

6

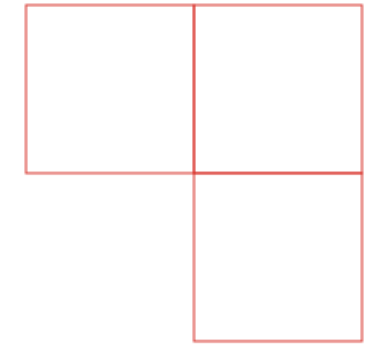
## Operadores y expresiones

Ve más allá



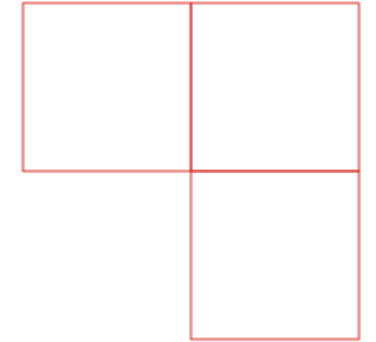
# CONTENIDOS

- **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**



# ÍNDICE

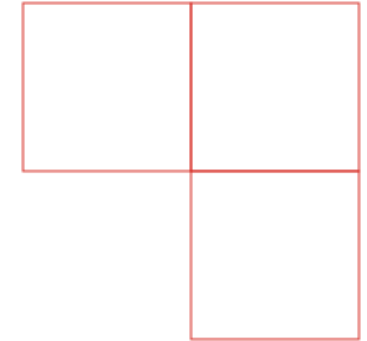
- Entrada y Salida
  1. Paquetes
  2. Entrada y salida estándar
  3. Entrada System.in
  4. Entrada Scanner
  5. Entrada StreamReader
  6. Salida Formateada
  7. Ejercicio





Programación/Entrada y salida

# Paquetes



Un paquete es el equivalente a una librería de C. En lugar de utilizar la orden de C:

```
# include
```

en Java utilizaremos la orden:

```
import paquete;
```





El conjunto de todos los paquetes oficiales en Java es lo que se denomina la Interfaz de Programación de Aplicaciones (API, del inglés Application Programming Interface) 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.





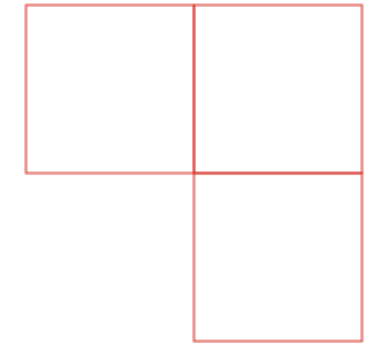
# Entrada y salida

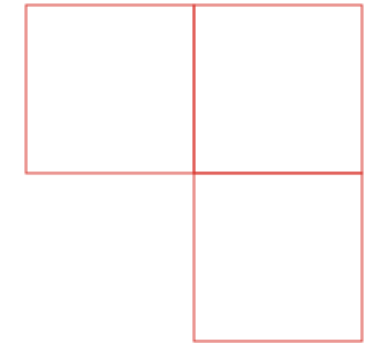
Hasta ahora has escrito varios programas interesantes, pero hasta ahora no le has podido pedir información al usuario. En Java hay varias librerías que te van a ayudar a hacerlo, por eso en el apartado anterior viste los paquetes, ya que es ahí donde las vas a encontrar.

Al igual que utilizas `System.out` para la salida (es decir, para hacer tu `print` o `println`), también puedes utilizar `System.in` para obtener la entrada del usuario. En general, en cualquier sistema operativo existen tres conceptos que conviene conocer:

1. Salida estándar. Se utiliza **`System.out`**
2. Entrada estándar. Se utiliza **`System.in`**
3. Salida de errores. Se utiliza **`System.err`**

Los tres se encuentran en que se encuentra en el paquete `java.lang`. Este paquete se considera que pertenece al núcleo de Java y por lo tanto será el único del que no tendremos que hacer `import`.





# Entrada System.in

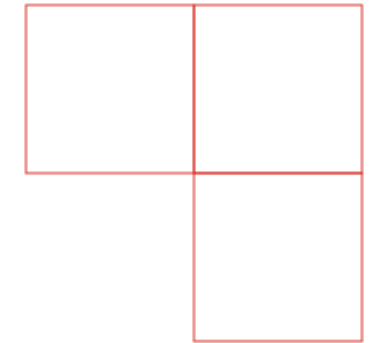
Si lo que deseamos leer es un único carácter, podemos utilizar el objeto System.in que tiene el método llamado read() de la siguiente manera:

```
char c = (char) System.in.read();
```

Ejemplo: Crea un programa que solicite al usuario un número del 1 al 5 y lo muestre por pantalla.

```
import java.io.IOException;

public class Test {
    public static void main(String[] args) throws IOException {
        System.out.println("Indica un número del 1 al 5");
        char num = (char) System.in.read();
        System.out.println("Has elegido el " + num);
    }
}
```



# Entrada Scanner

La manera anterior de leer del teclado es poco práctica y sería tedioso programar la lectura de una cadena de caracteres. En su lugar, es habitual hacer uso del paquete **java.util** que nos ofrece ayuda para acceder de manera más cómoda a la lectura del teclado. En concreto, la clase Scanner sirve para crear un objeto a partir de System.in y tiene muchos métodos que hacen muy cómoda la lectura de datos de diferente tipo.



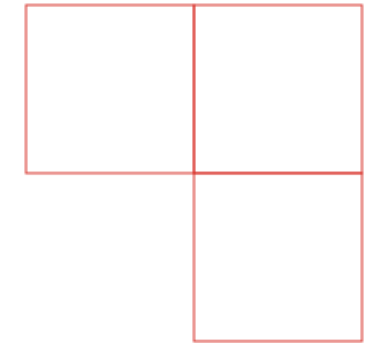




# Entrada y salida

Reading Input

Programación/Entrada y salida





# Entrada Scanner



Ejemplo: Crea un programa que solicite al usuario su nombre, peso y edad (String, int, double) y luego lo muestre por pantalla.

```
import java.util.Scanner;

public class Test {

    public static void main(String[] args) {

        Scanner entrada = new Scanner (System.in);

        System.out.print("Indica tu nombre: ");

        String nombre = entrada.nextLine();

        System.out.print("Indica tu edad: ");

        int edad = entrada.nextInt();

        System.out.print("Indica tu peso: ");

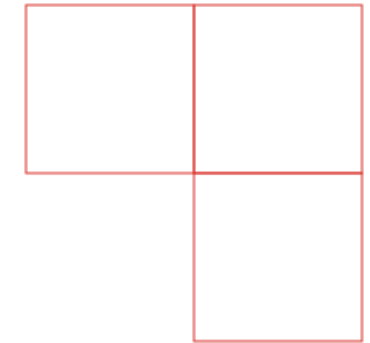
        double peso = entrada.nextDouble();

        System.out.println();

        System.out.println(nombre + " tiene " + edad + " años y pesa " + peso);

    }

}
```



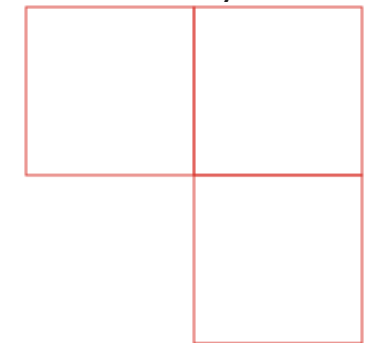
# Entrada Scanner

Por lo general, si quieres leer más de una entrada tiene sentido guardar el escáner en una variable. En el código anterior se llama “entrada”. Con la sintaxis objeto.método se pueden utilizar diferentes métodos como son:

- `nextLine`: Sirve para leer toda una línea en formato de texto (String). Si sólo queremos leer una palabra, también podemos utilizar `next`.
- `nextInt`: Sirve para leer un número entero.
- `nextDouble`: Sirve para leer un número real, con decimales. Ten cuidado al introducirlo ya que probablemente desde tu consola los decimales se introduzcan con `,` en lugar de `.`

Es interesante que siempre haya una pregunta antes de utilizar cualquiera de los métodos anteriores, para ayudar al usuario a saber qué es lo que se espera de él. Si te das cuenta, esas preguntas las hemos hecho con el método `print` en lugar de `println` para conseguir que la respuesta se escriba en la misma línea que la pregunta. Además, hemos dejado un espacio en blanco dentro del String para que no se escriba pegado a la pregunta.





# Entrada StreamReader

- Las clases de StreamReader, que está en el paquete java.io también nos sirven para leer de teclado y son interesantes cuando queremos leer mucha información. Veamos el siguiente ejemplo:

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class Test {
    public static void main(String[] args) throws IOException {
        String nombre;
        InputStreamReader isr = new InputStreamReader(System.in);
        BufferedReader br = new BufferedReader(isr);
        System.out.print("Escribe tu nombre: ");
        nombre = br.readLine();
        System.out.println("Hola " + nombre + ", ¿cómo estás?");
    }
}
```



# Entrada StreamReader

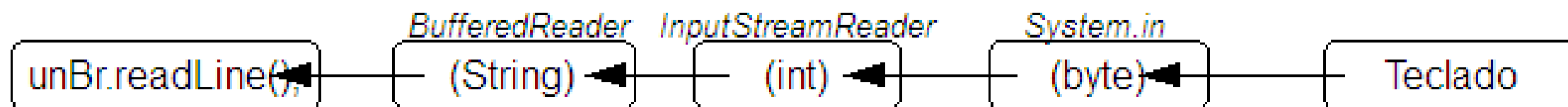
Dentro del paquete `java.io` se encuentra el método `readLine()` de una clase llamada `BufferedReader`.

Este método lee una cadena de caracteres y devuelve un objeto `String`.

Lo que ocurre es que necesitamos convertir la lectura del `System.in` (que lee bytes) en un objeto `Reader` (que es lo que necesita `BufferedReader`).

Así que utilizamos una clase intermedia, que se llama `InputStreamReader`, que nos permite hacer esta conversión.

Gráficamente:

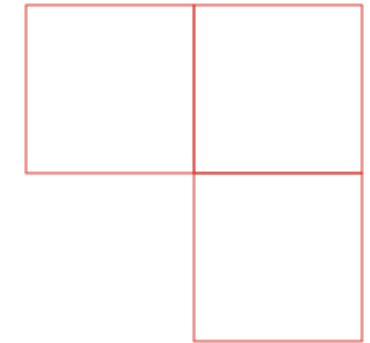




# Salida formateada

Ya conoces la salida habitual de Java con `print` o `println`, esa misma es la que viene utilizando desde tu primer programa `HolaMundo`. Dichos métodos son del objeto `System.Out` que ya sabes que está en el paquete `java.lang`.

Hay otro método junto a los anteriores que se llama `printf` o `format`, que nos va a permitir imprimir datos con formato.



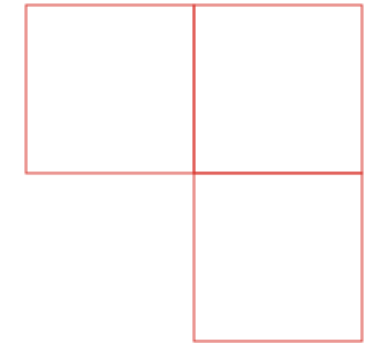




# Salida formateada

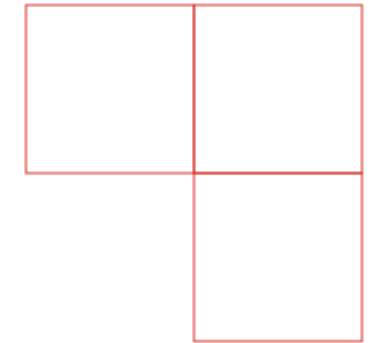
Solución:

```
public class Test {  
    public static void main(String[] args) {  
        int cantidad = 100;  
        double precioUnitario = 3.46;  
        double precioTotal = cantidad * precioUnitario;  
        System.out.printf("Coste: %7.2f\n", precioTotal);  
  
        double iva = 0.21;  
        double ivaTotal = precioTotal * iva;  
        System.out.printf("IVA: %9.2f\n", ivaTotal);  
    }  
}
```





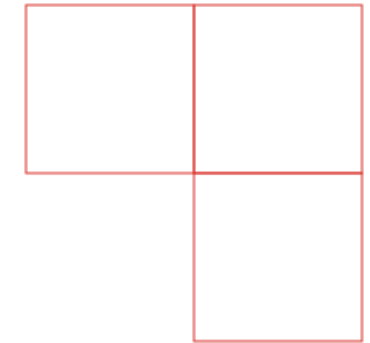
# Salida formateada



El método printf permite alternar texto y datos de una forma más cómoda. El signo de porcentaje

Indica que que ahora vamos a imprimir algo en ese lugar: %7.2, donde el 7 significa que queremos que tenga 7caracteres de ancho, el .2 significa que queremos 2 dígitos después del punto decimal. La f significa que es un punto flotante o número decimal. El \n significa que después queremos que pase a la línea siguiente. Como puedes ver, hay un mini lenguaje completo en estas cadenas de formato, pero las más importantes son:

- Números decimales: f
- Números enteros: d
- Cadenas de texto: s



# Salida formateada

## Formatted Output

```
int cookiesPerDay;  
double cerealBoxesPerDay;  
String name;
```



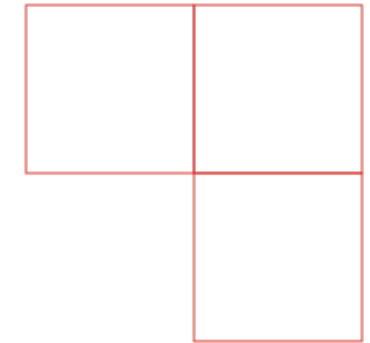
```
System.out.printf("% ", cookiesPerDay);  
                                     ↑      ↑  
                                     what goes here if I want to print 6 characters  
                                     wide?  
  
System.out.printf(, cerealBoxesPerDay);  
                  ↪ what format string goes here if I want 2 decimal  
                  places 4 characters wide?  
  
System.out.printf(, name);  
                  ↪ what format string goes here
```







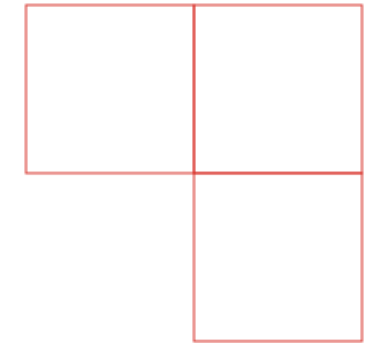
Programación/Entrada y salida



# Ejercicio

First Do It By Hand!





# Ejercicio

```
import java.util.Scanner;

public class Tiles
{
    public static void main(String[] args)
    {
        Scanner in = new Scanner(System.in);

        System.out.print("Width of wall: ");
        double width = in.nextDouble();

        System.out.print("Tile size: ");
        double tileSize = in.nextDouble();

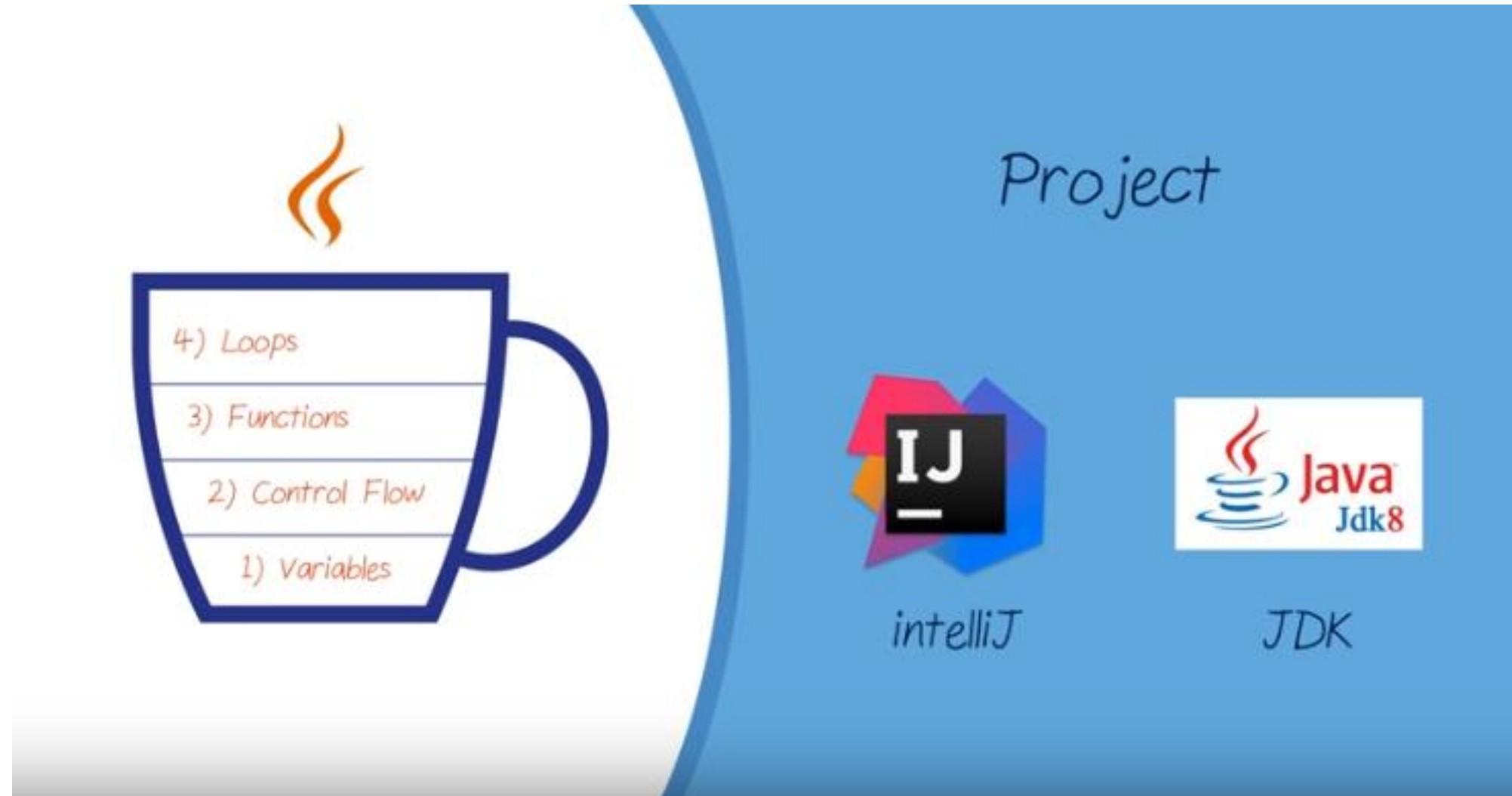
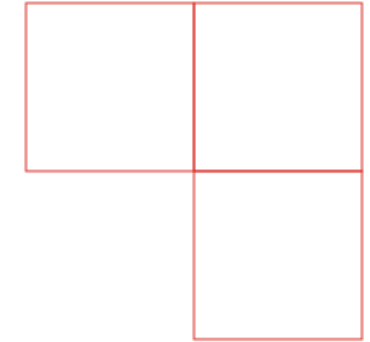
        // TODO: Complete these computations
        int pairs = ...;
        int redTiles = ...;
        int whiteTiles = ...;

        System.out.println("Red tiles: " + redTiles);
        System.out.println("Whites tiles: " + whiteTiles);
    }
}
```

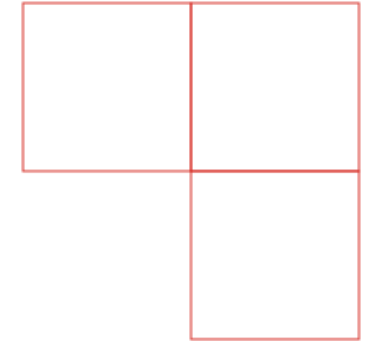




# APUNTE...







# APUNTE...

**Toodaim Biblioteca de Videojuegos**  
LIBROS SOBRE TODO LO RELACIONADO CON EL DESARROLLO DE VIDEOJUEGOS

LIBROS GRATUITOS EN PDF  
LIBROS COMERCIALES DESDE AMAZON

**GRATIS** PDF

**Libros sobre desarrollo de videojuegos**  
Una recopilación de libros sobre Diseño y Desarrollo de videojuegos. Guión, Arte, Programación, Marketing... Tanto libros gratuitos en PDF como libros comerciales desde Amazon.



**Universidad  
Europea**

**GRACIAS**

**Pedro J. Camacho**

**Universidadeuropea.com**

**Ve más allá**