Curso de Java

Estructura del Lenguaje

Prof. Ing. Guido Acosta

Contenido de la clase

-Variables

-Convención de código Java

-Tipos primitivos

-Operadores

Declaración de variables

Declaración

<tipo de variable> <nombre de variable>;

Declaración y asignación

<tipo de variable> <nombre de variable> = <valor>;

```
// Variable de tipo int
int numeroEntero;

//Múltiples declaraciones en una sola línea
int i, j, l;
```


Identificadores

Un identificador es un nombre que nos permite dirigirnos a las variables, funciones y otros elementos utilizados a la hora de escribir el código de un programa.

Reglas

-Los nombres deben empezar con una letra, con _ o $.

-Los siguientes caracteres pueden ser números, letras, _ o $.

Ejemplos

contador //válido

_siguiente //válido

1erContador //??

INT //??

Contenido de la clase

-Variables

-**Convención de código Java**

-Tipos primitivos

-Operadores

Convenciones

Los identificadores asociados a variables se suelen poner en minúsculas

int Contador; // mal
int contador; // bien

Cuando el identificador está formado por varias palabras, la primera palabra va en minúsculas y el resto de las palabras se una inicia con una letra mayúscula.

int contadorelementos; // mal
int contadorElementos; // bien


Convenciones

Número por línea

Se recomienda una declaración por línea ya que alienta a comentar.

```
int tamaño; // tamaño de la tabla
int cantidad; // cantidad de alumnos
```

Lugar

Declarar las variables al inicio del bloque

```
void myMethod() {
    int int1; // inicio del bloque del método
    if (condicion) {
        int int2; // inicio del bloque if
        ...
    }
}
```

Convención - Importancia

-El 80% del costo del ciclo de vida del software va a mantenimiento.

-Difícilmente un software es mantenido por su autor original.

-La convención de código mejora la lectura del programa


Definición de constante

final <tipo> identificador = <valor>;

Convenciones

Los identificadores asociados a constantes se suelen poner en mayusculas

final int COTIZACION = 4600;

Si el identificador esta formado por varias palabras, las distintas palabras se separan por un guion bajo.

final int COTIZACION_DOLAR = 4600;


Contenido de la clase

-Variables

-Convención de código Java

-Tipos primitivos

-Operadores


Tipos primitivos

Tipo numérico

Números enteros: **byte, short, int, long**

Números en coma flotante: **float, double**

Tipo carácter

char

Tipo booleano

boolean

Números enteros

Tipo de dato	Espacio en memoria	Valor mínimo	Valor máximo
byte	8 bits	-128	127
short	16 bits	-32768	32767
int	32 bits	-2147483648	2147483647
long	64 bits	-9223372036854775808	9223372036854775807

Números enteros

Tipo de dato	Espacio en memoria	Valor mínimo	Valor máximo
byte	8 bits	-128	127
short	16 bits	-32768	32767
int	32 bits	-2147483648	2147483647
long	64 bits	-9223372036854775808	9223372036854775807

¿long o int?

47483648


Literales enteros

- Los literales enteros son de tipo int por defecto.
- Un literal entero es de tipo long si va acompañado del sufijo l o L.
- 237665L es de tipo long.

Desbordamiento de número enteros

```
short numero = 32767;

numero + 1
```


Desbordamiento de número enteros

Tipo	Operación	Resultado
byte	127 + 1	-128
short	32767+1	-32768
int	2147483647+1	-2147483648


Números en coma flotante

Tipo de dato	Espacio en memoria
float	32 bits
double	64 bits

Literales reales

- Cadenas de dígitos con un punto decimal
  234.32 .001

- En notación científica (mantisa.$10^{exponente}$)
  123e45 123E+45 1E-6

- Por defecto, los literales reales representan valores de tipo double.

- Para representar un valor de tipo float, añadir el sufijo f o F.
  123.87F


Caracteres

Tipo de dato	Espacio en memoria	Codificación
char	16 bits	UNICODE

Literales de tipo carácter

Valores entre comillas simples
‘a’ ‘1’ ‘*’


Secuencias de escape para representar caracteres especiales

Secuencia de escape	Descripción
\t	Tabulador
\n	Nueva línea
\r	Retorno de carro
\b	Retroceso
\’	Comillas simples
\”	Comillas dobles
\\	Barra invertida

Secuencias de escape para representar caracteres especiales

Secuencia de escape	Descripción
\t	Tabulador
\n	Nueva línea
\r	Retorno de carro
\b	Retroceso
\’	Comillas simples
\”	Comillas dobles
\\	Barra invertida

```
System.out.println("Bienvenidos alumnos al "primer" día de clase");
```

Secuencias de escape para representar caracteres especiales

Secuencia de escape	Descripción
\t	Tabulador
\n	Nueva línea
\r	Retorno de carro
\b	Retroceso
\’	Comillas simples
\”	Comillas dobles
\\	Barra invertida

System.*out.println("Bienvenidos alumnos al* \"primer\" *día de clase");*


Tipo boolean

Tipo de dato	Espacio en memoria	Valores
boolean	1 bit	Verdadero o falso

- **Los literales son true y false**

Conversión entre tipos primitivos

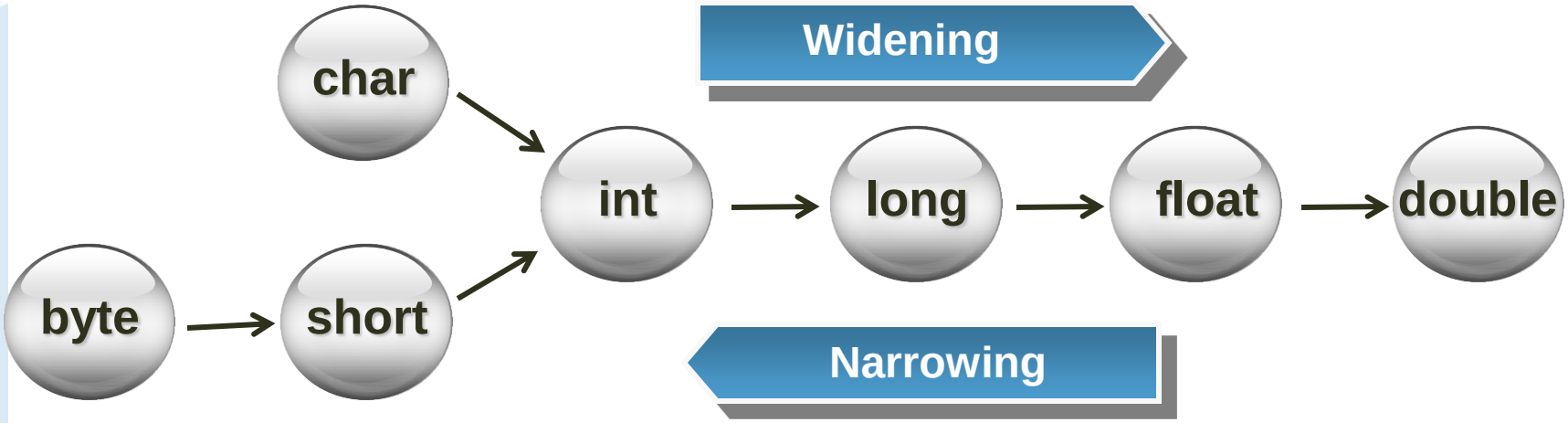
Narrowing: conversión de un tipo primitivo mayor a un tipo primitivo menor. La conversión tiene que ser explícita.

```
double d = 1.5;
int i = (int) d;       // el valor de d queda truncado a 1
```

Widening: conversión de un tipo primitivo menor a un tipo primitivo mayor. La conversión puede ser implícita.

```
short s = 15;
float f = s;
```

Conversión entre tipos primitivos



Promoción aritmética

Se dá cuando se realiza una operación aritmética entre dos tipos primitivos numéricos diferentes. El compilador realiza una operación widening.

```
int intDato = 5;
double  doubleDato = 3;
```

```
doubleDato = intDato * doubleDato;
```

```
intDato = intDato * doubleDato;
```

Promoción aritmética

Se dá cuando se realiza una operación aritmética entre dos tipos primitivos numéricos diferentes. El compilador realiza una operación widening.

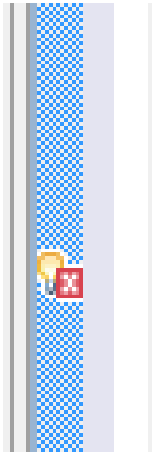
```
int intDato = 5;
double  doubleDato = 3;
```

```
doubleDato = intDato * doubleDato;
```

```
intDato = intDato * doubleDato;
```

```
intDato = (int) (intDato * doubleDato);
```

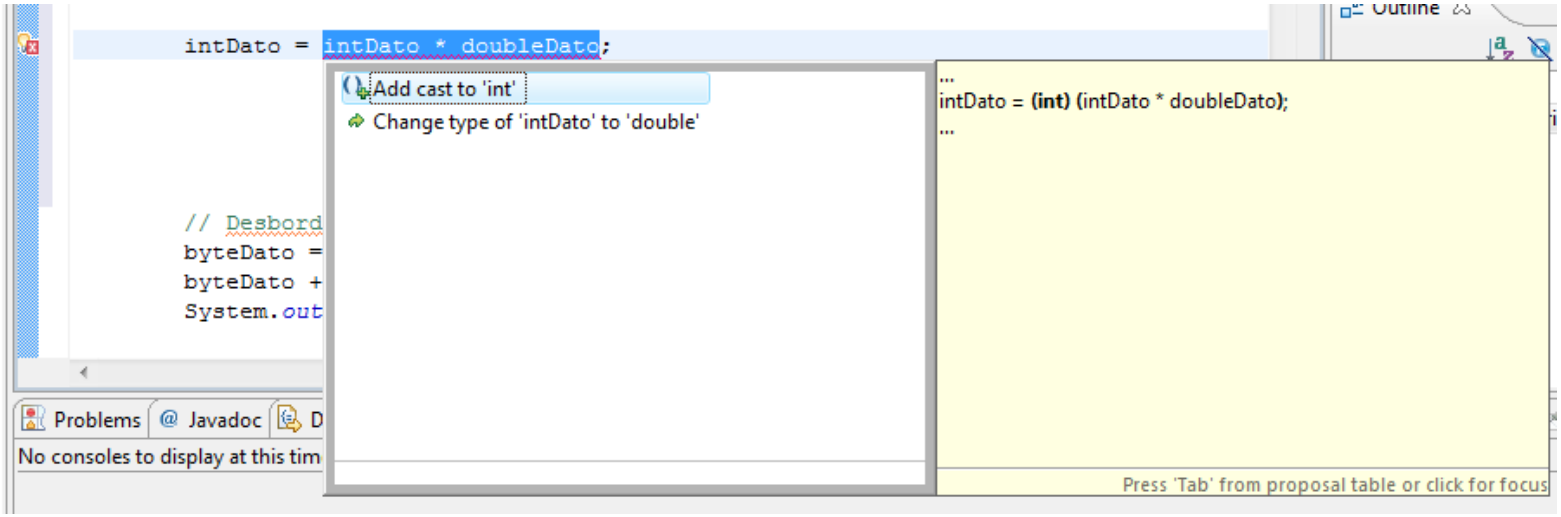
Ayuda Eclipse



```
int intDato = 5;
double doubleDato = 3;

intDato = intDato * doubleDato;
```

Ayuda Eclipse



Contenido de la clase

-Variables

-Convención de código Java

-Tipos primitivos

-Operadores


Incremento y decremento

Descripción	Operador
Incremento	++
Decremento	--

Modos de uso

```
int a = 0;
int b = a++; // incremento después de asignar
int c = ++a; // incremento antes de asignar

b = a--; // decremento después de asignar
c = --a; // decremento antes de asignar
```


Operadores aritméticos

Operador	Operación
+	Suma
-	Resta o cambio de signo
*	Multiplicación
/	División
%	Módulo (resto de la división)

-Si los operandos son enteros se realizan operaciones enteras.
Ejemplo

Operación	Tipo	Resultado
8/3	int	2

Operadores de comparación

Los operadores de comparación son válidos para números y caracteres

Operador	Significado
==	Igual
!=	Distinto
<	Menor
>	Mayor
<=	Menor o igual
>=	Mayor o igual


Operadores lógicos

Operador	Nombre	Significado
!	NOT	Negación lógica
&&	AND	
||	OR	
^	XOR	

Los operadores && || son de corto circuito.

Operadores de asignación

```
int  valor = 10;

valor += 5;          // valor = valor + 5;

valor -= 5;          // valor = valor – 5;

valor *= 5;          valor = valor * 5;

valor  /= 2;         valor = valor / 2;
```