

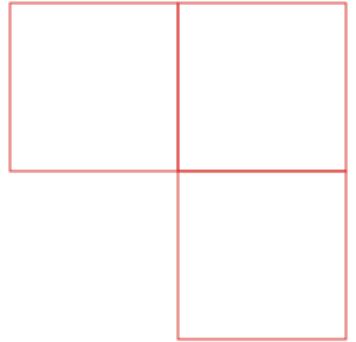
## UF3: Programación básica

3.2: Fundamentos del  
lenguaje Java II

Ve más allá



# Índice



- Paquetes (API)
- Tipos de datos por Referencia
- Envoltorios
- String
- Entrada/Salida



# Paquetes

Los paquetes son la manera de agrupar clases (ficheros) en java.

- **package paquete;**

Para indicar a qué paquete pertenece una clase.

- **import paquete;**

Para utilizar las clases y sus propiedades de aquellas que pertenezcan a dicho paquete.

# Paquetes

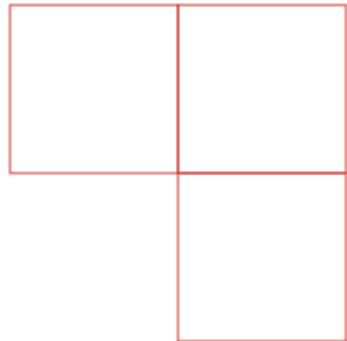
Los paquetes (librerías) facilitan extraordinariamente la labor del programador, ya que existen miles de líneas de código que otros programadores ya han hecho por nosotros.

El conjunto de todos los paquetes oficiales en Java es lo que se denomina la Interfaz de Programación de Aplicaciones (**API**) de Java.

En cada versión de la API se incluye un mayor número de paquetes, y también existe documentación exhaustiva para que cualquier programador pueda consultar cómo hacer uso de ella.

<https://docs.oracle.com/en/java/javase/17/docs/api/java.base/module-summary.html>

# Tipos de datos por referencia



TIPOS OBJETO O DE REFERENCIA (con métodos, necesitan una invocación para ser creados)	Tipos de la biblioteca estándar de Java	String (cadenas de texto) Muchos otros (p.ej. Scanner, TreeSet, ArrayList...)
	Tipos definidos por el programador / usuario	Cualquiera que se nos ocurra (ej: Taxi, Autobus, Tranvia...)
	arrays	Serie de elementos o formación tipo vector o matriz. Lo consideraremos un objeto especial que carece de métodos.
	Tipos envoltorio o wrapper (Equivalentes a los tipos primitivos pero como objetos.)	Byte
		Short
		Integer
		Long
		Float
		Double
		Character
		Boolean

# Tipos de datos por referencia

## Envoltorios (wrappers): ¿qué son?

Además de los tipos de datos primitivos, Java sigue con su vocación de utilizar objetos y nos ofrece un envoltorio de los tipos de datos primitivos para convertirlos en objetos.

Clases envoltorios:	<b>Integer</b> (int)	<b>Long</b> (long)
	<b>Float</b> (float)	<b>Double</b> (double)
	<b>Short</b> (short)	<b>Byte</b> (byte)
	<b>Character</b> (char)	<b>Boolean</b> (boolean)

Los nombres son prácticamente idénticos a los tipos básicos, pero la primera letra es **mayúscula** (recordad que Java es sensible a mayúsculas).

# Tipos de datos por referencia

## Envoltorios (wrappers): ¿para qué sirven?

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.

Ej: Un usuario introduce un número entero por teclado. Lo que un usuario introduce por teclado es un String siempre. ¿Cómo lo cambiamos a un entero?

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

Método: `parseInt()` de la clase `Integer`. Este método recibe como parámetro un String y devuelve el número entero correspondiente de hacer la conversión de ese String.

# String

La clase **String** está orientada a manejar cadenas de caracteres. Pertenece al paquete **java.lang**, y por lo tanto no hay que importarla.

Los objetos de la clase String se pueden crear a partir de cadenas constantes, definidas entre dobles comillas.

```
String saludo = "Hola ";
```

Podemos usar el operador de concatenación (+) para cambiar el valor de una variable de tipo String.

```
saludo = "Hola " + "Mundo";
```

# Entrada / Salida de datos

- **Salida estándar.** Se utiliza `System.out` que se encuentra en el paquete `java.lang`.
- **Entrada estándar.** Se utiliza `System.in` que se encuentra en el paquete `java.lang`.
- **Salida de errores.** Se utiliza `System.err` que se encuentra en el paquete `java.lang`.

Para imprimir por pantalla utilizaremos los métodos `print()` o `println()`.

```
System.out.print("Mensaje sin el retorno de carro");
```

```
System.out.println("Mensaje con un retorno de carro");
```

# Entrada / salida de datos

## Antes de java 1.5

Se usaban las clases del paquete `java.io` `InputStreamReader` y `BufferedReader` conjuntamente. La primera es capaz de convertir los bytes de entrada a caracteres. La segunda es capaz de leer hasta un fin de línea.

```
InputStreamReader isr = new InputStreamReader(System.in);
BufferedReader br = new BufferedReader(isr);
System.out.print("Escribe tu nombre: ");
String nombre = br.readLine();
System.out.println("Hola " + nombre + ", ¿cómo estás?");
```

# Entrada / salida de datos

## Desde java 1.5

Se usa la clase **Scanner** del paquete **java.util**. Basta instanciar esta clase pasándole el **System.in**

```
Scanner sc = new Scanner(System.in);
```

```
String cadena = sc.nextLine();
```

Con el método **nextLine()** leemos la línea introducida por teclado.

Otro método que nos puede ser útil es **nextInt()** para leer un entero en vez de una cadena de caracteres. Pero cuidado con este método, porque a veces da problemas.

# Actividad

## Calculadora

Realiza una calculadora en Java de manera que el programa solicite al usuario los datos y la operación, tras lo cual deberá mostrar el resultado por pantalla.



