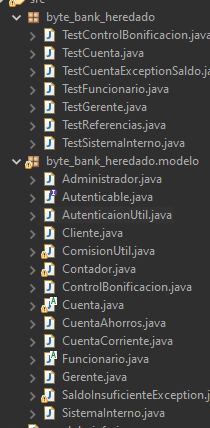
Curso de  
Java y java.lang: Programe con la clase

Object y String

**Organizando clases con paquetes:**

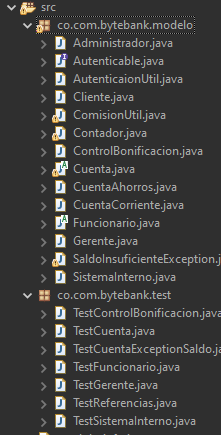
Usando Paquetes:



Actualizando Clases



Importando Clases:



Estructura de Clases:

Correcto, el FQN (Full Qualified Name) es el nombre completo de la clase, que consta del nombre del paquete y el nombre de la clase.

FQN = Nombre del paquete. Nombre de clase simple

**Lo que Aprendimos:**

¿Qué aprendimos?

* Los packages sirven para organizar nuestro código.
* Los packages son parte del FQN (*Full Qualified Name*) de la clase.
* El nombre completo de la clase (FQN) consta de: PACKAGE.NOMBRE\_SIMPLE\_DE\_LA\_CLASE
* La definición de package debe ser la primera declaración en el código fuente
* Para facilitar el uso de clases de otros packages, podemos importarlas
* Los *import* son justo después de la declaración del package
* La nomenclatura padrón es usar el nombre de dominio en la web al revés con el nombre del proyecto, por ejemplo:

br.com.caelum.geradornotas

br.com.alura.gnarus

br.gov.rj.notas

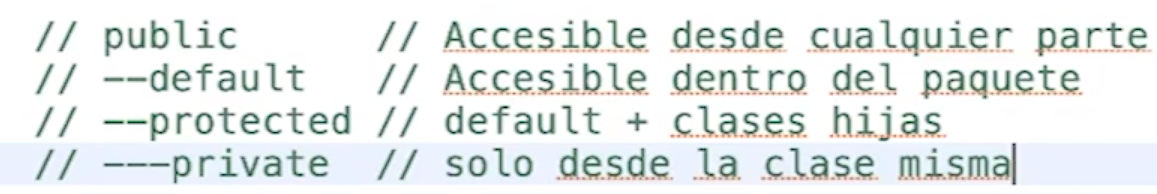
de.adidas.lagerCOPIA EL CÓDIGO

Una vez organizadas nuestras clases, podemos revisar los modificadores de visibilidad que dependen de los paquetes.

**Todos los modificadores de acceso:**

Modificadores de acceso:

* Default: Solo es visible dentro del mismo paquete.
* Public: No importa donde estes, serás visible para todos.
* Protected: Visibilidad a nivel de paquete y de herencia.
* Private: No puedes acceder fuera de la clase.



**Lo que Aprendimos:**

¿Qué aprendimos?

En esta clase volvimos a hablar sobre visibilidad y aprendimos:

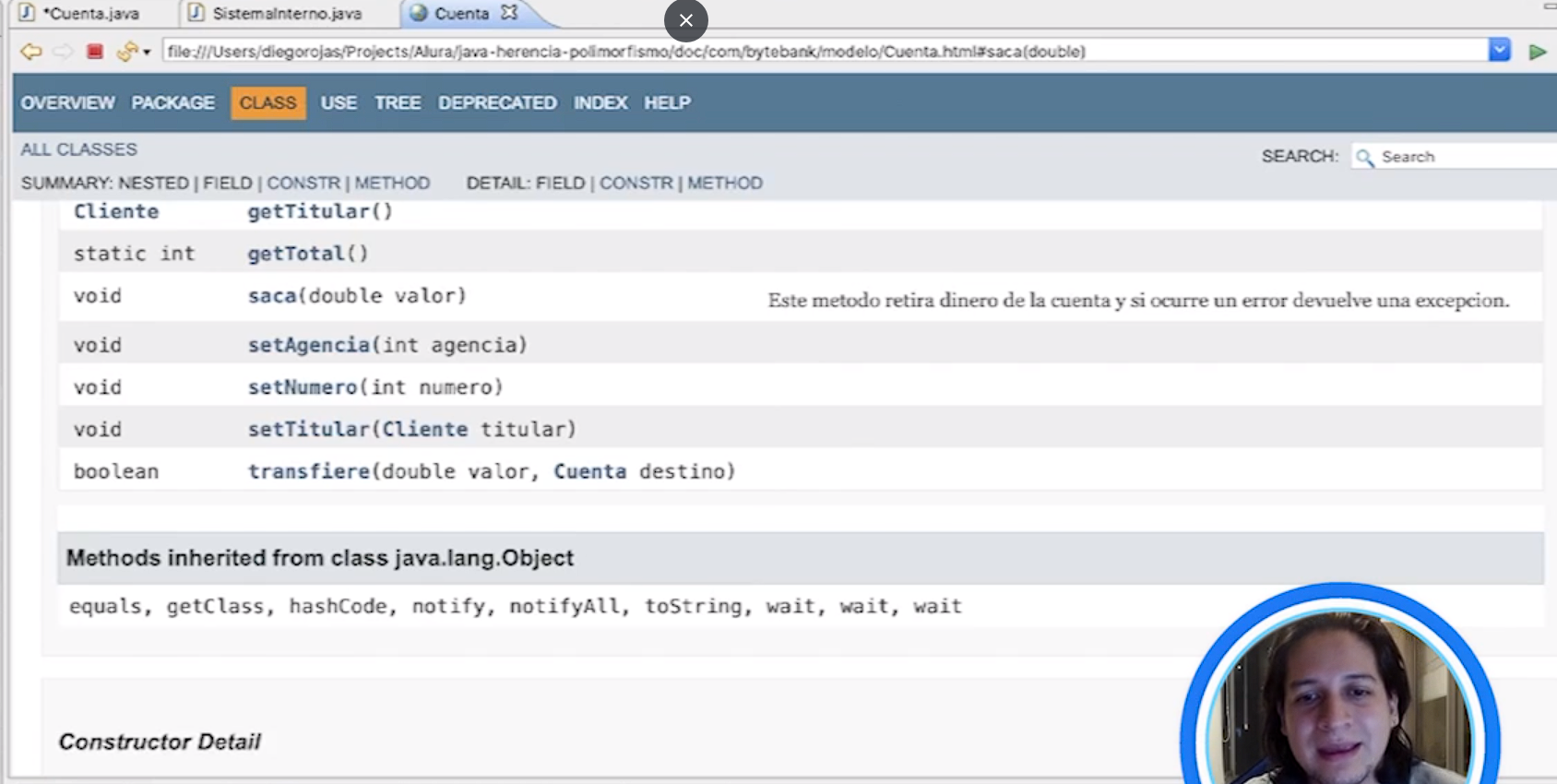
* Hay 3 palabras clave relacionadas con la visibilidad: *private, protected, public*
* Hay 4 niveles de visibilidad (de menor a mayor):
  + private (visible solo en clase)
  + <<package private>> (visible en la clase y en cualquier otro miembro del mismo paquete, que puede ser llamado de default)
  + protected (visible en la clase y en cualquier otro miembro del mismo paquete y para cualquier hijo)
  + public (visible en cualquier paquete)
* Los modificadores pueden ser usados en la definición de la clase, atributo, constructor y método.

**Distribucion de código:**

Documentacion Javadoc:

Los comentarios en azul son los que al final quedaran en el documento de mi programa.

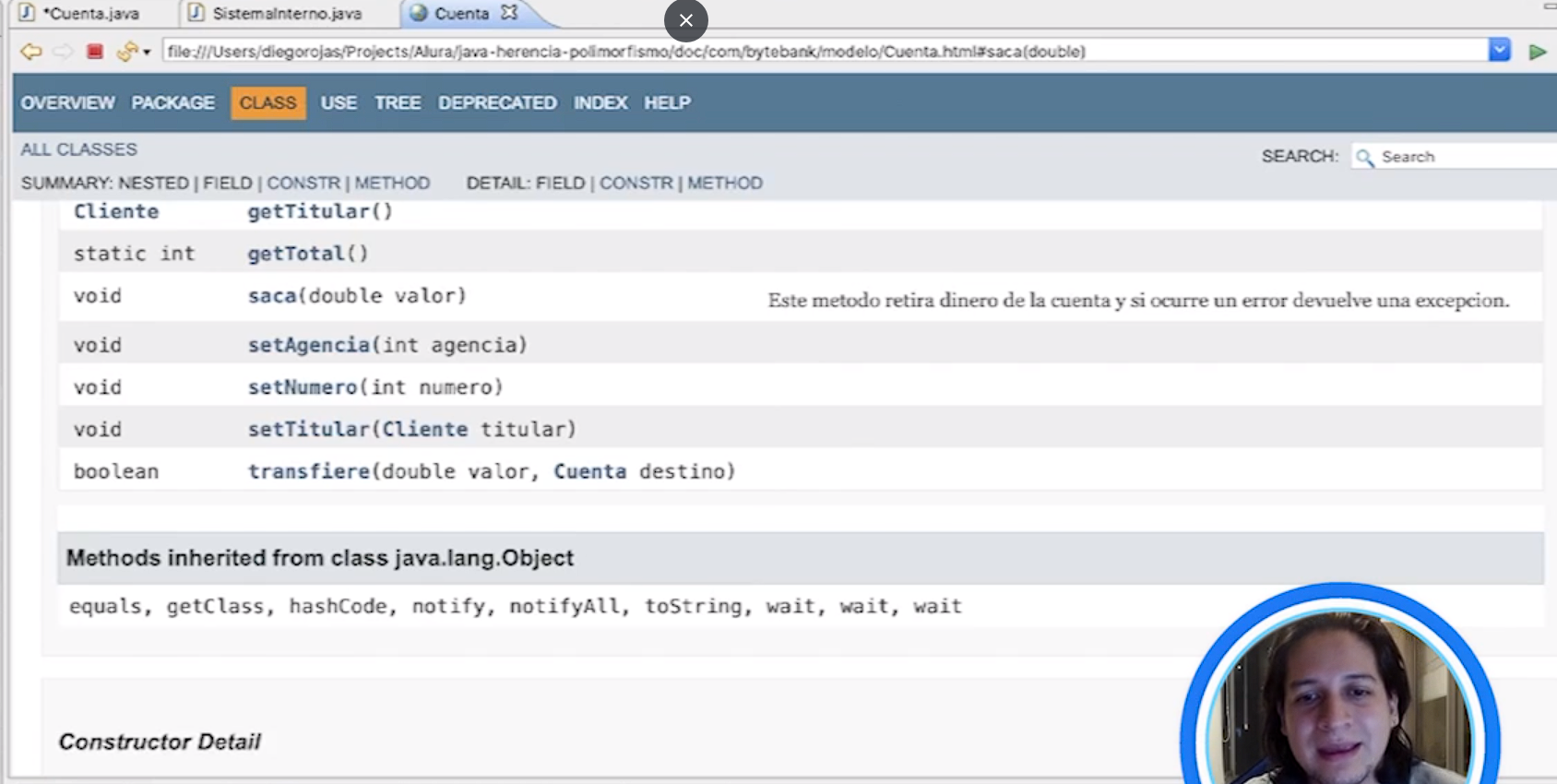
Generando Javadoc:



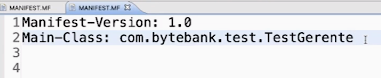
Archivos JAR:

Export, Java, JAR File.

Usando Bibliotecas con JAR:



JAR ejecutables:



**Lo que Aprendimos:**

¿Qué aprendimos?

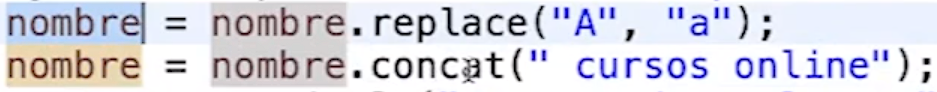
En esta sección más ligera vimos y aprendimos:

* Qué comentarios y tags (anotaciones) usar para definir el javadoc
* Cómo generar javadoc en Eclipse
* Que javadoc es una documentación para desarrolladores
* Que las clases estándar de Java también usan javadoc
* Cómo crear nuestra propia librería a través de JAR (J\*ava \*ARchive)
* Cómo importar librerías al nuevo proyecto
* Cómo crear un JAR ejecutable

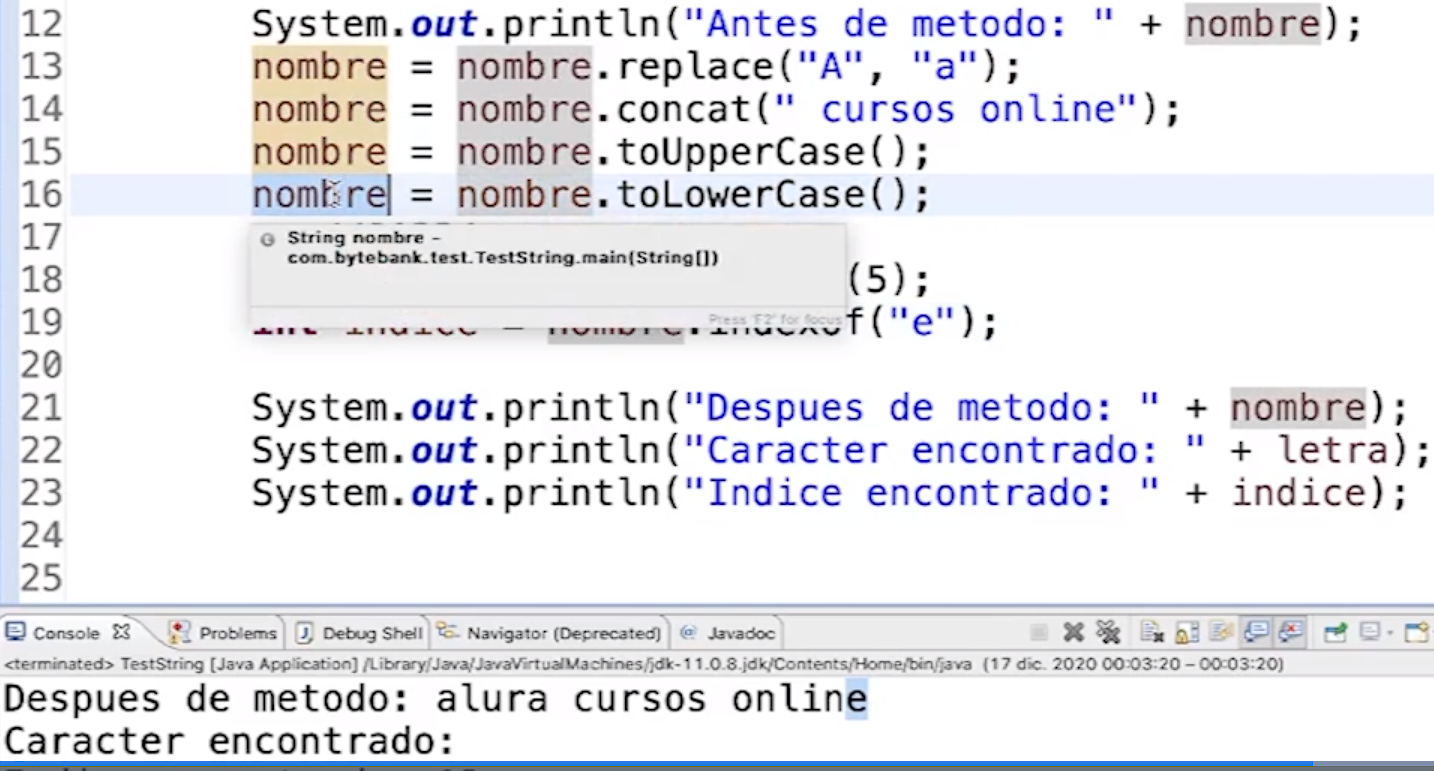
**El paquete java.lang:**

Explorando String:

* STRING ES INMUTABLE.



Metodos String:



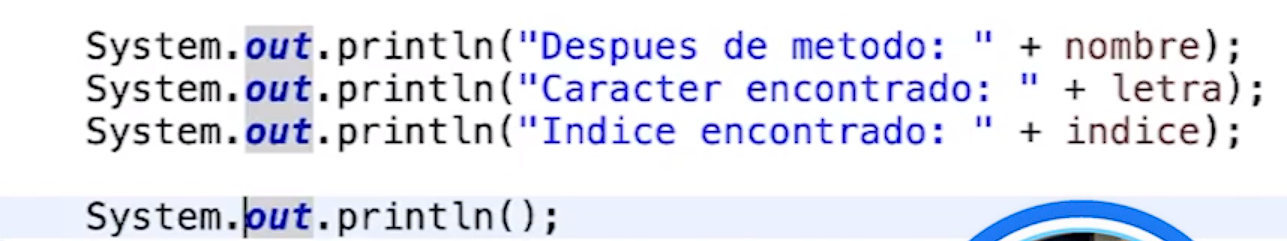
**Lo que Aprendimos:**

En esta clase aprendimos y sabemos:

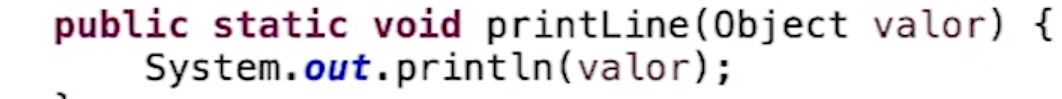
* El package java.lang es el único paquete que no necesita ser importado
* Tiene clases fundamentales que cualquier aplicación necesita, como la clase String y System
* Los objetos de la clase String son inmutables y usamos una sintaxis literal para crear (objeto literal)
* Cualquier método para cambiar la clase String devuelve un nuevo String que representa el cambio
* La clase String es una CharSequence
* Si necesitamos concatenar muchos String debemos usar la clase StringBuilder
* Vimos varios métodos de la clase String como trim, charAt, contains, isEmpty, length, indexOf, replace

**La clase object:**

Explorando System:



Clase Object:



Metodo toString():

