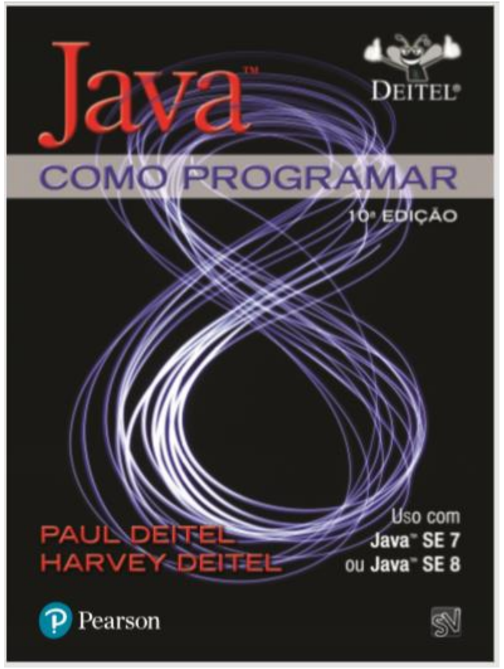


Java

Notas de Aula

Prof. André Bernardi
andrebernardi@unifei.edu.br

Setembro - 21

Strings e Caracteres

Strings e Caracteres

Os objetivos desta aula são:

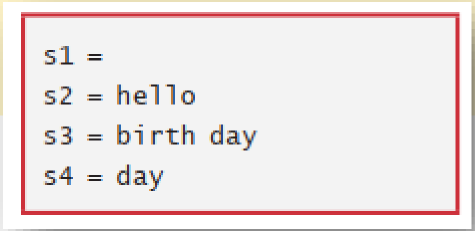
- Criar e manipular conjuntos de caracteres não modificáveis, objetos da classe ***String*** ;
- Criar e manipular conjuntos de caracteres modificáveis, objetos da classe ***StringBuffer*** e ***StringBuilder*** ;
- Criar e manipular Objetos da classe ***Character*** ;
- Compreender a utilização da classe ***StringTokenizer*** ;

Classe **String** - Construtores

```
String();
String( String );
String( charArray );
String( charArray, inicio, numero);
String( byteArray, inicio, numero);
String ( StringBuffer );
```

Construtores - String

```java
// Figura 14.1: StringConstructors.java
// construtores da classe String.

public class StringConstructors
{
      public static void main(String[] args)
      {
            char[] charArray = {'b', 'i', 'r', 't', 'h', ' ', 'd', 'a', 'y'};
            String s = new String("hello");

            // utiliza os construtores String
            String s1 = new String();
            String s2 = new String(s);
            String s3 = new String(charArray);
            String s4 = new String(charArray, 6, 3);

            System.out.printf(
                  "s1 = %s%ns2 = %s%ns3 = %s%ns4 = %s%n", s1, s2, s3, s4);
      }
} // fim da classe StringConstructors
```

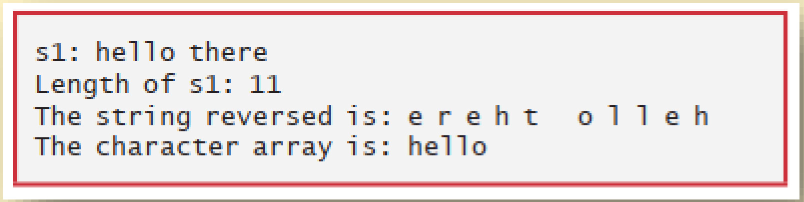


```
s1 =
s2 = hello
s3 = birth day
s4 = day
```

Classe **String**
Métodos *length, charAt, getChars*

```
long length()

char charAt(posicao)

void getChars(inicio, fim,

  charArray, inicio)
```

```java
// Figura 14.2: StringMiscellaneous.java
// Esse aplicativo demonstra os métodos da classe String
// length, charAt e getChars.

public class StringMiscellaneous
{
    public static void main(String[] args)
    {
        String s1 = "hello there";
        char[] charArray = new char[5];

        System.out.printf("s1: %s", s1);

        // testa o método length
        System.out.printf("%nLength of s1: %d",  s1.length());

// faz loop pelos caracteres em s1 com charAt e os exibe na ordem inversa
        System.out.printf("%nThe string reversed is: ");

        for (int count = s1.length() - 1; count >= 0; count--)
            System.out.printf("%c ",  s1.charAt(count));

        // copia caracteres a partir de string para charArray
        s1.getChars(0, 5, charArray, 0);
        System.out.printf("%nThe character array is: ");

        for (char character : charArray)
            System.out.print(character);

        System.out.println();
    }
} // fim da classe StringMiscellaneous
```



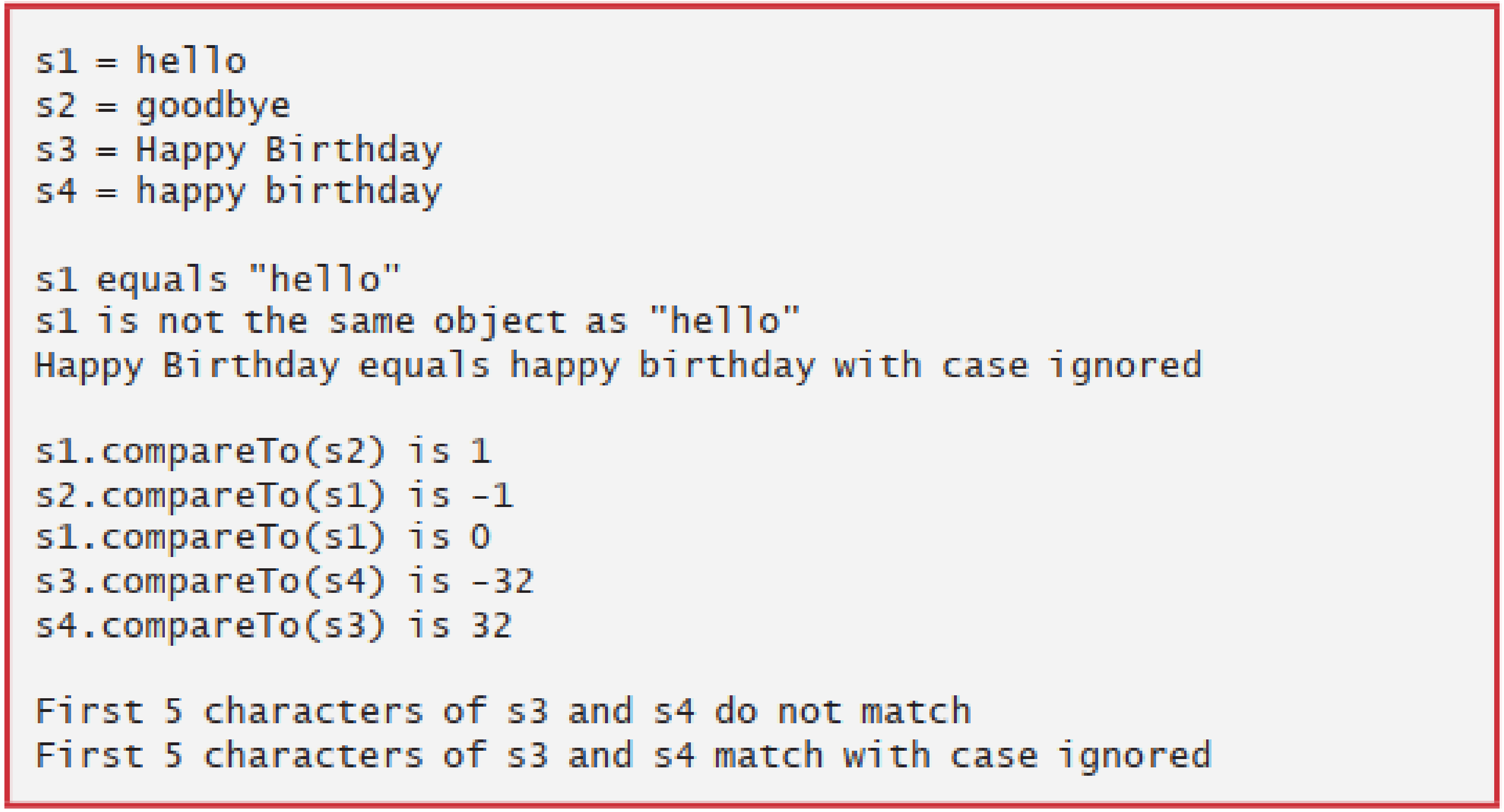
```
s1: hello there
Length of s1: 11
The string reversed is: e r e h t   o l l e h
The character array is: hello
```

Classe **String** Métodos de Comparação

```
boolean equals(String); // true se iguais
boolean equalsIgnoreCase(String);
int compareTo(String);  //( <0, ==0, >0 )
boolean regionMatches(inicio, string,
  inicio, cont);              // true se iguais
boolean String.regionMatches(caseIgnore,
  inicio, string, inicio, cont);


boolean String.startWith(String, offset);
boolean String.endWith(String);
```

```java
// Figura 14.3: StringCompare.java
// Métodos String equals, equalsIgnoreCase, compareTo e regionMatches.

public class StringCompare
{
     public static void main(String[] args)
     {
         String s1 = new String("hello"); // s1 é uma cópia de "hello"
         String s2 = "goodbye";
         String s3 = "Happy Birthday";
         String s4 = "happy birthday";

         System.out.printf(
            "s1 = %s%ns2 = %s%ns3 = %s%ns4 = %s%n%n", s1, s2, s3, s4);

         // teste para igualdade
         if ( s1.equals("hello")) // true
            System.out.println("s1 equals \"hello\"");
         else
            System.out.println("s1 does not equal \"hello\"");

         // testa quanto à igualdade com ==
         if ( s1 == "hello") // false; eles não são os mesmos objetos
            System.out.println("s1 is the same object as \"hello\"");
         else
            System.out.println("s1 is not the same object as \"hello\"");

         // testa quanto à igualdade (ignora maiúsculas e minúsculas)
         if ( s3.equalsIgnoreCase(s4)) // true
            System.out.printf("%s equals %s with case ignored%n", s3, s4);
         else
            System.out.println("s3 does not equal s4");
```

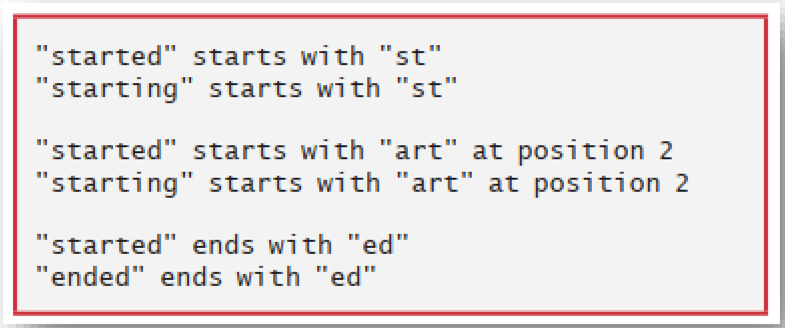
```java

      // testa compareTo
      System.out.printf(
            "%ns1.compareTo(s2) is %d",  s1.compareTo(s2));
      System.out.printf(
            "%ns2.compareTo(s1) is %d",  s2.compareTo(s1));
      System.out.printf(
            "%ns1.compareTo(s1) is %d",  s1.compareTo(s1));
      System.out.printf(
            "%ns3.compareTo(s4) is %d",  s3.compareTo(s4));
      System.out.printf(
            "%ns4.compareTo(s3) is %d%n%n",  s4.compareTo(s3));

      // testa regionMatches (distingue maiúsculas e minúsculas)
      if ( s3.regionMatches(0, s4, 0, 5))
         System.out.println("First 5 characters of s3 and s4 match");
      else
         System.out.println(
            "First 5 characters of s3 and s4 do not match");

      // testa regionMatches (ignora maiúsculas e minúsculas)
      if ( s3.regionMatches(true, 0, s4, 0, 5))
         System.out.println(
            "First 5 characters of s3 and s4 match with case ignored");
      else
         System.out.println(
            "First 5 characters of s3 and s4 do not match");
   }
} // fim da classe StringCompare
```



```
s1 = hello
s2 = goodbye
s3 = Happy Birthday
s4 = happy birthday

s1 equals "hello"
s1 is not the same object as "hello"
Happy Birthday equals happy birthday with case ignored

s1.compareTo(s2) is 1
s2.compareTo(s1) is -1
s1.compareTo(s1) is 0
s3.compareTo(s4) is -32
s4.compareTo(s3) is 32

First 5 characters of s3 and s4 do not match
First 5 characters of s3 and s4 match with case ignored
```

```java
// Figura 14.4: StringStartEnd.java
// métodos String startsWith e endsWith.

public class StringStartEnd
{
     public static void main(String[] args)
     {
         String[] strings = {"started", "starting", "ended", "ending"};

         // testa o método startsWith
         for (String string : strings)
         {
             if (string.startsWith( "st" ))
                 System.out.printf("\"%s\" starts with \"st\"%n", string);
         }

         System.out.println();

         // testa o método startsWith iniciando da posição 2 de string
         for (String string : strings)
         {
             if ( string.startsWith( "art" , 2))
                 System.out.printf(
                     "\"%s\" starts with \"art\" at position 2%n", string);
         }

```

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

        // testa o método endsWith
        for (String string : strings)
        {
            if ( string.endsWith("ed"))
                System.out.printf("\"%s\" ends with \"ed\"%n", string);
        }
    }
} // fim da classe StringStartEnd
```



```
"started" starts with "st"
"starting" starts with "st"

"started" starts with "art" at position 2
"starting" starts with "art" at position 2

"started" ends with "ed"
"ended" ends with "ed"
```

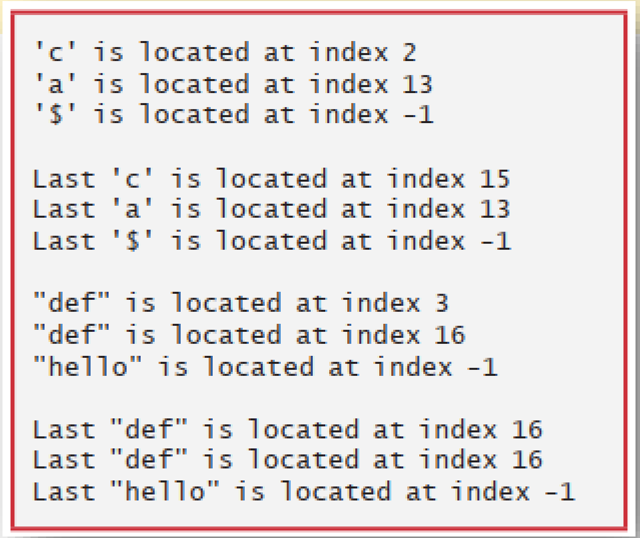
Classe **String**

Localizando caracteres e substrings

```
int indexOf(String);
int indexOf(String, inicio);
int indexOf(char);
int indexOf(char, inicio);

int lastIndexOf(String);
int lastIndexOf(String, inicio);
int lastIndexOf(char );
int lastIndexOf(char, inicio);
```

```java
// Figura 14.5: StringIndexMethods.java
// Métodos de pesquisa de String indexOf e lastIndexOf.

public class StringIndexMethods
{
 public static void main(String[] args)
 {
     String letters = "abcdefghijklmabcdefghijklm";

     // testa indexOf para localizar um caractere em uma string
     System.out.printf(
           "'c' is located at index %d%n",  letters.indexOf('c'));
     System.out.printf(
           "'a' is located at index %d%n",  letters.indexOf('a', 1));
     System.out.printf(
           "'$' is located at index %d%n%n",  letters.indexOf('$'));

     // testa lastIndexOf para localizar um caractere em uma string
     System.out.printf("Last 'c' is located at index %d%n",
             letters.lastIndexOf('c'));
     System.out.printf("Last 'a' is located at index %d%n",
             letters.lastIndexOf('a', 25));
     System.out.printf("Last '$' is located at index %d%n%n",
             letters.lastIndexOf('$'));

```

```java
      // testa indexOf para localizar uma substring em uma string
      System.out.printf("\"def\" is located at index %d%n",
              letters.indexOf("def"));
      System.out.printf("\"def\" is located at index %d%n",
              letters.indexOf("def", 7));
      System.out.printf("\"hello\" is located at index %d%n%n",
              letters.indexOf("hello"));

      // testa lastIndexOf para localizar uma substring em uma string
      System.out.printf("Last \"def\" is located at index %d%n",
              letters.lastIndexOf("def"));
      System.out.printf("Last \"def\" is located at index %d%n",
              letters.lastIndexOf("def", 25));
      System.out.printf("Last \"hello\" is located at index %d%n",
              letters.lastIndexOf("hello"));
   }
} // fim da classe StringIndexMethods
```



```
'c' is located at index 2
'a' is located at index 13
'$' is located at index -1

Last 'c' is located at index 15
Last 'a' is located at index 13
Last '$' is located at index -1

"def" is located at index 3
"def" is located at index 16
"hello" is located at index -1

Last "def" is located at index 16
Last "def" is located at index 16
Last "hello" is located at index -1
```

Classe **String**

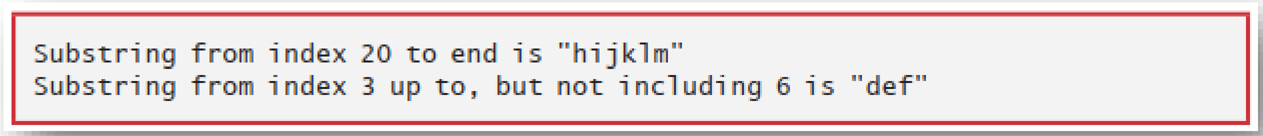
- Extraindo substrings

```
String substring(indice);
String substring(inicio,
  final);
```

- Concatenando strings

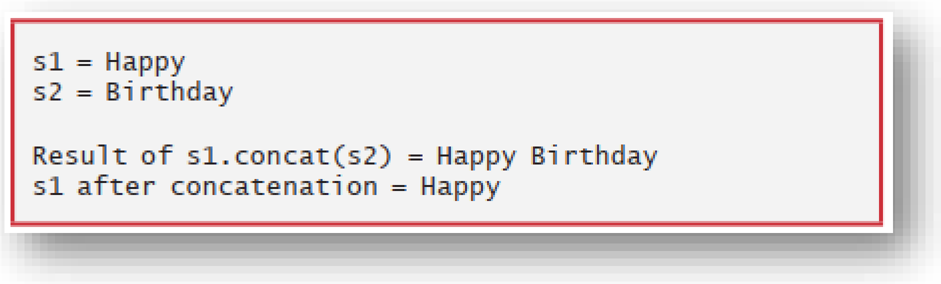
```
String String.concat( String );
```

```java
// Figura 14.6: SubString.java
// métodos substring da classe String.

public class SubString
{
 public static void main(String[] args)
 {
      String letters = "abcdefghijklmabcdefghijklm";

      // testa métodos substring
      System.out.printf("Substring from index 20 to end is \"%s\"%n",
                        letters.substring(20));
      System.out.printf("%s \"%s\"%n",
                    "Substring from index 3 up to, but not including 6 is",
                     letters.substring(3, 6));
 }
} // fim da classe Substring
```



```
Substring from index 20 to end is "hijklm"
Substring from index 3 up to, but not including 6 is "def"
```

```java
// Figura 14.7: StringConcatenation.java
// Método string concat.

public class StringConcatenation
{
  public static void main(String[] args)
  {
      String s1 = "Happy ";
      String s2 = "Birthday";

      System.out.printf("s1 = %s%ns2 = %s%n%n",s1, s2);
      System.out.printf(
             "Result of s1.concat(s2) = %s%n",  s1.concat(s2));
      System.out.printf("s1 after concatenation = %s%n", s1);
  }
} // fim da classe StringConcatenation
```



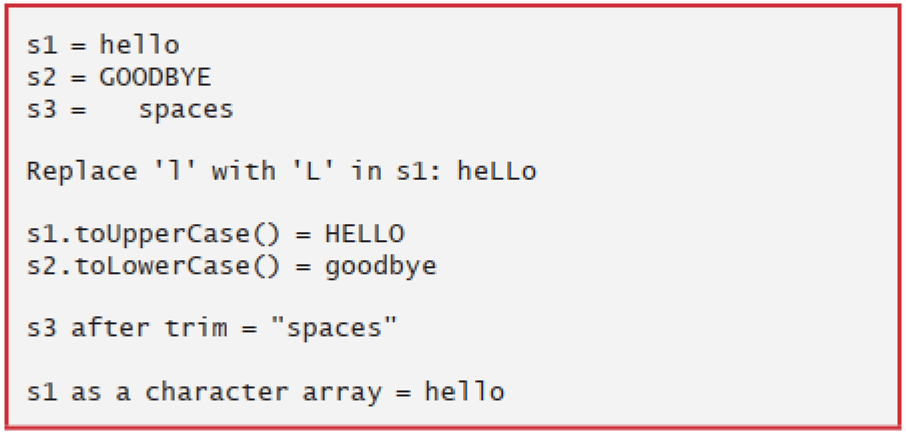
```
s1 = Happy
s2 = Birthday

Result of s1.concat(s2) = Happy Birthday
s1 after concatenation = Happy
```

Classe **String**
Métodos Variados

```
String replace(charFrom, charTo)
String toLowerCase()
String toUpperCase()
String trim()
String toString()
char[ ] toCharArray()
```

```java
// Figura 14.8: StringMiscellaneous2.java
// Métodos String replace, toLowerCase, toUpperCase, trim e toCharArray.

public class StringMiscellaneous2
{
 public static void main(String[] args)
 {
     String s1 = "hello";
     String s2 = "GOODBYE";
     String s3 = " spaces ";

     System.out.printf("s1 = %s%ns2 = %s%ns3 = %s%n%n", s1, s2, s3);

     // testa o método replace
     System.out.printf(
            "Replace 'l' with 'L' in s1: %s%n%n",  s1.replace('l', 'L'));

     // testa o toLowerCase e toUpperCase
     System.out.printf("s1.toUpperCase() = %s%n",  s1.toUpperCase());
     System.out.printf("s2.toLowerCase() = %s%n%n",  s2.toLowerCase());

     // testa o método trim
     System.out.printf("s3 after trim = \"%s\"%n%n",  s3.trim());

```

```java
      // testa o método toCharArray
       char[] charArray = s1.toCharArray();
      System.out.print("s1 as a character array = ");

      for (char character : charArray)
         System.out.print(character);

      System.out.println();
   }
} // fim da classe StringMiscellaneous2
```



```
s1 = hello
s2 = GOODBYE
s3 =   spaces

Replace 'l' with 'L' in s1: heLLo

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

s3 after trim = "spaces"

s1 as a character array = hello
```

Classe **StringBuffer**

- **Construtores**

```
StringBuffer();
StringBuffer(int);
StringBuffer(String);
StringBuffer(StringBuffer);
```

- Métodos **length, capacity**

```
StringBuffer.length();
StringBuffer.capacity();
```

Classe **StringBuffer**

```
StringBuffer.charAt(int);
StringBuffer.setCharAt(posicao, char);

StringBuffer.getChars(inicio, fim,
  destino, inicio);
StringBuffer.reverse();
```

Classe **StringBuffer**

```
StringBuffer.append(Object);
StringBuffer.append(String);
StringBuffer.append(char);
StringBuffer.append(char[]);
StringBuffer.append(char[], start, end );
StringBuffer.append(int);
StringBuffer.append(long);
StringBuffer.append(float);
StringBuffer.append(double);
```

Classe **StringBuffer**

```
StringBuffer.insert(pos,Object);
StringBuffer.insert(pos,String);
StringBuffer.insert(pos,char);
StringBuffer.insert(pos,charArray);
StringBuffer.insert(pos,int);
StringBuffer.insert(pos,long);
StringBuffer.insert(pos,float);
StringBuffer.insert(pos,double);
StringBuffer.insert(pos,String);

StringBuffer.deleteCharAt(pos);
StringBuffer.delete(inicio,fim);
```

Classe **StringBuilder**

- **Construtores**

```
StringBuilder();
StringBuilder(int);
StringBuilder(String);
StringBuilder(StringBuilder);
```

- Métodos **length, capacity**

```
StringBuilder.length();
StringBuilder.capacity();
```

```java
// Figura 14.10: StringBuilderConstructors.java
// Construtores StringBuilder.

public class StringBuilderConstructors
{
     public static void main(String[] args)
     {
         StringBuilder buffer1 = new StringBuilder();
         StringBuilder buffer2 = new StringBuilder(10);
         StringBuilder buffer3 = new StringBuilder("hello");

         System.out.printf("buffer1 = \"%s\"%n", buffer1);
         System.out.printf("buffer2 = \"%s\"%n", buffer2);
         System.out.printf("buffer3 = \"%s\"%n", buffer3);
     }
} // fim da classe StringBuilderConstructors
```

```
buffer1 = ""
buffer2 = ""
buffer3 = "hello"
```

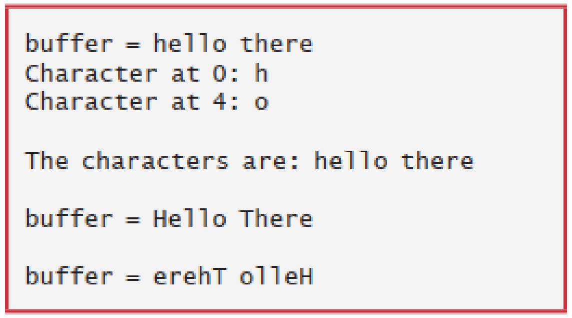
```java
// Figura 14.11: StringBuilderCapLen.java
// Métodos StringBuilder length, setLength, capacity e ensureCapacity.

public class StringBuilderCapLen
{
     public static void main(String[] args)
     {
        StringBuilder buffer = new StringBuilder("Hello, how are you?");

        System.out.printf("buffer = %s%nlength = %d%ncapacity = %d%n%n",
               buffer.toString(), buffer.length(), buffer.capacity());

         buffer.ensureCapacity(75);
        System.out.printf("New capacity = %d%n%n",  buffer.capacity());

         buffer.setLength(10));
        System.out.printf("New length = %d%nbuffer = %s%n",
                buffer.length(), buffer.toString());
     }
} // fim da classe StringBuilderCapLen
```

```
buffer = Hello, how are you?
length = 19
capacity = 35

New capacity = 75

New length = 10
buffer = Hello, how
```

Classe **StringBuilder**

```
StringBuilder.charAt(int);
StringBuilder.setCharAt(posicao, char);

StringBuilder.getChars(inicio, fim,
  destino, inicio);
StringBuilder.reverse();
```

```java
// Figura 14.12: StringBuilderChars.java
// Métodos StringBuilder charAt, setCharAt, getChars e reverse.

public class StringBuilderChars
{
    public static void main(String[] args)
    {
        StringBuilder buffer = new StringBuilder("hello there");

        System.out.printf("buffer = %s%n", buffer.toString());
        System.out.printf("Character at 0: %s%nCharacter at 4: %s%n%n",
            buffer.charAt(0),  buffer.charAt(4));

        char[] charArray = new char[buffer.length()];
        buffer.getChars(0, buffer.length(), charArray, 0);
        System.out.print("The characters are: ");

        for (char character : charArray)
            System.out.print(character);

        buffer.setCharAt(0, 'H');
        buffer.setCharAt(6, 'T');
        System.out.printf("%n%nbuffer = %s", buffer.toString());

        buffer.reverse();
        System.out.printf("%n%nbuffer = %s%n", buffer.toString());
    }
} // fim da classe StringBuilderChars
```



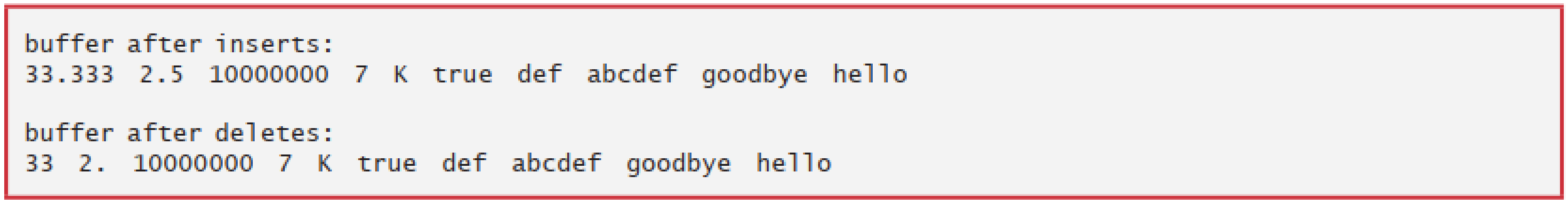
```
buffer = hello there
Character at 0: h
Character at 4: o

The characters are: hello there

buffer = Hello There

buffer = erehT olleH
```

Classe **StringBuilder**

```
StringBuilder.insert(pos,Object);
StringBuilder.insert(pos,String);
StringBuilder.insert(pos,char);
StringBuilder.insert(pos,charArray);
StringBuilder.insert(pos,int);
StringBuilder.insert(pos,long);
StringBuilder.insert(pos,float);
StringBuilder.insert(pos,double);
StringBuilder.insert(pos,String);

StringBuilder.deleteCharAt(pos);
StringBuilder.delete(inicio,fim);
```

```java
// Figura 14.14: StringBuilderInsertDelete.java
// Métodos StringBuilder insert, delete e deleteCharAt.

public class StringBuilderInsertDelete
{
     public static void main(String[] args)
     {
         Object objectRef = "hello";
         String string = "goodbye";
         char[] charArray = {'a', 'b', 'c', 'd', 'e', 'f'};
         boolean booleanValue = true;
         char characterValue = 'K';
         int integerValue = 7;
         long longValue = 10000000;
         float floatValue = 2.5f; // o sufixo f indica que 2.5 é um tipo float
         double doubleValue = 33.333;

         StringBuilder buffer = new StringBuilder();

          buffer.insert(0, objectRef);
          buffer.insert(0, "  "); // cada um desses contém dois espaços
          buffer.insert(0, string);
          buffer.insert(0, "  ");
          buffer.insert(0, charArray);
          buffer.insert(0, "  ");
          buffer.insert(0, charArray, 3, 3);
          buffer.insert(0, "  ");
          buffer.insert(0, booleanValue);
          buffer.insert(0, "  ");
          buffer.insert(0, characterValue);
          buffer.insert(0, "  ");
```

```java
        buffer.insert(0, integerValue);
        buffer.insert(0, " ");
        buffer.insert(0, longValue);
        buffer.insert(0, " ");
        buffer.insert(0, floatValue);
        buffer.insert(0, " ");
        buffer.insert(0, doubleValue);

        System.out.printf(
               "buffer after inserts:%n%s%n%n", buffer.toString());

        buffer.deleteCharAt(10); // exclui 5 em 2.5
        buffer.delete(2, 6); // exclui .333 em 33.333

        System.out.printf(
               "buffer after deletes:%n%s%n", buffer.toString());
    }
} // fim da classe StringBuilderInsertDelete
```



```
buffer after inserts:
33.333  2.5  10000000  7  K  true  def  abcdef  goodbye  hello

buffer after deletes:
33  2.  10000000  7  K  true  def  abcdef  goodbye  hello
```

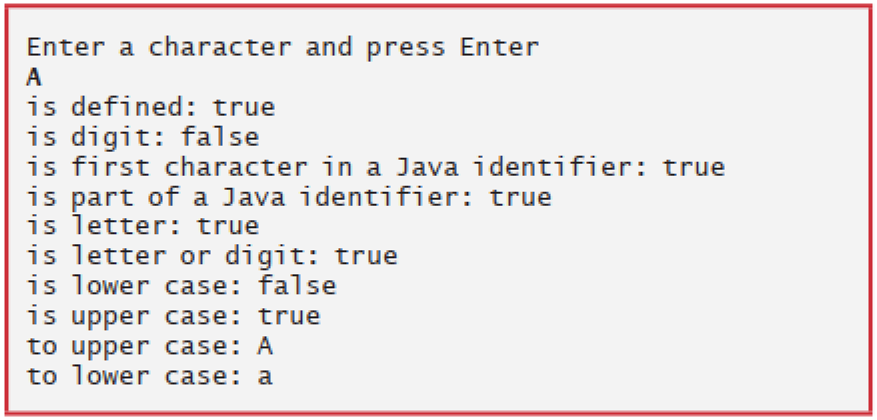
Classe **Character**

```
boolean Character.isDefined(char)
boolean Character.isJavaIdentifierStart(char)
boolean Character.isJavaIdentifierPart(char)
boolean Character.isDigit(char)
boolean Character.isLetter(char)
boolean Character.isLetterOrDigit(char)
boolean Character.isLowerCase(char)
boolean Character.isUpperCase(char)
boolean Character.isWhitespace(char)
```

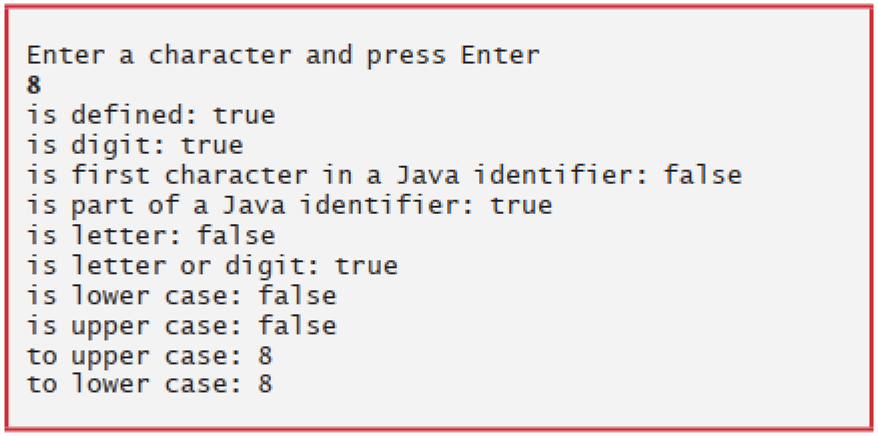
- Faixa de valores do char

```
int Character.MIN_VALUE
int Character.MAX_VALUE
```

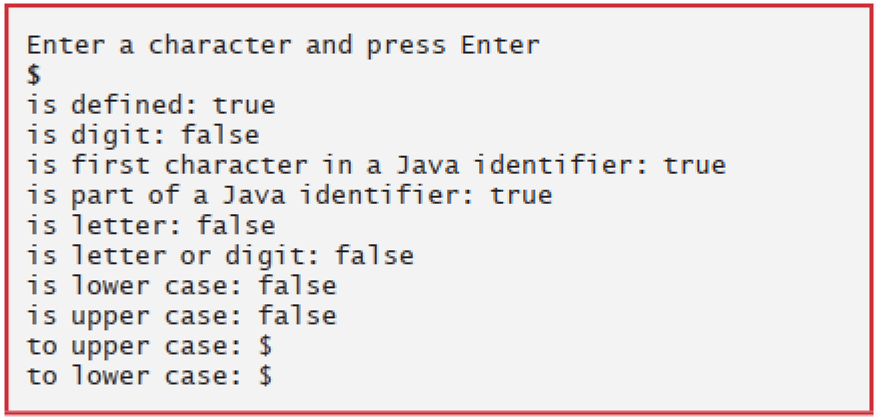
```java
// Figura 14.15: StaticCharMethods.java
// Métodos estáticos Character para testar caracteres e converter entre maiúsculas e minúsculas.
import java.util.Scanner;

public class StaticCharMethods
{
  public static void main(String[] args)
  {
      Scanner scanner = new Scanner(System.in); // cria scanner
      System.out.println("Enter a character and press Enter");
      String input = scanner.next();
       char c = input.charAt(0); // obtém caractere de entrada

      // exibe informações de caractere
      System.out.printf("is defined: %b%n",  Character.isDefined(c));
      System.out.printf("is digit: %b%n",  Character.isDigit(c));
      System.out.printf("is first character in a Java identifier: %b%n",
                 Character.isJavaIdentifierStart(c));
      System.out.printf("is part of a Java identifier: %b%n",
                 Character.isJavaIdentifierPart(c));
      System.out.printf("is letter: %b%n",  Character.isLetter(c));
      System.out.printf(
                "is letter or digit: %b%n",  Character.isLetterOrDigit(c));
      System.out.printf(
                "is lower case: %b%n",  Character.isLowerCase(c));
      System.out.printf(
               "is upper case: %b%n",  Character.isUpperCase(c));
      System.out.printf(
               "to upper case: %s%n",  Character.toUpperCase(c));
      System.out.printf(
               "to lower case: %s%n",  Character.toLowerCase(c));
   }
} // fim da classe StaticCharMethods
```



```
Enter a character and press Enter
A
is defined: true
is digit: false
is first character in a Java identifier: true
is part of a Java identifier: true
is letter: true
is letter or digit: true
is lower case: false
is upper case: true
to upper case: A
to lower case: a
```



```
Enter a character and press Enter
8
is defined: true
is digit: true
is first character in a Java identifier: false
is part of a Java identifier: true
is letter: false
is letter or digit: true
is lower case: false
is upper case: false
to upper case: 8
to lower case: 8
```



```
Enter a character and press Enter
$
is defined: true
is digit: false
is first character in a Java identifier: true
is part of a Java identifier: true
is letter: false
is letter or digit: false
is lower case: false
is upper case: false
to upper case: $
to lower case: $
```

Classe **StringTokenizer**

- `long StringTokenizer.countTokens()`
- `boolean StringTokenizer.hasMoreTokens()`
- `String StringTokenizer.nextToken()`

// Exemplo

```
StringTokenizer strT;
strT = new StringTokenizer
       ("Uma frase com cinco palavras");
while(strT.hasMoreTokens())
  System.out.println(strT.nextToken());
```

```java
// Fig. 30.18: TokenTest.java
// StringTokenizer class.
import java.util.Scanner;
import java.util.StringTokenizer;

public class TokenTest
{
      // execute application
      public static void main( String args[] )
      {
          // get sentence
          Scanner scanner = new Scanner( System.in );
          System.out.println( "Enter a sentence and press Enter" );
          String sentence = scanner.nextLine();

          // process user sentence
          StringTokenizer tokens = new StringTokenizer( sentence );
          System.out.printf( "Number of elements: %d\nThe tokens are:\n",
                               tokens.countTokens() );

          while ( tokens.hasMoreTokens() )
              System.out.println( tokens.nextToken() );
      } // end main
} // end class TokenTest
```

```
Enter a sentence and press Enter
This is a sentence with seven tokens
Number of elements: 7
The tokens are:
This
is
a
sentence
with
seven
tokens
```

```java
// Figura 14.18: TokenTest.java
// Método split da classe String usado para tokenizar strings.
import java.util.Scanner;
import java.util.StringTokenizer;

public class TokenTest
{
    // executa o aplicativo
    public static void main(String[] args)
    {
        // obtém a frase
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter a sentence and press Enter");
        String sentence = scanner.nextLine();

        // processa a frase do usuário
         String[] tokens = sentence.split(" ");
        System.out.printf("Number of elements: %d%nThe tokens are:%n",
                     tokens.length);

         for (String token : tokens)
             System.out.println(token);
    }
} // fim da classe TokenTest
```

```
Enter a sentence and press Enter
This is a sentence with seven tokens
Number of elements: 7
The tokens are:
This
is
a
sentence
with
seven
tokens
```

Exercícios

Referências

- Java How to Program 3, 4, 5, 6, 7, 8, 9, 10 ed. - Paul Deitel and Harvey Deitel
- Sun (http://java.sun.com)
- Oracle (http://www.oracle.com/technetwork/java)