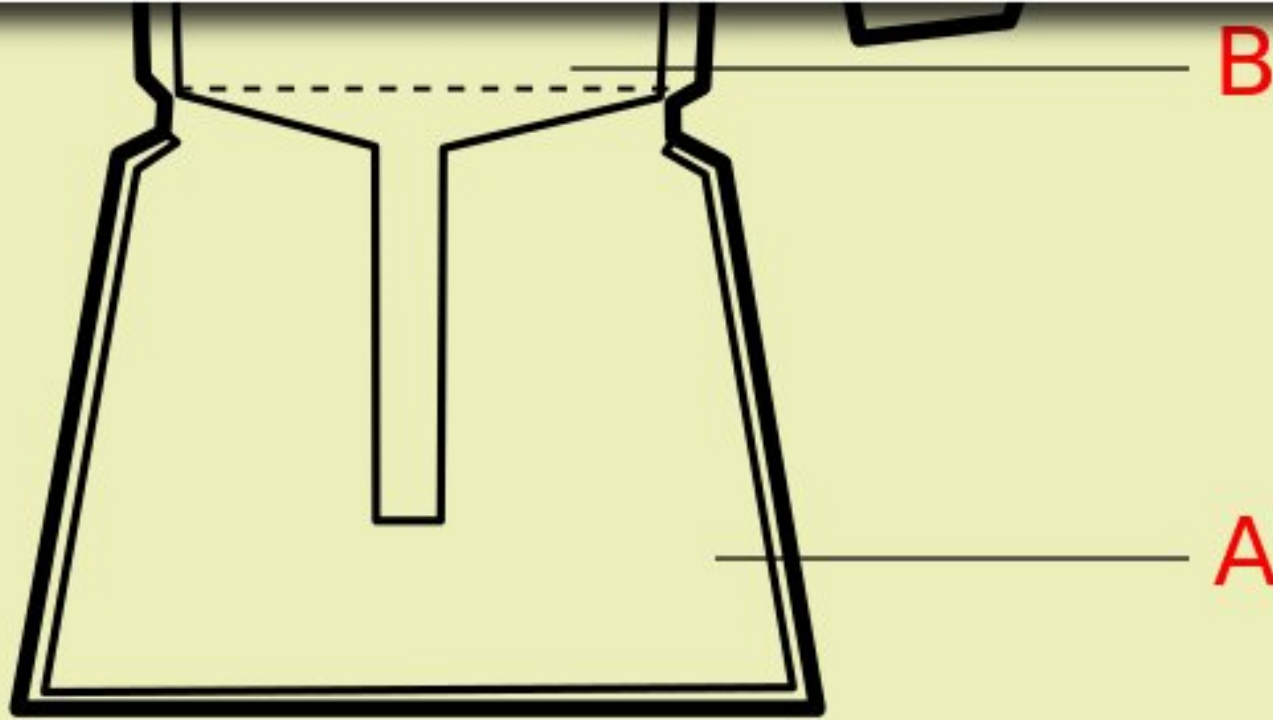


# Programação Orientada a Objetos

## Fundamentos de Java

André Santanchè  
Instituto de Computação - UNICAMP  
Março 2019

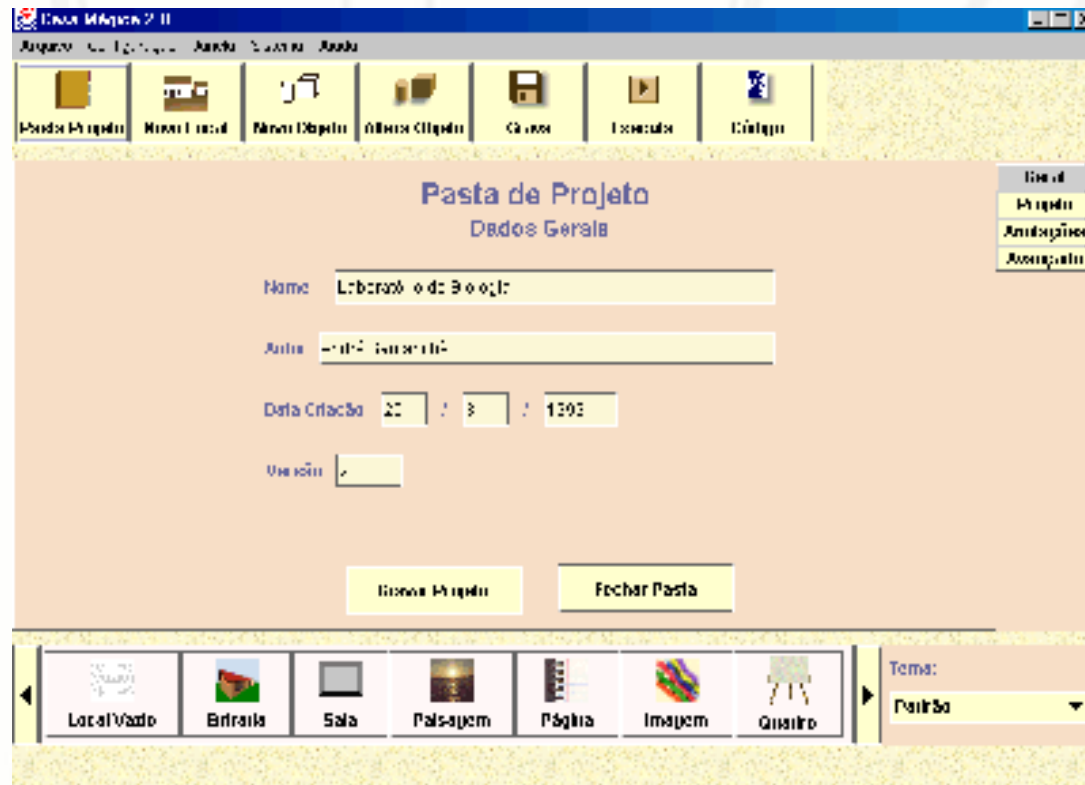


# Java

- Orientado a Objetos
- Baseado na Linguagem C++
- Independente de plataforma
- Código independente de arquitetura
- Seguro
- Robusto
- Compacto
- Suporte a Multiprocessamento
- Pode ser usado em páginas HTML

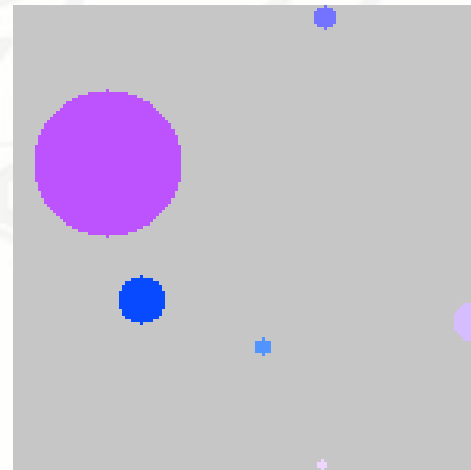
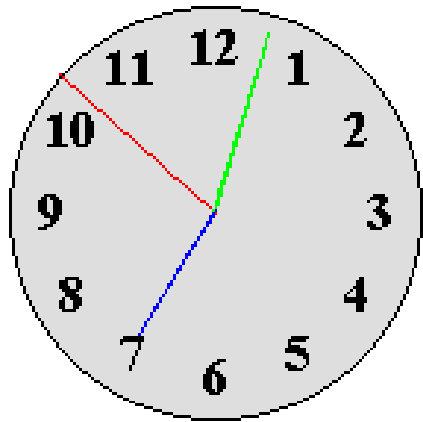
# Application

- Programa independente - pode ser de grande porte - interpretado por um módulo run-time.

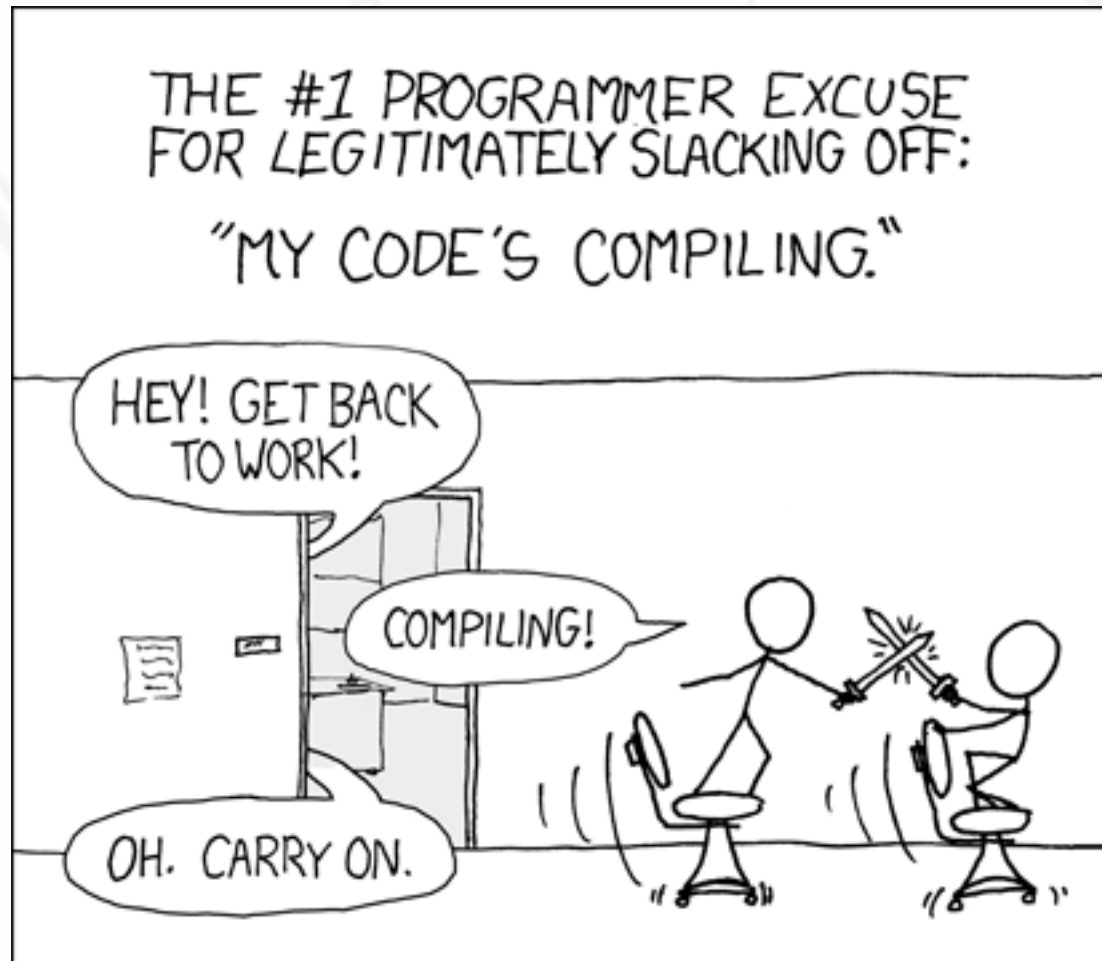


# Applet

- Programa geralmente de pequeno porte que pode ser acoplado a uma página HTML e é executado pelo Browser dentro da página.

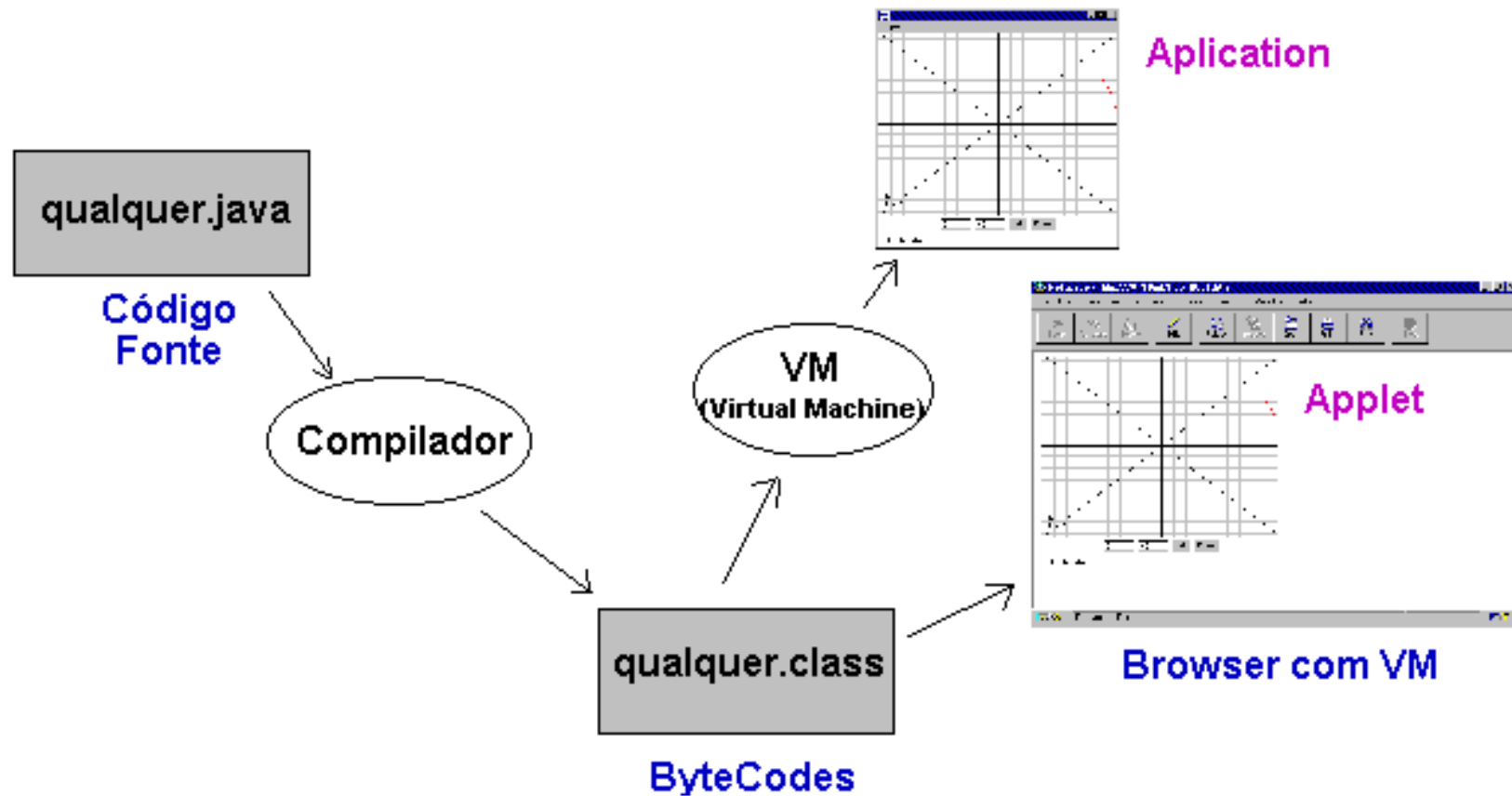


# Compilação e Bytecodes



By xkcd: <http://xkcd.com/303/>

# Esquema de Funcionamento





**Pacotes**

# Estrutura Mínima de um Programa Java

```
package pt.c01basico.s00hello;
```

```
public class HelloWorld {
```

```
    public static void main(String args[]) {
```

```
        System.out.println("0 dinossauro pulou na lama.");
```

```
    }
```

```
}
```



# Estrutura de um Programa

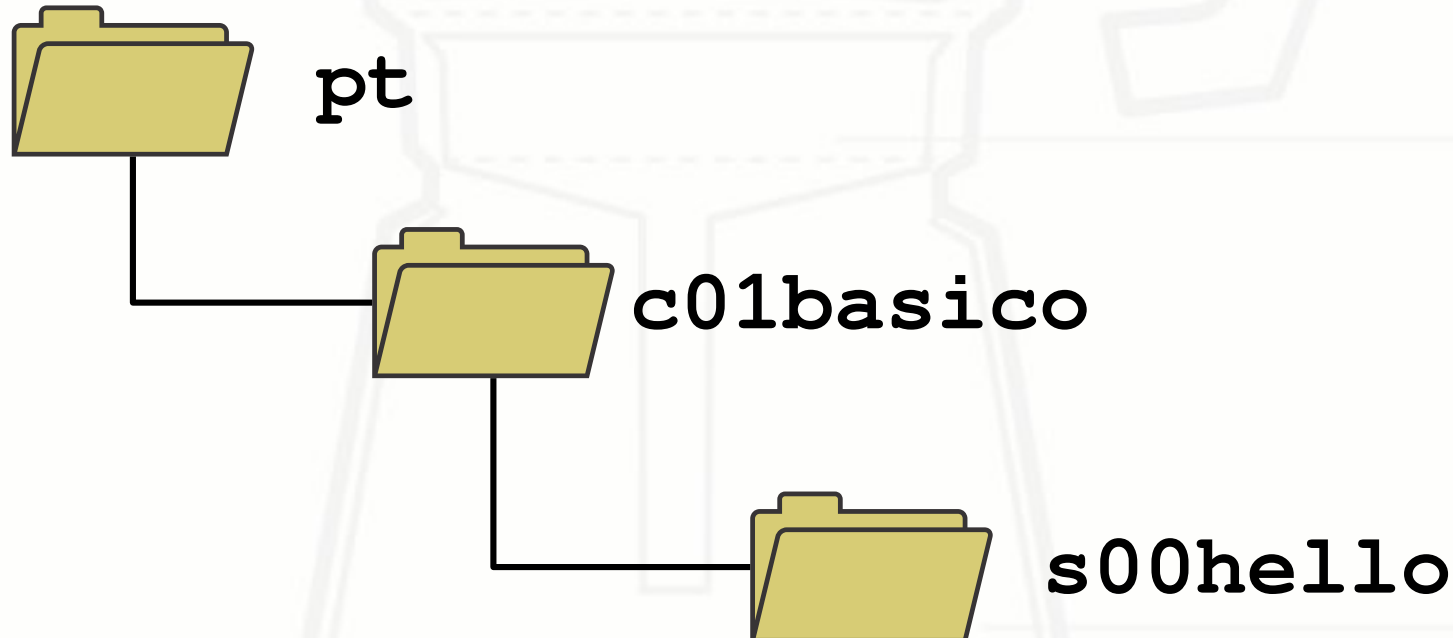
- As coisas em Java acontecem dentro de classes
- Todo o programa inicia em um método de alguma classe declarado com esta assinatura:

```
public static void main(String args[])
```

# Hierarquia de Pacotes

- Hierarquia de pacotes deve ser igual a estrutura de pastas:

**package** pt.c01basico.s00hello



# Estrutura de Pastas

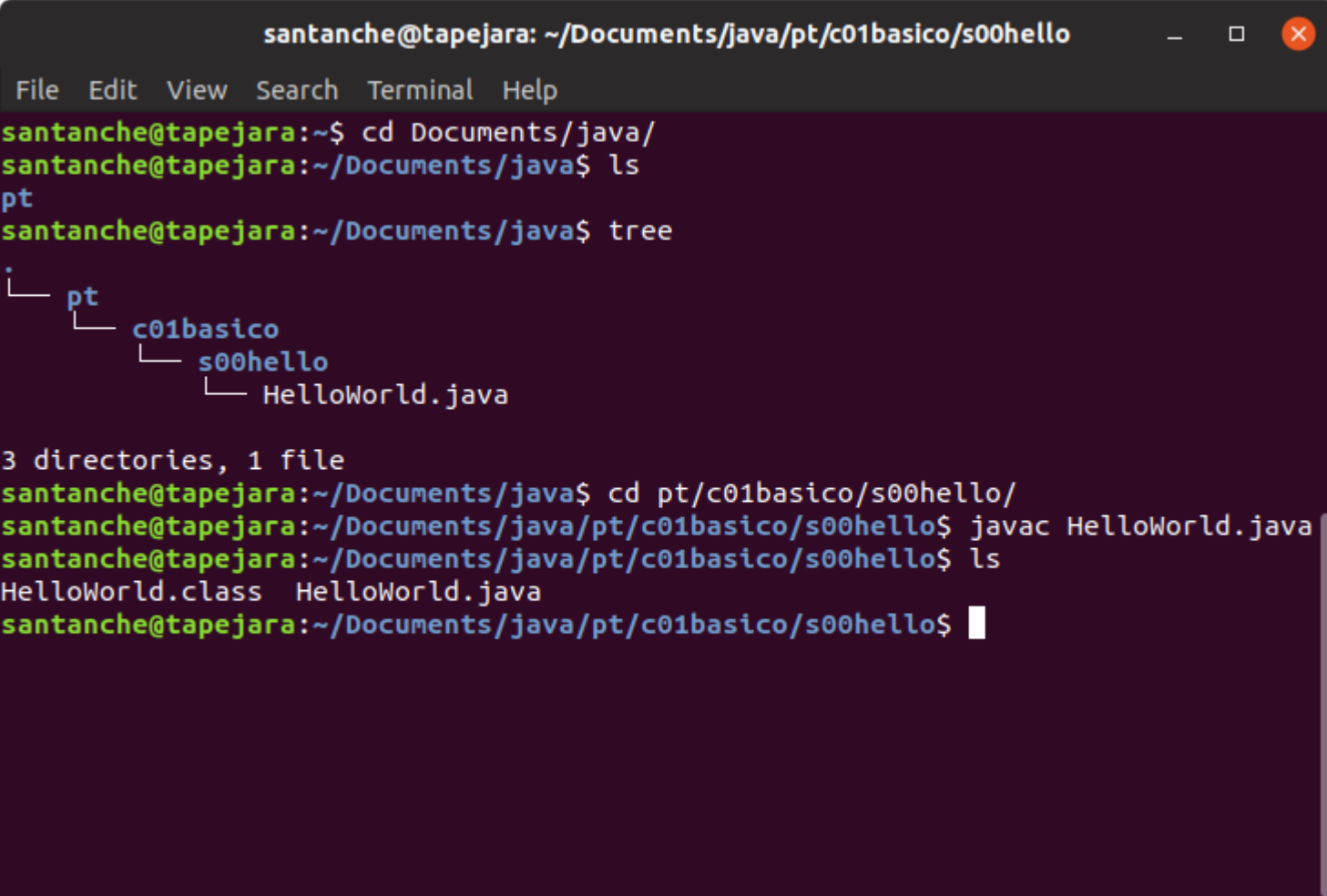
```
santanche@tapejara: ~/Documents/java
File Edit View Search Terminal Help
santanche@tapejara:~$ cd Documents/java/
santanche@tapejara:~/Documents/java$ ls
pt
santanche@tapejara:~/Documents/java$ tree
.
├── pt
│   ├── c01basico
│   │   └── s00hello
│   │       └── HelloWorld.java
└──
```

3 directories, 1 file  
santanche@tapejara:~/Documents/java\$ █

# Compilação

- No diretório do arquivo fonte

```
javac <arquivo fonte>
```



```
santanche@tapejara: ~/Documents/java/pt/c01basico/s00hello
File Edit View Search Terminal Help
santanche@tapejara:~$ cd Documents/java/
santanche@tapejara:~/Documents/java$ ls
pt
santanche@tapejara:~/Documents/java$ tree
.
├── pt
│   ├── c01basico
│   │   └── s00hello
│   │       └── HelloWorld.java
└──
```

3 directories, 1 file

```
santanche@tapejara:~/Documents/java$ cd pt/c01basico/s00hello/
santanche@tapejara:~/Documents/java/pt/c01basico/s00hello$ javac HelloWorld.java
santanche@tapejara:~/Documents/java/pt/c01basico/s00hello$ ls
HelloWorld.class HelloWorld.java
santanche@tapejara:~/Documents/java/pt/c01basico/s00hello$
```

# Execução

- No diretório acima da raiz do pacote:

```
java <hierarquia pacotes>.<classe>
```

```
santanche@tapejara: ~/Documents/java
File Edit View Search Terminal Help
santanche@tapejara:~$ cd Documents/java/
santanche@tapejara:~/Documents/java$ ls
pt
santanche@tapejara:~/Documents/java$ tree
.
├── pt
│   ├── c01basico
│   │   └── s00hello
│   │       └── HelloWorld.java
└──
3 directories, 1 file
santanche@tapejara:~/Documents/java$ cd pt/c01basico/s00hello/
santanche@tapejara:~/Documents/java/pt/c01basico/s00hello$ javac HelloWorld.java
santanche@tapejara:~/Documents/java/pt/c01basico/s00hello$ ls
HelloWorld.class HelloWorld.java
santanche@tapejara:~/Documents/java/pt/c01basico/s00hello$ cd ../../..
santanche@tapejara:~/Documents/java$ java pt.c01basico.s00hello.HelloWorld
O dinossauro pulou na lama.
santanche@tapejara:~/Documents/java$
```

# Argumentos

```
public static void main (String args[])
```

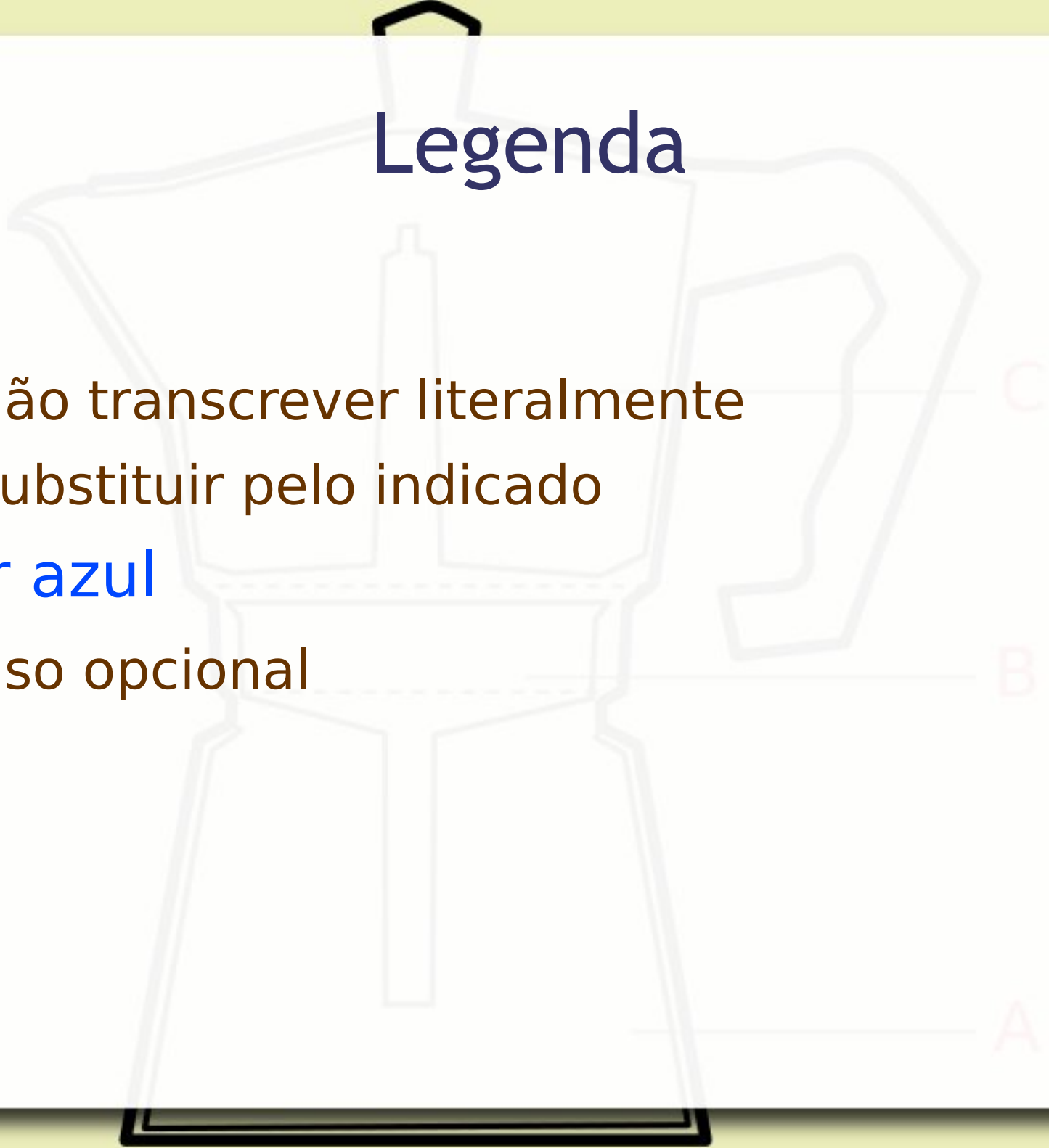
- `args[]` - vetor de argumentos
- `args.length` - número de argumentos

# Aplicação Básica - Application

<pre>public class HelloWorld {</pre>	Início da classe basicoX
<pre>    public static void main(String argumentos[])     {         System.out.println("Piteco e Tecodonte.");     }</pre>	Método principal - primeiro a ser acionado
<pre>}</pre>	Fechamento da classe

# Legenda

- < >
  - não transcrever literalmente
  - substituir pelo indicado
- Cor azul
  - uso opcional







**Argumentos da linha de comandos**

# Declaração de Variável

`<tipo> <declaração1>, ..., <declaraçãon>;`

- `<tipo>`

- tipo das variáveis

- `<declaração>`

- Sintaxe: `<nomeVariável> = <inicialização>`
- nome da variável usualmente inicia com minúsculas
- `<inicialização>` corresponde a uma expressão com o valor inicial da variável

# Tipos de Dados Simples

## ■ Inteiros:

<u>tipo</u>	<u>bits</u>	<u>faixa valores</u>
byte	8	-128..127
short	16	-32,768..32,767
int	32	-2,147,483,648.. 2,147,483,647
long	64	-9,223,372,036,854,775,808.. 9,223,372,036,854,775,807

## ■ Ponto flutuante (real):

float	32	3.4e-0.38.. 3.4e+0.38
double	64	1.7e-308.. 1.7e+308

## ■ Caractere:

char	16	conjunto de caracteres Unicode
------	----	--------------------------------

## ■ Booleano:

boolean		true, false
---------	--	-------------

# Tipo String

- String é uma classe
- Valores tipo strings são instâncias desta classe
- Tipo de classe especial onde instância pode ser declarada como tipos simples:
  - Ex.:  
String nome = "Asdrubal";
- Comparação
  - `<string1>.equals(<string2>)`
  - `<string1>.equalsIgnoreCase(<string2>)`

# Funções de Conversão de Tipos

- Úteis quando a conversão não é automática
  - `Short.parseShort(<argumento>)`
  - `Integer.parseInt(<argumento>)`
  - `Long.parseLong(<argumento>)`
  - `Float.parseFloat(<argumento>)`
  - `Double.parseDouble(<argumento>)`

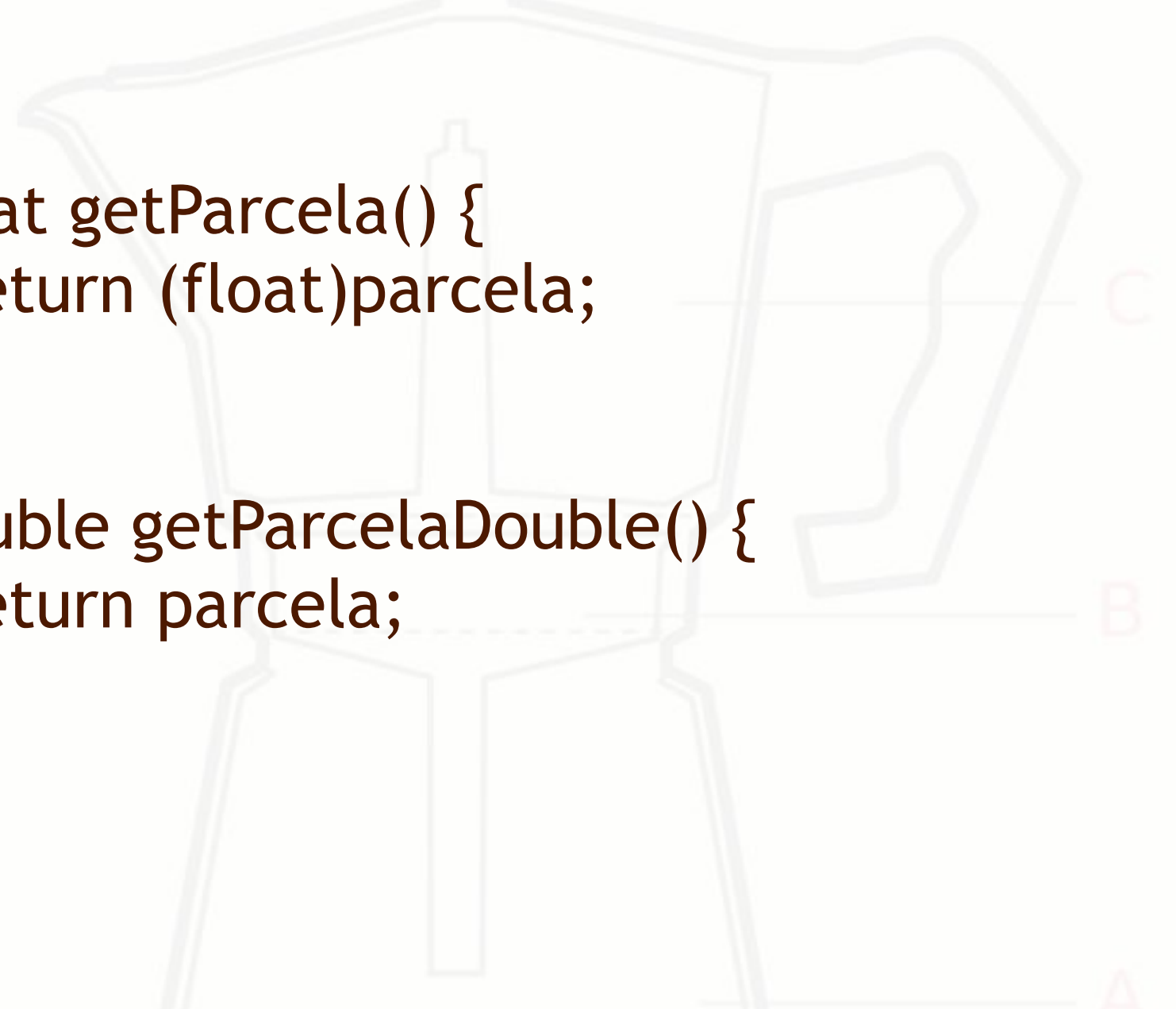
# Cast

- `double x = 50;`  
`float y;`  
`y = (float)x;`

C

B

A



```
■ float getParcela() {  
    return (float)parcela;  
}
```

```
double getParcelaDouble() {  
    return parcela;  
}
```

# Console

## Saída de Dados

`System.out.print(<expressão>)`

- Imprime no console e mantém o cursor na mesma linha

`System.out.println(<expressão>)`

- Imprime no console e pula para a próxima linha

- Constante string entre aspas duplas
- Outros elementos devem ser concatenados usando “+”



# Console

## Entrada de Dados

- Sequência de instruções

- Criação do objeto de entrada de dados

```
Scanner <entrada> = new Scanner(System.in);
```

- Para cada leitura teclado  
teclado.nextLine()

- Função nextLine() retorna String

# Condiciona if

```
if (<condição>)  
  <bloco>
```

```
if (<condição>)  
  <bloco>  
else  
  <bloco>
```

C

B

A

# Partes da Estrutura

- (<condição>)
  - Parênteses são obrigatórios
- <bloco>
  - Apenas uma instrução
    - terminada por ponto-e-vírgula
  - Mais de uma instrução
    - delimitada por chaves { }
    - cada instrução dentro das chaves é encerrada por ponto-e-vírgula

# Condicional switch

```
switch ( <expressão> )  
{  
    case <constante> : <instruções>  
                      break;  
    ...  
    case <constante> : <instruções>  
                      break;  
    default : <instruções>  
}
```

- Desvia para o “case” cujo valor da <constante> é igual ao valor da <expressão>; senão desvia para o default.

# Partes do switch

- (<expressão>)
  - tem que resultar em um valor:
    - char, byte, short, int, Character, Byte, Short, Integer, ou um tipo enumerado
- <constante>
  - tem que ser:
    - char, byte, short, int, Character, Byte, Short, Integer, ou um tipo enumerado
- break
  - interrompe seqüência de instruções; se não for usado a seqüência seguinte é invadida

# Repetição while

```
while (<condição>)  
    <bloco>
```

- testa condição no início

```
do  
    <bloco>  
while (<condição>);
```

- testa condição no final

# Repetição for

```
for (<inicialização>;<condição>;<incremento>)  
  <bloco>
```

- <inicialização>
  - executada antes de entrar no for
  - usualmente inicializa variável de controle
- <condição>
  - testada na entrada e a cada ciclo completo
  - se verdadeira prossegue a repetição
- <incremento>
  - executada a cada ciclo completo
  - usualmente incrementa variável de controle

# Método Estático

```
static <tipo> <nome> ( <param1>, ..., <paramn> )  
{  
    <instruções>  
}
```

- <tipo>
  - tipo de retorno do método
  - “void” (vazio) indica que método não retorna nada
- <nome>
  - nome do método usualmente inicia com minúscula
- <param>
  - parâmetro de entrada do método
  - Sintaxe: <tipo\_parâmetro> <nome\_parâmetro>



# Método Retornando Valores

**return** <expressão>

- retorna resultado da expressão pelo método

# Vetor

## ■ Declaração

```
<tipo>[] <declaração1>, ..., <declaraçãon>;  
<tipo> <declaração1>[], ..., <declaraçãon>[];
```

### ▫ <declaração>


- Sintaxe: <nome> = <inicialização>
- Chaves são usadas para inicializar cada dimensão
- Ex.: `int primos[] = {1, 2, 3, 5, 7};`

## ■ Quando a inicialização não é inline o vetor ou matriz precisa ser instanciado

```
<nome> = new <tipo>[<tamanho>]
```


### ▫ Ex.:

```
int primos[];  
primos = new int[5];
```



André Santanchè

<http://www.ic.unicamp.br/~santanche>



# License

- These slides are shared under a Creative Commons License. Under the following conditions: Attribution, Noncommercial and Share Alike.
- See further details about this Creative Commons license at: <http://creativecommons.org/licenses/by-nc-sa/3.0/>