



# Java Foundations

**3-4**

## **Conversão entre Tipos de Dados**

**ORACLE**  
Academy



Copyright © 2020, Oracle e/ou suas empresas afiliadas. Todos os direitos reservados.

# Objetivos

- Esta lição abrange os seguintes objetivos:
  - Tirar proveito da promoção automática
    - E quando ser cauteloso com as promoções
  - Converter variáveis em outros tipos de dados
    - E quando ser cauteloso com a conversão
  - Fazer parse de Strings como valores numéricos



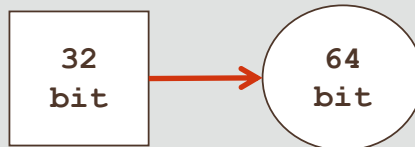
# Parabéns!



- Parabéns por ter chegado até aqui neste curso!
- Uma promoção está chegando no seu caminho!



- Sua promoção:



**ORACLE**  
Academy

JFo 3-4  
Convertendo entre Tipos de Dados

Copyright © 2020, Oracle e/ou suas empresas afiliadas. Todos os direitos reservados.

4

# Indução ao Erro em double

- O que já vimos:

```
double x = 9/2;           //Deve ser 4.5
System.out.println(x);    //imprime 4.0
```

- O Java soluciona a expressão, trunca o 0,5 e, em seguida, transforma a resposta em um double

- Simplificando o cenário, vemos:

```
double x = 4;
System.out.println(x);    //imprime 4.0
```

- Estamos atribuindo um valor inteiro a uma variável dupla
- O Java promove o valor inteiro para um duplo

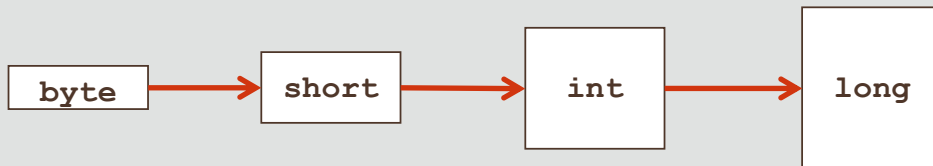
32 bits

64 bits

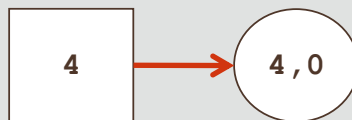
# Promoção

- Promoções automáticas:

- Se você atribuir um tipo pequeno a um tipo maior:



- Se você atribuir um valor inteiro a um tipo de ponto flutuante:



- Exemplos de promoções automáticas:

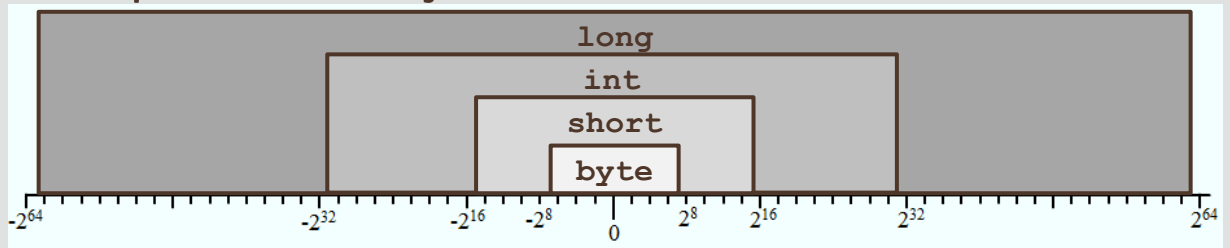
- `long intToLong = 6;`
  - `double intToDouble = 4;`

Em algumas circunstâncias, o compilador altera o tipo de uma variável para um tipo que suporte um valor de tamanho maior. Essa ação denomina-se promoção. Algumas promoções são feitas automaticamente pelo compilador caso isso não ocasione a perda dos dados.

Essas promoções incluem o seguinte:

- Se você atribuir um tipo menor (à direita do sinal de igualdade) a um tipo maior (à esquerda do sinal de igualdade)
- Se você atribuir um tipo integral a um tipo de ponto flutuante (porque não há casas decimais para serem perdidas)

## Por que a Promoção Funciona?



- Um `byte` poderia variar de -128 a 127
- Todos os valores `byte` possíveis podem estar contidos em `short`
- Todos os valores `short` possíveis podem estar contidos em `int`
- Todos os valores `int` possíveis podem estar contidos em `long`
- Todos os valores `int` possíveis podem estar contidos em `double` sem perder a precisão

## Cuidado com a Promoção, Exemplo 1

- Equação:  $55555 * 66666 = 3703629630$
- Exemplo de um possível problema:

```
int num1 = 55555;  
int num2 = 66666;  
long num3;  
num3 = num1 * num2;
```

- Exemplo de uma possível solução:

```
int num1 = 55555;  
long num2 = 66666; ——— Alterado de int para long  
long num3;  
num3 = num1 * num2;
```

Antes de ser atribuído a uma variável, o resultado de uma equação é armazenado em um local temporário na memória. O tamanho desse local sempre é igual ao tamanho de um tipo `int` ou ao tamanho do maior tipo de dados usado na expressão ou na instrução. Por exemplo, se sua equação multiplicar dois tipos `int`, o tamanho do recipiente será um tipo `int` em tamanho ou 32 bits.

Se os dois valores que você multiplica resultarem em um valor que está além do escopo de um tipo `int` (como  $55555 * 66666 = 3.703.629.630$ , que é muito grande para caber em um tipo `int`), o valor `int` deverá ser truncado para que o resultado caiba no local temporário na memória. Esse cálculo acaba resultando em uma resposta incorreta porque a variável da sua resposta recebe um valor truncado, independentemente do tipo usado para sua resposta. Para resolver esse problema, defina pelo menos uma das variáveis na sua equação como o tipo `long` para garantir o maior tamanho de recipiente temporário possível.



## Cuidado com a Promoção, Exemplo 2

- Equação:  $7 / 2 = 3.5$
- Exemplo de um possível problema:

```
int num1 = 7;  
int num2 = 2;  
double num3;  
num3 = num1 / num2;           //num3 é 3.0
```

- Exemplo de uma possível solução:

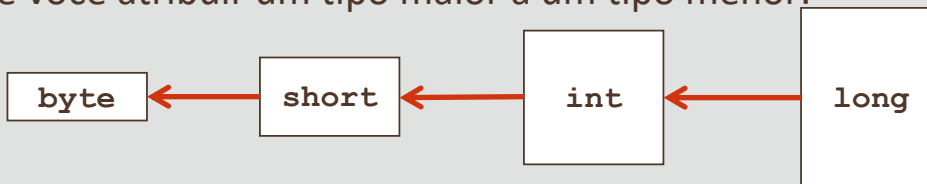
```
int num1 = 7;  
double num2 = 2;               ——— Alterado de int para double  
double num3;  
num3 = num1 / num2;           //num3 é 3.5
```

Quando um operador leva como operandos tipos de dados diferentes de ser atribuído a uma variável, o resultado de uma equação é armazenado em um local temporário na memória. O tamanho do local é sempre igual ao tamanho do maior tipo de dados usado na expressão ou na instrução. Por exemplo, se sua equação dividir dois tipos `int`, o tamanho do recipiente será um tipo `int` em tamanho ou 32 bits.

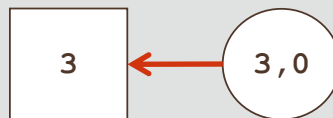
# Conversão de Tipo

- Quando converter:

- Se você atribuir um tipo maior a um tipo menor:



- Se você atribuir um tipo de ponto flutuante a um tipo integral:



- Exemplos de conversão:

- `int longToInt = (int) 20L;`
- `short doubleToShort = (short) 3.0;`

**ORACLE**  
Academy

JFo 3-4  
Convertendo entre Tipos de Dados

Copyright © 2020, Oracle e/ou suas empresas afiliadas. Todos os direitos reservados.

10

A conversão de tipo reduz o intervalo de um valor, quase que literalmente limitando esse valor a um tamanho menor por meio da alteração do tipo do valor (por exemplo, convertendo um valor `long` em `int`). Você faz isso para usar métodos que aceitem apenas determinados tipos de argumentos. Então, você pode atribuir valores a uma variável de um tipo de dados menor ou pode economizar memória.

A sintaxe para a conversão de tipo de um valor é **identificador = (tipo\_alvo) valor**.

Na sintaxe:

- **identificador** é o nome que você atribui à variável.
- **valor** é o valor que você deseja atribuir ao identificador.
- **(tipo\_alvo)** é o tipo para o qual você deseja converter o tipo do valor. Note que **tipo\_alvo** deve ser colocado entre parênteses.

## Cuidado com a Conversão de Tipo

- Esteja ciente sobre a perda de precisão
- Exemplo de um possível problema:

```
int myInt;  
double myPercent = 51.9;  
myInt = (int)myPercent; // O número é "cortado"  
                        // myInt é 51
```

Se você converter um tipo de um valor `float` ou `double` com uma parte fracionária para um tipo integral, como um `int`, todos os valores decimais serão perdidos. No entanto, esse método de conversão de tipo às vezes é útil se você quer truncar o número para baixo para o número inteiro (por exemplo, 51,9 torna-se 51).

## Cuidado com a Conversão de Tipo

- Exemplo de um possível problema:

```
int myInt;  
long myLong = 123987654321L;  
myInt = (int)myLong; //O número é "cortado"  
                    //myInt é -566397263
```

- Exemplo mais seguro de conversão:

```
int myInt;  
long myLong = 99L;  
myInt = (int)myLong; //Não há perda de dados; só zeros.  
                    //myInt é 99
```

A perda de precisão com a conversão às vezes pode levar a situações em que os números são truncados, ocasionando erros nos cálculos.

# Cortando um Integral

- Os exemplos que vimos levantam algumas questões:
  - O que significa "cortar" um integral?
  - Por que estamos recebendo valores negativos?
- É o momento de começar outra investigação com...
  - Conversão
  - Math





## Exercício 1

- Importe e edite o projeto `Casting01`
- Declare e inicialize um `byte` com o valor 128:
  - Observe a reclamação do NetBeans
  - Remova essa linha problemática
- Declare e inicialize um `short` com o valor 128:
  - Crie uma instrução de impressão que converta esse `short` em um `byte`
- Declare e inicialize um `byte` com o valor 127
  - Some 1 a essa variável e imprima-a
  - Some 1 a essa variável e imprima-a novamente



## Resultados da Investigação

- Um `byte` pode ter um valor entre -128 e 127
  - 128 é o primeiro valor positivo que pode estar contido dentro de `short`, mas não dentro de `byte`
  - Tentar converter uma variável com o valor 128 em `byte` é o mesmo que atribuir a `byte` o valor 127 e somar +1
- Tentar incrementar uma variável além de seu valor máximo resulta em seu valor mínimo
  - O espaço do valor de uma variável é disposto ao redor
  - Quando isso ocorre, diz-se que houve o estouro de uma variável
- 127 em binário é 01111111; 128 em binário é 10000000
  - O Java usa o primeiro bit em um número para indicar (+/-)

Considere essas informações básicas. Não é necessário lembrar esses fatos para terminar conjuntos de problemas. É mais importante entender como promover e converter.

## Suposições do Compilador para Tipos de Dados Integral e de Ponto Flutuante

- A maioria das operações resulta em `int` ou `long`
  - Os valores `byte`, `short` e `char` são promovidos automaticamente para `int` antes de uma operação
  - Se uma expressão contiver `long`, toda ela será promovida para `long`
- Se uma expressão contiver um tipo flutuante, toda ela será promovida para um tipo flutuante
- Todos os valores de pontos flutuantes literais são vistos como `double`

O compilador da tecnologia Java faz suposições específicas quando calcula expressões. Você deve entender essas suposições para fazer as conversões de tipo apropriadas ou outras acomodações. Os próximos slides dão exemplos.



## Opções para Corrigir Problemas

- Exemplo de um possível problema:

```
int num1 = 53;           //32 bits de memória para conter o valor
int num2 = 47;           //32 bits de memória para conter o valor
byte num3;               //8 bits de memória reservados
num3 = (num1 + num2);    //causa erro do compilador
```

- Um byte deve ser capaz de conter o valor 100
- Mas o Java recusa-se a fazer a atribuição e emite um erro de "possível perda de precisão"
- O Java considera que adicionar variáveis int resultará em um valor que estouraria o espaço alocado para um byte

## Opções para Corrigir Problemas

- Solução usando um tipo de dados maior:

```
int num1 = 53;  
int num2 = 47;  
int num3; ——— Alterado de byte para int  
num3 = (num1 + num2);
```

- Solução usando a conversão:

```
int num1 = 53;           //32 bits de memória para conter o valor  
int num2 = 47;           //32 bits de memória para conter o valor  
byte num3;               //8 bits de memória reservados  
num3 = (byte)(num1 + num2); //não há perda de dados
```

Para corrigir um erro de “possível perda de precisão”, você pode (a) declarar a variável no lado esquerdo (num3) como sendo um tipo de dados maior como `int` ou (b) converter o tipo de dados à direita para corresponder ao tipo de dados à esquerda.

# Promoção Automática

- Exemplo de um possível problema:

```
short a, b, c;  
a = 1 ;  
b = 2 ;  
c = a + b ;
```

a and b are automatically promoted to integers  
Compiler error results

- Exemplo de possíveis soluções:
  - Declare c como um tipo int na declaração original:  
- `int c;`
  - Converta o tipo do resultado (a+b) na linha de atribuição:  
- `c = (short) (a+b) ;`

No exemplo do slide, um erro ocorre porque dois dos três operandos (a e b) são promovidos automaticamente de um tipo `short` para um tipo `int` antes de serem adicionados.

Na última linha, os valores de a e b são convertidos em tipos `int`, e os valores convertidos são adicionados a um resultado `int`. Então, o operador de atribuição (=) tenta atribuir o resultado `int` à variável `short` (c). No entanto, essa atribuição é inválida e causa um erro no compilador.

# Usando um Longo

```
public class Person {
```

```
    public static void main(String[] args){  
        int ageYears = 32;  
        int ageDays = ageYears * 365;  
        long ageSeconds = ageYears * 365 * 24L * 60 * 60;
```

O uso do L para indicar um longo fará com que o compilador reconheça o resultado total como um longo

```
        System.out.println("Você já tem " + ageDays  
                            + " dias de idade.");  
        System.out.println("Você já tem " + ageSeconds  
                            + " segundos de idade.");
```

```
    } //fim do método main  
} //fim da classe Person
```

O exemplo de código a seguir usa princípios desta seção para calcular a idade de uma pessoa em dias e em segundos. Como a variável `ageSeconds` é declarada como `long`, um dos valores literais usados como operando na expressão atribuída deve ser inicializado como um valor `long` ('L') para que o compilador permita a atribuição.

# Usando Pontos Flutuantes

- Exemplo de possível problema:

```
int num1 = 1 + 2 + 3 + 4.0;
int num2 = (1 + 2 + 3 + 4) * 1.0;
```

//erro do compilador  
//erro do compilador

As expressões são promovidas

automaticamente a pontos flutuantes

- Exemplo de possíveis soluções:
  - Declare num1 e num2 como tipos double:

```
double num1 = 1 + 2 + 3 + 4.0; //10.0
double num2 = (1 + 2 + 3 + 4) * 1.0; //10.0
```

- Converta num1 e num2 como tipos int na linha de atribuição:

```
int num1 = (int)(1 + 2 + 3 + 4.0); //10
int num2 = (int)((1 + 2 + 3 + 4) * 1.0); //10
```

Se uma expressão contiver um tipo flutuante, toda ela será promovida para um tipo flutuante.

# Atribuição e Tipos de Dados de Ponto Flutuante

- Exemplo de possível problema:

```
float float1 = 27.9; //erro do compilador
```

- Exemplo de possíveis soluções:

- O F avisa ao compilador que 27,9 é um valor float:

```
float float1 = 27.9F;
```

- 27,9 é conversão para um tipo float:

```
float float1 = (float) 27.9;
```

Da mesma forma que os tipos integrais assumem `int` como padrão em algumas circunstâncias, os valores atribuídos a tipos de pontos flutuantes sempre assumem como padrão um tipo `double`, a menos que você informe especificamente que o valor é um tipo `float`. Isso é feito inserindo um F maiúsculo depois do valor de um número. Caso contrário, 27,9 será considerado como sendo um tipo `double`. Ocorre um erro do compilador porque um tipo `double` não pode se ajustar em uma variável `float`.



## Exercício 2

- Importe e edite o projeto `Casting02`
- Existem vários erros neste programa
- Você deve ser capaz de corrigir esses erros usando...
  - Seu conhecimento sobre tipos de dados
  - Seu conhecimento sobre promoção
  - Seu conhecimento sobre conversão

# O Sublinhado

- Você deve ter percebido o uso de sublinhados (\_):
  - A partir do Java SE7, você pode incluir sublinhados ao atribuir valores numéricos
  - Os sublinhados ajudam a tornar os números grandes mais legíveis
  - Os sublinhados não afetam o valor de uma variável
- As duas instruções a seguir são equivalentes:

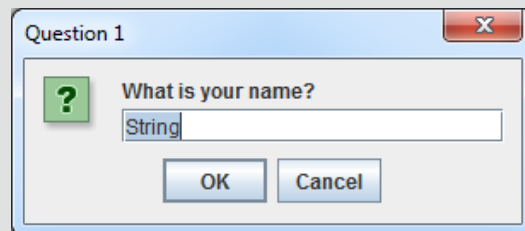
```
int x = 123_456_789;
```

```
int x = 123456789;
```



# Convertendo Strings em Dados Numéricos

- Quando você convida um usuário a digitar em uma caixa de diálogo...
  - Ele pode digitar o que quiser
  - Este texto é melhor representado por uma String
- Às vezes, você precisará fazer cálculos com entradas de usuário
  - Se projetar um programa que aceite entrada de texto, pode ser que você precise converter a String em um tipo de dados numérico



## Fazendo Parse de Strings

- A conversão de texto em dados numéricos é uma forma de parse
- Como converter uma String em int:

```
int intVar1 = Integer.parseInt("100");
```

- Como converter uma String em double:

```
double doubleVar2 = Double.parseDouble("2.72");
```



## Exercício 3, Parte 1

- Importe e edite o projeto `Parsing01`
- Declare e inicialize 3 Strings com os seguintes dados:

Variável de String	Descrição	Exemplo de Valores
shirtPrice	O texto pode ser convertido em int:	"15"
taxRate	O texto pode ser convertido em double:	"0.05"
gibberish	Gibberish	"887ds7nds87dsfs"



## Exercício 3, Parte 2

- Faça parse e multiplique `shirtPrice*taxRate` para encontrar o imposto
  - Imprima esse valor
- Tente fazer parse de `taxRate` como `int`
  - Observe a mensagem de erro
- Tente fazer parse de `gibberish` como `int`
  - Observe a mensagem de erro

# Problema com Entrada de Usuário

- NumberFormatException

- Ocorre porque não é possível fazer parse de um valor
- É arriscado quando os usuários podem inserir qualquer valor que desejarem



```
int intVar1 = Integer.parseInt("Puppies!");
```

- O software não deve travar devido a uma entrada do usuário

- Mas ignore isso por enquanto
- Primeiro, vamos imaginar como obter a entrada do usuário na próxima lição
- Aprenderemos a tratar erros e exceções na Seção 8

# Resumo

- Nesta lição, você deverá ter aprendido a:
  - Tirar proveito da promoção automática
    - E quando ser cauteloso com as promoções
  - Converter variáveis em outros tipos de dados
    - E quando ser cauteloso com a conversão
  - Fazer parse de Strings como valores numéricos



