PALABRA CLAVE THIS EN JAVA. CONTENIDO NULL POR DEFECTO DE UN OBJETO.

El uso más frecuente de this en Java tiene lugar en este contexto: **cuando existe sobrecarga de nombre**s. La sobrecarga de nombres se da cuando tenemos una variable local de un método o constructor, o un parámetro formal de un método o constructor, con un nombre idéntico al de un campo (propiedad) de la clase.

Este sería un mal ejemplo de sobrecarga de nombres, tanto con parámetros como con variables locales:

```
class Mensaje {
   //Campos
   private String remitente;
   private String para;
   private String texto;

   //Constructor con sobrecarga de nombres al coincidir nombres de parámetros con los de campos
   Mensaje (String remitente, String para, String texto) {
      remitente = remitente; //¿Cómo va a saber Java cuál es el parámetro y cuál el campo?
      para = para; //¿Cómo va a saber Java cuál es el parámetro y cuál el campo?
      texto = texto; //¿Cómo va a saber Java cuál es el parámetro y cuál el campo?
   } //Cierre del constructor

   //Método con sobrecarga de nombres al coincidir un parámetro con un campo
   public void setRemitente (String remitente) {
      remitente = remitente; //¿Cómo va a saber Java cuál es el parámetro y cuál el campo?
   } //Cierre del método

   //Método con sobrecarga de nombres al coincidir una variable local con un campo
   public void extraerFraccionTexto () {
      String texto = ""; //Esto supone declarar una variable local que "tapa" al campo
      texto = texto.substring (0, 5);  //¿Cómo va a saber Java si nos referimos al campo?
   } //Cierre del método
} // Cierre de la clase
```

Escribe y compila el código anterior. El código, efectivamente compila. ¿Por qué? Porque Java tiene previstos mecanismos para resolver conflictos de nombres y aplica una reglas. En concreto, la regla de que "un nombre hace referencia a la variable más local de entre las disponibles". Y el carácter de local se interpreta de la siguiente manera:

Variable local > Parámetro formal > Campo de clase

un método verMensaje y pruébalo

Interpretemos ahora por qué nos aparece null como contenido de los atributos del objeto. En el constructor hemos definido tres parámetros: remitente, para y texto. Luego hemos indicado que remitente = remitente;. ¿Qué variable usa Java? Tiene que elegir entre usar el campo o usar el parámetro del método. No puede usar ambos porque no podría saber

cuándo usar uno y cuándo usar otro. El conflicto lo resuelve utilizando el parámetro del método, es decir, interpretando que “el parámetro es igual al parámetro”. Esto no tiene ningún efecto, lo que significa que el atributo remitente se queda sin inicializar. Lo hemos declarado, pero no lo hemos inicializado.

En nuestra clase tenemos tres campos o variables de instancia cuyo ámbito es toda la clase. El ámbito de los parámetros o variables locales es exclusivamente el constructor o método al cual se aplican.

Recordemos que un String es un objeto en Java. Un objeto no inicializado carece de contenido y esto nos lo informa Java indicándonos un contenido aparente null. La palabra clave null indica que un objeto se encuentra carente de contenido.

Volvamos ahora al código de nuestra clase Mensaje. El conflicto de nombres vamos a solventarlo haciendo uso de la palabra clave this. Escribiendo this.nombreDelCampo le indicaremos a Java que nos referimos al atributo de la clase en vez de a una variable local o parámetro. Veámoslo aplicado en el código:

```
public class Mensaje {
    private String remitente;
    private String para;
    private String texto;

    //Constructor con sobrecarga de nombres al coincidir nombres de parámetros con los de campos
    public Mensaje (String remitente, String para, String texto) {
        this.remitente = remitente; //this.remitente es el campo y remitente el parámetro
        this.para = para; //this.para es el campo y para el parámetro
        this.texto = texto; //this.texto es el campo y texto el parámetro
    }

    //Método con sobrecarga de nombres al coincidir un parámetro con un campo
    public void setRemitente (String remitente) {
        this.remitente = remitente; //this.remitente es el campo y remitente el parámetro
    }

    //Método con sobrecarga de nombres al coincidir una variable local con un campo
    public String extraerFraccionTexto () {
        String texto = ""; //texto es una variable local
        texto = this.texto.substring (0, 5);  //this.texto es el campo de los objetos de la clase
        return texto;
    }  }
```

Crea ahora un objeto de tipo Mensaje e inicialízalo introduciendo valores para los parámetros requeridos por el constructor. El resultado es que ahora sí se produce una inicialización correcta porque hemos definido adecuadamente cómo han de gestionarse los nombres de variables.

EJERCICIO

1.Define una clase Profesor considerando los siguientes atributos de clase: nombre (String), apellidos (String), edad (int), especialista (boolean). Define un constructor que reciba los parámetros necesarios para la inicialización y otro constructor que no reciba parámetros. El nombre de los parámetros debe ser el mismo que el de los atributos y usar this para asignar los parámetros recibidos a los campos del objeto. Crea los métodos para poder establecer y obtener los valores de los atributos, con sobrecarga de nombres y uso de this en los métodos setters (los que sirven para establecer el valor de los atributos). Compila el código para comprobar que no presenta errores, crea un objeto usando el constructor con sobrecarga. Comprueba que se inicializa correctamente consultando el valor de sus atributos después de haber creado el objeto. Usa los métodos setters y comprueba que funcionen correctamente.

2. Crea una clase llamada Cuenta que tendrá los siguientes atributos: titular y cantidad (puede tener decimales).

El titular será obligatorio y la cantidad es opcional. Crea dos constructores que cumpla lo anterior.
Crea sus métodos get, set .
Tendrá dos métodos especiales:
•ingresar(double cantidad): se ingresa una cantidad a la cuenta, si la cantidad introducida es negativa, no se hará nada.
•retirar(double cantidad): se retira una cantidad a la cuenta, si restando la cantidad actual a la que nos pasan es negativa, la cantidad de la cuenta pasa a ser 0.

3) Haz una clase llamada **Password** que siga las siguientes condiciones:
•Que tenga los atributos **longitud** y **clave** . Por defecto, la longitud sera de 8.
•Los constructores serán los siguiente:
•Un constructor por defecto.
•Un constructor con la longitud que nosotros le pasemos. Generara una clave aleatoria con esa longitud.
•Los métodos que implementa serán:
•**esFuerte()**: devuelve un booleano si es fuerte o no, para que sea fuerte debe tener más de 2 mayúsculas, más de 1 minúscula y más de 5 números.
•**generarPassword()**: genera la clave del objeto con la longitud que tenga.
•Método get para clave y longitud.
•Método set para longitud.
Ahora, crea una clase clase ejecutable:
•Crea un array de Passwords con el tamaño que tu le indiques por teclado.
•Crea un bucle que cree un objeto para cada posición del array.
•Indica también por teclado la longitud de los Passwords (antes de bucle).
•Crea otro array de booleanos donde se almacene si el password del array de Password es o no fuerte (usa el bucle anterior).
•muestra la clave generada y si es o no fuerte (usa el bucle anterior). Usa este simple formato:
clave1 valor_booleano1
clave2 valor_bololeano2