



Introducción a Lenguaje Java

2C 2023 – Ing. Elizabeth Barrera



Temas

- 👑 *Tipos de Datos*
- 👑 *Variables*
- 👑 *Operadores*
- 👑 *Estructuras de Control*
- 👑 *Métodos*
- 👑 *Clases*
- 👑 *Paquetes*
- 👑 *Estructuras de Datos*

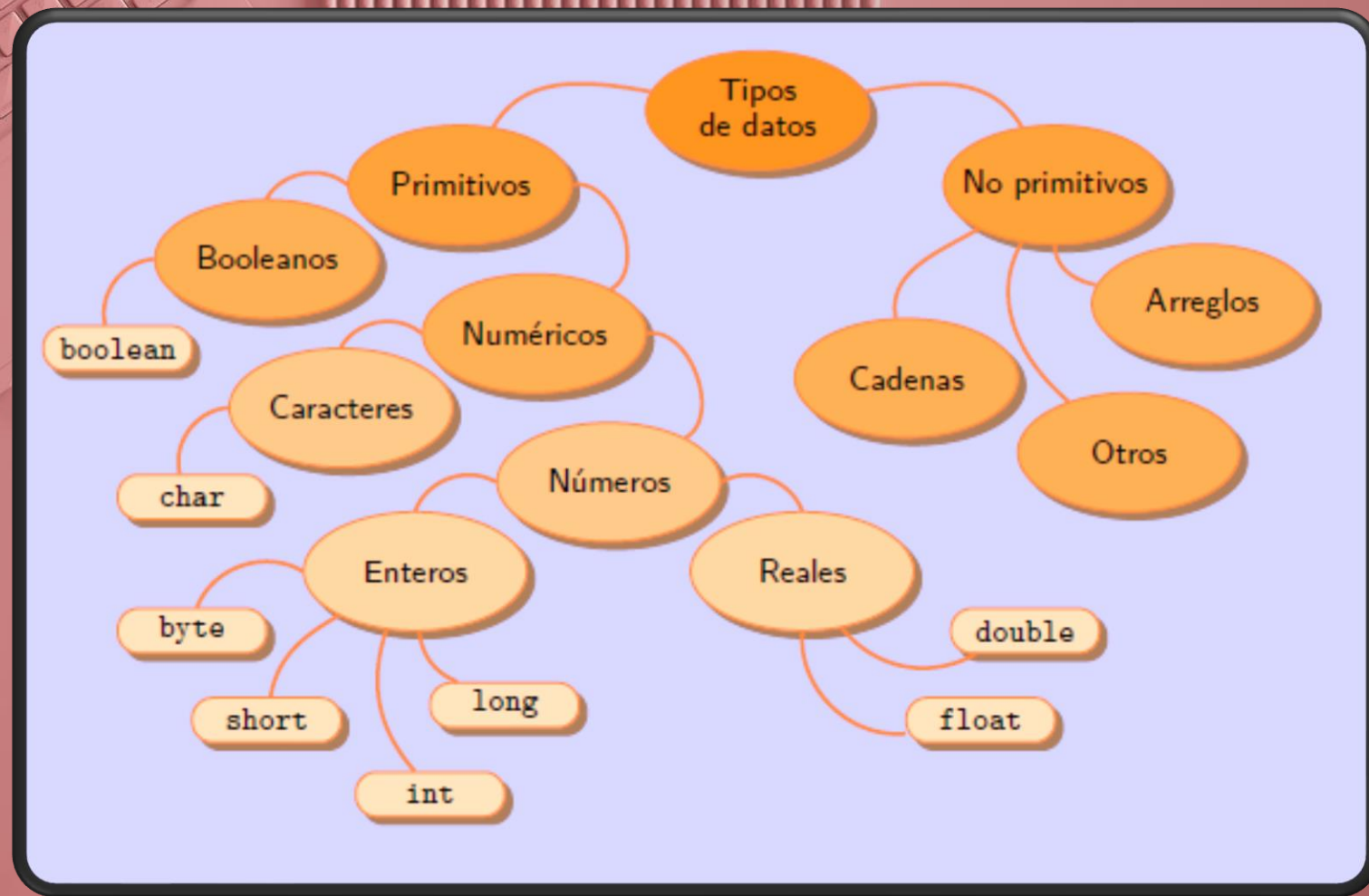
Lenguaje Java

Java comenzó a desarrollarse a comienzos de los años noventa. Es un *lenguaje orientado a objetos* con una sintaxis similar a la de C.

Es un *lenguaje compilado estáticamente tipado*. Cada variable debe ser declarada con su tipo de datos correspondiente antes de ser utilizada, y no se puede cambiar el tipo de datos de una variable durante la ejecución de un programa.



Tipos de Datos



| Tipo de dato | Tamaño | Rango de valores | Valor por defecto |
|--------------|-----------------|--------------------------------|-------------------|
| boolean | 1 bit | false → true | false |
| byte | 1 byte (8 bits) | $-2^7 \rightarrow 2^7-1$ | 0 |
| short | 2 bytes | $-2^{15} \rightarrow 2^{15}-1$ | 0 |
| int | 4 bytes | $-2^{31} \rightarrow 2^{31}-1$ | 0 |
| long | 8 bytes | $-2^{63} \rightarrow 2^{63}-1$ | 0L |
| float | 4 bytes | ilimitado | 0f |
| double | 8 bytes | ilimitado | 0d |
| char | 2 bytes | \u0000 → \uffff | \u0000 |



Variables

Declaración

- *de una variable sola:*

`<tipo> <nombre_variable>;`

`int cantidad;`

- *de más de una variable del mismo tipo:*

`<tipo> <nombre_variable_1>, <nombre_variable_2>;`

`int cantidad, contador, resultado;`

Declaración

- *declaración y asignación de valor a una variable:*

<tipo> <nombre_variable> = <valor>;

int contador = 0;

Aclaraciones

- Para la **asignación** de valores se utiliza el operador `=`.
- El **alcance** está dado por el bloque en el que se declara. Es visible dentro del bloque en el que fue declarada y dentro de todos los sub-bloques que se definan dentro de éste.
- El lenguaje es sensible a mayúsculas y minúsculas.



Operadores

Aritméticos

- $+$ operador de suma
- $-$ operador de la resta y de número negativo
- $*$ operador de multiplicación
- $/$ operador de división
- $\%$ operador de resto o módulo (devuelve el resto de una división)
- $++$ operador de incremento en uno
- $--$ operador de decremento en uno

Relacionales

- $<$ y \leq operador menor y menor o igual
- $>$ y \geq operador mayor y mayor o igual
- $==$ operador de igualdad
- $!=$ operador distinto

Lógicos

- **!** *operador de negación*
- **||** *operador O lógico*
- **&&** *operador Y lógico*



Estructuras de Control

Bloques

- Los bloques de sentencias se delimitan con *llaves* {}.
- La *Indentación* se utiliza a modo simbólico para facilitar la lectura del código (el compilador lo ignora totalmente).

A top-down view of a desk with a red background. In the top left, there is a red keyboard and a red mouse. In the top center, there is a red pen holder filled with many red pens. In the bottom left, there is a red notebook. In the bottom center, there is a large, blank, light-colored rectangular piece of paper. In the bottom right, there is a stack of red notebooks with a red pen resting on top of them.

Secuencia “;”

A top-down view of a desk with a red background. In the top left, there is a red keyboard and a red mouse. In the top center, there is a red pen holder filled with many red pens. In the bottom left, there is a red notebook. In the bottom center, there is a large, blank, light-colored rectangular piece of paper. In the bottom right, there is a stack of red notebooks with a red pen resting on top of them.

Condicional

Condicional Simple

```
if (<condición_booleana>) {  
    instrucción 1;  
    instrucción 2;  
    instrucción 3;  
    ...  
} else {  
    instrucción 4;  
    instrucción 5;  
    instrucción 6;  
    ...  
}
```

Condicional Múltiple

```
switch (<expresión_entera>) {  
    case valor1:  
        ...  
        break;  
    case valor2:  
        ...  
        break;  
    ...  
    default:  
        ...  
}
```




Ciclos

```
while (<condición_booleana>) {  
    expresión 1;  
    expresión 2;  
    ...  
}
```

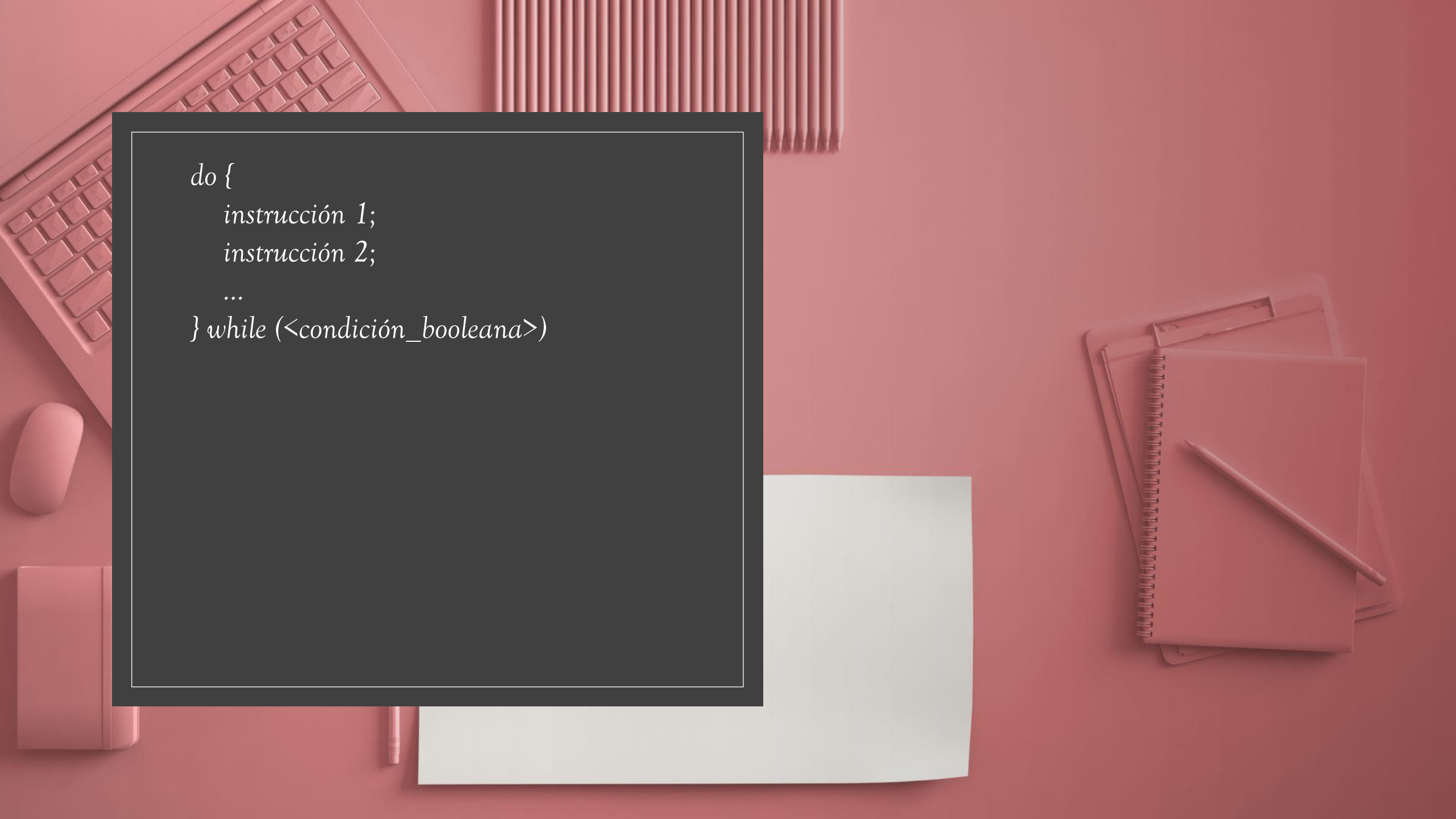
```
for (<inicialización_variable>;  
    <condición_booleana> ;  
    <variación_variable>) {  
    expresión 1;  
    expresión 2;  
    ...  
}
```

Comentarios

- *La variable puede ser declarada previo al for o dentro del mismo.*
- *De acuerdo a dónde se declare será el alcance que tenga (si se declara en el for sólo se podrá utilizar en las instrucciones dentro del ciclo for).*
- *La variación de la variable puede ser un incremento o un decremento.*

Ejemplo

```
int suma = 0;  
for (int i = 1; i <= 10 ; i++) {  
    suma = suma + i;  
}
```


A top-down view of a desk with a red background. In the top left, there is a red keyboard and a red mouse. In the top center, there are several red pens. In the bottom right, there are several red notebooks and a red pencil. A large, dark gray rectangular box with a thin white border is centered on the desk, containing white text.

```
do {  
    instrucción 1;  
    instrucción 2;  
    ...  
} while (<condición_booleana>)
```



Métodos

Métodos

- En Java todas las *operaciones* se llevan a cabo a través de métodos.
- Se puede hacer una similitud a *funciones* de los lenguajes estructurados.

```
<tipo_retorno_método> <nombre_método>  
(<tipo_parámetro> <parámetro_1>, ...,  
<tipo_parámetro> <parámetro_n>) {  
    instrucción1;  
    instrucción2;  
    ...  
}
```

Ejemplo

```
int sumar (int x, int y) {  
    int sum = 0;  
    sum = x + y;  
    return sum;  
}
```

```
int sumar (int x, int y) {  
    return (x + y);  
}
```




Pasaje de Variables

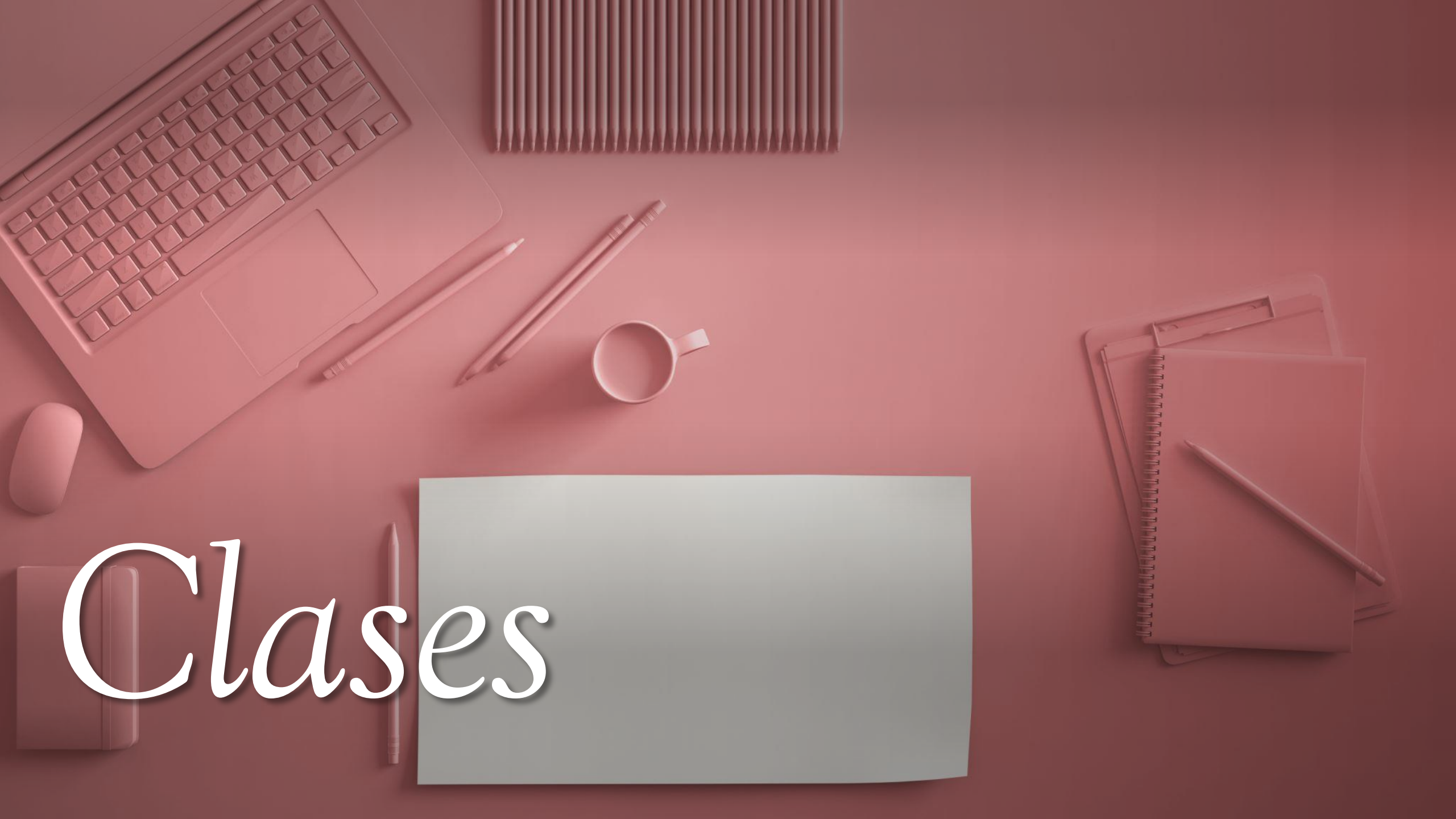
Pasaje por Valor

- *En el pasaje por valor, se pasa una copia del valor de la variable al método.*
- *El método trabaja sobre este valor, la variable original no sufre ninguna modificación.*
- *Los tipos de datos primitivos se pasan en Java siempre por valor.*

Pasaje por Referencia

- *En el pasaje por referencia, se pasa la dirección de memoria de la variable al método.*
- *El método trabaja directamente sobre la variable, que sufre los cambios que el llamado al método provoque.*
- *Los objetos se pasan en Java por referencia.*





Classes

Clases

- En los lenguajes Orientados a Objetos se estructuran los programas mediante *clases*.
- Las clases son la definición de un *objeto*. Un objeto representa a alguna entidad de la vida real.
- Sus componentes son *variables* y *métodos*.
- Para indicar que es una clase se utiliza la palabra reservada "*class*".


```
class <nombre_clase> {  
    <tipo_variable> <nombre_variable>;  
    <tipo_método> <nombre_método>  
    (<parámetros_método>) {  
        instrucción 1;  
        instrucción 2;  
        ...  
    }  
}
```

A top-down view of a desk with a pink background. In the top left, a portion of a pink keyboard is visible. Below it is a pink mouse. To the right of the mouse is a pink vertical pen holder containing several pens. In the bottom left, there is a pink rectangular object, possibly a book or a small box. On the right side of the desk, there are several pink spiral-bound notebooks stacked, with a pink pen resting on top of them. A large, light pink rectangular object, possibly a piece of paper or a folder, is partially visible in the bottom center.

Uso:

```
<nombre_clase> <nombre_variable> = new  
<nombre_clase>();
```

Acceder a componentes:

```
<nombre_variable>.<nombre_componente>
```

Ejemplo

Declaración:

```
class Punto {  
    int x;  
    int y;  
}
```

Uso:

```
Punto p = new Punto();
```

Acceder a componentes:

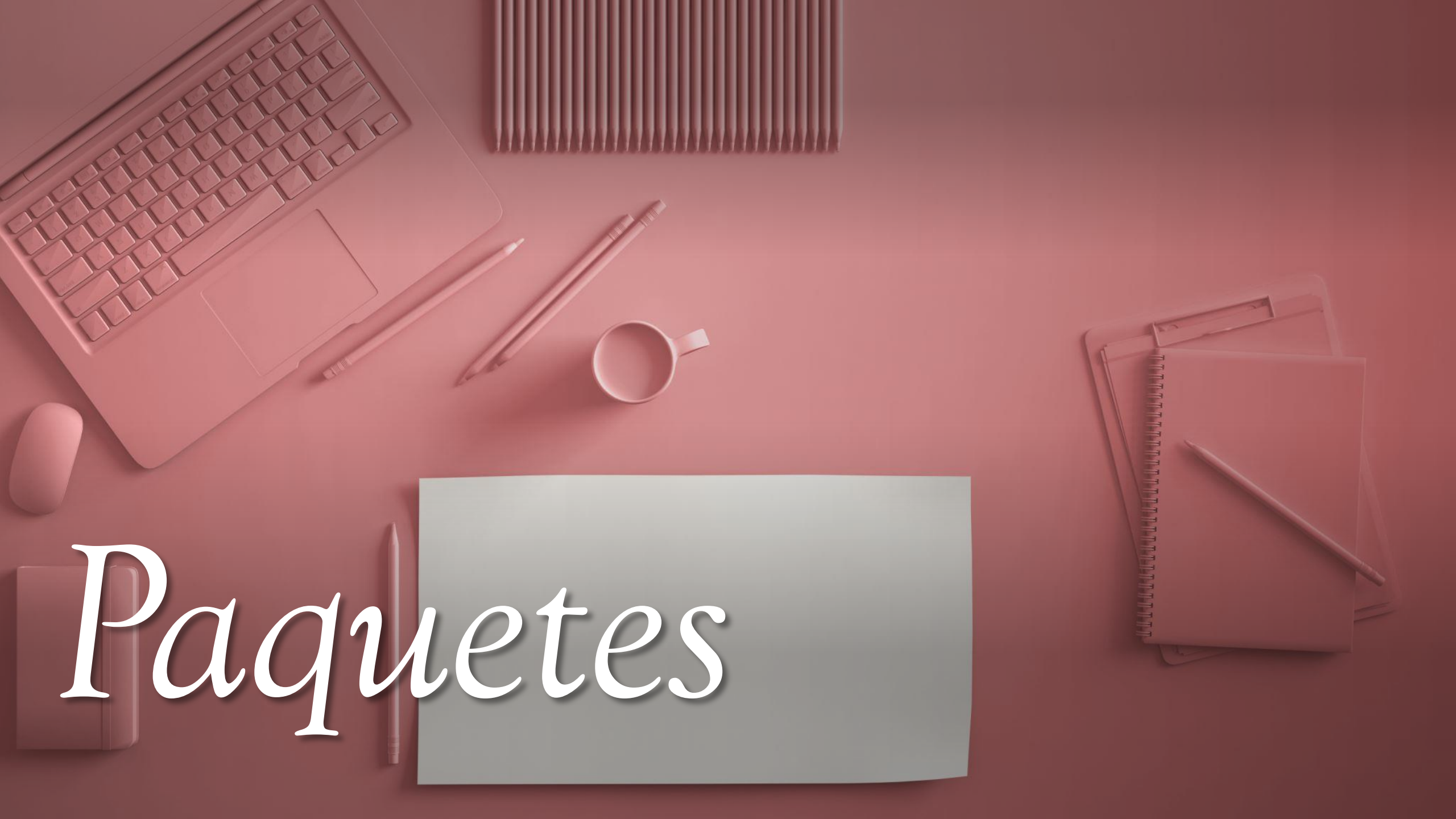
```
p.x = 5;
```

Alcance de métodos y variables

- Propiedad que permite indicar por quién podrán ser utilizados.
- Si un método o variable es **privado** solo se podrá utilizar dentro de la clase, si es **público** podrá ser usado tanto dentro de la clase como por fuera.
- Para indicar que son públicos se utiliza la palabra reservada **public**, y para indicar si son privados la palabra **private**.

Ejemplo

```
class Persona {  
    String nombre;  
    String apellido;  
    String calle;  
    int numero;  
    String ciudad;  
    public void setearNombre  
    (String n, String a) {  
        nombre = n;  
        apellido = a;  
    }  
  
    public void setearDireccion  
    (String c, int n, String ciu)  
    {  
        calle = c;  
        numero = n;  
        ciudad = ciu;  
    }  
}
```

Paquetes

Paquetes

- Las *bibliotecas* de Java se llaman paquetes, que pueden contener sub-paquetes.
- Se utiliza la *notación de punto* para invocar métodos de los paquetes.

Salida

- Un paquete que utilizaremos casi siempre es el paquete `java.lang`, que contiene la clase ***System***.
- Esta clase contiene el objeto ***out***, que representa un espacio de salida (por defecto, la pantalla).
- Este objeto ofrece, entre otros, los métodos ***println*** y ***print***.

Ejemplo

```
int n = 3;  
char c = 'f';  
System.out.println(n);  
System.out.println(c);  
System.out.println("Hola!");
```



Estructuras de Datos

Estructuras Estáticas

- Tamaño en *memoria fijo*, no se puede modificar.
- Ubicación de una posición en forma directa, existencia de *índices*.
- Ejemplo: arreglos o vectores.



Estructuras Dinámicas

- Tamaño en *memoria modificable*, se va asignando memoria mientras se vaya necesitando.
- No existe ubicación por índices, debe recorrerse la estructura para llegar a cierta posición.
- Ejemplo: listas enlazadas.

A top-down view of a desk with a keyboard, mouse, pens, and notebooks. The background is a solid reddish-pink color. In the top left, a portion of a keyboard is visible. To its right is a row of pens. In the bottom left, a mouse and a small notebook are visible. In the bottom center, a large, light-colored rectangular object, possibly a piece of paper or a folder, is partially visible. On the right side, there are several spiral-bound notebooks stacked together, with a pen resting on top of them.

Arreglos

Arreglo

| | | | | | | |
|-------|-------|-------|-------|-------|-----|-------|
| 5 | 7 | 22 | 8 | 34 | ... | ... |
| i = 0 | i = 1 | i = 2 | i = 3 | i = 4 | | i = n |

- ***Declaración:***

`<tipo_datos_arreglo>[] <nombre_arreglo>;`

`int[] valores;`

- Los arreglos, por tratarse de una estructura de tamaño fijo, antes de poder usarlos se le debe asignar la cantidad de elementos que va a contener, es decir, ***dimensionarlo:***

`<nombre_arreglo> = new <tipo_datos_arreglo> [<dimensión>]`

`valores = new int[100];`

Ejemplo

Inicialización:

```
for (int i = 0; i < 100; i++) {  
    valores[i] = i + 2;  
}
```

Búsqueda (x):

```
boolean encontrado = false;  
int i = 0;  
while (!encontrado && i < 100) {  
    if (valores[i] == x) {  
        encontrado = true;  
    }  
    i++;  
}
```


A top-down view of a desk with a red background. In the top left, there is a red keyboard and a red mouse. In the top center, there is a red pen holder filled with many red pens. In the bottom left, there is a red notebook. In the bottom center, there is a large, blank, light green rectangular piece of paper. In the bottom right, there is a stack of red notebooks with a red pen resting on top of them.

Listas

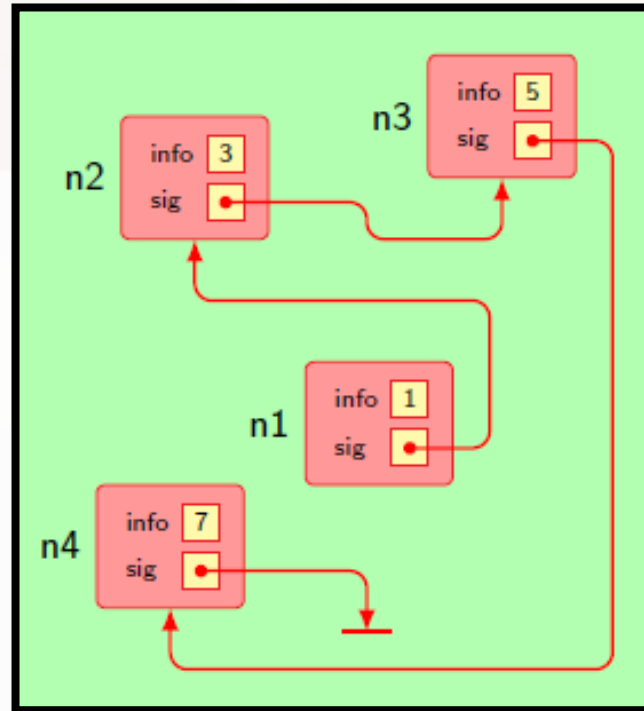
Lista

- Las estructuras enlazadas utilizan variables que son *referencias a otros objetos*.
- Estas variables almacenan la *dirección en memoria* de un objeto.

Lista

- Puede crearse una estructura enlazada con la siguiente clase:

```
public class Nodo {  
    int info;  
    Nodo sig;  
}
```



¡Muchas Gracias!
Nos vemos en clase





Bibliografía

- 👑 *Programación II – Apuntes de
Cátedra – V1.3 – Cuadrado
Trutner – UADE*
- 👑 *Programación II – Apuntes de
Cátedra – Wehbe – UADE*

Ejemplo pasaje por valor

```
public static void cambiar (int  
u) {  
    System.out.println("Valor  
inicial de la variable u: " +  
u);  
    u = u * 5;  
    System.out.println("Valor  
final de la variable u: " +  
u);  
}
```

```
public static void main(String[]  
args) {  
    int u = 5;  
    System.out.println("Valor de  
u antes del llamado: " + u);  
    cambiar(u);  
    System.out.println("Valor de  
u después del llamado: " +  
u);  
}
```

Salida:

Valor de u antes del llamado: 5;
Valor inicial de la variable u: 5;
Valor final de la variable u: 25;
Valor de u después del llamado: 5;

Ejemplo pasaje por referencia

```
public static class num {  
    int info;  
}  
public static void cambiar(num  
u) {  
    System.out.println("Valor  
inicial de la variable u: " +  
u.info);  
    u.info = u.info * 5;  
    System.out.println("Valor  
final de la variable u: " +  
u.info);  
}
```

```
public static void main(String[]  
args) {  
    num u = new num();  
    u.info = 5;  
    System.out.println("Valor de  
u antes del llamado: " +  
u.info);  
    cambiar(u);  
    System.out.println("Valor de  
u después del llamado: " +  
u.info);  
}
```

Salida:

Valor de u antes del llamado: 5;
Valor inicial de la variable u: 5;
Valor final de la variable u: 25;
Valor de u después del llamado: 25;