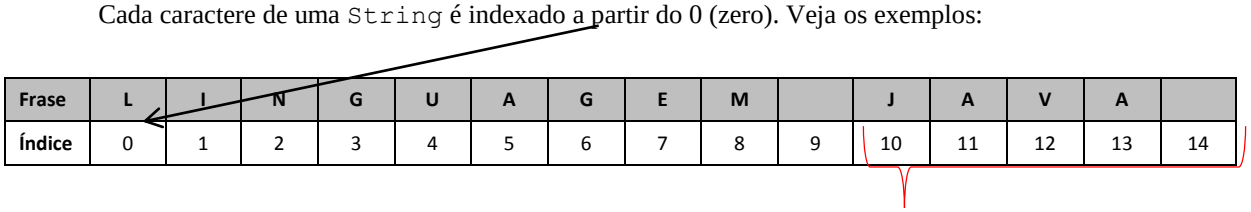
A forma de uso é:

```
<string>.substring(<índice_inicial>,[<índice_final>])
```

O primeiro argumento especifica o índice do início da cópia dos caracteres da `string` (da mesma forma que `charAt`), em qualquer `String` o índice inicia-se em 0 (zero). O segundo argumento é opcional e define o índice final, em que termina a cópia dos caracteres. Entretanto o índice final deve ser um índice além do último caractere.

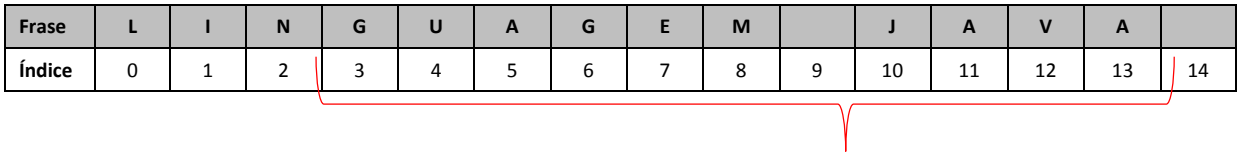
Para entendermos melhor, considere a variável `String` chamada frase com o seguinte conteúdo: "Linguagem Java".

Cada caractere de uma `String` é indexado a partir do 0 (zero). Veja os exemplos:



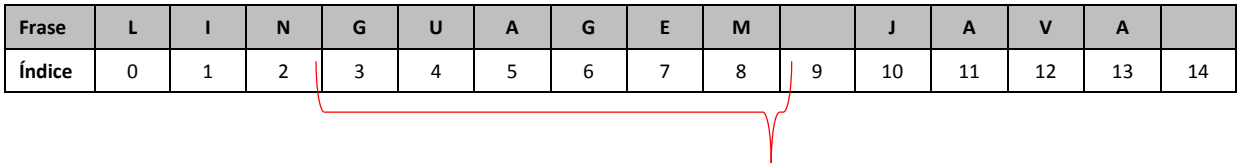
Frase	L	I	N	G	U	A	G	E	M		J	A	V	A	
Índice	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

1 – `String x = frase.substring(10)`, Ou seja, x recebe o conteúdo **"JAVA",** pois ao passar apenas o primeiro argumento para o método substring, ele retorna o caractere a partir do índice informado (no caso 10, aposição J) **até o último caractere da String**.



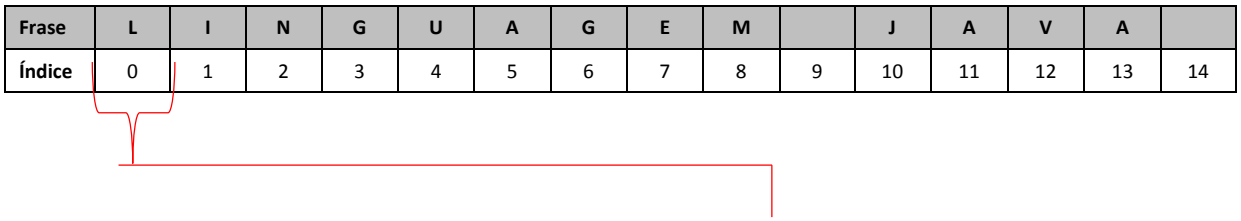
Frase	L	I	N	G	U	A	G	E	M		J	A	V	A	
Índice	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

2 – `String x = frase.substring(3)`, recebe o conteúdo "**GUAGEM JAVA**", isto e, do caractere do índice 3 até o último caractere da String frase.



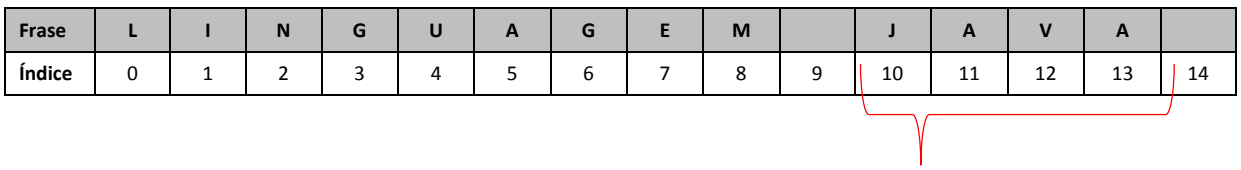
Frase	L	I	N	G	U	A	G	E	M		J	A	V	A	
Índice	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

3 – `String x = frase.substring(3,9)`, recebe o conteúdo "**GUAGEM**", isto é, do caractere de índice 3 até o caractere de índice 8.



Frase	L	I	N	G	U	A	G	E	M		J	A	V	A	
Índice	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

4 – `String x = frase.substring(0,1)`, recebe o conteúdo "**L**", isto é, do caractere de índice 0 até o caractere de índice 0. Isto mesmo, como precisamos apenas da letra L devemos informar além do índice do caractere desejado. Se informarmos (0,0) não teríamos caractere algum.

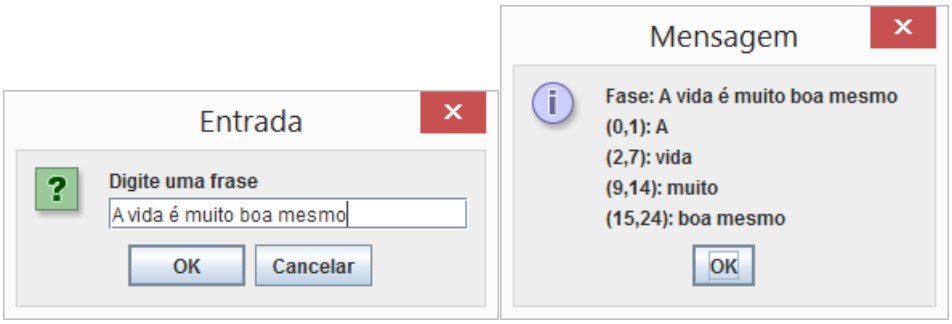


Frase	L	I	N	G	U	A	G	E	M		J	A	V	A	
Índice	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

5 – `String x = frase.substring(10,14)`, ou seja, recebe o conteúdo "**JAVA**", isto é, do caractere de índice 10 até o caractere de índice 13.

Observe este exemplo prático:

```
3 import javax.swing.JOptionPane;
4
5 public class Substring {
6
7     public static void main(String[] args) {
8         try {
9             String frase =  JOptionPane.showInputDialog("Digite uma frase");
10
11            JOptionPane.showMessageDialog(null, "Fase: " + frase +
12                    "\n(0,1): "+frase.substring(0,1) +
13                    "\n(2,7): "+frase.substring(2,7) +
14                    "\n(9,14): "+ frase.substring(9,14) +
15                    "\n(15,24): "+ frase.substring(15, 24));
16
17            }catch (StringIndexOutOfBoundsException erro) {
18                JOptionPane.showMessageDialog(null, "A frase deve conter pelo menos 23 caracteres");
19            }
20    }
21 }
```



11.6 – MÉTODO `trim` –"podar/aparar"

Um problema no desenvolvimento de sistemas de bom desempenho e confiabilidade está no fato de manter a consistência dos dados. Imagine uma tela de cadastro onde o usuário ao digitar o nome de uma pessoa, inicie alinha colocando espaços em branco. No pior dos cenários teríamos problemas ao executar um comando de busca de informações para a futura localização deste registro. Imagine ainda se outro usuário do sistema realizasse outro cadastro da mesma pessoa, mas com dois ou três espaços em branco. Para evitar este tipo de situação é que podemos fazer uso do método trim. Sua função é retirar os espaços em branco existente dentro de uma `String` antes e depois do texto. Lembro que este método apenas altera a visualização do conteúdo da `String`, mas podemos nos utilizar do artifício: `frase = frase.trim().` Isso tornaria a variável livre de espaços em branco.

Veja o exemplo:

```
3 import javax.swing.JOptionPane;
4
5 public class StringTrim {
6
7     public static void main(String[] args) {
8         String frase =  JOptionPane.showInputDialog("Entre com a frase: ");
9
10        JOptionPane.showMessageDialog(null, "Com espaços:" + frase+
11                "\nFrase exibida sem espaços antes e depois:"+frase.trim());
12
13
14        frase = frase.trim();
15
16        JOptionPane.showMessageDialog(null, "Variável frase remontada sem espaços antes e depois: "+frase.trim());
17
18    }
19
20 }
```

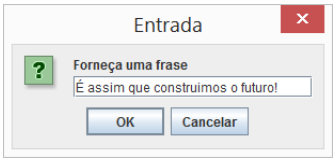
11.7 MÉTODO `replace`

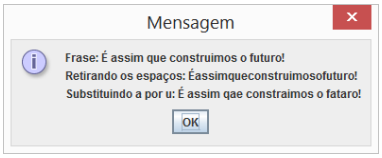
O método `replace` é utilizado para substituição de caracteres, ou grupo de caracteres, em uma determinada String. Para isso é necessário informar o(s) caractere(s) que deseja substituir e por qual(is) caractere(s) ele(s) será(ão) substituído(s). Caso não haja na String nenhuma ocorrência do caractere a ser substituído, a String original é retornada, isto é, não ocorre nenhuma alteração.

Veja sua forma de uso:

```
<string>.replace(<caracteres_a_serem_substituidos>,<substituição>>
```

```
3 import javax.swing.JOptionPane;
4
5 public class StringReplace {
6
7     public static void main(String[] args) {
8         String frase =  JOptionPane.showInputDialog("Forneça uma frase");
9
10        JOptionPane.showMessageDialog(null, "Frase: "+frase +
11                                      "\nRetirando os espaços: " + frase.replace(" ", "") +
12                                      "\nSubstituindo a por u: " + frase.replace("u", "a"));
13    }
14
15 }
```





11.8 MÉTODO `replaceAll`

O método replaceAll se utiliza das expressões regex (regular expressions) muito utilizado na construção de compiladores. Sendo assim utilizaremos o método `trim()` juntamente com o método replaceAll para garantir que se o usuário digitar nomes com vários espaços antes, depois e entre as palavras somente tenhamos a frase sem espaços antes, depois e somente um espaço entre as palavras.

A forma de uso do método replaceAll é:

```
<string>.replaceAll(<valor a ser procurado>,<novo valor a ser atribuído>)
```

Vale apena lembrar que o método `replaceAll` é mais flexível do que o replace pois podemos inserir uma gama de regular expressions em seus parâmetros. A ciência das expressões regulares foge dos objetivos deste material mas poderá ser mais explorado em http://aurelio.net/regex/

```
3 public class StringTrimReplaceAllTudo {
4
5     public static void main(String[] args) {
6         String frase = "    É   Chegado   o    dia   ";
7
8         System.out.println("Frase armazenada na variável:"+frase);
9
10        frase = frase.replaceAll("\u0020++"," ").trim();;
11
12        System.out.println("Frase após replaceALL + trim:"+frase);
13
14    }
15
16 }
```

```
Console
<terminated> StringTrimReplaceAllTudo [Java Application] C:\Program Files (x86)\Java\jre1.8.0_31\bin\javaw.
Frase armazenada na variável:    É   Chegado   o    dia
Frase após replaceALL + trim:É Chegado o dia
```

Indica que o replaceAll irá procurar por "\u0020"(espaço) dentro da string e o ++ indica que se houver mais de um espaço.

Dica! Para entender o \u0020, experimente digitar:
`System.out.print("Ass\u0020im que se faz");`
E veja o resultado no console!

O replaceAll irá trocar tudo por um espaço apenas.

11.9 – MÉTODO `ValueOf`

No capítulo 04 no ítem 4.5 vimos este método que é usado para converter tipos de dados em String. Esse método aceita vários tipos de argumentos(números ou cadeia de caracteres). Uma das formas de uso para o método valueOf é:

String.valueOf(<nome_da_variável_a_ser_convertida>);

Demonstrarei abaixo alguns tipos de conversão de tipos numéricos.

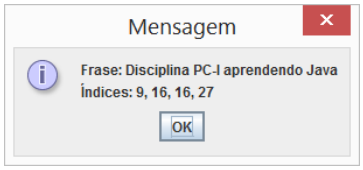
```
3 import javax.swing.JOptionPane;
4
5 public class StringValueOf {
6
7     public static void main(String[] args) {
8         int a = 1000;
9         long b = 50000;
10        float c = 20.45f;
11        double d = 15.432;
12        String x = String.valueOf(a) +" "+ String.valueOf(b) +" "+ String.valueOf(c) +" "+  String.valueOf(d);
13
14        JOptionPane.showMessageDialog(null, x);
15
16    }
17
18 }
```

11.10 MÉTODO `indexOf`

Com este método conseguimos saber qual é a posição de um determinado caractere procurado. Seria semelhante a procurar uma palavra em um texto utilizando por exemplo o `if` ou `switch case`. Porém com o `indexOf` passamos como o valor procurado um caractere. Caso haja sucesso na busca, é retornado um número inteiro referente à posição do texto (o índice) onde o caractere foi encontrado, ou a posição do texto onde se inicia a substring localizada. Caso o `indexOf` não encontre o caractere procurado ele retornará o valor inteiro (-1). De qualquer forma, o retorno de `indexOf` sempre será um número inteiro (o valor do índice encontrado ou -1). Sua forma de uso é a seguinte:

`String.indexOf(<caractere ou string a ser localizada>, [posição inicial>])`

```
3 import javax.swing.JOptionPane;
4
5 public class StringIndexOF {
6
7     public static void main(String[] args) {
8         String frase = "Disciplina PC-I aprendendo Java";
9
10        char caractere = 'a';
11
12        JOptionPane.showMessageDialog(null, "Frase: " + frase + "\nÍndices: " + frase.indexOf(caractere) + ", " +
13                                                                                  frase.indexOf(caractere, 10) + ", "+
14                                                                                  frase.indexOf("aprendendo")+", "+
15                                                                                  frase.indexOf("Java", 15));
16
17    }
18
19 }
```



Outro exemplo. Busca utilizando o `indexOf`:

```
import javax.swing.JOptionPane;

public class Pesquisa {
    public static void main(String[] args) {
        String frase = JOptionPane.showInputDialog("Forneça uma frase");
        String palavra = JOptionPane.showInputDialog("Forneça uma palavra");
        int pos = 0;
        int qtde = 0;
        while (true){
            pos = frase.indexOf(palavra,pos);
            if (pos!=-1){
                qtde++;
                pos++;
            }
            else
                break;
        }
        JOptionPane.showMessageDialog(null,
            "\nFrase fornecida: " + frase +
            "\nPalavra fornecida: " + palavra +
            "\nQuantidade de Ocorrências: " + qtde);
    }
}
```

11.11 Método Concat

O método .concat retorna a junção de duas strings.

Veja a forma de uso:

```
String.concat("frase para ser adicionada");
```

```
public class StringConcat {

    public static void main(String[] args) {
        String frase = "Plataforma Java é bom ";

    System.out.println(frase.concat("demais"));

    }
}
```

```
Console
<terminated> StringConcat [Java Application] C:\Program Files (x86)\Java\jre1.8.
Plataforma Java é bom demais
```

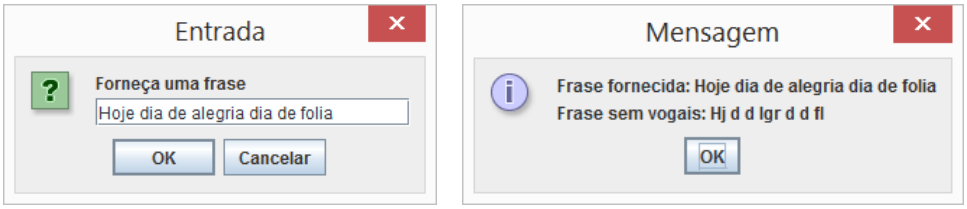
11.12 – Método Split

Este método é utilizado para separar valores de uma String. Também chamado de Tokenização ou token. Exemplo imagine a seguinte String “XHTML; Java; CSharp; Visual Basic” na verdade temos uma String com todas estas palavras mas perceba que elas estão separadas por um ; “ponto-e-vírgula” o que nos permite utilizar este sinal como separador e criar um vetor com cada uma destas palavras separadas por um índice.

Este método será demostrado melhor no capítulo 13 no ítem 13.11.

ATIVIDADES – 11

1) Construa uma classe que receba uma frase qualquer e mostre esta frase sem nenhuma vogal como indica o resultado abaixo.



2) Altere o exemplo do tópico **11.4** deste capítulo para fazer a mesma coisa porém com a classe **`Thread`** ao invés do for com o valor de i=999.999.999.

3) Construa uma classe que receba uma frase qualquer e mostre-a de forma invertida.

4) Elabore uma classe que mostre o efeito representado na figura abaixo para uma determinada palavra que o usuário digitar.