

4

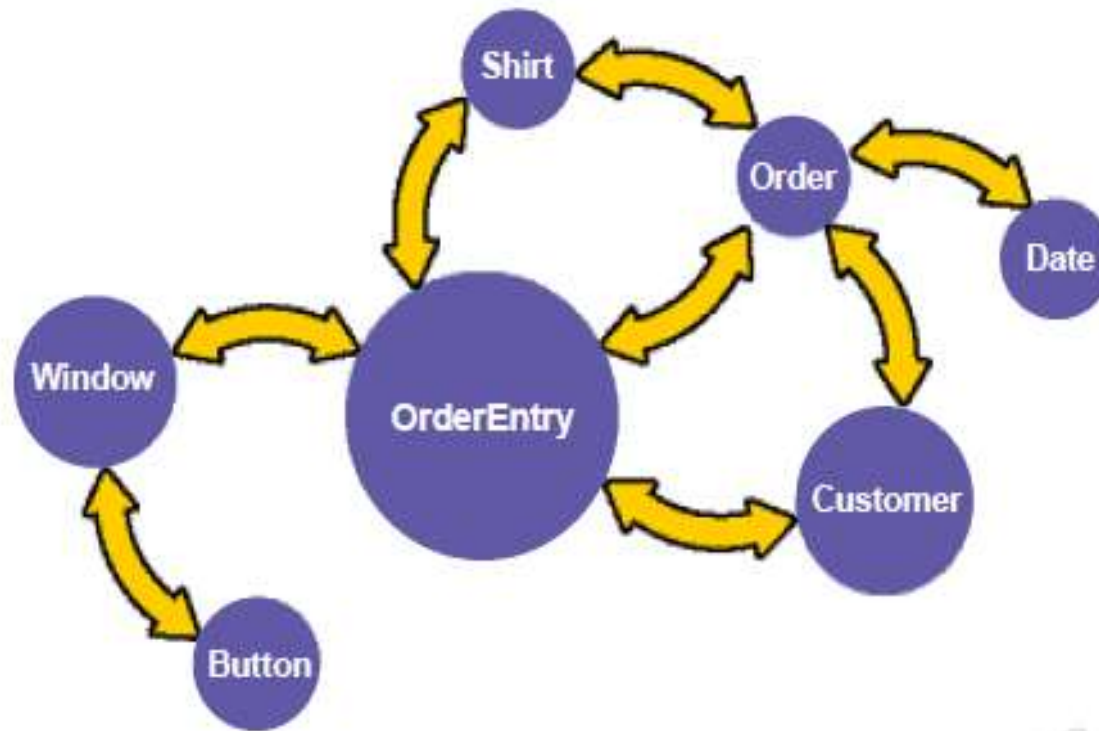
Introducción al lenguaje Java

Importancia

¿Cómo prueba algo que ha creado, como una casa, un mueble o un programa?

(fundacion@proydesa.org) has a
to use this Student Guide.

Identificación de los componentes de una clase

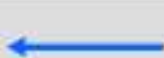


Estructuración de clases

- Declaración de clase
- Declaraciones de campo (los atributos de clase se denominan “campos”)
 - Los campos también se pueden inicializar en el momento de la declaración.
- Métodos (opcionales)
- Comentarios (opcionales)

Estructuración de clases

```
public class Shirt {
```

 **Declaración de clase**

```
    public int shirtID = 0; // Default ID for the shirt
    public String description = "-description required-"; // default
    // The color codes are R=Red, B=Blue, G=Green, U=Unset
    public char colorCode = 'U';
    public double price = 0.0; // Default price for all shirts
    public int quantityInStock = 0;
```

Declaraciones de campo



```
    // This method displays the values for an item
    public void displayInformation() {
        System.out.println("Shirt ID: " + shirtID);
        System.out.println("Shirt description:" + description);
        System.out.println("Color Code: " + colorCode);
        System.out.println("Shirt price: " + price);
        System.out.println("Quantity in stock: " + quantityInStock);
```

Método



```
    } // end of display method
} // end of class
```

Símbolos utilizados en la definición de un origen Java

- Llaves **{ }**
- Paréntesis **()**
- Puntos y comas **;**
- Comas **,**
- Comillas simples **' '**
- Comillas dobles **" "**
- Comentario de una línea **//**

Unión de todo

- Sintaxis para declarar una clase:

```
[modifiers] class class_identifier
```

- Ejemplo de clase:

```
public class Shirt{  
    public double price;
```

```
    public void setPrice(double priceArg)  
    {  
        price = priceArg;  
    }
```

```
}
```

Llaves de
apertura y
de cierre
para la
clase Shirt

Declaraciones y asignaciones de campos

```
public int shirtID = 0;  
public String description = "-description required-";  
public char colorCode = 'U';  
public double price = 0.0;  
public int quantityInStock = 0;
```

adacion@proydesa.org) has a
this Student Guide.

Comentarios

- Una sola línea:

```
public int shirtID = 0; // Default ID for the shirt
public double price = 0.0; // Default price for all shirts

// The color codes are R=Red, B=Blue, G=Green
```

- Tradicional:

```
/******
 *
 * Attribute Variable Declaration Section
 *****/
```

Métodos

- Sintaxis:

```
[modifiers] return_type method_identifier ([arguments]){  
    method_code_block  
}
```

- Ejemplo:

```
public void displayInformation() {  
  
    System.out.println("Shirt ID: " + shirtID);  
    System.out.println("Shirt description:" + description);  
    System.out.println("Color Code: " + colorCode);  
    System.out.println("Shirt price: " + price);  
    System.out.println("Quantity in stock: " + quantityInStock);  
  
} // end of display method
```

Palabras clave

<code>abstract</code>	<code>default</code>	<code>for</code>	<code>package</code>	<code>synchronized</code>
<code>assert</code>	<code>do</code>	<code>if</code>	<code>private</code>	<code>this</code>
<code>boolean</code>	<code>double</code>	<code>implements</code>	<code>protected</code>	<code>throw</code>
<code>break</code>	<code>else</code>	<code>import</code>	<code>public</code>	<code>throws</code>
<code>byte</code>	<code>enum</code>	<code>instanceof</code>	<code>return</code>	<code>transient</code>
<code>case</code>	<code>extends</code>	<code>int</code>	<code>short</code>	<code>true</code>
<code>catch</code>	<code>false</code>	<code>interface</code>	<code>static</code>	<code>try</code>
<code>char</code>	<code>final</code>	<code>long</code>	<code>strictfp</code>	<code>void</code>
<code>class</code>	<code>finally</code>	<code>native</code>	<code>super</code>	<code>volatile</code>
<code>continue</code>	<code>float</code>	<code>new</code>	<code>switch</code>	<code>while</code>

Creación y uso de una clase de prueba

Ejemplo:

```
class ShirtTest {  
  
    public static void main (String[] args) {  
  
        Shirt myShirt;  
        myShirt= new Shirt();  
  
        myShirt.displayInformation();  
  
    }  
}
```

Método main

- Método especial que JVM reconoce como punto de inicio de cada programa de tecnología Java que se ejecuta desde una línea de comandos.
- Sintaxis:

```
public static void main (String[] args)
```

Compilación de un programa

1. Vaya al directorio donde están almacenados los archivos de código fuente.
2. Introduzca el siguiente comando para cada archivo `.java` que desee compilar.

- Sintaxis:

```
javac filename
```

- Ejemplo:

```
javac Shirt.java
```


Ejecución (prueba) de un programa

1. Vaya al directorio en el que están almacenados los archivos de clase.
2. Introduzca lo siguiente para el archivo de clase que contiene el método `main`:

- Sintaxis:

```
java classname
```

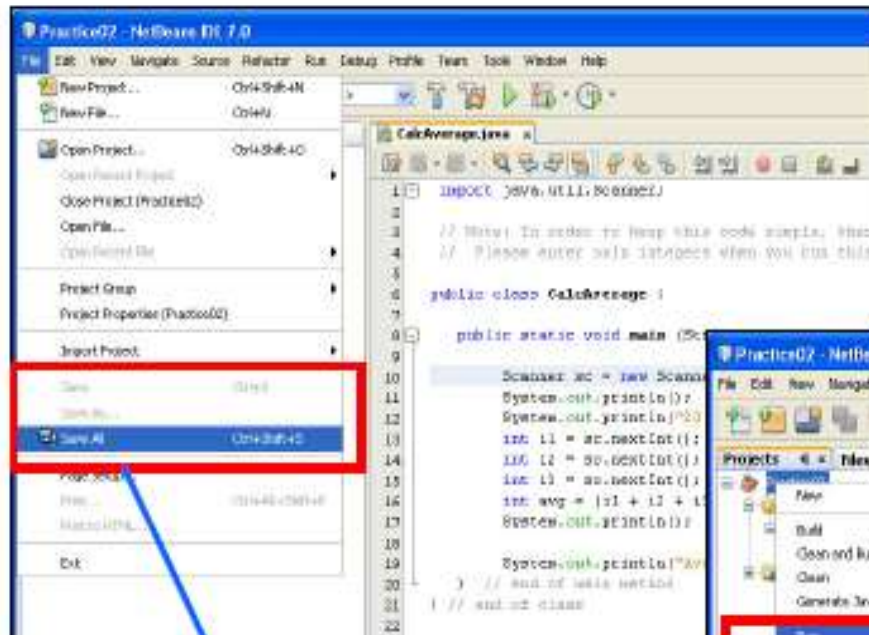
- Ejemplo:

```
java ShirtTest
```

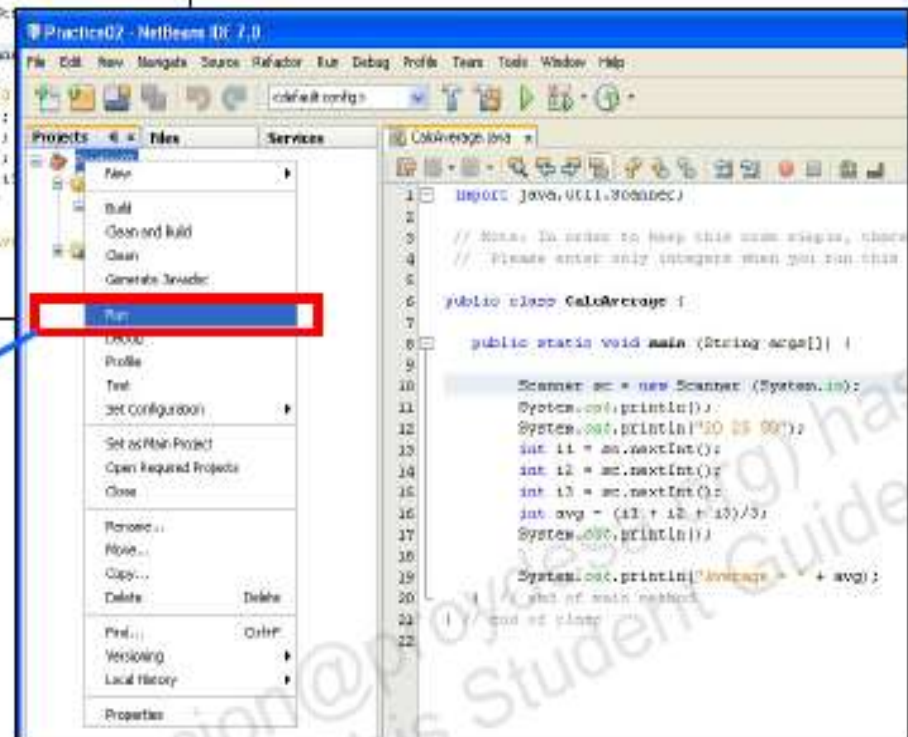
- Salida:

```
Shirt ID: 0  
Shirt description:-description required-  
Color Code: U  
Shirt price: 0.0  
Quantity in stock: 0
```

Compilación y ejecución de un programa mediante un IDE

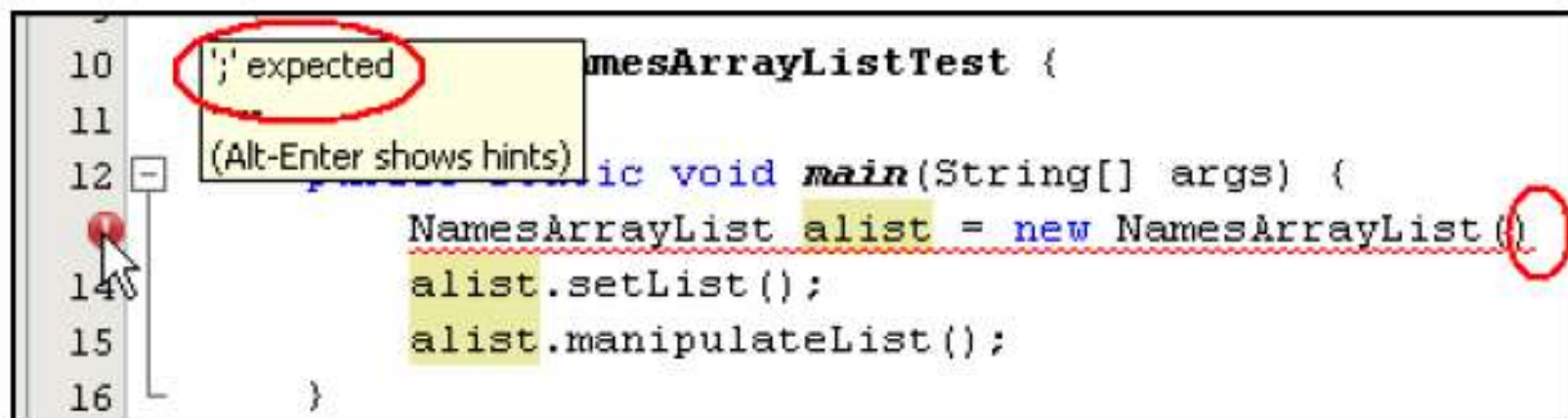


Save equivale a javac.



Run equivale a java.





















Cómo evitar problemas de sintaxis



```
10      NamesArrayListTest {  
11  
12      public void main(String[] args) {  
13          NamesArrayList alist = new NamesArrayList()  
14          alist.setList();  
15          alist.manipulateList();  
16      }
```

Trabajar con un depurador de IDE

```
20 public void displayShirtInformation() {  
21     System.out.println("Shirt ID: " + shirtID);  
22     System.out.println("Shirt description: " + description);  
23     System.out.println("Color Code: " + colorCode);  
24     System.out.println("Shirt price: " + price);  
25     System.out.println("Quantity in stock: " + quantityInStock);  
}
```

Variables	Breakpoints	Output	Tasks
 Name Type Value			
<Enter new watch> 			
  this		Shirt	 #58
  shirtID		int	 0
  description		String	 "-description required-"
  colorCode		char	 'U'
  price		double	 0.0
  quantityInStock		int	 0

5

Declaración, inicialización y uso de variables

Importancia

- Una variable hace referencia a algo que puede cambiar. Las variables pueden contener un valor de un juego de valores. ¿Dónde ha visto variables con anterioridad?
- ¿Qué tipos de dato cree que pueden contener variables?

(fundacion@proydesa.org) has a
to use this Student Guide.

Identificación del uso y la sintaxis de las variables

Ejemplo:

```
public class Shirt {  
  
    public int shirtID = 0; // Default ID for the shirt  
  
    public String description = "-description required-"; // default  
    // The color codes are R=Red, B=Blue, G=Green, U=Unset  
    public char colorCode = 'U';  
  
    public double price = 0.0; // Default price for all shirts  
  
    public int quantityInStock = 0; // Default quantity for all shirts  
  
    // This method displays the values for an item  
    public void displayInformation() {  
  
        System.out.println("Shirt ID: " + shirtID);  
    }  
}
```

Identificación del uso y la sintaxis de las variables

Ejemplo:

```
public void displayDescription {  
    String displayString = "";  
    displayString = "Shirt description: " + description;  
    System.out.println(displayString);  
}
```

Usos de las variables

- Contener datos únicos para una instancia de objeto
- Asignar el valor de una variable a otra
- Representar valores en una expresión matemática
- Imprimir los valores en la pantalla
- Contener referencias a otros objetos

Declaración e inicialización de variables

- Sintaxis (campos):

```
[modifiers] type identifier [= value];
```

- Sintaxis (variables locales):

```
type identifier [= value];
```

- Ejemplos:

```
public int shirtID = 0;  
public String description = "-description required-";  
public char colorCode = 'U';  
public double price = 0.0;  
public int quantityInStock = 0;
```

Descripción de tipos de dato primitivos

- Tipos integrales (`byte`, `short`, `int` y `long`)
- Tipos de coma flotante (`float` y `double`)
- Tipo textual (`char`)
- Tipo lógico (`boolean`)

Tipos primitivos integrales

Tipo	Longitud	Rango	Ejemplos de valores literales permitidos
byte	8 bits	De -2^7 a $2^7 - 1$ (de -128 a 127, o 256 posibles valores)	2 -114 0b10 (número binario)
short	16 bits	De -2^{15} a $2^{15} - 1$ (de -32.768 a 32.767, o 65.535 posibles valores)	2 -32699
int (tipo por defecto para literales integrales)	32 bits	De -2^{31} a $2^{31} - 1$ (de -2.147.483.648 a 2.147.483.647, o 4.294.967.296 posibles valores)	2 147334778 123_456_678

Tipos primitivos integrales

Tipo	Longitud	Rango	Ejemplos de valores literales permitidos
long	64 bits	De -2^{63} a $2^{63} - 1$ (de $-9.223.372.036.854.775.808$ a $9.223.372.036.854.775.807$, o $18.446.744.073.709.551,616$ posibles valores)	2 $-2036854775808L$ 1L

Tipos primitivos de coma flotante

Tipo	Longitud Float	Ejemplos de valores literales permitidos
float	32 bits	99F -327456,99.01F 4.2E6F (notación de ingeniería para $4,2 * 10^6$)
double (tipo por defecto de los literales de coma flotante)	64 bits	-1111 2.1E12 99970132745699.999

```
public double price = 0.0; // Default price for all shirts
```

Tipo primitivo textual

- El único tipo de dato textual primitivo es `char`.
- Se utiliza para un único carácter (16 bits).
- Ejemplo:
 - `public char colorCode = 'U';`

Tipo primitivo lógico

- El único tipo de dato es `boolean`.
- Solo puede almacenar `true` o `false`.
- Contiene el resultado de una expresión que se evalúa en `true` o `false`.

Asignación de nombres a variables

Reglas:

- Los identificadores de variables deben empezar por una letra mayúscula o minúscula, un carácter de subrayado (_) o un signo de dólar (\$).
- Los identificadores de variables no pueden contener puntuación, espacios ni guiones.
- No se pueden utilizar las palabras clave de la tecnología Java.

Asignación de nombres a variables

Instrucciones:

- Empezar cada variable por una letra minúscula. Las siguientes palabras deben tener la inicial mayúscula (por ejemplo, `myVariable`).
- Seleccionar nombres que sean nemotécnicos y que indiquen al observador casual la intención de la variable.

Asignación de un valor a una variable

- Ejemplo:
 - `double price = 12.99;`
- Ejemplo (booleano):
 - `boolean isOpen = false;`

fundacion@proydesa.org) has a
this Student Guide.

Declaración e inicialización de varias variables en una línea de código

- Sintaxis:
 - `type identifier = value [, identifier = value];`
- Ejemplo:
 - `double price = 0.0, wholesalePrice = 0.0;`

Métodos adicionales para declarar variables y asignar valores a variables

- Asignación de valores literales:
 - `int ID = 0;`
 - `float pi = 3.14F;`
 - `char myChar = 'G';`
 - `boolean isOpen = false;`
- Asignación del valor de una variable a otra:
 - `int ID = 0;`
 - `int saleID = ID;`

adacion@proydesa.org) has a
this Student Guide.

Métodos adicionales para declarar variables y asignar valores a variables

- Asignación del resultado de una expresión a variables integrales, de coma flotante o booleanas:
 - `float numberOrdered = 908.5F;`
 - `float casePrice = 19.99F;`
 - `float price = (casePrice * numberOrdered);`
 - `int hour = 12;`
 - `boolean isOpen = (hour > 8);`
- Asignación del valor de retorno de una llamada a método a una variable

adacion@proydesa.org) has a
this Student Guide.

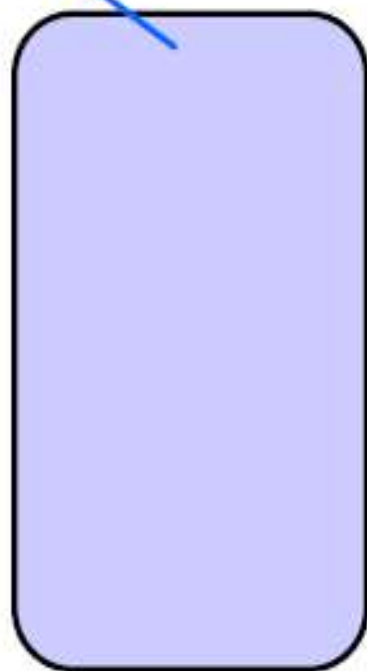
Constantes

- Variable (puede cambiar):
 - `double salesTax = 6.25;`
- Constante (no puede cambiar):
 - `final int NUMBER_OF_MONTHS = 12;`
- Instrucciones: las constantes deben ir en mayúscula, con las palabras separadas con un carácter de subrayado (_).

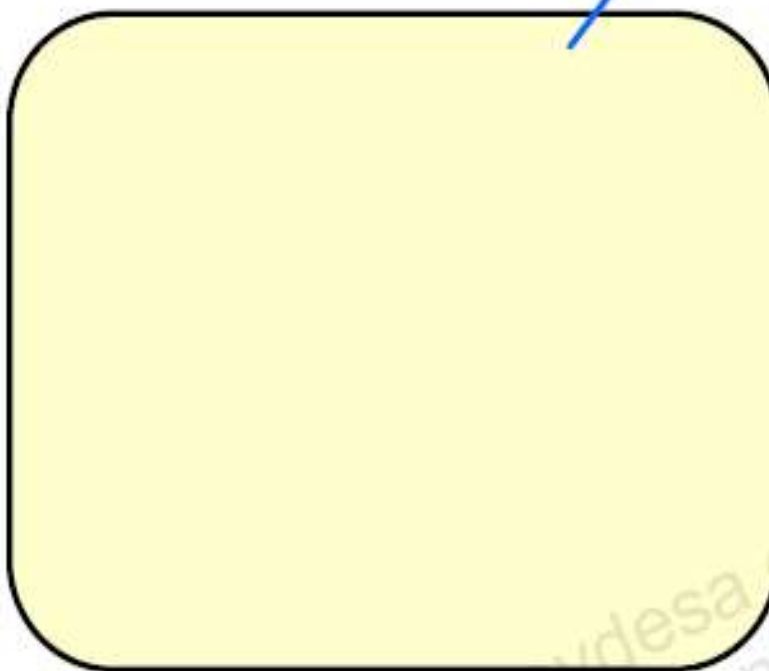
Almacenamiento de primitivos y constantes en memoria

Variable local declarada en un método

Objetos con campos



Memoria de pila



Memoria de montón

Operadores matemáticos estándar

Objetivo	Operador	Ejemplo	Comentarios
Suma	+	<code>sum = num1 + num2;</code> Si num1 es 10 y num2 es 2, sum es 12.	
Resta	-	<code>diff = num1 - num2;</code> Si num1 es 10 y num2 es 2, diff es 8.	
Multiplicación	*	<code>prod = num1 * num2;</code> Si num1 es 10 y num2 es 2, prod es 20.	
División	/	<code>quot = num1 / num2;</code> Si num1 es 31 y num2 es 6, quot es 5.	La división devuelve un valor entero (sin resto).

Operadores matemáticos estándar

Objetivo	Operador	Ejemplo	Comentarios
Resto	%	<code>mod = num1 % num2;</code> Si num1 es 31 y num2 es 6, mod es 1.	<p>El resto busca el resto del primer número dividido entre el segundo número.</p> <div><div>5 R ①</div><div>6 31 30 ---- 1</div></div> <p>El resto siempre da una respuesta con el mismo signo como primer operando.</p>

Operadores de aumento y disminución (++ y --)

Forma extendida:

```
age = age + 1;
```

o bien

```
count = count - 1;
```

Operadores de aumento y disminución (++ y --)

Forma breve:

Operador	Objetivo	Ejemplo	Notas
++	Aumento previo (++ <i>variable</i>)	<pre>int i = 6; int j = ++i; i is 7, j is 7</pre>	
	Aumento posterior (<i>variable</i> ++)	<pre>int i = 6; int j = i++; i is 7, j is 6</pre>	El valor i se asigna a j antes de aumentar i. Por lo tanto, a j se asigna 6.

Operadores de aumento y disminución (++ y --)

Operador	Objetivo	Ejemplo	Notas
--	Disminución previa (-- <i>variable</i>)	<pre>int i = 6; int j = --i; i is 5, j is 5</pre>	
	Disminución posterior (<i>variable</i> --)	<pre>int i = 6; int j = i--; i is 5, j is 6</pre>	El valor <i>i</i> se asigna a <i>j</i> antes de disminuir <i>i</i> . Por lo tanto, a <i>j</i> se asigna 6.

Operadores de aumento y disminución (++ y --)

Ejemplos:

```
int count=15;  
int a, b, c, d;  
a = count++;  
b = count;  
c = ++count;  
d = count;  
System.out.println(a + ", " + b + ", " + c + ", " + d);
```

Prioridad de operadores

A continuación se presenta un ejemplo de la necesidad de reglas de prioridad.

¿La respuesta del siguiente problema es 34 o 9?

$$c = 25 - 5 * 4 / 2 - 10 + 4;$$

fundacion@proydesa.org) has a
use this Student Guide.

Prioridad de operadores

Reglas de prioridad:

1. Operadores delimitados por un par de paréntesis
2. Operadores de aumento y disminución
3. Operadores de multiplicación y división, evaluados de izquierda a derecha
4. Operadores de suma y resta, evaluados de izquierda a derecha

Uso de paréntesis

Ejemplos:

```
c = (((25 - 5) * 4) / (2 - 10)) + 4;
```

```
c = ((20 * 4) / (2 - 10)) + 4;
```

```
c = (80 / (2 - 10)) + 4;
```

```
c = (80 / -8) + 4;
```

```
c = -10 + 4;
```

```
c = -6;
```

Uso de ampliación y conversión de tipo

- 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
```

- Ejemplo de una posible solución:

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

Ampliación

- Ampliaciones automáticas:
 - Si asigna un tipo más pequeño a un tipo mayor.
 - Si asigna un tipo integral a un tipo de coma flotante.
- Ejemplo de ampliaciones automáticas:

```
long big = 6;
```

(fundacion@proydesa.org) has a
to use this Student Guide.

Conversión de tipo

- Sintaxis:

```
identifier = (target_type) value
```

- 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
```

- Ejemplo de una posible solución:

```
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
```

Conversión de tipo

Ejemplos:

```
int myInt;  
long myLong = 99L;  
myInt = (int) (myLong); // No data loss, only zeroes.  
                        // A much larger number would  
                        // result in data loss.  
  
int myInt;  
long myLong = 123987654321L;  
myInt = (int) (myLong); // Number is "chopped"
```

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

- Ejemplo de un posible problema:

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

- 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);
```


Tipos de dato 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 a un tipo float:

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


Ejemplo

```
public class Person {  
  
    public int ageYears = 32;  
  
    public void calculateAge() {  
  
        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 calculateAge method  
} // end of class
```