**Sesión 2: “Prepararse para atender la problemática a resolver” – Java**

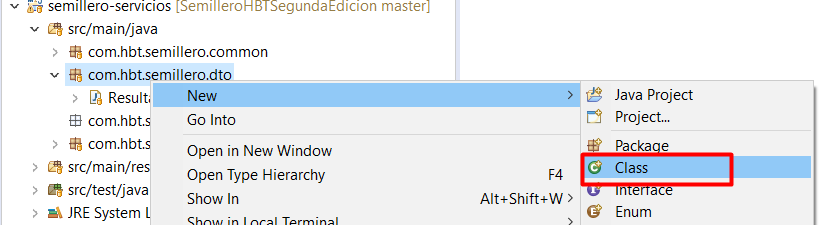
* Java básico
* Definición de la estructura de una clase
* Crear y ejecutar clases, uso de importaciones, paquetes, modificadores de acceso, clases e interfaces abstractas, static.
* Tipos de datos
* Métodos y encapsulación
* Clases del api, arrays
* Flujos de control
* Control de errores

**Objetivos de la sesión:**

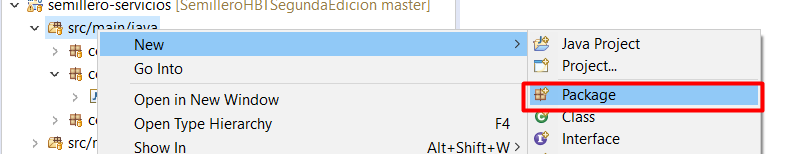
1. Entender el uso, el almacenamiento y el manejo de estructuras de datos.

**Desarrollo de la sesión**

* ¿Como se va a representar los datos requeridos para manejar un comic?: Para presentar los datos requeridos se va a usar dos clases, una clase ComicDTO.java la cual va a permitir definir el contenedor de la información que debe viajar desde la capa de servicios (negocio) a capa de web (presentación) y una clase Comic.java la que va a permitir representar información que está en la base de datos en la tabla DB\_SEMILLERO.COMIC.
* ¿Dónde se va a crear DTO? se debe crear un paquete “com.hbt.semillero.dto” y posteriormente crear una nueva clase que se llame ComicDTO.java



* ¿Dónde se va a crear la entidad?: se debe crear un paquete “com.hbt.semillero.entidad” posteriormente crear una nueva clase que se llame Comic.java



* ¿Qué atributos debe tener la clase ComicDTO.java?: Primero se debe tener claro el mapeo de los datos que se deben tener, como base se proponen unos campos presentes en el sql \utiles\ SQL\_TABLAS.sql

Inicialmente se va a crear los siguientes atributos:

**private** String id;

**private** String nombre;

**private** String editorial;

**private** String tematica;

**private** String coleccion;

**private** Integer numeroPaginas;

**private** BigDecimal precio;

**private** String autores;

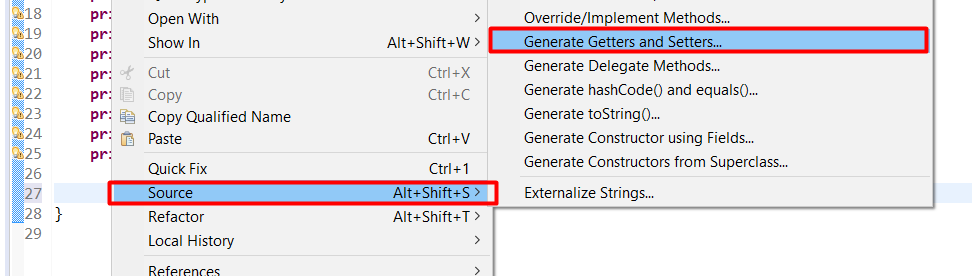
**private** Boolean color;

**private** LocalDate fechaVenta;

**private** String estado;

**private** Long cantidad;

* ¿Como se puede acceder a un atributo privado? Se deben crear métodos set (para poner el valor de un dato) y get (para obtener el valor del atributo): Estos métodos se pueden autogenerar, dando clic derecho en la clase y seleccionando lo siguiente:



* ¿Como se crea una entidad? A diferencia de los DTO’s las entidades son clases con mas significado ya que requieren usar la librería javax.persistence por medio de sus métodos permite mapear la tabla, sus campos y más funcionalidades. Se debe agregar la dependencia en el archivo pom.xml

<!-- Dependencia para el manejo de la persistencia -->

<dependency>

<groupId>org.hibernate</groupId>

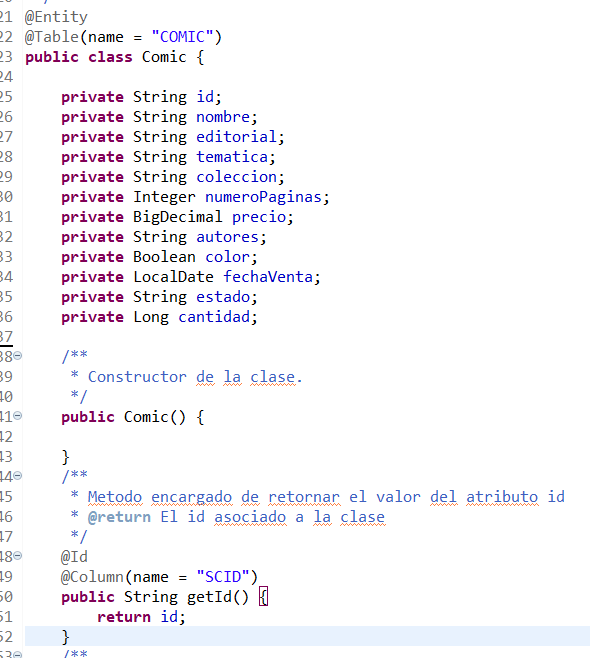
<artifactId>hibernate-core</artifactId>

<version>5.2.12.Final</version>

</dependency>

* Se debe tener presente las siguientes reglas para crear una entidad.

La clase debe estar anotada con @Entity, se debe tener un constructor (el constructor por defecto es suficiente), si la tabla tiene un nombre diferente a la entidad se debe colocar la anotación @Table(name = "COMIC"), cada campo de la tabla debe estar mapeado o puesto en la entidad con su correspondiente anotación @Column(name = " SCID")



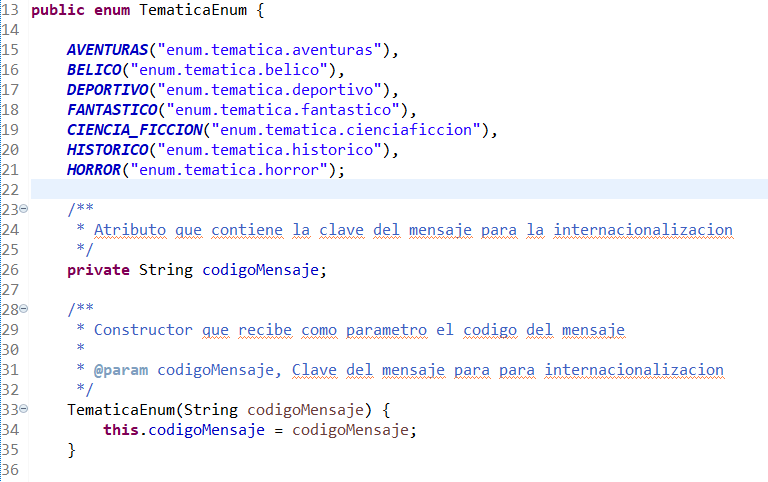
* ¿Como se puede obtener un dato almacenado preveniente para ser manipulado en la capa de negocio en java?: Se puede utilizar la librería javax.persistence la cual brinda las clases y métodos indicados para usar la persistencia.
* Dado que el campo id es un identificador único para el comic, se debe garantizar que no se registre varias veces el mismo id para diferentes comics, para esto se puede usar una secuencia la cual automáticamente genera números consecutivos, simplemente se debe garantizar que la secuencia este creada en la base de datos y luego anotar el método getId() con lo siguiente

@SequenceGenerator(allocationSize = 1, name = "COMIC\_SCID\_GENERATOR", sequenceName = "SEQ\_COMIC")

@GeneratedValue(strategy = GenerationType.***SEQUENCE***, generator = "COMIC\_SCID\_GENERATOR")

* ¿Como se puede trabajar con los campos que tienen condiciones o restricciones definidas previamente en la base de datos? Los campos SCTEMATICA y SCESTADO presentan ciertas condiciones las cuales se pueden modelar usando campos predefinidos para su representación, cambiar nombre de atributo temática por tematicaEnum, crear el enum TematicaEnum.class en él paquete “com.hbt.semillero.entidad”, cambiar las firmas de los métodos get y set agregar anotación @Enumerated(value = EnumType.STRING) en el get del campo tematicaEnum.

La clase TematicaEnum.class debe tener las opciones validas, un constructor y en este caso se agrega un atributo con un código de mensaje que debe tener su correspondiente get y set.



* En la clase Comic.java se le pueden sobrescribir los métodos toString(), hashCode() y equals(Object obj)

**Ejercicio sesión 2**

* Realizar la enumeración para el campo estado EstadoEnum.java y hacer los correspondientes ajustes a la clase Comic.java
* Modificar la clase AppTest.java y crear un método test que permita manipular el EstadoEnum, y validando las siguientes condiciones:

Instanciar un enum de la clase EstadoEnum

Devolver un String con el nombre de la constante (ACTIVO)

Devolver un String con el nombre de la constante (ACTIVO)

Devolver un entero con la posición del enum según está declarada

Comparar el enum con el parámetro según el orden en el que están declarados lo enum

Devolver un array que contiene todos los enum

* Lectura “Conceptos básicos de Modelado.docx”

**Conceptos adicionales**

* “Data Transfer Object (DTO) – Patrón de diseño, tiene como finalidad de crear un objeto plano (POJO) con una serie de atributos que puedan ser enviados o recuperados del servidor en una sola invocación, de tal forma que un DTO puede contener información de múltiples fuentes o tablas y concentrarlas en una única clase simple.”
* Entidades son clases que representa al modelo de datos
* Las capas de negocio, presentación permiten determinar una división lógica de las tecnologías y artefactos a usar en la construcción de aplicaciones.
* Un enumerado (o Enum) es una clase "especial" (tanto en Java como en otros lenguajes) que limitan la creación de objetos a los especificados explícitamente en la implementación de la clase.
* java.lang.Object#toString() Metodo que permite asociar al objeto un texto representativo
* java.lang.Object#hashCode() Este método viene a complementar al método equals y sirve para comparar objetos de una forma más rápida en estructuras Hash ya que únicamente nos devuelve un número entero. CuandoJava compara dos objetos en estructuras de tipo hash (HashMap, HashSet etc) primero invoca al método hashcode y luego el equals
* java.lang.Object#equals(java.lang.Object) Metodo que permite comparar objetos

**Enlaces de interés**

* Documento “Material-JPA.docx”