



ORACLE

Academy



Java Foundations

3-4

Conversión entre Tipos de Datos

ORACLE
Academy



Objetivos

- En esta lección se abordan los siguientes objetivos:
 - Aprovechar las ventajas de las promociones automáticas
 - Y saber cuándo tener cuidado con los promociones
 - Convertir variables en otros tipos de Datos
 - Y saber cuándo tener cuidado con las conversiones
 - Analizar cadenas como valores numéricos



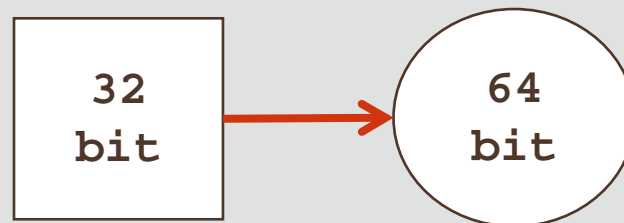
Enhorabuena.



- Enhorabuena por ir tan avanzado en el curso
- ¡Se avecina una promoción!



- Su promoción:



Doble Decepción

- Esto es lo que hemos visto antes:

```
double x = 9/2;           //Should be 4.5
System.out.println(x);    //prints 4.0
```

- Java resuelve la expresión, trunca el .5 y, a continuación, convierte la respuesta en un valor `double`

- Dicho de forma más sencilla:

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

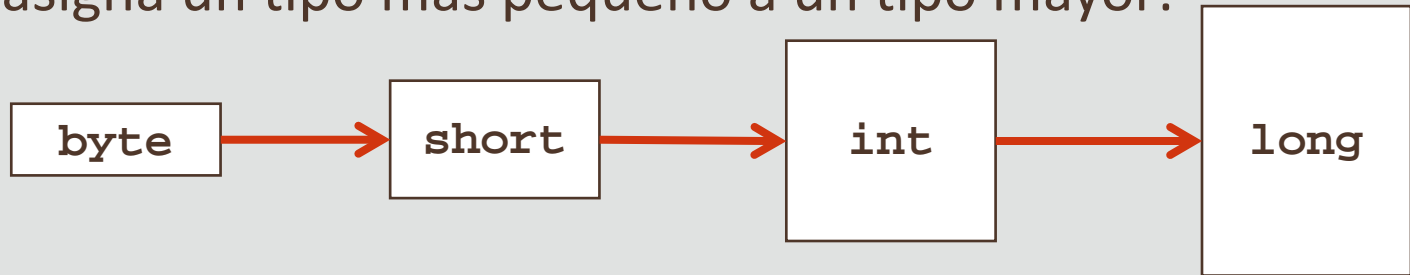
- Estamos asignando un valor entero a una variable `double`
- Java asciende el valor entero a una variable `double`

32 bits

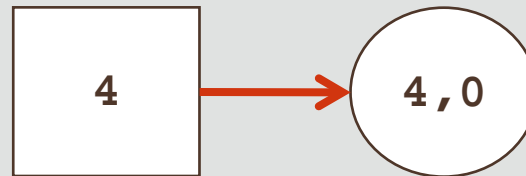
64 bits

Promotion

- Ampliaciones automáticas:
 - Si asigna un tipo más pequeño a un tipo mayor:

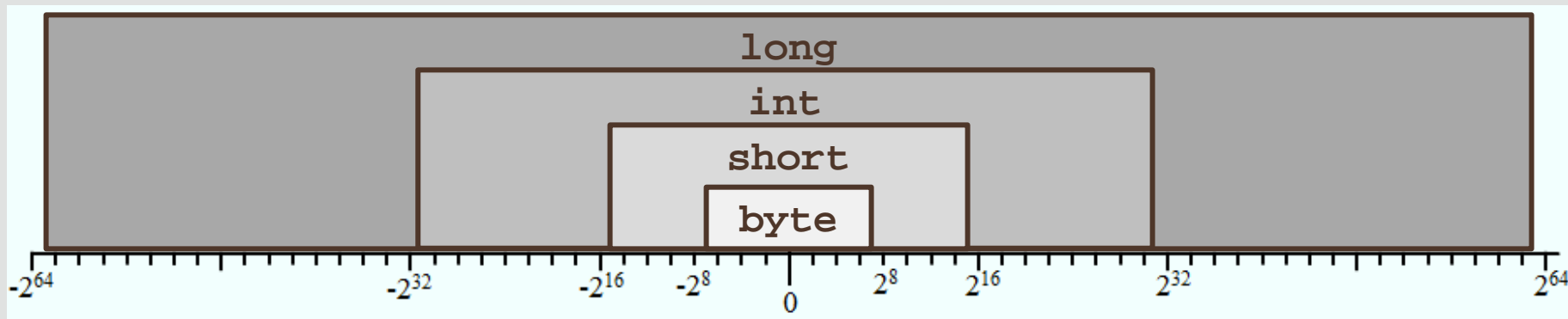


- Si asigna un valor integral a un tipo de coma flotante:



- Ejemplos de promociones automáticas:
 - `long intToLong = 6;`
 - `double intToDouble = 4;`

¿Por qué funcionan las promociones?



- Un valor **byte** podría estar comprendido entre -128 y 127
- Todos los valores **byte** posibles pueden estar incluidos en un valor **short**
- Todos los valores **short** posibles pueden estar incluidos en un valor **int**
- Todos los valores **int** posibles pueden estar incluidos en un valor **long**
- Todos los valores **int** posibles pueden estar incluidos en un valor **double** sin perder precisión



Cuidado con las promociones, Ejemplo 1

- Ecuación: $55555 * 66666 = 3703629630$
- Ejemplo de un posible problema:

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

- Ejemplo de una posible solución:

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

———— Ha cambiado de int a long



Cuidado con las promociones, Ejemplo 2

- Ecuación: $7 / 2 = 3.5$
- Ejemplo de un posible problema:

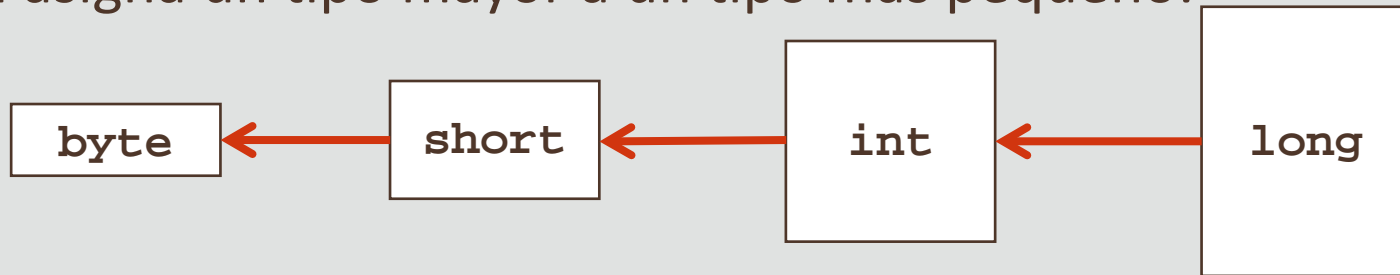
```
int num1 = 7;  
int num2 = 2;  
double num3;  
num3 = num1 / num2;           //num3 is 3.0
```

- Ejemplo de una posible solución:

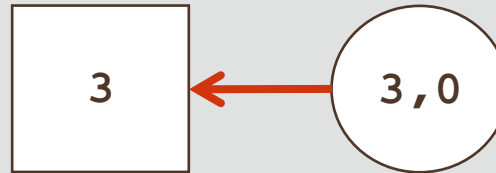
```
int num1 = 7;  
double num2 = 2;              ————— Se ha cambiado de int a double  
double num3;  
num3 = num1 / num2;           //num3 is 3.5
```

Conversión de tipo

- Cuándo realizar una conversión:
 - Si asigna un tipo mayor a un tipo más pequeño:



- Si asigna un tipo de coma flotante a un tipo integral:



- Ejemplos de conversión:
 - `int longToInt = (int) 20L;`
 - `short doubleToShort = (short) 3.0;`



Cuidado con la Conversión de Tipo

- Preste especial atención a la pérdida de precisión
- Ejemplo de un posible problema:

```
int myInt;  
double myPercent = 51.9;  
myInt = (int)myPercent; // Number is "chopped"  
                        // myInt is 51
```



Cuidado con la Conversión de Tipo

- Ejemplo de un posible problema:

```
int myInt;  
long myLong = 123987654321L;  
myInt = (int)myLong; // Number is "chopped"  
                      // myInt is -566397263
```

- Ejemplo de conversión más seguro:

```
int myInt;  
long myLong = 99L;  
myInt = (int)myLong; // No data loss, only zeroes.  
                      // myInt is 99
```

Cortar una Integral

- Los ejemplos que hemos visto pueden plantear algunas dudas:
 - ¿Qué significa "cortar" una integral?
 - ¿Por qué estamos obteniendo valores negativos?
- Es el momento de iniciar otra investigación utilizando...
 - Conversiones
 - Matemáticas



Ejercicio 1

- Cree un nuevo proyecto y agréguele el archivo `Casting01.java`
- Declare e inicialice un tipo `byte` con un valor de 128:
 - Observe la queja de NetBeans
 - Comente esta línea problemática
- Declare e inicialice un tipo `short` con un valor de 128:
 - Cree una sentencia print que convierta este `short` en un valor `byte`
- Declare e inicialice un tipo `byte` con un valor de 127
 - Agregue 1 a esta variable e imprímala
 - Vuelva a agregar 1 a esta variable e imprímala de nuevo



Resultados de la Investigación

- Un tipo `byte` puede tener un valor comprendido entre -128 y 127
 - 128 es el primer valor positivo que puede estar incluido en un tipo `short`, pero no en un tipo `byte`
 - Intentar convertir una variable con un valor de 128 en un `byte` es como asignar a un tipo `byte` un valor de 127 e incrementarlo en +1
- Al intentar incrementar una variable por encima de su valor máximo, se obtiene como resultado su valor mínimo
 - El espacio de valor de una variable se ajusta
 - Cuando esto sucede, se dice que la variable ha sufrido un desbordamiento
- 127 en binario es 01111111; 128 en binario es 10000000
 - Java utiliza el primer bit de un número para indicar el signo (+/-)

Suposiciones del compilador para tipos de Datos integrales y de coma flotante

- La mayoría de las operaciones dan como resultado un valor `int` o `long`
 - Los valores `byte`, `short` y `char` se ascienden automáticamente a `int` antes de llevar a cabo una operación
 - Si una expresión contiene un valor `long`, la expresión entera se asciende a `long`
- Si una expresión contiene una coma flotante, la expresión entera se asciende a una coma flotante
- Todos los valores literales con coma flotante se ven como `double`

Opciones para solucionar problemas

- Ejemplo de un posible problema:

```
int num1 = 53;           // 32 bits of memory to hold the value
int num2 = 47;           // 32 bits of memory to hold the value
byte num3;               // 8 bits of memory reserved
num3 = (num1 + num2);    // causes compiler error
```

- Un `byte` debe poder tener un valor de 100
- Pero Java se niega a realizar esta asignación y emite un error de "posible pérdida de precisión"
- Java asume que al agregar variables `int`, se obtendría un valor que desbordaría el espacio asignado a un `byte`



Opciones para solucionar problemas

- Solución basada en un tipo de Datos mayor:

```
int num1 = 53;  
int num2 = 47;  
int num3;  
num3 = (num1 + num2);
```

Ha cambiado de byte a int

- Solución basada en conversiones:

```
int num1 = 53;           // 32 bits of memory to hold the value  
int num2 = 47;           // 32 bits of memory to hold the value  
byte num3;               // 8 bits of memory reserved  
num3 = (byte)(num1 + num2); // no data loss
```

Promociones automáticas

- Ejemplo de un posible problema:

```
short a, b, c;  
a = 1 ;  
b = 2 ;  
c = a + b ; //compiler error
```

a y b se ascienden automáticamente a valores enteros

- Ejemplo de posibles soluciones:

- Declarar c como tipo `int` en la declaración original:
 - `int c;`
- Convertir el tipo del resultado de (a+b) en la línea de asignación:
 - `c = (short)(a+b);`

Uso de un Valor Long

```
public class Person {  
  
    public static void main(String[] args){  
        int ageYears = 32;  
        int ageDays = ageYears * 365;  
        long ageSeconds = ageYears * 365 * 24L * 60 * 60;  
  
        System.out.println("You are " + ageDays + " days old.");  
        System.out.println("You are " + ageSeconds + " seconds old.");  
  
    } //end of main method  
} //end of class
```

Usar L para indicar un valor long hará que el compilador reconozca el resultado total como un valor long

Uso de Comas Flotantes

- Ejemplo de un posible problema:

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

Las expresiones se ascienden

- Ejemplo de posibles soluciones: automáticamente a comas flotantes

- Declarar num1 y num2 como tipos `double`:

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

- Convertir el tipo num1 y num2 en tipos `int` en la línea de asignación:

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

Tipos de Datos de coma flotante y asignación

- Ejemplo de un posible problema:

```
float float1 = 27.9; //compiler error
```

- Ejemplo de posibles soluciones:

- La F notifica al compilador que 27.9 es un valor `float`:

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

- 27.9 se convierte en un tipo `float`:

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


Ejercicio 2

- Cree un nuevo proyecto y agréguele el archivo `Casting02.java`
- Este programa presenta varios errores
- Debería poder corregir estos errores usando sus conocimientos sobre...
 - Tipos de Datos
 - Promociones
 - Conversiones

Carácter de subrayado

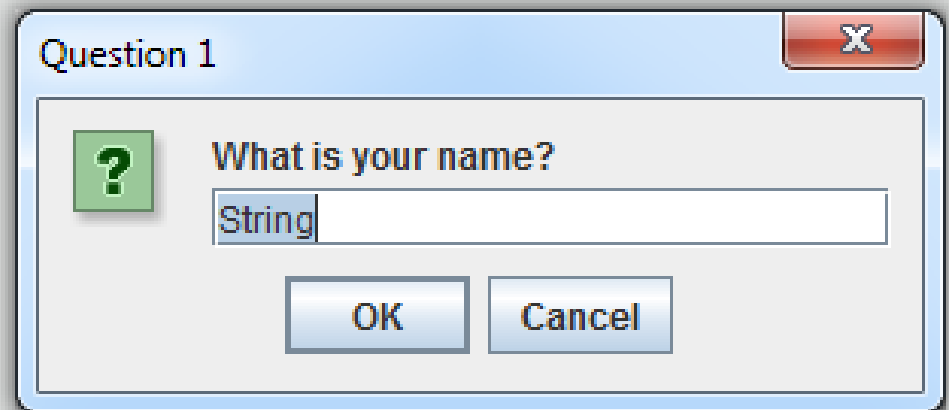
- Es posible que haya notado la presencia de caracteres de subrayado (_):
 - A partir de Java SE7, se pueden incluir caracteres de subrayado al asignar valores numéricos
 - Los caracteres de subrayado facilitan la lectura de los números grandes
 - Los caracteres de subrayado no afectan al valor de una variable
- Las dos sentencias siguientes son equivalentes:

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

```
int x = 123456789;
```

Conversión de Cadenas en Datos Numéricos

- Cuando le pide a un usuario que escriba en un cuadro de diálogo...
 - Pueden escribir el texto que quieran
 - La mejor forma de representar este texto es mediante un valor String
- Pero a veces tendrá que hacer operaciones matemáticas con las entradas de los usuarios
 - Si diseña un programa que acepte la entrada de texto, es posible que tenga que convertir el valor String en tipos de Datos numéricos



Análisis de cadenas

- Convertir texto en Datos numéricos es una forma de análisis
- Cómo convertir un valor String en `int`:

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

- Cómo convertir un valor String en `double`:

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

Ejercicio 3, parte 1

- Cree un nuevo proyecto y agregue el archivo `Parsing01.java` al proyecto
- Declare e inicialice 3 valores `String` con los Datos siguientes:

Variable de String	Descripción	Valores de ejemplo
shirtPrice	Texto que se va a convertir en un valor <code>int</code> :	"15"
taxRate	Texto que se va a convertir en un valor <code>double</code> :	"0.05"
gibberish	Texto sin sentido	"887ds7nds87dsfs"

Ejercicio 3, parte 2

- Analice y multiplique `shirtPrice*taxRate` para averiguar el impuesto
 - Imprima este valor
- Intente analizar `taxRate` como un valor `int`
 - Lea el mensaje de error
- Intente analizar `gibberish` como un valor `int`
 - Lea el mensaje de error

Problemas con las Entradas de los Usuarios

- NumberFormatException

- Se produce porque un valor no se puede analizar
- Este es un riesgo que se corre si los usuarios pueden introducir el texto que quieran



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

- El software no debería fallar como consecuencia de las entradas de los usuarios
 - Pero vamos a ignorar esto de momento
 - En primer lugar, vamos a ver cómo obtener entradas de usuarios en la siguiente lección
 - Aprenderemos a manejar errores y excepciones en la sección 8

Resumen

- En esta lección, debe haber aprendido lo siguiente:
 - Aprovechar las ventajas de los ascensos automáticos
 - Y saber cuándo tener cuidado con los ascensos
 - Convertir variables en otros tipos de Datos
 - Y saber cuándo tener cuidado con las conversiones
 - Analizar cadenas como valores numéricos





ORACLE

Academy

