La palabra reservada new

- Sirve para crear o instanciar objetos de una clase.
- Asigna memoria y llama al constructor para inicializar el objeto.

Ejemplo:

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

Aquí, **new Scanner(System.in)** crea un objeto de la clase Scanner.

Al principio puede parecer una "regla mágica", pero es necesaria para usar clases y objetos en Java.

Constructores

- Son métodos especiales que se ejecutan al crear o instanciar un objeto con **new**.
- Sirven para **inicializar** atributos o **preparar** el estado del objeto.
- Siempre tienen el **mismo nombre** que la clase y **no** tienen un **tipo de retorno.**

Ejemplo:

```
class Persona {
    String nombre;
    Persona(String nombre) {
         // Inicializa el atributo
        this.nombre = nombre;
    }
}
// al instanciar este objeto se llama al constructor con parámetros

Persona objetoPersona = new Persona("Juan");
```

Llamar a un método de un objeto

Se usa esta estructura:

nombreDelObjeto.nombreDelMétodo();

Ejemplo:

```
objetoPersona.saludar();
```

- `objetoPersona` es el objeto.
- `saludar()` es el método que pertenece a la clase de ese objeto.

Visibilidad (public, private, protected)

private: Cuando declaras un atributo o método como private, solo puede ser accedido o modificado desde dentro de la misma clase donde se declaró.

Ninguna otra clase, ni siquiera aquellas que estén en el mismo paquete o que hereden de esta clase, podrán acceder directamente a un miembro private. La idea es que los datos sean seguros y que las interacciones con ellos se hagan de manera controlada, generalmente a través de métodos públicos como getters y setters.

Ejemplo:

```
class Persona {
    // Atributos privados (protegidos del acceso externo)
    private String nombre;
    private int edad;
    // Constructor para inicializar los atributos
    public Persona(String nombre, int edad) {
        this.nombre = nombre;
        this.edad = edad;
    }
```

```
// Métodos públicos para acceder a los atributos (Getters)
    public String getNombre() {
        return nombre;
    }

    public int getEdad() {
        return edad;
    }

    // Métodos públicos para modificar los atributos
(Setters)

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public void setEdad(int edad) {
        if (edad > 0) {
         // Control adicional: Edad no puede ser negativa
            this.edad = edad;
        }
    }
}
```

Análisis del ejemplo:

Los atributos nombre y edad son privados (private), lo que significa que:

- No se pueden acceder directamente desde fuera de la clase.
- Protegemos estos datos para evitar que sean manipulados sin control.
- Proporcionamos métodos públicos (getNombre, setNombre, etc.) para leer y modificar los atributos:

Esto permite agregar lógica adicional al modificar un atributo. Por ejemplo, en *setEdad*, verificamos que la **edad no sea negativa antes de asignarla**. Si no tuviéramos estos métodos… y los atributos fueran públicos, se podría hacer algo como lo siguiente:

```
Persona persona = new Persona("Juan", 30);
persona.edad = -10; // ERROR: La edad puede ser inválida.
```

Esto sería **peligroso**, porque permitiría datos incorrectos o inconsistentes en el objeto.

protected: Accesible dentro de la misma clase, clases hijas y el mismo paquete.

Útil en herencia.

public: Accesible desde cualquier clase.

Ejemplo: métodos que quieres que usen otras clases.

Regla básica:

- Atributos: casi siempre private.
- Métodos: generalmente public, salvo que sean auxiliares.
- Clases: suelen ser public, salvo en casos específicos.

Reflexión sobre la programación

- No siempre tendrás claro el "por qué" de todo al principio, ¡no te preocupes!
- Muchas reglas y conceptos se entienden mejor con la práctica y la experiencia.
- La programación es un aprendizaje constante: incluso los expertos no lo saben todo.
- Lo importante es investigar por tu cuenta (esto lo vas a tener que hacer muy a menudo), practicar, experimentar y aceptar que entender todo lleva tiempo.

En la próxima **unidad 4**, se explicarán estos conceptos **con más detalle**, de momento lo importante es **saber cómo se usan.**