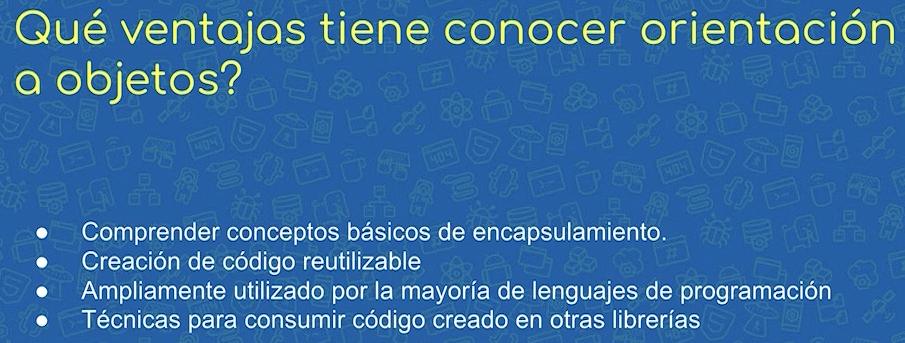
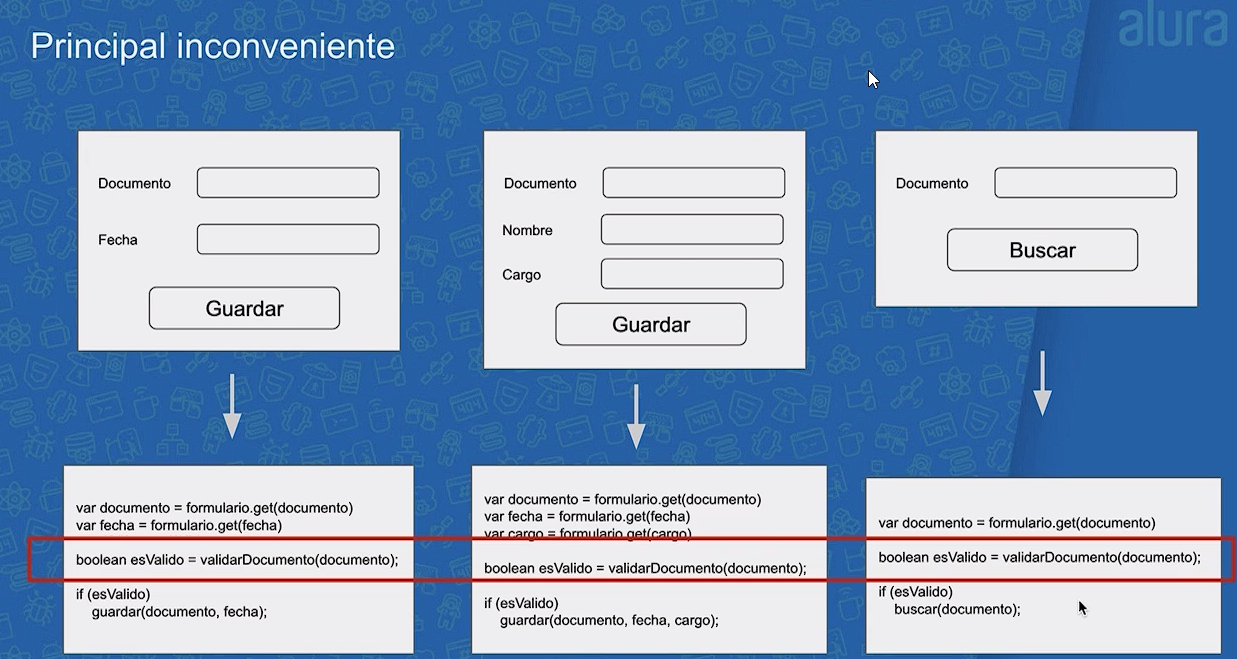
Curso de  
**Java OO: Entendiendo la Orientación a Objetos**



**El problema del paradigma procedimental:**

Introducción a orientación de objetos:



Lo que aprendimos:

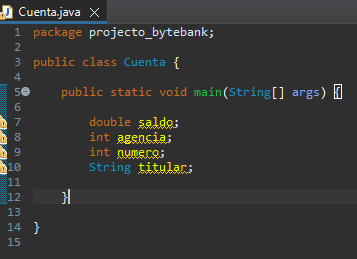
En esta clase conocimos el paradigma procedural, que se utilizó como práctica de programación antes de la introducción de lenguajes orientados a objetos. La necesidad de validar el numeroIdentidad en un formulario se utilizó como ejemplo para discutir los principales problemas que pueden aparecer en este paradigma.

En particular, a medida que otros formularios y desarrolladores necesitan la misma validación de numeroIdentidad, no fue fácil ver que ya había procedimientos y funciones que hicieron este trabajo, ya que los datos y las funciones no tenían un vínculo tan fuerte. Esto podría dar lugar a otra nueva función o fragmento de código con una responsabilidad similar.

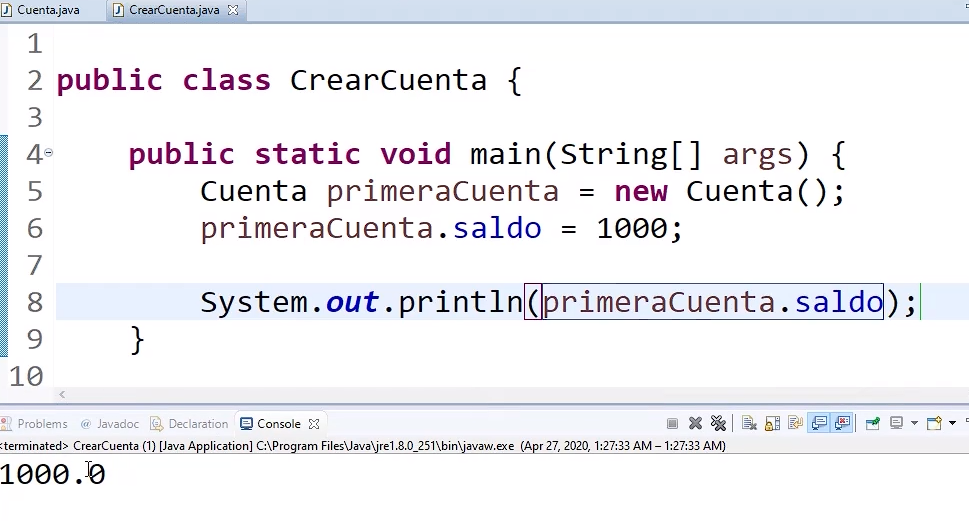
Además, conocemos la idea central del paradigma orientado a objetos, que es crear unidades de código que combinen los datos asociados con cierta información con las funcionalidades aplicadas a esos datos (por ejemplo, numeroIdentidad + validación). Son los atributos y métodos.

**Introducción a orientación a objetos:**

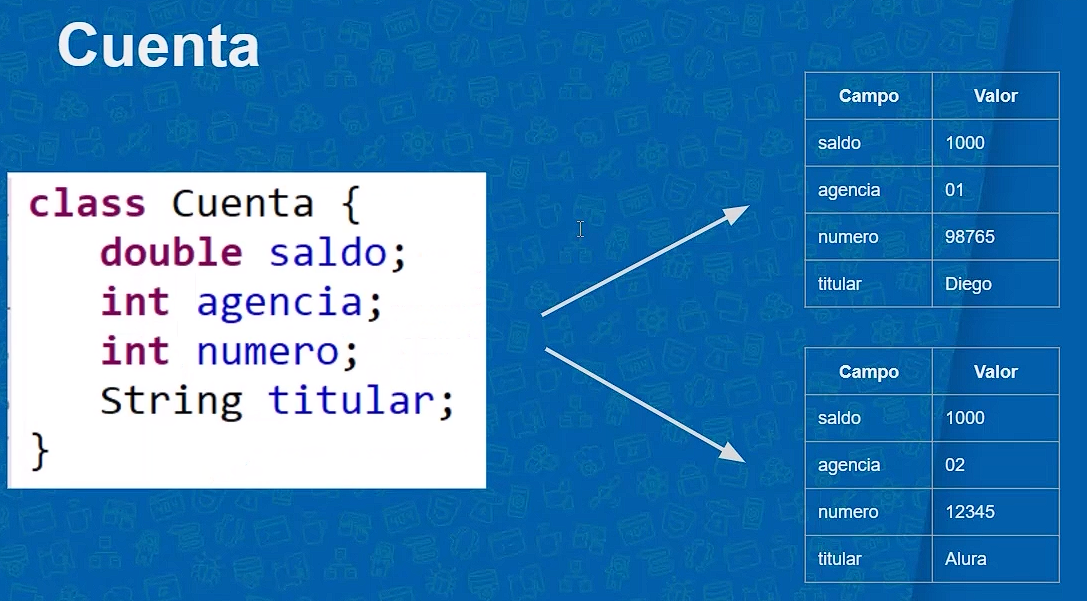
Clase cuenta:



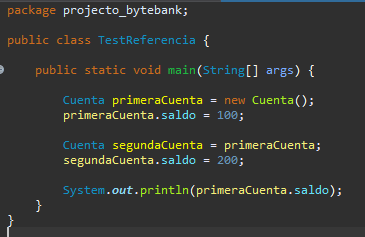
Instancias 1:



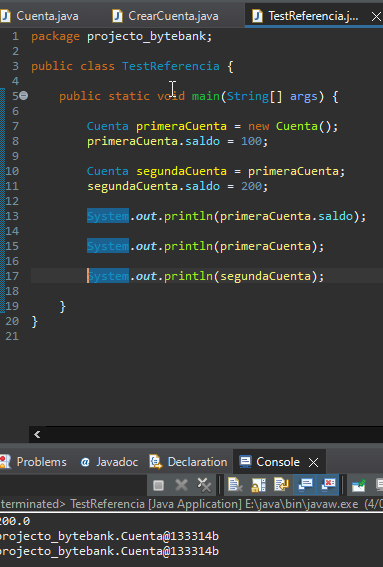
Instancias 2:



Referencias 1:



Referencias 2:



Lo que aprendimos:

En esta clase aprendimos sobre atributos y creación de objetos.

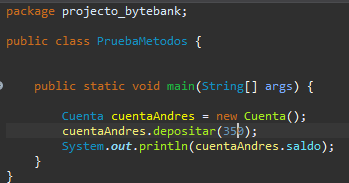
Vimos:

¿Qué es una clase? Cómo crear una clase Cómo crear un objeto o instancia de una clase ¿Qué son los atributos? Cómo definir y ver los valores de los atributos ¿Cómo funciona la asignación de una referencia a una variable?

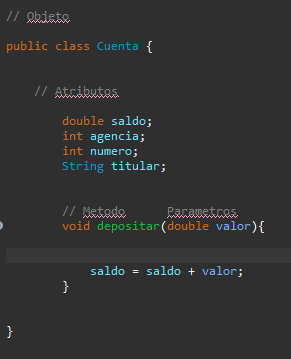
¡Veremos más sobre las clases en la próxima clase! Ahora no solo sus características sino también su comportamiento. ¡Solo comencemos! ¡Te espero allí!

**Definiendo comportamientos:**

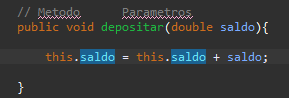
Creación método:



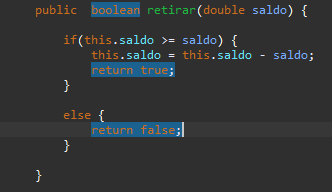
Implementación método:



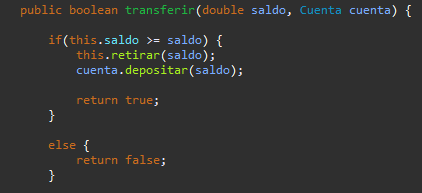
Uso this:

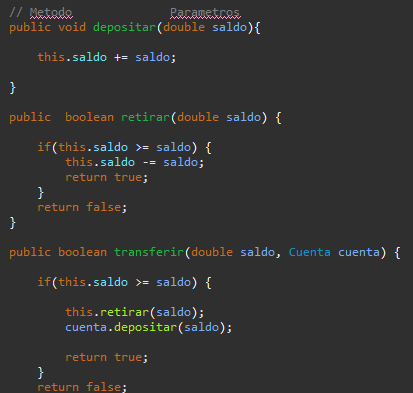


Método con retorno:



Referenciando objetos:







Lo que aprendimos:

En esta clase hablamos sobre el comportamiento que son los métodos.

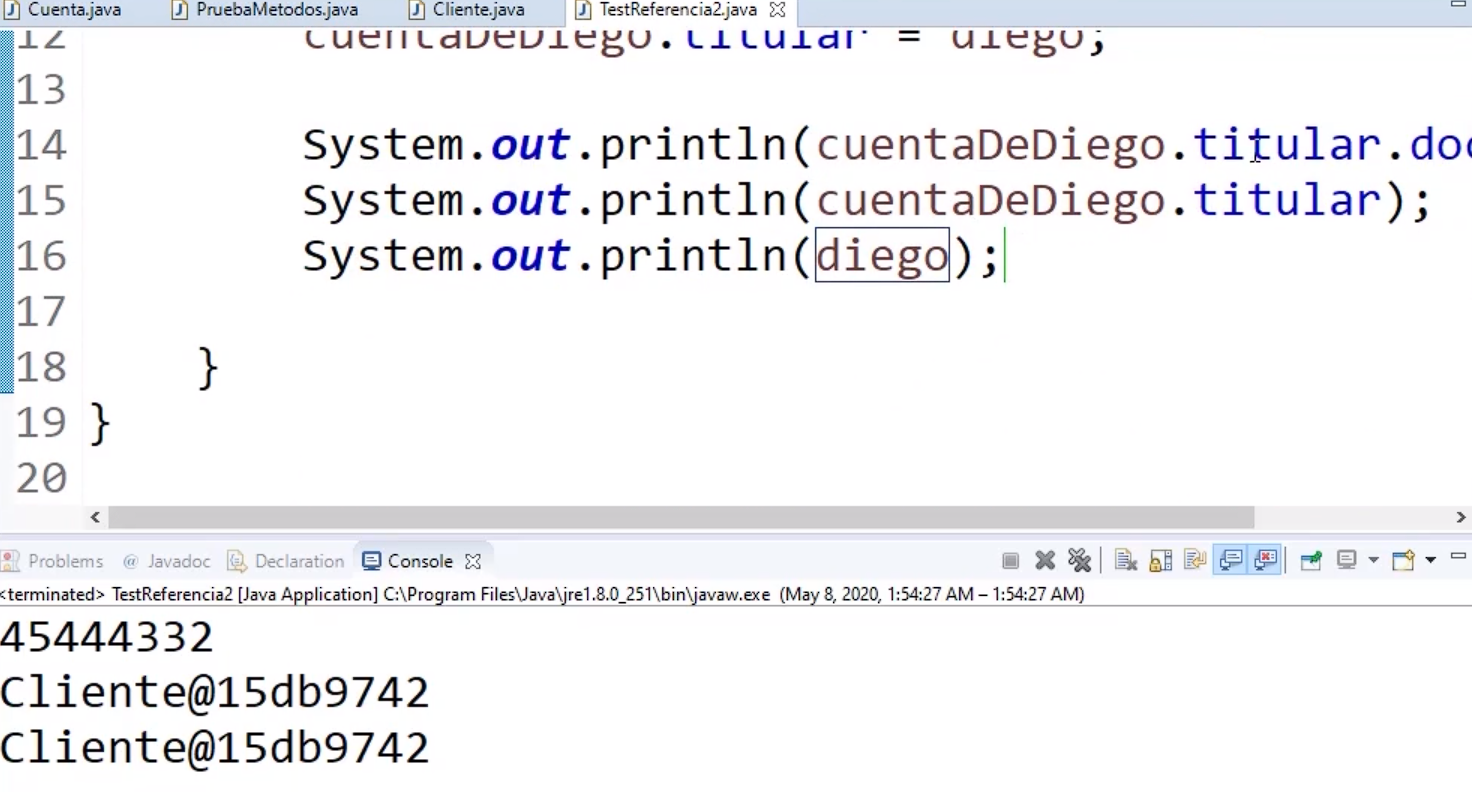
Vimos:

cómo definir métodos con parámetros y retorno cómo devolver algo usando la palabra clave return cómo usar la referencia thispara acceder a un atributo que podemos pasar una referencia como parámetro de método los métodos se invocan desde la referencia utilizando el operador .

Si aún tiene dudas sobre las referencias, this y el uso de métodos, tenga la seguridad de que en las próximas clases (y cursos) revisaremos los conceptos y practicaremos mucho más. ¿Continuamos?

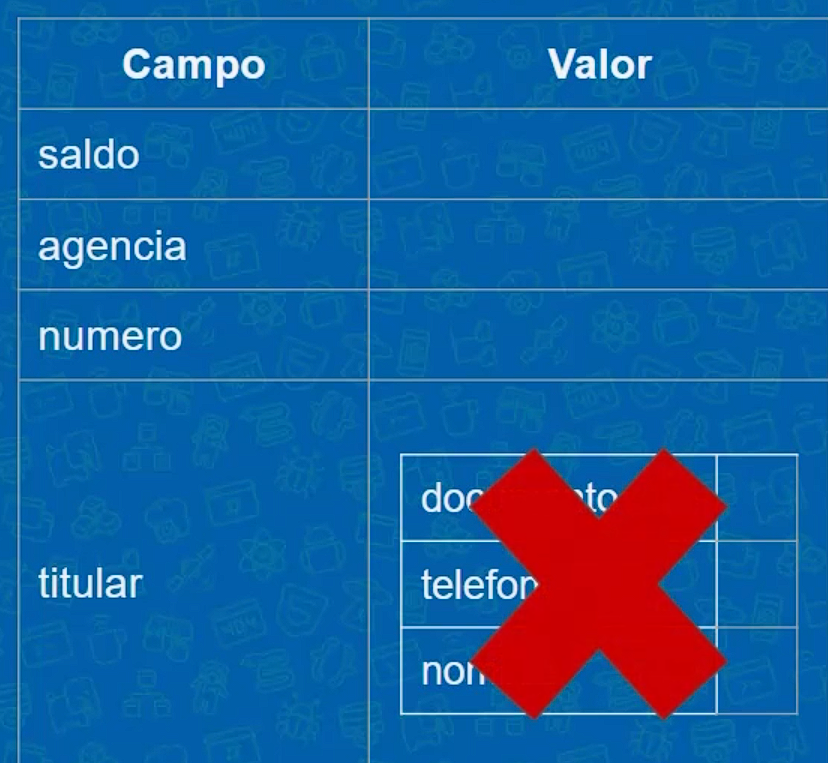
**Composición de objetos:**

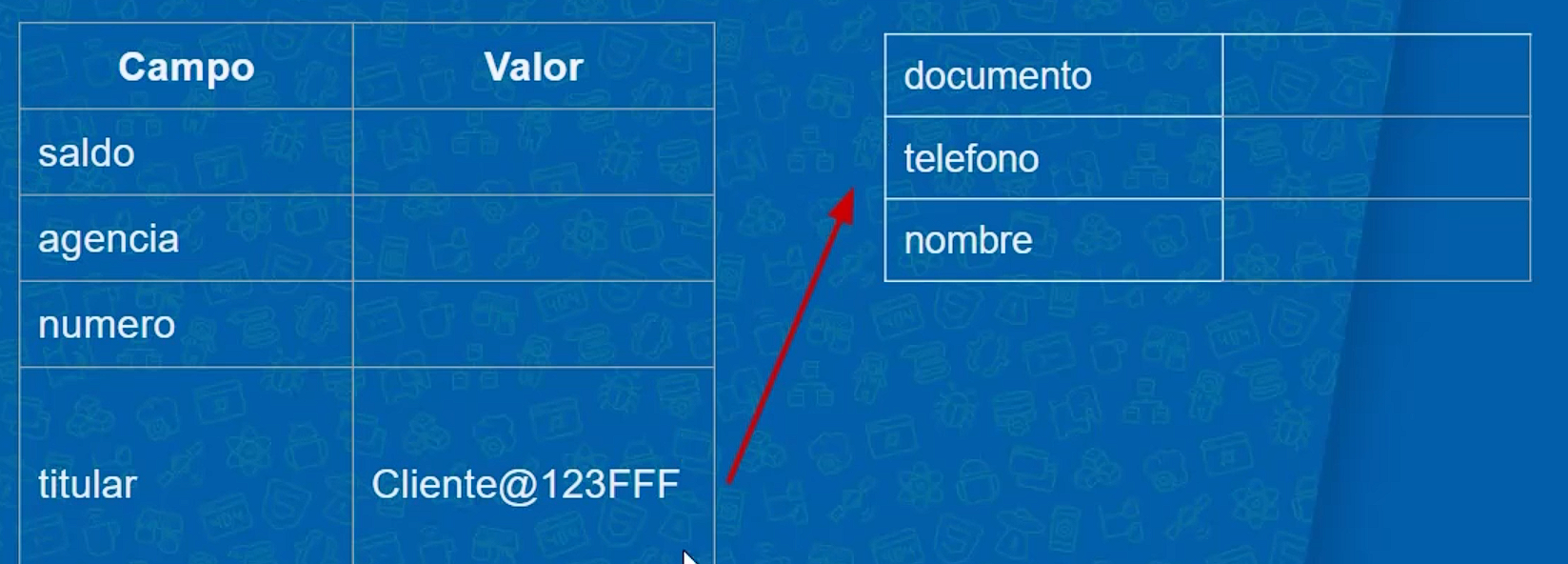
Referencia objetos:



Valores null:

Referencias de inicialización:





Lo que aprendimos:

Aprendimos en este capítulo:

• Darse cuenta de la relación entre clases a través de la composición;

• Ventajas de aislar información repetida en otra clase;

• Detalles de la composición;

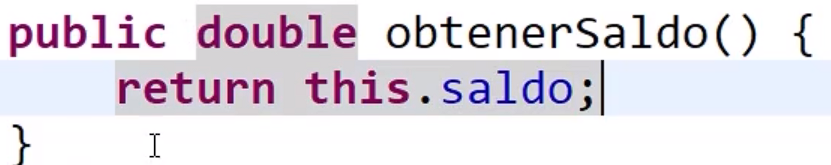
• NullPointerException para atributos no inicializados y su cuidado.

**Encapsulamiento y visibilidad:**

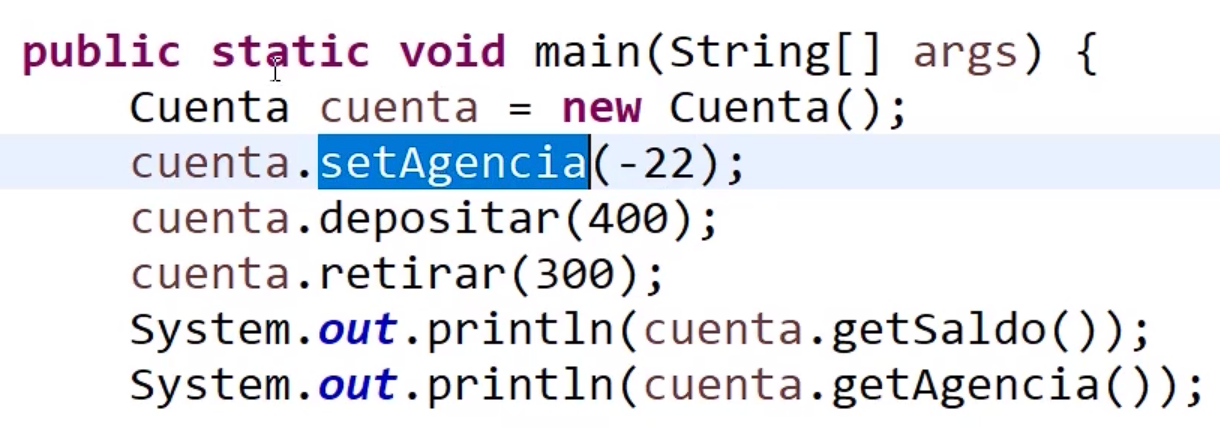
Encapsulamiento:

PRIVATE Y PUBLIC = ACCESO

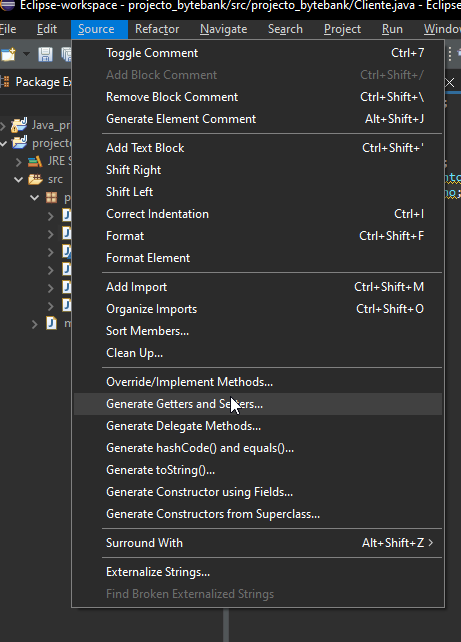
Getter:



Setter:



Generando Getter Setter:



Referencias:

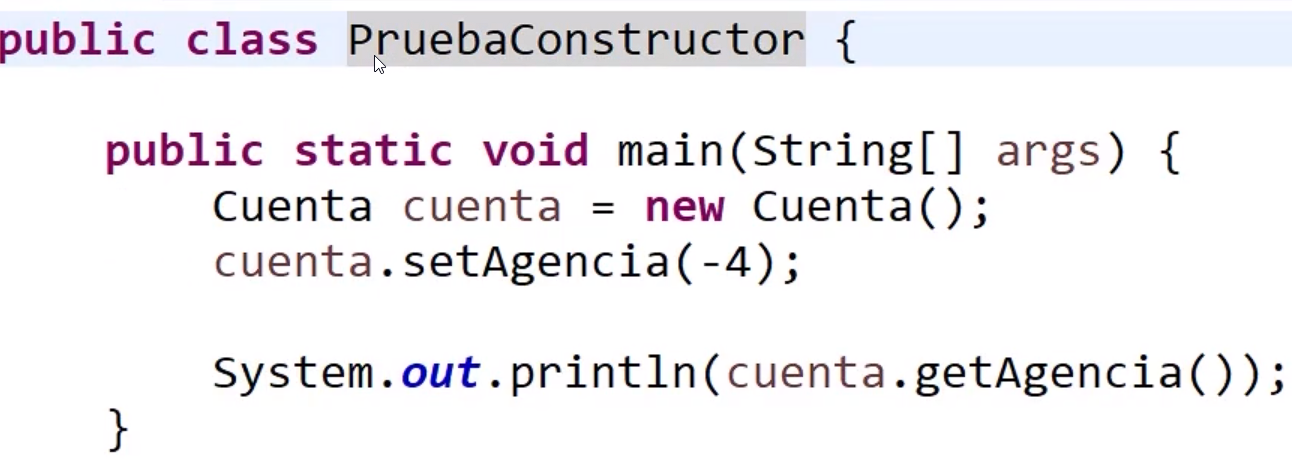
Lo que aprendimos:

Aprendimos en esta clase:

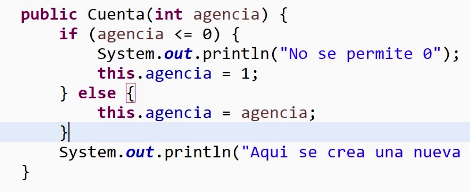
Atributos privados, restringiendo el acceso a los atributos. Encapsulación de código Métodos de lectura de atributos, los getters Métodos para modificar atributos, los setters Getters y Setters de referencia.

**Constructores y miembros estáticos:**

Constructor:



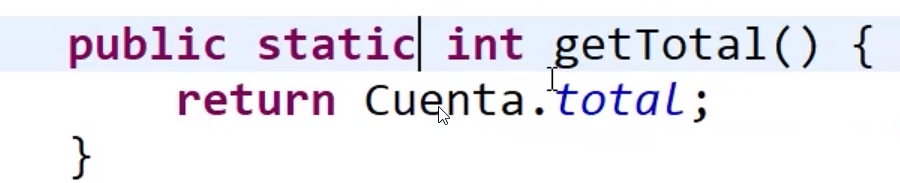
Constructor personalizado:



Variables estáticas:



Miembros estáticos:



Lo que aprendimos:

Aprendimos en esta clase:

Constructor de clases, que permite recibir argumentos e inicializar atributos desde la creación de un objeto. Con esto, la inicialización de los atributos recibidos en el constructor se vuelve obligatoria. Atributos de clase, atributos estáticos. Métodos de clase, métodos estáticos. Ausencia de referencia, del this, dentro de los métodos estáticos.