

# Java Starter

[www.t2ti.com](http://www.t2ti.com)

# Curso Java Starter

---

## **Apresentação**

O Curso Java Starter foi projetado com o objetivo de ajudar àquelas pessoas que têm uma base de lógica de programação e desejam entrar no mercado de trabalho sabendo Java,

A estrutura do curso é formada por módulos em PDF e por mini-cursos em vídeo. O aluno deve baixar esse material e estudá-lo. Deve realizar os exercícios propostos. Todas as dúvidas devem ser enviadas para a lista de discussão que está disponível para inscrição na página do Curso Java Starter no site [www.t2ti.com](http://www.t2ti.com). As dúvidas serão respondidas pelos instrutores Albert Eije, Cláudio de Barros e Miguel Kojjio, além dos demais participantes da lista.

Nosso objetivo é que após o estudo do Curso Java Starter o aluno não tenha dificuldades para acompanhar um curso avançado onde poderá aprender a desenvolver aplicativos para Web, utilizando tecnologias como Servlets e JSP e frameworks como Struts e JSF, além do desenvolvimento para dispositivos móveis.

Albert Eije trabalha com informática desde 1993. Durante esse período já trabalhou com várias linguagens de programação: Clipper, PHP, Delphi, C, Java, etc. Atualmente mantém o site [www.alberteije.com](http://www.alberteije.com).

Cláudio de Barros é Tecnólogo em Processamento de Dados.

Miguel Kojjio é bacharel em Sistemas de Informação, profissional certificado Java (SCJP 1.5).

O curso Java Starter surgiu da idéia dos três amigos que trabalham juntos em uma instituição financeira de grande porte.

### Módulo

# 05

## Pacotes `java.lang` e `java.util`

### Definição de Pacotes

O que são pacotes em Java? Um pacote (**package**) em Java é um diretório em que está armazenada uma ou mais classes. Os pacotes costumam agrupar classes de mesmas afinidades, por exemplo, observe a classe *Aluno*, ela está no pacote *javainiciante* e deve ser declarada da seguinte maneira:

```
package javainiciante;

public class Aluno {
    //Estrutura da classe
}
```

A palavra reservada **package** define em qual pacote se encontra a classe.

Para utilizar classes de um pacote, é usada a palavra reservada **import** do seguinte modo:

```
import nome-do-pacote.nome-da-classe
```

Ex1.: classe *Teste* importando a classe *Aluno*:

```
import javainiciante.Aluno;

public class Teste {
    Aluno aluno = new Aluno();
    //Estrutura da classe
}
```

Ex2.: **import** `java.util.Date`; --> indica que será usada a classe *Date* do pacote *java.util* que está no diretório `\java\util`.

Ex3.: **import** `java.util.*`; --> Neste exemplo, está sendo usado o símbolo "\*" ao

## Curso Java Starter

---

invés do nome da classe. Isto significa que estaremos importando todas as classes que pertencem ao pacote *java.util*.

A utilização do `import` só é necessária, caso a classe que está utilizando as funcionalidades de outra classe estiver em um pacote diferente. Ex.: classe *Professor* que está no mesmo pacote da classe *Aluno*:

```
package javainiciante;

public class Professor {
    Aluno aluno = new Aluno(); //não é necessário importar a classe Aluno
    //Estrutura da classe
}
```

Uma classe está somente em um `package` e pode ter vários `imports`.

### O pacote java.lang

Por padrão, o pacote *java.lang* é importado automaticamente pelo Java. É o único pacote que possui esta característica. Isso explica o fato de, nos exemplos que utilizamos nos módulos anteriores, não precisarmos utilizar `imports` em nossas classes.

As principais classes que fazem parte deste pacote são:

*Boolean, Byte, Character, Double, Float, Integer, Long, Math, Number, Object, Short, String, System, Thread, etc.*

### java.lang.String

*String* é uma classe em Java. Instâncias de uma *String* correspondem à união de um conjunto de caracteres. Essas *strings* podem ser manipuladas de diversas maneiras. Apresentaremos aqui, alguns métodos desta classe:

#### Método lenght()

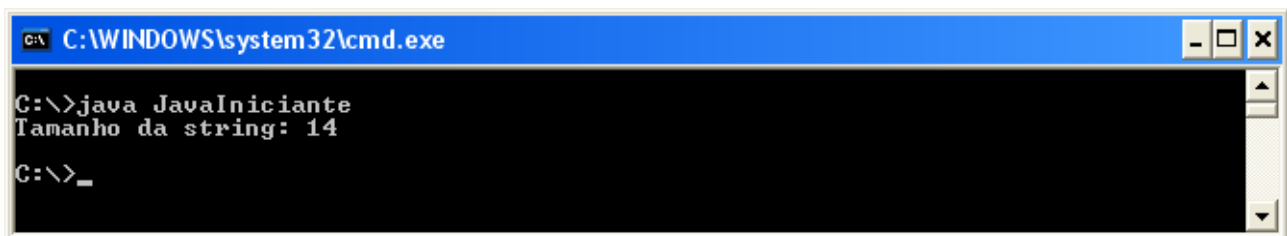
O método *length()* é utilizado para retornar o tamanho de uma determinada string, incluindo também os espaços em branco. Este método sempre retorna um valor do tipo *int*. Tem a seguinte sintaxe:

*nome-da-string.length()*

# Curso Java Starter

Segue um exemplo e o resultado de execução da classe:

```
class JavaIniciante {  
  
    public static void main (String args[]){  
        String s = "Java Iniciante";  
        System.out.println("Tamanho da string: " + s.length());  
    }  
}
```



```
C:\WINDOWS\system32\cmd.exe  
C:\>java JavaIniciante  
Tamanho da string: 14  
C:\>_
```

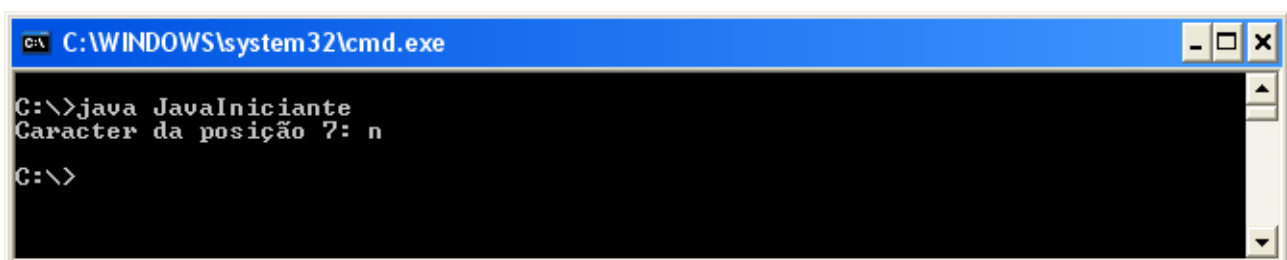
## Método charAt()

O método *charAt()* é utilizado para retornar um caractere de uma determinada string de acordo com um índice especificado entre parênteses (parâmetro), sendo que o primeiro caractere tem o índice 0, o segundo tem o índice 1 e assim por diante. Tem a seguinte sintaxe:

*nome-da-string.charAt(indice)*

Observe que no exemplo a seguir, o índice informado no parâmetro é 6, retornando o 7º caractere da string.

```
class JavaIniciante {  
  
    public static void main (String args[]){  
        String s = "Java Iniciante";  
        System.out.println("Caracter da posição 7: " + s.charAt(6));  
    }  
}
```



```
C:\WINDOWS\system32\cmd.exe  
C:\>java JavaIniciante  
Caracter da posição 7: n  
C:\>
```

### Métodos `toUpperCase()` e `toLowerCase()`

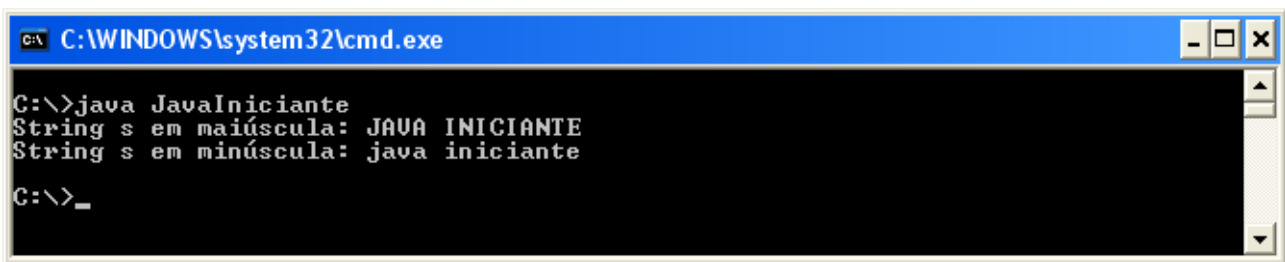
Os métodos `toUpperCase()` e `toLowerCase()` são utilizados para transformar todas as letras de uma determinada string em maiúsculas ou minúsculas, respectivamente. Tem a seguinte sintaxe:

*nome-da-string.toUpperCase()*

*nome-da-string.toLowerCase()*

Veja no exemplo a seguir:

```
class JavaIniciante {  
  
    public static void main (String args[]){  
        String s = "Java Iniciante";  
        System.out.println("String s em maiúscula: " + s.toUpperCase());  
        System.out.println("String s em minúscula: " + s.toLowerCase());  
    }  
}
```



```
C:\WINDOWS\system32\cmd.exe  
C:\>java JavaIniciante  
String s em maiúscula: JAVA INICIANTE  
String s em minúscula: java iniciante  
C:\>_
```

### Método `substring()`

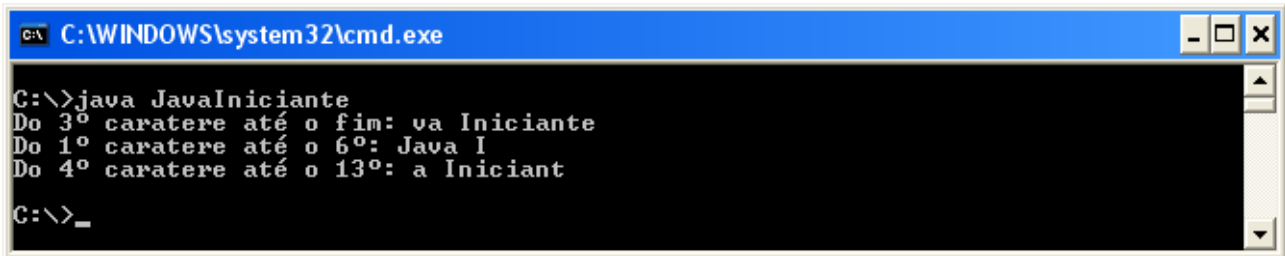
O método `substring()` é utilizado para retornar uma cópia de caracteres de uma string a partir de dois índices inteiros especificados. Tem a seguinte sintaxe:

***nome-da-string.substring(indice1, indice2)***

O primeiro parâmetro especifica o índice a partir do qual se inicia a cópia dos caracteres. O segundo parâmetro é opcional (se não for inserido será considerado como o comprimento máximo da string) e especifica o índice final em que termina a cópia dos caracteres, porém o índice final deve especificar um índice além do último caractere. Caso queira uma cópia do 1º ao 5º caracteres, o índice inicial deve ser 0 e o índice final 6. Veja no exemplo a seguir:

# Curso Java Starter

```
class JavaIniciante {
    public static void main (String args[]){
        String s = "Java Iniciante";
        System.out.println("Do 3º caractere até o fim: " + s.substring(2));
        System.out.println("Do 1º caractere até o 6º: " + s.substring(0, 6));
        System.out.println("Do 4º caractere até o 13º: " + s.substring(3, 13));
    }
}
```



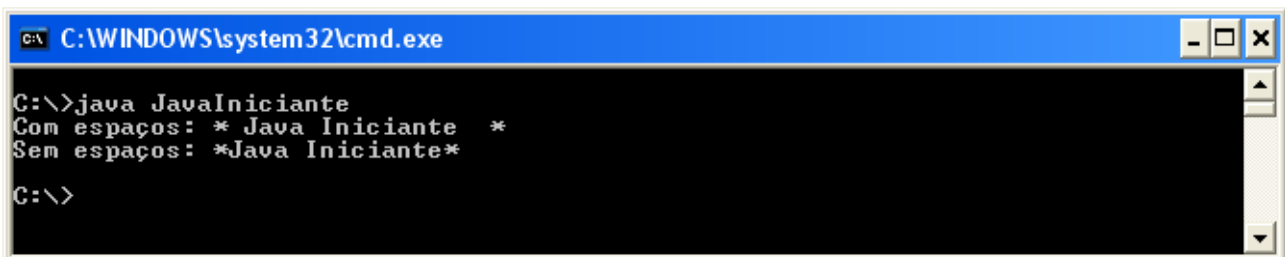
```
C:\WINDOWS\system32\cmd.exe
C:\>java JavaIniciante
Do 3º caractere até o fim: va Iniciante
Do 1º caractere até o 6º: Java I
Do 4º caractere até o 13º: a Iniciant
C:\>_
```

## Método trim()

O método *trim()* tem por objetivo remover todos os espaços em branco que aparecem no início e no final de uma determinada string. Observe que serão removidos apenas os espaços do início e do fim da string – não serão removidos os espaços entre as palavras. Tem a seguinte sintaxe:

***nome-da-string.trim()***

```
class JavaIniciante {
    public static void main (String args[]){
        String s = " Java Iniciante ";
        System.out.println("Com espaços: *" + s + "*");
        System.out.println("Sem espaços: *" + s.trim() + "*");
    }
}
```



```
C:\WINDOWS\system32\cmd.exe
C:\>java JavaIniciante
Com espaços: * Java Iniciante *
Sem espaços: *Java Iniciante*
C:\>
```

## Método replace()

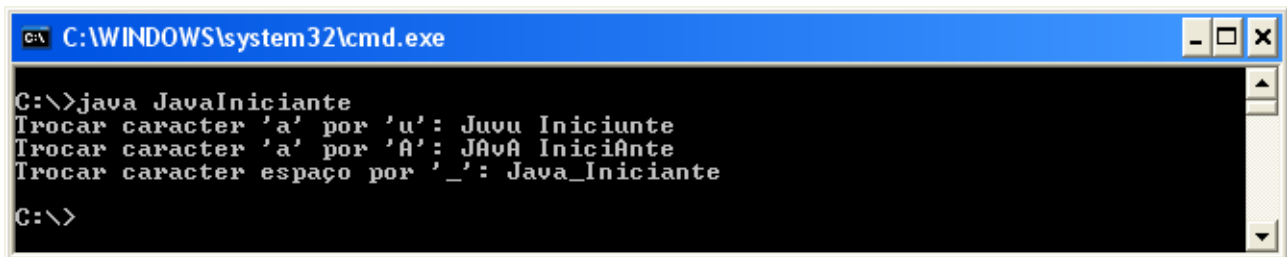
O método *replace()* é utilizado para substituição de caracteres individuais de

## Curso Java Starter

uma string. Deverão ser informados como parâmetros, o caracter a ser substituído e por qual caractere será substituído. Tem a seguinte sintaxe:

*nome-da-string.replace(caracter-a-ser-substituído, substituição)*

```
class JavaIniciante {
    public static void main (String args[]){
        String s = "Java Iniciante";
        System.out.println("Trocar caracter 'a' por 'u': " + s.replace("a", "u"));
        System.out.println("Trocar caracter 'a' por 'A': " + s.replace("a", "A"));
        System.out.println("Trocar caracter espaço por '_': " + s.replace(" ", "_"));
    }
}
```



```
C:\WINDOWS\system32\cmd.exe
C:\>java JavaIniciante
Trocar caracter 'a' por 'u': Juvu Iniciunte
Trocar caracter 'a' por 'A': JÁvÁ IniciÁnte
Trocar caracter espaço por '_' : Java_Iniciante
C:\>
```

### Método `String.valueOf()`

O método `String.valueOf()` é usado para converter diversos tipos de dados em strings. Para isso, esse método aceita vários tipos de argumento e tranforma-os em strings. Tem a seguinte sintaxe:

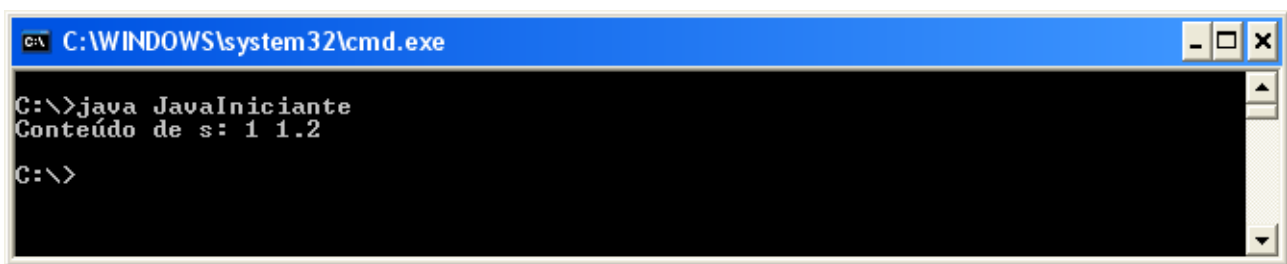
*String.valueOf(nome-da-variável)*

```
class JavaIniciante {

    public static void main (String args[]){
        int a = 1;
        double b = 1.2;
        String s = String.valueOf(a) + " " + String.valueOf(b);
        System.out.println("Conteúdo de s: " + s);
    }
}
```



## Curso Java Starter



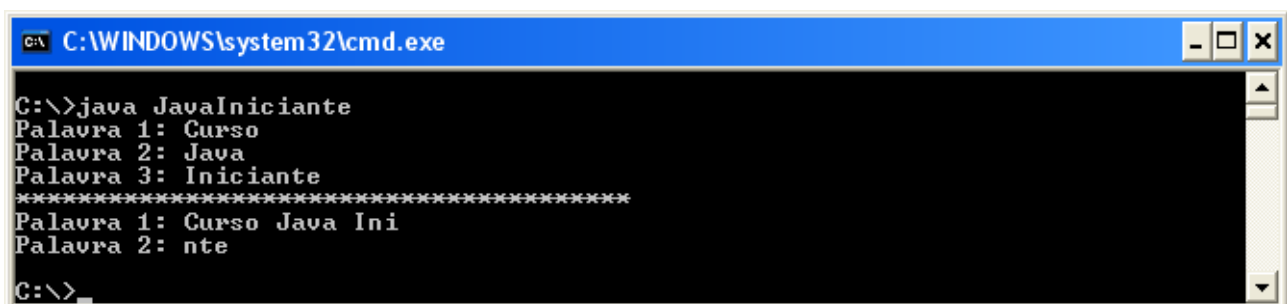
```
C:\WINDOWS\system32\cmd.exe
C:\>java JavaIniciante
Conteúdo de s: 1 1.2
C:\>
```

### Método split()

O método *split()* é usado para dividir uma string em um array de strings. É passado como parâmetro qual o caractere ou caracteres que delimitam a divisão. Tem a seguinte sintaxe:

**nome-da-string.split(caractere);**

```
class JavaIniciante {
    public static void main (String args[]){
        String s = "Curso Java Iniciante";
        String[] palavras = s.split(" ");
        for (int i = 0; i < palavras.length; i++){
            System.out.println("Palavra " + (i + 1) + ": " + palavras[i]);
        }
        System.out.println("*****");
        palavras = s.split("cia");
        for (int i = 0; i < palavras.length; i++){
            System.out.println("Palavra " + (i + 1) + ": " + palavras[i]);
        }
    }
}
```



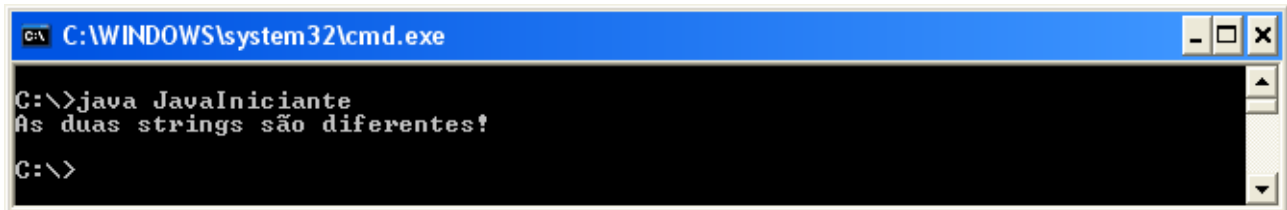
```
C:\WINDOWS\system32\cmd.exe
C:\>java JavaIniciante
Palavra 1: Curso
Palavra 2: Java
Palavra 3: Iniciante
*****
Palavra 1: Curso Java Ini
Palavra 2: nte
C:\>
```

### Comparando Strings

Veja o exemplo seguinte:

# Curso Java Starter

```
class JavaIniciante {  
  
    public static void main (String args[]){  
        String s1 = new String("Curso Java Iniciante");  
        String s2 = new String("Curso Java Iniciante");  
        if (s1 == s2){  
            System.out.println("As duas strings são iguais!");  
        }  
        else{  
            System.out.println("As duas strings são diferentes!");  
        }  
    }  
}
```



```
C:\WINDOWS\system32\cmd.exe  
C:\>java JavaIniciante  
As duas strings são diferentes!  
C:\>
```

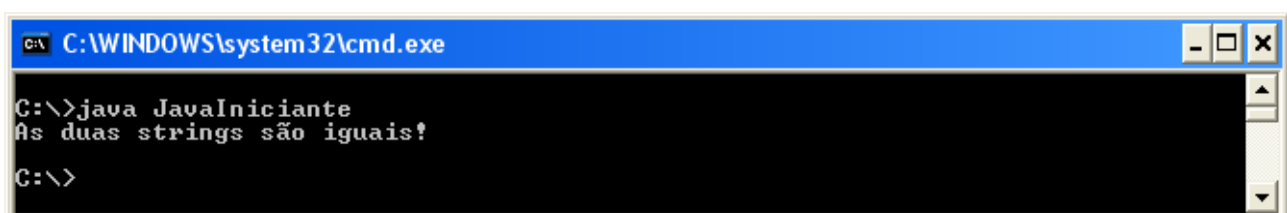
O que há de errado neste resultado? Declaramos duas strings e a instanciamos com o mesmo conteúdo, e ao compará-las temos como resultado que elas são diferentes.

Quando quisermos comparar duas strings, não devemos utilizar "==" e sim o método *equals()* que possui a seguinte sintaxe:

***string1.equals(string2)***

Agora vamos utilizar este método e ver o que acontece:

```
class JavaIniciante {  
  
    public static void main (String args[]){  
        String s1 = new String("Curso Java Iniciante");  
        String s2 = new String("Curso Java Iniciante");  
        if (s1.equals(s2)){  
            System.out.println("As duas strings são iguais!");  
        }  
        else{  
            System.out.println("As duas strings são diferentes!");  
        }  
    }  
}
```



```
C:\WINDOWS\system32\cmd.exe  
C:\>java JavaIniciante  
As duas strings são iguais!  
C:\>
```

## Curso Java Starter

Agora sim, temos o resultado esperado!

A classe `java.lang.String` possui vários métodos. É uma boa prática consultar o javadoc (esse assunto será abordado no próximo módulo) desta classe para aprender mais sobre a mesma.

### java.lang.Math

A classe `java.lang.Math` possui vários métodos que permitem efetuar diversos tipos de cálculos matemáticos. Para utilizar os métodos desta classe basta chamar o método precedido do nome `Math`. Ela não pode ser estendida (é uma classe *final*) e possui o construtor *private* (não pode ser instanciada);

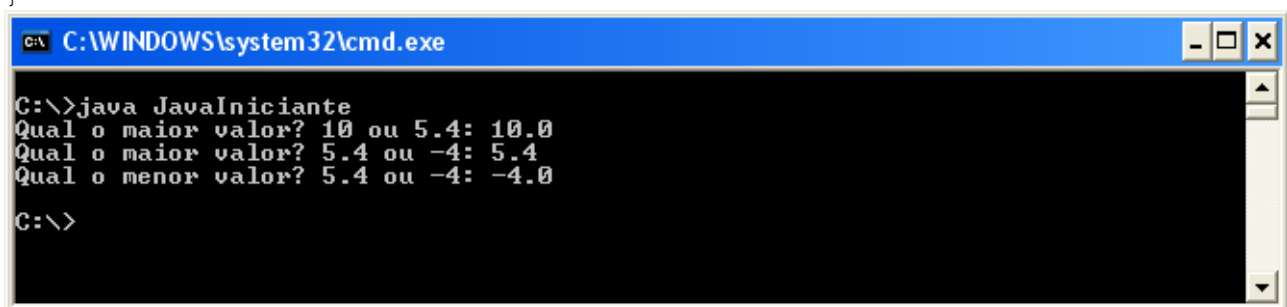
### Métodos max() e min()

O método `max()` é utilizado para verificar o maior valor entre dois números, e o método `min()` é utilizado para verificar o menor valor entre dois números, que podem ser do tipo `double`, `float`, `int` ou `long`. Observe a seguir a sintaxe destes métodos:

**`Math.max(valor1, valor2);`**

**`Math.min(valor1, valor2);`**

```
class JavaIniciante {  
  
    public static void main (String args[]){  
        int a = 10;  
        double b = 5.4;  
        int c = -4;  
        System.out.println("Qual o maior valor? 10 ou 5.4: " + Math.max(a, b));  
        System.out.println("Qual o maior valor? 5.4 ou -4: " + Math.max(b, c));  
        System.out.println("Qual o menor valor? 5.4 ou -4: " + Math.min(b, c));  
    }  
}
```



```
C:\WINDOWS\system32\cmd.exe  
C:\>java JavaIniciante  
Qual o maior valor? 10 ou 5.4: 10.0  
Qual o maior valor? 5.4 ou -4: 5.4  
Qual o menor valor? 5.4 ou -4: -4.0  
C:\>
```

## Método `random()`

O método `random()` é utilizado para gerar valores de forma aleatória. Toda vez que este método é chamado, será gerado um valor do tipo `double` entre 0.0 e 1.0 (o valor 1.0 nunca é gerado). Veja a sintaxe deste método:

### **`Math.random()`**

```
class JavaIniciante {  
  
    public static void main (String args[]){  
        System.out.println("Número aleatório: " + Math.random());  
        System.out.println("Número aleatório: " + Math.random());  
        System.out.println("Número aleatório: " + Math.random());  
    }  
}
```

```
C:\WINDOWS\system32\cmd.exe  
C:\>java JavaIniciante  
Número aleatório: 0.2726702015295974  
Número aleatório: 0.05899563640405092  
Número aleatório: 0.4857618203716825  
C:\>_
```

Execute a aplicação várias vezes e perceba que o resultado será sempre diferente.

Agora, se quisermos gerar números aleatórios inteiros? Por exemplo, minha aplicação precisa de números entre 0 e 99. O exemplo seguinte mostra como implementar isto.

```
class JavaIniciante {  
  
    public static void main (String args[]){  
        System.out.println("Número aleatório: " + (int) (Math.random() * 100));  
        System.out.println("Número aleatório: " + (int) (Math.random() * 100));  
        System.out.println("Número aleatório: " + (int) (Math.random() * 100));  
    }  
}
```

```
C:\WINDOWS\system32\cmd.exe  
C:\>java JavaIniciante  
Número aleatório: 74  
Número aleatório: 57  
Número aleatório: 17  
C:\>
```

## Curso Java Starter

A classe `java.lang.Math` possui vários métodos. É uma boa prática consultar o javadoc (esse assunto será abordado no próximo módulo) desta classe para aprender mais sobre a mesma.

### Classes Wrapper

Classes wrapper são classes usadas para embutir tipos primitivos para que possam ser utilizados como objetos. Cada tipo primitivo tem a sua classe wrapper correspondente. As classes wrapper são: *Character*, *Byte*, *Short*, *Integer*, *Long*, *Float*, *Double* e *Boolean*.

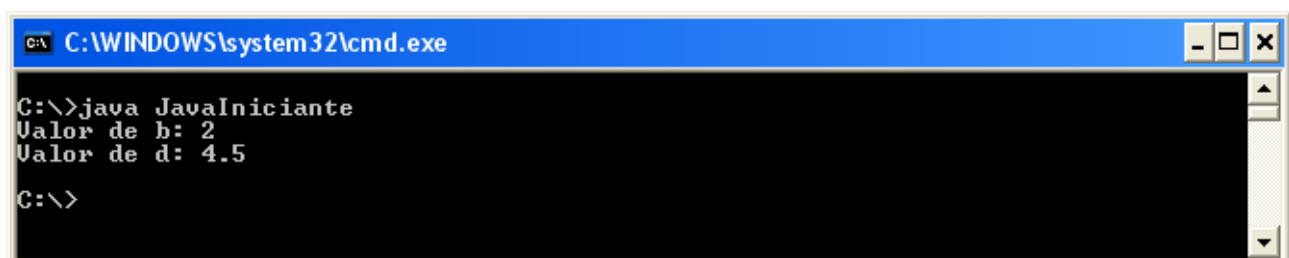
Cada classe wrapper permite manipular tipos primitivos como objetos da classe *Object*. Desta forma, valores do tipo primitivo podem ser processados de forma polimórfica (assunto será abordado no módulo 09) se forem mantidos como objetos das classes wrapper.

Como fazer para obter um *int* que está armazenado em sua classe wrapper?

Como fazer para obter um *double* que está armazenado em sua classe wrapper?

O exemplo a seguir mostra a resposta para estas duas perguntas:

```
class JavaIniciante {  
  
    public static void main (String args[]){  
        Integer a = new Integer(2);  
        int b = a.intValue();  
        Double c = new Double(4.5);  
        double d = c.doubleValue();  
        System.out.println("Valor de b: " + b);  
        System.out.println("Valor de d: " + d);  
    }  
}
```



```
C:\WINDOWS\system32\cmd.exe  
C:\>java JavaIniciante  
Valor de b: 2  
Valor de d: 4.5  
C:\>
```

O pacote `java.lang` possui várias classes. É uma boa prática consultar a API (esse assunto será abordado no próximo módulo) deste pacote para aprender mais sobre o mesmo.

### O pacote java.util

O pacote *java.util* possui várias classes muito utilizadas e importantes, e algumas delas foram abordadas em módulos anteriores. Neste módulo iremos falar sobre a classe *Calendar*. Não falaremos sobre a classe *Date*, pois a maioria dos métodos desta classe estão "**deprecated**" e a classe *Calendar* a substituiu.

Quando um método está marcado como **deprecated**, significa que este foi substituído por outro. Isso acontece quando é detectado um *bug* ou ineficiência. Então, o método é mantido como está e marcado como **deprecated**, para que aplicações legadas continuem utilizando o mesmo sem problemas. Você até pode usar um método "**deprecated**", mas isso não é recomendado.

### java.util.Calendar

Com a classe *Calendar* podemos realizar vários tipos de operações envolvendo datas, como ano, mês, dia, hora, minuto, etc. Esta classe tem o tempo representado em milissegundos.

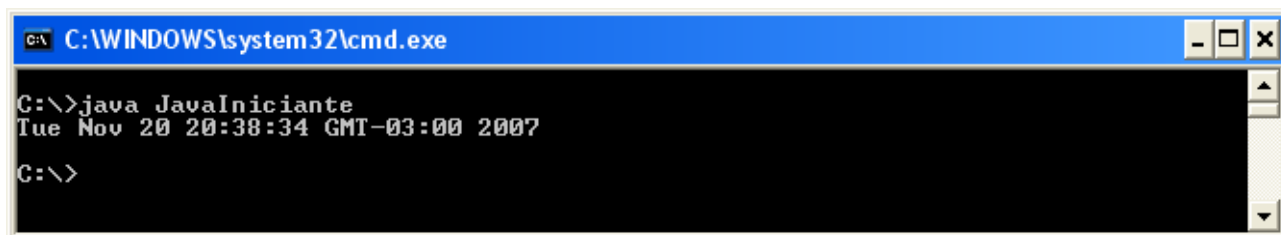
Para obter uma instância desta classe, basta chamar o método estático *getInstance()*. Quando este método é chamado, é obtido um *Calendar* com data e hora atuais. Depois de obtido este objeto, utilizamos o método *get()* para buscar o ano, dia, etc.

Primeiramente, vamos utilizar o método *getTime()* para mostrar a data e hora atuais.

```
import java.util.Calendar; //necessário este import para utilização da classe

class JavaIniciante {

    public static void main (String args[]){
        Calendar data = Calendar.getInstance();
        System.out.println(data.getTime());
    }
}
```



```
C:\WINDOWS\system32\cmd.exe
C:\>java JavaIniciante
Tue Nov 20 20:38:34 GMT-03:00 2007
C:\>
```

## Curso Java Starter

No exemplo anterior, a data e hora foram mostradas de uma forma diferente da que estamos acostumados. Vamos analisá-lo:

1 - Tue --> representa o dia da semana. Neste caso "Tuesday" - Terça-feira

2 - Nov --> representa o mês. Neste caso "November" - Novembro

3 - 20 --> representa o dia do mês.

4 - 20:38:34 --> representa a hora, minuto e segundo.

5 - GMT -03:00 --> representa o Tempo Médio de Greenwich. Neste caso a hora oficial do brasil.

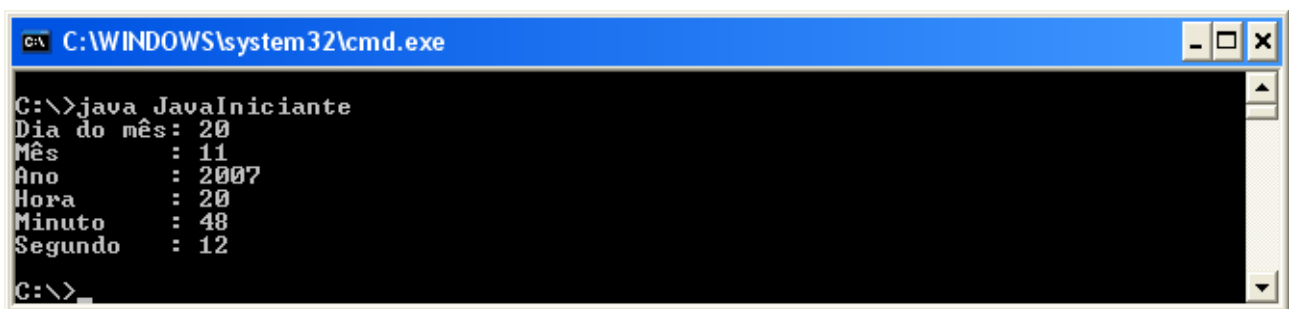
6 - 2007 --> representa o ano.

Talvez não seja muito útil a data representada desta maneira, então vamos obter cada campo separadamente. Vamos utilizar agora o método *get()*.

```
import java.util.Calendar;

class JavaIniciante {

    public static void main (String args[]){
        Calendar data = Calendar.getInstance();
        System.out.println("Dia do mês: " + data.get(Calendar.DAY_OF_MONTH));
        System.out.println("Mês      : " + (data.get(Calendar.MONTH) + 1));
        System.out.println("Ano      : " + data.get(Calendar.YEAR));
        System.out.println("Hora    : " + data.get(Calendar.HOUR_OF_DAY));
        System.out.println("Minuto  : " + data.get(Calendar.MINUTE));
        System.out.println("Segundo : " + data.get(Calendar.SECOND));
    }
}
```



```
C:\WINDOWS\system32\cmd.exe
C:\>java JavaIniciante
Dia do mês: 20
Mês      : 11
Ano      : 2007
Hora     : 20
Minuto   : 48
Segundo  : 12
C:\>
```

O método *get()* da classe *Calendar* sempre retorna um valor do tipo *int*. Observe que para o mês, incrementamos o valor obtido em 1. A constante *Calendar.MONTH* que representa o mês inicia em 0(zero), ou seja, janeiro = 0, fevereiro = 1 e assim por diante.

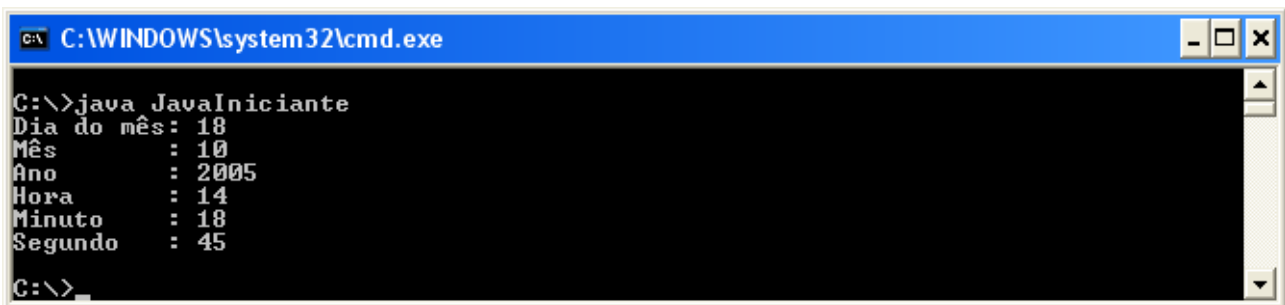
Agora iremos atribuir novos valores para os campos utilizando o método *set()*. Este método necessita de dois argumentos: o primeiro define qual campo vamos alterar e o segundo o novo valor do campo.

## Curso Java Starter

```
import java.util.Calendar;

class JavaIniciante {

    public static void main (String args[]){
        Calendar data = Calendar.getInstance();
        data.set(Calendar.DAY_OF_MONTH, 18);
        data.set(Calendar.MONTH, 10);
        data.set(Calendar.YEAR, 2005);
        data.set(Calendar.HOUR_OF_DAY, 14);
        data.set(Calendar.MINUTE, 18);
        data.set(Calendar.SECOND, 45);
        System.out.println("Dia do mês: " + data.get(Calendar.DAY_OF_MONTH));
        System.out.println("Mês      : " + data.get(Calendar.MONTH));
        System.out.println("Ano      : " + data.get(Calendar.YEAR));
        System.out.println("Hora    : " + data.get(Calendar.HOUR_OF_DAY));
        System.out.println("Minuto  : " + data.get(Calendar.MINUTE));
        System.out.println("Segundo : " + data.get(Calendar.SECOND));
    }
}
```



```
C:\WINDOWS\system32\cmd.exe

C:\>java JavaIniciante
Dia do mês: 18
Mês      : 10
Ano      : 2005
Hora     : 14
Minuto   : 18
Segundo  : 45

C:\>
```

No exemplo anterior, definimos a data para 18/11/2005, 14:18:45. E se quisermos “somar” 20 dias a esta data? Para isto, utilizamos o método *add()*. Da mesma forma que o *set()*, informamos dois argumentos: o primeiro define qual campo vamos “somar” e o segundo quanto iremos “somar”.

```
import java.util.Calendar;

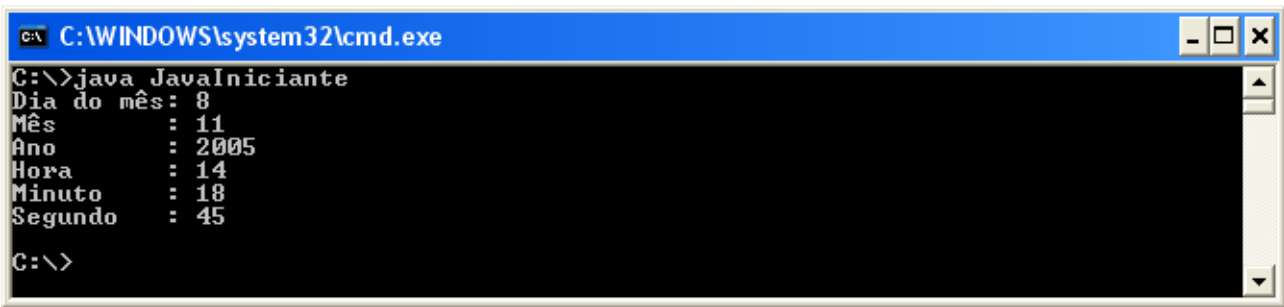
class JavaIniciante {

    public static void main (String args[]){
        Calendar data = Calendar.getInstance();
        data.set(Calendar.DAY_OF_MONTH, 18);
        data.set(Calendar.MONTH, 10);
        data.set(Calendar.YEAR, 2005);
        data.set(Calendar.HOUR_OF_DAY, 14);
        data.set(Calendar.MINUTE, 18);
        data.set(Calendar.SECOND, 45);
        data.add(Calendar.DAY_OF_MONTH, 20);
        System.out.println("Dia do mês: " + data.get(Calendar.DAY_OF_MONTH));
        System.out.println("Mês      : " + data.get(Calendar.MONTH));
        System.out.println("Ano      : " + data.get(Calendar.YEAR));
        System.out.println("Hora    : " + data.get(Calendar.HOUR_OF_DAY));
        System.out.println("Minuto  : " + data.get(Calendar.MINUTE));
        System.out.println("Segundo : " + data.get(Calendar.SECOND));
    }
}
```



## Curso Java Starter

```
}  
}
```



```
C:\WINDOWS\system32\cmd.exe  
C:\>java JavaIniciante  
Dia do mês: 8  
Mês : 11  
Ano : 2005  
Hora : 14  
Minuto : 18  
Segundo : 45  
C:\>
```

Como resultado, obtivemos a data 8/12/2005, 14:18:45. Lembre-se que o campo "mês" começa com 0(zero).

Outros métodos muito importantes da classe *java.util.Calendar* são: *before()* e *after()*. *before()* nos permite saber se uma determinada data é anterior a outra data. *after()* é o inverso, retorna se uma data é posterior a outra data. Tem a seguinte sintaxe:

***data1.before(data2)***

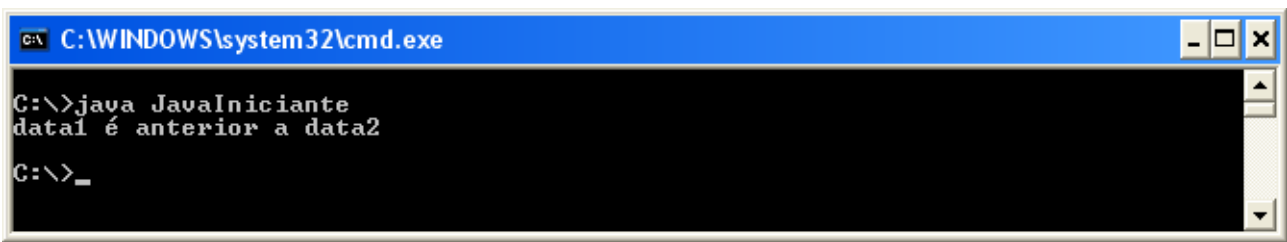
***data1.after(data2)***

*before()* retorna verdadeiro, se e somente se *data1 < data2*.

*after()* retorna verdadeiro, se e somente se *data1 > data2*.

```
import java.util.Calendar;  
  
class JavaIniciante {  
  
    public static void main (String args[]){  
        Calendar data1 = Calendar.getInstance();  
        data1.set(Calendar.DAY_OF_MONTH, 18);  
        data1.set(Calendar.MONTH, 10);  
        data1.set(Calendar.YEAR, 2005);  
        Calendar data2 = Calendar.getInstance();  
        data2.set(Calendar.DAY_OF_MONTH, 12);  
        data2.set(Calendar.MONTH, 8);  
        data2.set(Calendar.YEAR, 2007);  
        if (data1.before(data2)){  
            System.out.println("data1 é anterior a data2");  
        }  
        else{  
            System.out.println("data1 não é anterior a data2");  
        }  
    }  
}
```

## Curso Java Starter



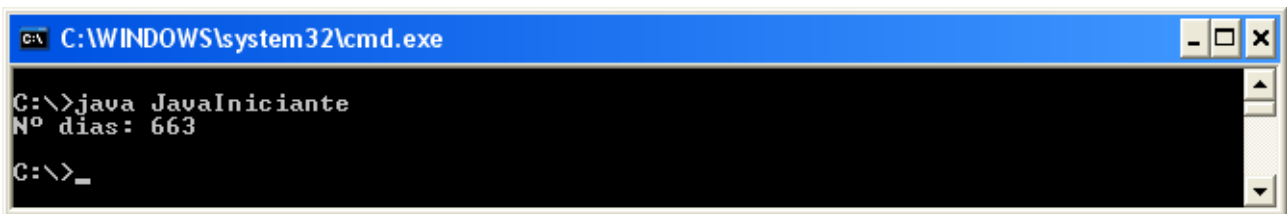
```
C:\WINDOWS\system32\cmd.exe
C:\>java JavaIniciante
data1 é anterior a data2
C:\>_
```

Agora vamos utilizar o método `getTimeInMillis()`, para calcularmos o intervalo de dias entre as datas utilizadas no exemplo anterior. O cálculo é feito pegando a diferença entre a data2 de data1 e dividindo-se pelo valor de milissegundos em um dia.

```
import java.util.Calendar;

class JavaIniciante {

    public static void main (String args[]){
        Calendar data1 = Calendar.getInstance();
        data1.set(Calendar.DAY_OF_MONTH, 18);
        data1.set(Calendar.MONTH, 10);
        data1.set(Calendar.YEAR, 2005);
        Calendar data2 = Calendar.getInstance();
        data2.set(Calendar.DAY_OF_MONTH, 12);
        data2.set(Calendar.MONTH, 8);
        data2.set(Calendar.YEAR, 2007);
        long d1 = data1.getTimeInMillis();
        long d2 = data2.getTimeInMillis();
        System.out.println("N° dias: " + (int) ((d2 - d1) / (24*60*60*1000)));
    }
}
```



```
C:\WINDOWS\system32\cmd.exe
C:\>java JavaIniciante
N° dias: 663
C:\>_
```

Agora você deve tentar resolver a lista de exercícios abaixo. Qualquer dificuldade envie para a lista de discussão:

### Exercícios

Aprenda com quem também está aprendendo, veja e compartilhe as suas respostas no nosso [Fórum](#):

[Exercícios – Módulo 05 – Pacotes \(java.lang – java.util\)](#)

## Curso Java Starter

---

01 – Elabore um programa em Java que possua um método que recebe uma *String* como parâmetro e caso esta *String* tenha menos que 8 caracteres, emita uma mensagem de erro.

02 – Aproveitando a classe implementada no exercício anterior, defina também que a *String* deve começar com a letra "A".

03 – Elabore um programa em Java que possua um método que recebe uma *String* como parâmetro e mostra ao usuário os quatro primeiros caracteres em Maiúsculas.

04 – Utilizando a classe elaborada no exercício anterior, verifique se a *String* contem números, emitindo uma mensagem ao usuário.

05 – Elabore um programa em Java que gere dois números aleatoriamente (entre 0 e 1000) e verifique qual dos números gerados é maior.

06 – Elabore um programa em Java que mostre ao usuário a quantidade de anos, meses, dias, horas, minutos e segundos que faltam para o ano de 2014.

07 – Elabore um programa em Java que possua um método que receba como parâmetro duas datas, e mostre a maior data. Caso as datas sejam iguais, emitir mensagem informando o usuário.

08 – Aproveitando o programa do exercício anterior, emita uma mensagem ao usuário caso a diferença entre as datas seja maior que 20 dias.