Universidad Europea

4

Tipos de datos

Ve más allá

CONTENIDOS

- **Unidad 1: Fundamentos del lenguaje Java**
  1. Nuestro primer programa
  2. Sentencias y errores
  3. Variables y constantes
  4. **Tipos de datos**
  5. Operadores y expresiones
  6. Entrada y Salida

Universidad Europea

ÍNDICE

- Tipos de datos
  1. ¿Qué son los tipos de datos?
  2. Clasificación
  3. Tipos primitivos
  4. Valor inicial y cambios de valor
  5. Ejercicios
  6. Conversiones implícitas
  7. Conversiones explícitas (cast)
  8. Envoltorios

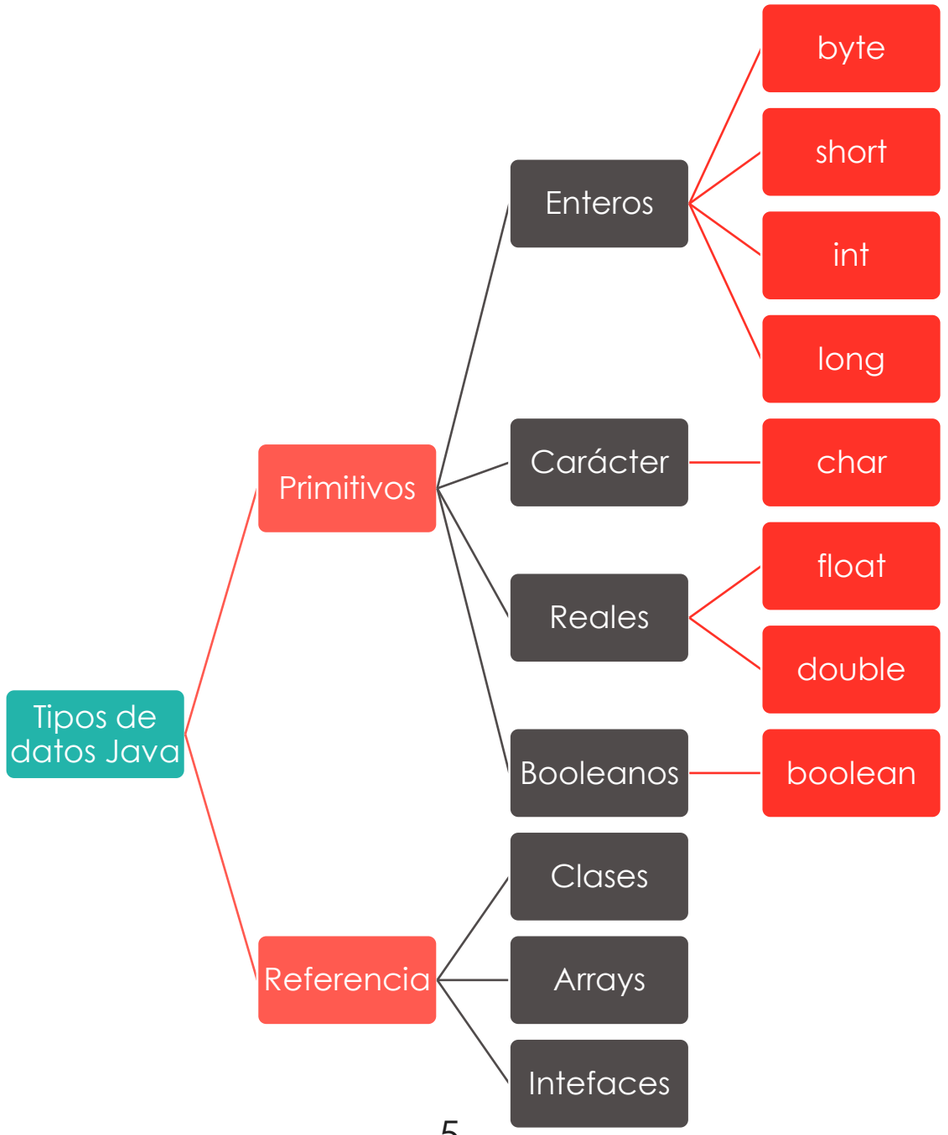
Universidad Europea

¿Qué son los tipos de datos?

Elegir bien el tipo de datos es esencial para construir buenos programas. Imagina que estás construyendo un parking para coches, motos y camiones. Como es lógico, el tamaño de cada plaza debe ser diferente para que cada tipo de vehículo pueda caber sin problemas en su interior. Con las variables pasa lo mismo. En este apartado veremos los diferentes tipos de datos que tiene Java, cómo inicializarlos, así como las conversiones que se pueden hacer de un tipo a otro.

Java es muy tipado, es decir, que necesitamos declarar una variable (indicar el tipo de dato) antes de utilizarla. Además, una vez elegido no podemos cambiarlo.

Clasificación



Tipos primitivos

Tipo de datos	Información representada	Rango	Descripción
byte	Datos enteros	-128 ⟷ +127	Se utilizan 8 bits (1 byte) para almacenar el dato.
short	Datos enteros	-32768 ⟷ +32767	Dato de 16 bits de longitud (independientemente de la plataforma).
int	Datos enteros	-2147483648 ⟷ +2147483647	Dato de 32 bits de longitud (independientemente de la plataforma).
long	Datos enteros	-9223372036854775808 ⟷ +9223372036854775807	Dato de 64 bits de longitud (independientemente de la plataforma).
char	Datos enteros y caracteres	0 ⟷ 65535	Este rango es para representar números en unicode, los ASCII se representan con los valores del 0 al 127. ASCII es un subconjunto del juego de caracteres Unicode.
float	Datos en coma flotante de 32 bits	Precisión aproximada de 7 dígitos	Dato en coma flotante de 32 bits en formato IEEE 754 (1 bit de signo, 8 para el exponente y 24 para la mantisa).
double	Datos en coma flotante de 64 bits	Precisión aproximada de 16 dígitos	Dato en coma flotante de 64 bits en formato IEEE 754 (1 bit de signo, 11 para el exponente y 52 para la mantisa).
boolean	Valores booleanos	true/false	Utilizado para evaluar si el resultado de una expresión booleanas es verdadero (true) o falso(false).

Tipos primitivos

Tipo de dato	Código
byte	byte a;
short	short b, c=3;
int	int d = -30; int e = 0xC125;
long	long b=434123 ; long b=5L ; /* la L en este caso indica Long*/
char	char car1='c'; char car2=99; /*car1 y car2 son lo mismo porque el 99 en decimal es la 'c' */
float	float pi=3.1416; float pi=3.1416F; /* la F en este caso indica Float*/ float medio=1/2F; /*0.5*/
double	double millón=1e6; /* 1x106 */ double medio1/2D; /*0.5 la D en este caso indica Double*/

Valor inicial y cambios de valor

Cuando se declara una variable, es un buen momento para que le des un valor inicial (si lo conoces en ese momento). Si tratas de utilizar una variable que todavía no has inicializado, el compilador te dará un error. Mira esta variable, su nombre es velocidad y su tipo es de tipo real (doble).

```
double velocidad = 98.15;
```

Aunque todavía no hayamos visto el tipo Date (que en realidad es una clase) podemos entender la siguiente instrucción:

```
Date nacimiento = new Date ();
```

Donde la variable “nacimiento” es de tipo “Date” y se le ha dado un valor (con new Date()).

Valor inicial y cambios de valor

Valor inicial y cambios de valor

Las variables se llaman variables porque pueden variar. Observa este ejemplo:

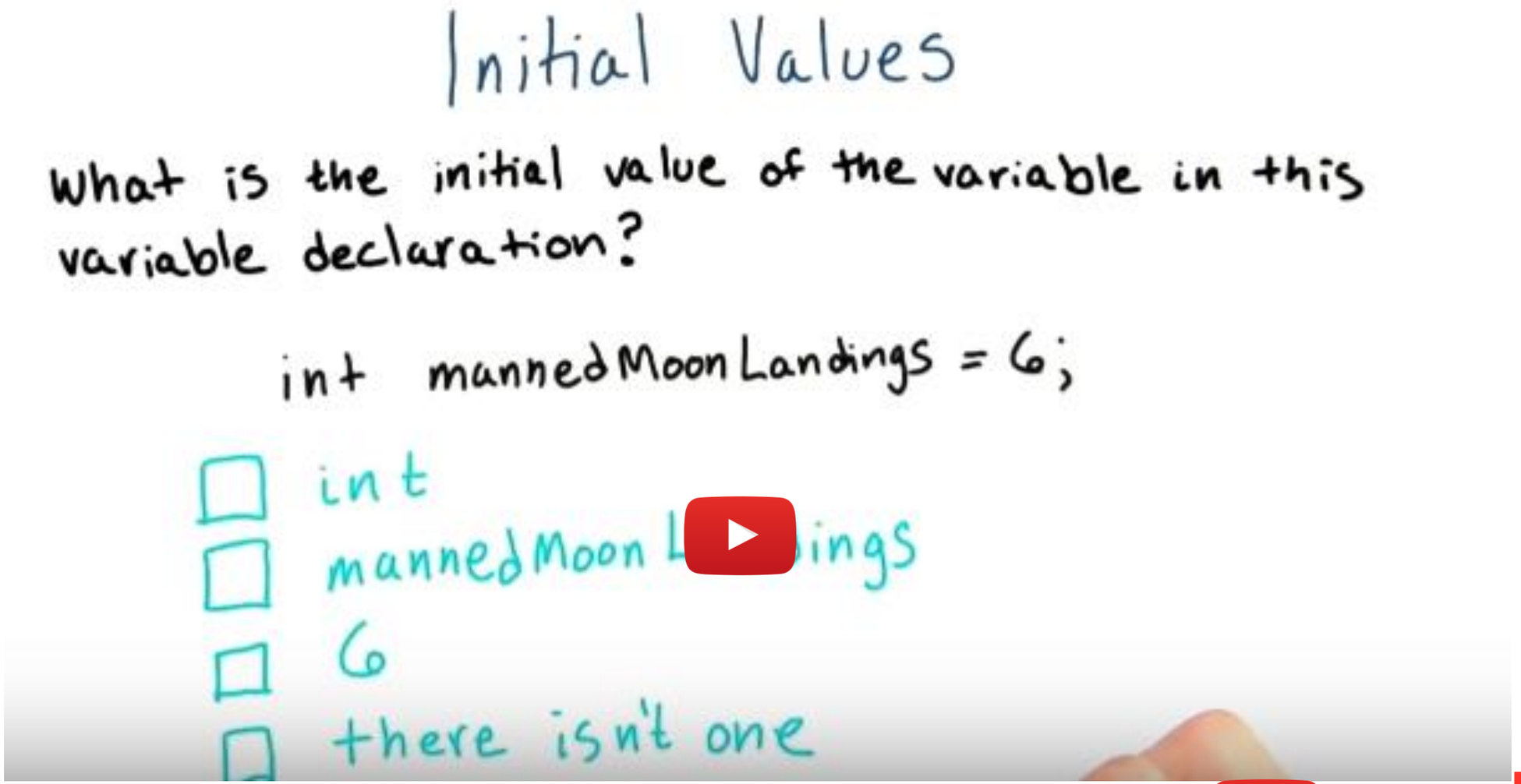
```
int cantidadManzanas = 25;
```

Esta es una variable cuyo valor inicial es 25, pero puedo variar esto si surge la necesidad. Tal vez más tarde quieras reducir su valor en 5 unidades, para lo que tendrías que hacer:

```
cantidadManzanas = cantidadManzanas – 5;
```

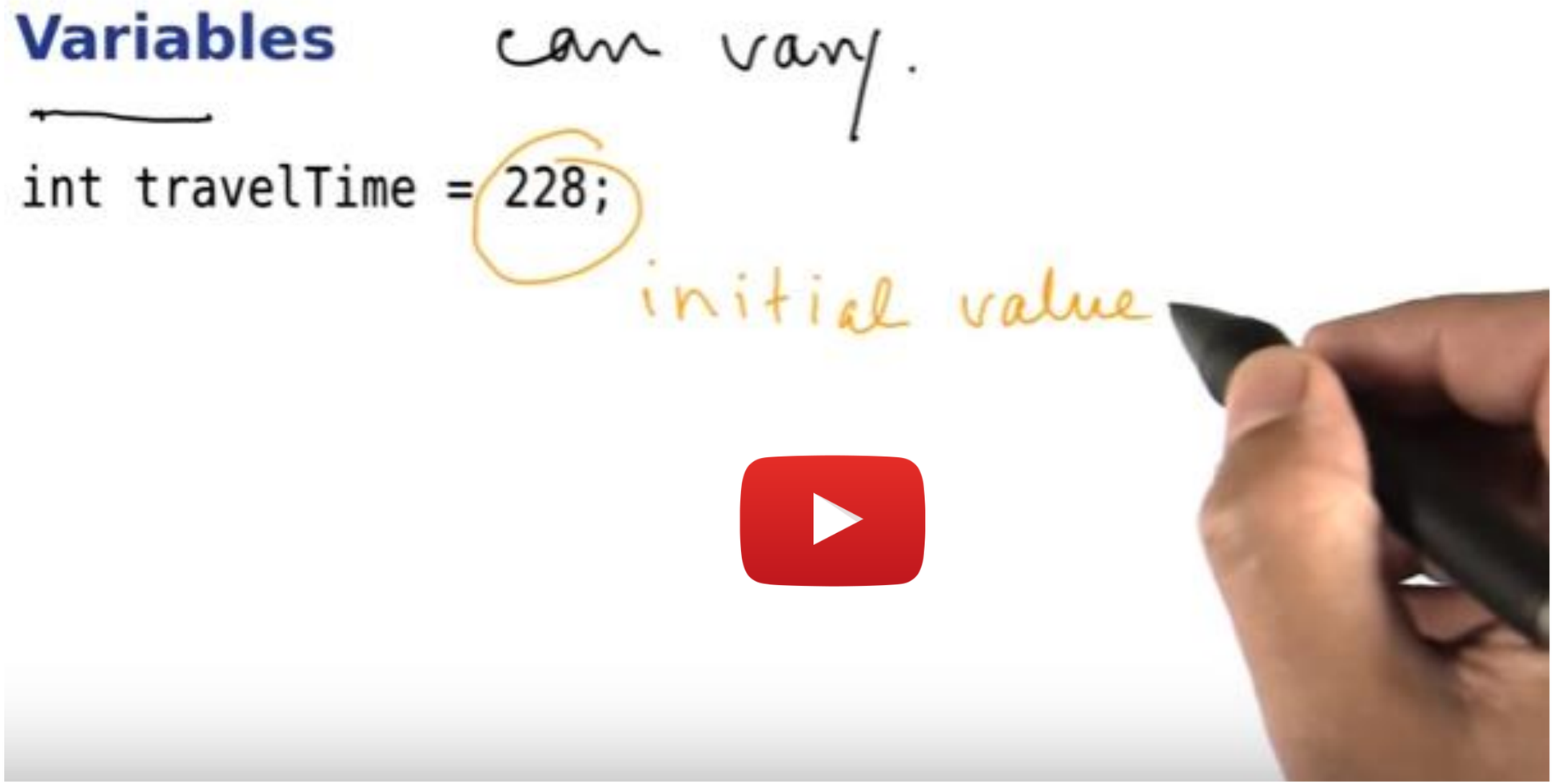
Lo que también tienes que saber, es que una vez que modificas su valor, el anterior se pierde.

Valor inicial y cambios de valor

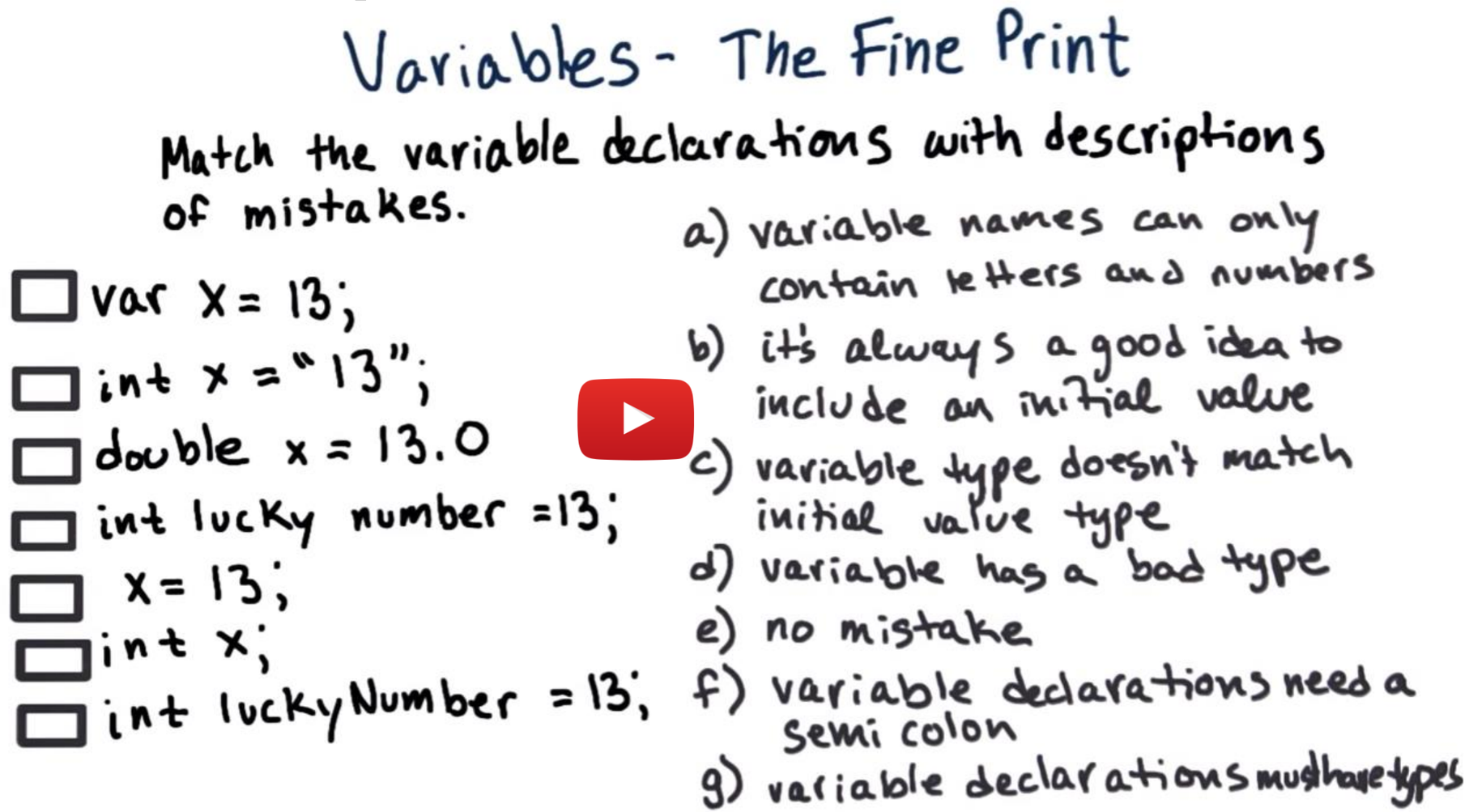


<- solución

Valor inicial y cambios de valor



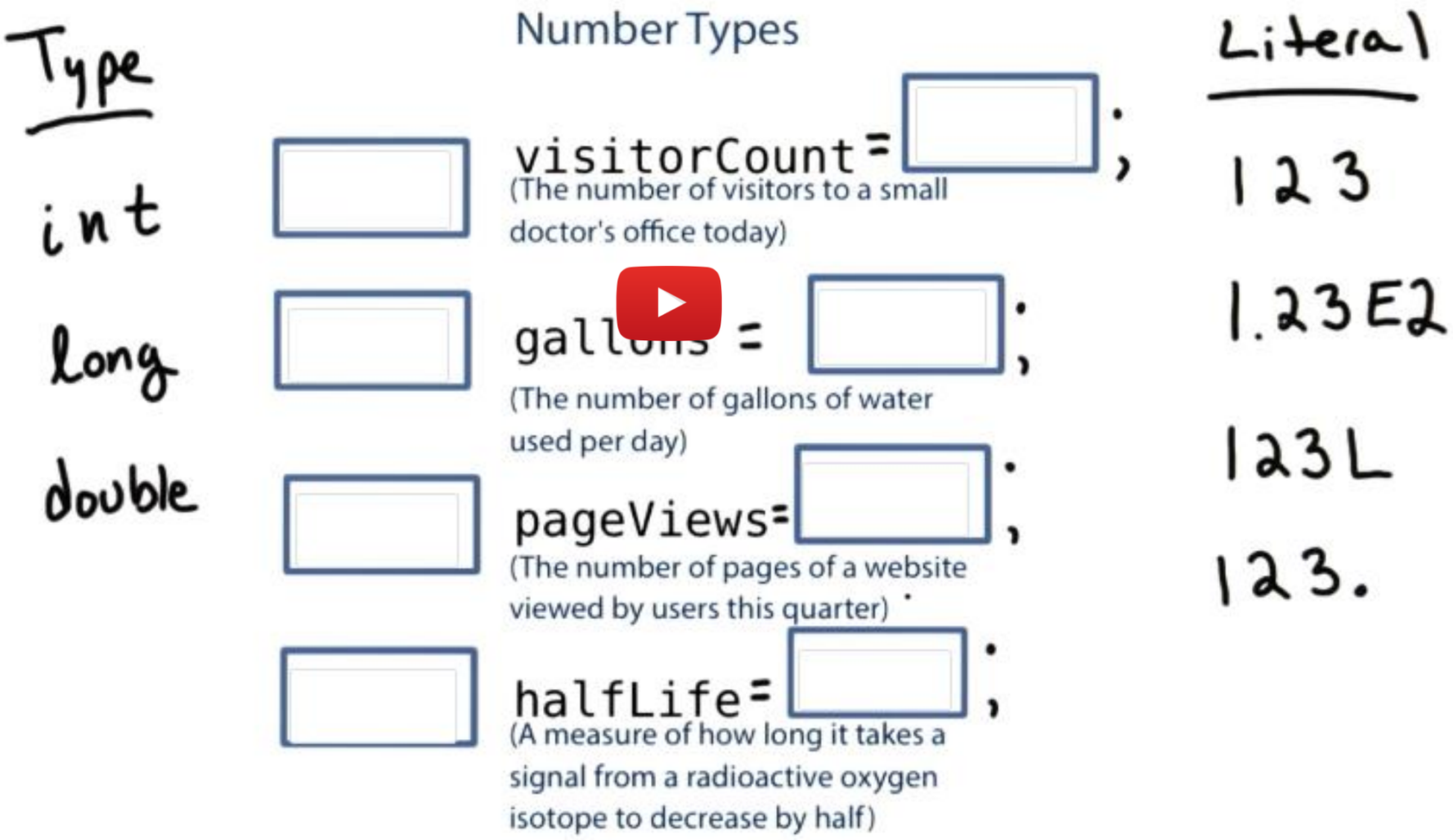
Valor inicial y cambios de valor



solución ->

Universidad Europea

Valor inicial y cambios de valor



<- solución

Universidad Europea

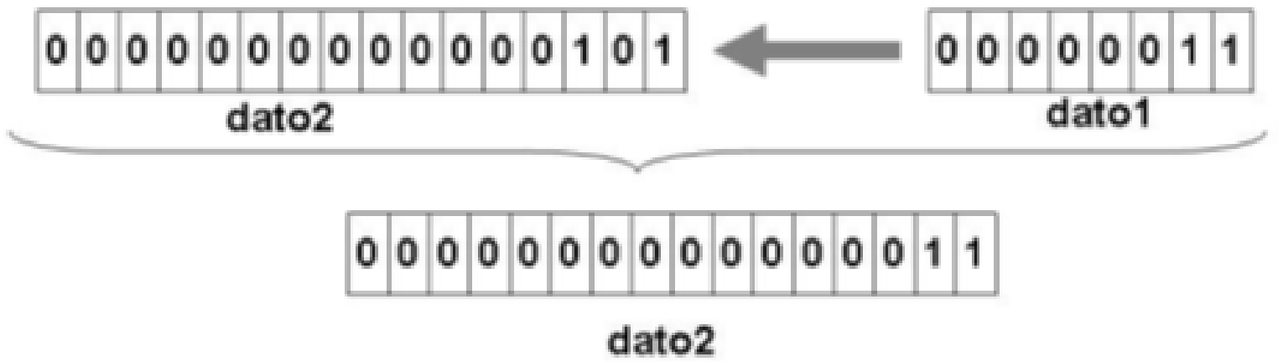
Conversiones implícitas

Se realiza de forma automática entre dos tipos de datos diferentes. Requiere que la variable destino (la colocada a la izquierda) tenga más precisión que la variable origen (situada a la derecha).

Por ejemplo:

```
byte dato1 = 3;
short dato2 = 5;
dato2 = dato1;
```

Internamente lo que ha pasado es lo siguiente:



Conversiones explícitas

También se conocen como cast o casting. Veamos el siguiente ejemplo:

```
double precio = 4.25;

int centimos = precio * 100;
```

Casts

```
double price = 4.35;

int pennies = 4.35 * 100;
```

- Yes, it works – I get 435 pennies
- No, I get a compiler error
- No, I get 434 pennies
- No, I get 434.999999994 pennies

Conversiones explícitas

Una forma de solucionar este error es utilizar la conversión mediante la operación llamada cast, que tiene el siguiente formato:

```
(tipo) expresión
```

Lo que estamos haciendo es “forzar” que la expresión que se encuentra a la derecha del tipo entre paréntesis sea de ese tipo. Como es lógico, no se pueden realizar conversiones entre enteros y booleanos o reales y booleanos.

Conversiones explícitas

Casts

```
double price = 4.35;

int pennies = 4.35 * 100;
```

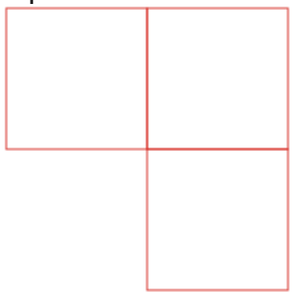
double

double

```
int pennies = (int)(4.35 * 100);
```

double

<- solución



Conversiones explícitas

Otro ejemplo de conversión explícita sería el siguiente:

```
int idato=5;

byte bdato;

bdato = (byte) idato;

System.out.println(bdato); // sacará 5 por pantalla
```

Envoltorios

Fijaos que todos ellos tienen nombres prácticamente idénticos a los tipos básicos, pero con la diferencia de que la primera letra es mayúscula (recordad que Java es sensible a mayúsculas).

Los envoltorios son muy útiles para realizar conversiones entre tipos de manera sencilla, ya que la API nos ofrece métodos para hacer casi todo lo que deseemos.

Por ejemplo, la conversión de un número entero guardado como una cadena de caracteres, es una tarea muy sencilla en Java:

```
int numeroConvertido = Integer.parseInt(numeroString);
```

Fijaos que hemos utilizado el método parseInt() de un objeto Integer. Este método recibe como parámetro un String y devuelve el número entero correspondiente de hacer la conversión de ese String.

Nota: En este ejemplo podéis apreciar que, como no teníamos ningún objeto Integer, hemos podido utilizar el mismo nombre de la clase para hacer uso de su método. Son facilidades que nos da Java.

REFERENCIAS

BUENAFUENTE - INFORMÁTICA

21

Universidad Europea

Universidad Europea

GRACIAS

Pedro J. Camacho

Universidadeuropea.com

Ve más allá