

## Tipo flutuante: double

### Transcrição

Também queremos trabalhar com outros tipos de variáveis, pois se tentarmos colocar no lugar da idade, em `idade = 37;`, um valor como `37.5`, a compilação não irá ocorrer. O erro que se lê ao passarmos o mouse em cima, é "*Type mismatch: cannot convert from double to int*", isto é, a conversão não é possível. Lembrando que no Java nunca usaremos a vírgula para separar o decimal no código fonte.

Vamos criar uma nova classe para testar os números com **ponto flutuante**, clicando em "(default package)" com o lado direito do mouse e em "New > Class", nomeando-a de "TestaPontoFlutuante". Teremos, então:

```
public class TestaPontoFlutuante {  
  
    public static void main(String[] args) {  
        double salario;  
        salario = 1250.70;  
        System.out.println("meu salário é " + salario);  
    }  
}
```

Há dois tipos de variáveis em que cabem o tal de ponto flutuante, sendo que a mais utilizada é o `double`, como visto acima. Salvaremos o código e o rodaremos! Na aba "Console", obteremos o resultado:

```
meu salário é 1250.7
```

Dica: as *views* podem ser customizadas de acordo com sua necessidade, sendo possível fechar aquelas que não estão em uso, por exemplo.

O `0` (zero) referente aos centavos não apareceram porque é assim que a variável `double` é convertida para se juntar à *string*. Poderemos formatar para aparecerem duas, três casas decimais, ou apenas uma, por meio dos *formatters* do Java, inclusive na biblioteca. Não veremos isto neste curso, porém há diversos recursos disponíveis, como o `printf`, da linguagem C, para colocarmos porcentagens e afins.

No `double` cabem variáveis do tipo inteiro, isto é, poderemos fazer o caminho inverso, indicando que temos uma variável que guarda `idade`, com número `37`. Reparem que este valor não possui decimal. Não tem problema, um número de tipo inteiro cabe em um tipo `double`. O inverso, um decimal em um `int`, é que não é compatível.

O Java tem regras um tanto rígidas, portanto não aceitará `3.0` como `int`, já que não aceita pontos flutuantes. Ele é uma linguagem com peculiaridades que algumas pessoas podem estranhar, sendo vantajoso para se trabalhar em equipe pois reforça uma padronização, e as pessoas trabalham de formas parecidas.

Para enxergarmos a forma como o `double` funciona, podemos fazer uma conta de divisão, por exemplo:

```
double divisao = 3.14 / 2;  
System.out.println(divisao);
```

Rodando o código acima, teremos:

1.57

Parece básico para quem já conhece linguagem estaticamente tipadas, mas mesmo nessa parte mais básica da linguagem, nos aprofundaremos mais, para sentirmos algumas das características do Java.

O que aconteceria no caso de digitarmos `int outraDivisao = 5 / 2;` ? Alguns podem pensar que isso não é compilado, que dará erro, pois o resultado é `2.5` , o que não cabe em `int` .

No Java, entretanto, há uma regra: quando há uma divisão entre dois números inteiros, ele "forçará" um número inteiro como resultado. Se printarmos esta divisão, obteremos como resultado o valor `2` . Estranho, não?

E se quiséssemos que o resultado fosse `2.5` de fato, poderíamos tentar `double novaTentativa = 5 / 2;` , e pediríamos sua impressão, que traria `2.0` . Piorou! O Java irá ler apenas o lado que vem antes da atribuição na linha de código relativa ao `double` . Ou seja, primeiro, ele irá executar a divisão `5 / 2` , e depois ele tentará colocar o resultado em um `double` .

Na verdade, o que gostaríamos é que a conta tivesse sido feita partindo-se do `double` e, neste caso, bastaria que um dos valores da divisão fosse deste tipo, como em `5.0 / 2` . Desta forma, como trata-se de um `double` dividido por um `int` , a conta é feita levando-se em consideração o ponto flutuante.

Esta divisão, sim, trará `2.5` como resultado. Parece pegadinha, mas são características de linguagem que vão te deixar mais a par de como o Java funciona.

Fizemos um truque: estávamos sempre declarando a variável primeiro, e na linha seguinte, fazendo a atribuição. Depois, fizemos `double idade = 37;` . Ao declararmos variáveis, é muito comum inicializá-las, porque é estranho declararmos uma variável, digitarmos um monte de código e só depois inicializarmos um valor.

O comum é fazermos tudo na mesma linha. Desse modo, as linhas abaixo,

```
double salario;  
salario = 1250.70;
```

pela proximidade, equivalem a escrevermos isto:

```
double salario = 1250.70;
```

Declaramos a variável informando seu tipo, e a atualizamos, isto é, inicializamos ela, fazendo uma atribuição.

Vamos salvar o código!