

**Objetivos:**

- Comprender la diferencia entre el concepto de clase e instancia.
- Herencia de Clases, Introducción a Colecciones, DOO

**Ejercicio 1.**

**Introducción Clases:** *Una clase es una descripción, un molde, de un conjunto de objetos que manifiestan la misma semántica, operaciones, atributos, y relaciones. Comparten comportamiento y estructura común.*

**Introducción Instancias:** *Es un ejemplo concreto de un objeto que respeta el molde de la clase: semántica, operaciones, atributos y relaciones.*

a) En la siguiente lista identifica Clases y sus instancias (cada ítem en la lista se refiere a una Clase o a una instancia de alguna de las clases en la lista), por ejemplo: Materia es una Clase, metodologías de programación es una instancia de esta Clase. Docente es una Clase, Leandro Antonelli es una instancia de esta Clase.

Carrera universitaria, UNNOBA, abogacía, IPI, metodologías de programación, IOO, facultad, decano, Sebastian Sotille, materia, paradigmas de programación, Docente, universidad, facultad de Derecho, Luciana Balbi, lenguaje de programación, Pascal, Smalltalk, IPOO, Redes, Guillermo Tamarit, Carlos Di Cicco, facultad de Ciencias Exactas, UNLP, licenciatura en informática, UBA, Federico Naso, C++, Lisp, Python, BBDD.

b) En la siguiente lista identifica Clases y subClases. Arme la jerarquía correspondiente. Como, por ejemplo, un “Círculo es una Elipse”.

Figura, Figura Abierta, Segmento, Punto, Polígono, Triángulo, Rectángulo, Cuadrado, Círculo Elipse, Hexágono, Figura Cerrada, Rombo, Pentágono

**Ejercicio 2.**

Una afamada compañía que se dedica a innovar en aplicaciones de World Wide Web está interesada en modelar álbumes de fotos de forma electrónica.

Interesados por los resultados que esperan obtener, nos piden que modelemos e implementemos álbumes de fotos. Las fotos no indican un lugar determinado en el álbum. El álbum de fotos puede usarse de la siguiente manera:

- Agregando las fotos de a una.
- Agregando una colección de fotos al álbum.


- Eliminando una foto de un álbum.
- Eliminando todas las fotos de un álbum.

Los álbumes no tienen límite. Es decir, son infinitos.

- a) Implemente en JAVA-LIKE las clases y métodos que crea necesarios para poder mandar a los álbumes los mensajes para agregar *una foto*, *varias fotos*, *ver cuántos elementos hay*, *limpiar álbum*, *asignar un nombre al álbum*, etc.

### Ejercicio 3.

Un convertidor de temperaturas es un objeto el cual se puede instanciar con una temperatura inicial y luego se le puede pedir que transforme la temperatura a las escalas según se desee (Celsius, Fahrenheit o Kelvin). En la Tabla 1 (Información extraída de Wikipedia <http://en.wikipedia.org/wiki/Fahrenheit>) podemos ver las formulas que se utilizan para pasar de una escala a otra.

Fahrenheit temperature conversion formulas		
Conversion from	to	Formula
Fahrenheit	Celsius	$^{\circ}\text{C} = (^{\circ}\text{F} - 32) / 1.8$
Celsius	Fahrenheit	$^{\circ}\text{F} = ^{\circ}\text{C} \times 1.8 + 32$
Fahrenheit	kelvin	$\text{K} = (^{\circ}\text{F} - 32) / 1.8 + 273.15$
kelvin	Fahrenheit	$^{\circ}\text{F} = (\text{K} - 273.15) \times 1.8 + 32$
Additional conversion formulas		
Conversion calculator for units of temperature 		

**Tabla 1 - Tabla de conversión.**

- a) Como nosotros estamos en la Argentina y por ende utilizamos la escala dada por Celsius se pide que implemente un objeto **Temperature** que pueda ser instanciado mediante el mensaje *new* y al no tener parámetro el valor inicial de la temperatura es 0. El objeto también debe responder a los mensajes *getCelsius()* y *setCelsius(float value)* los cuales en el primer caso retorna el valor de la temperatura en la escala Celsius y en el segundo caso se cambia el valor inicial del objeto por el nuevo valor que se envía en el parámetro.
- b) Extienda el objeto **Temperature** para que ahora sepa retornar el valor de la temperatura en la escala Kelvin y en la escala Fahrenheit. Mediante los mensajes *getKelvin()* y *getFahrenheit()*. Además mediante los mensajes *setKelvin(float value)*

y *setFahrenheit(float value)*: se pueda cambiar el valor interno de la temperatura. Tenga en cuenta que la temperatura pasada en el parámetro de los mensajes estará en su respectiva escala y que no se desea cambiar la representación interna del objeto. ¿Cómo modela esta situación?

- c) Suponga ahora que se desea cambiar la representación interna del objeto **Temperature** para representar la temperatura en la escala **Rankine**. Es decir, ahora el objeto **Temperature** guarda el valor en esta escala. ¿Que debería cambiar? ¿Como se utiliza ahora el objeto Temperature? ¿Qué pasa con el resto de los objetos que utilizan objetos de la clase Temperature?

**Nota:** Rankine [ $^{\circ}\text{Ra}$ ] = [ $^{\circ}\text{Celsius}$ ] \* 9/5 + 491.67 [ $^{\circ}\text{Celsius}$ ] = ([ $^{\circ}\text{Ra}$ ] – 491.67) \* 5/9

- d) Qué ocurre si la representación interna se da en la escala **Réaumur**? Que debería cambiar? Como se utiliza ahora el objeto **Temperature**? Qué pasa con el resto de los objetos que utilizan objetos de la clase **Temperature**?

**Nota:** Réaumur [ $^{\circ}\text{Re}$ ] = ([ $^{\circ}\text{Fahrenheit}$ ] – 32) / 2.25

- e) ¿De los puntos c) y d) que podemos concluir? Discuta con el ayudante.

#### **Ejercicio 4. (SUBIR a <https://plataformaed.unnoba.edu.ar>)**

Se desea modelar un **Editor de Figuras**. Un editor de figuras posee una colección de figuras, las cuales pueden ser: cuadrado, círculo, triángulo, líneas. Todas ellas tienen un color y los datos necesarios para calcular el área según qué figura es. Todas entienden los mensajes:

- void setColor(String color); // Setea el color de la figura
- String getColor(); // Retorna el color de la figura
- double getArea(); // Retorna el área de la figura
- void pintar(); // Imprime en consola “Pintando FIGURA COLOR” por ejemplo. “Pintando: círculo rojo”

El editor entiende los siguientes mensajes:

void pintar(); // Que pinta todas las figuras de la colección.  
double calcularArea(); //que retorna la suma de las áreas de todas las figuras.

Además, entiende los mensajes para agregar y borrar figuras de su colección.

- a) Realice el diagrama en UML
- b) Implemente en JAVA-LIKE

**Ejercicio 5.**

Sea una **Librería** que tiene una colección de **Libros**. De cada libro se conoce su título, autor, fecha de publicación, cada personaje y cantidad de copias en stock.

- a) Realice el diagrama de clases en UML
- b) Implemente al menos los siguientes mensajes
  - a. agregarLibro(Libro unLibro) //Agrega un libro a la colección
  - b. eliminarLibro(Libro unLibro) //Elimina un libro de la colección
  - c. cantidadDeLibros() //Retorna la cantidad de libros
  - d. cantidadTotalDeCopias() //Retorna la cantidad total de copias

**Ejercicio 6.**

Se desea modelar un **sistema de información sobre películas y series**. Las mismas cuentan con actores y directores. Además de personajes y en el caso de las series se tienen además los capítulos.

- a) Realice el diagrama de clases en UML
- b) Especifique todos los atributos de cada clase.

**Ejercicio 7.**

Se desea implementar **Conjuntos** en JAVA-LIKE. Los conjuntos son colecciones que pueden tener elementos repetidos.

Extracto de Wikipedia: [http://es.wikipedia.org/wiki/Teoría\\_de\\_conjuntos](http://es.wikipedia.org/wiki/Teoría_de_conjuntos)

El concepto de conjunto es intuitivo y se podría definir como una "colección de objetos"; así, se puede hablar de un conjunto de personas, ciudades, gafas, lapiceros o del conjunto de objetos que hay en un momento dado encima de una mesa. Un conjunto está bien definido si se sabe si un determinado elemento pertenece o no al conjunto.

Sean  $A$  y  $B$  dos conjuntos:

**Unión:** Los elementos que pertenecen a  $A$  o a  $B$  o a ambos  $A$  y  $B$ , forman otro conjunto, llamado **Unión** de  $A$  y  $B$ , escrito  $A \cup B$ . Así pues, se tiene

**Intersección:** Los elementos comunes entre  $A$  y  $B$  forman un conjunto denominado **Intersección** de  $A$  y  $B$ , representado por  $A \cap B$ :

$$A \cap B = \{x : x \in A \wedge x \in B\}$$

**Diferencia:** Los elementos de un conjunto que no se encuentran en otro conjunto  $B$ , forman otro conjunto llamado *diferencia de* y  $B$ , representado por,  $A - B$ :

**Diferencia simétrica:** Se define la *diferencia simétrica* de dos conjuntos por:

$$A \Delta B = (A - B) \cup (B - A)$$

**NOTA:** Para facilitar el trabajo vamos a trabajar con conjuntos de Personas. Esto significa que nuestro conjunto solo podrá contener personas.

Se pide que:

- a) Realice el diagrama en UML
- b) Implemente en JAVA-LIKE la Clase Conjuntos
- c) Los Conjuntos deben ser capaces de entender los siguientes mensajes:
  - a. add(Persona p)
  - b. remove(Persona p)
  - c. unión(Set aSet)
  - d. intersección(Set aSet)
  - e. diferencia(Set aSet)
  - f. diferenciaSimetrica(Set aSet)

- d) Supongamos que tenemos los siguientes conjuntos:

$$; B = \{4, 6, 8, 10\} ; C = \{10, 14, 16, 26\}$$

Muestre ejemplo de cómo sería:

- a. La unión de A y B
- b. La unión de A y C.
- c. La intersección de A y B.
- d. La intersección de A y C.
- e. La diferencia de A y B.
- f. La diferencia simétrica de A y B.

**Ejercicio 8. (SUBIR a <https://plataformaed.unnoba.edu.ar>)**

Especificar e implementar las clases **Persona**, **Empleado**, **Administrativo** y **Técnico**. Los datos relevantes de una persona son: nombre, DNI, dirección, año de nacimiento y sexo. Un empleado es una persona que desempeña una función en alguna empresa y recibe un salario por ello.

Un técnico es un empleado que posee un título y trabaja por contrato. Es decir, cobra siempre el mismo monto que establece el contrato.

Los empleados administrativos reciben su salario de acuerdo a la categoría (valores numérico 1, 2, 3 y 4) y antigüedad que posean (años de antigüedad \* categoría).

La clase **Empresa** tiene una variable de instancia llamada `empleados` que es una colección que contiene todos los empleados de la empresa.

Implementar el método *montoTotal* en la clase **Empresa**, que retorna el monto total que la empresa debe pagar, en concepto de sueldos, a sus empleados.

**Ejercicio 9. (SUBIR a <https://plataformaed.unnoba.edu.ar>)**

Se desea modelar un Navegador de Internet o más comúnmente llamado **Browser**, similar al Chrome, IE Edge, Firefox, etc. El Browser es sencillo, solamente puede navegar por una **Página** web por vez. Esa sería la página actual o **Current Page**, la cual conoce.

Al **Browser**, se le puede indicar cuál es la *current page*.

La página cuenta con una dirección web, una fecha de visita, el tamaño (peso de la misma) y el código HTML (solamente texto).

El browser, posee un **Historial** de páginas visitadas, la cual guarda por si el usuario desea volver a visitar. El historial tiene la funcionalidad de:

- Agregar páginas.
- Eliminar páginas.
- Conocer la última página visitada.
- Retornar la cantidad de páginas visitadas

Además del historial, también se cuenta con los **Favoritos** que son una colección de Direcciones Webs que el usuario guarda como direcciones preferidas.

- a) Diseñe en UML una solución al enunciado, Identifique las clases involucradas, atributos, responsabilidades y relaciones.

**Ejercicio 9.**

Se desea modelar un sistema de gestión de documentos online, similar a Google docs. El sistema está compuesto por muchos documentos.

De momento solo maneja un solo tipo de documentos.

Del sistema se conoce el dueño (una persona), el idioma y la fecha de creación del mismo.

De cada documento se conoce el nombre, el texto, la fecha de creación, el dueño y una colección de editores (estos son personas que pueden editar el documento). Los documentos se pueden compartir entre varios editores.

El sistema de gestión de documentos tiene la siguiente funcionalidad:

- Cantidad de documentos -> retorna la cantidad total de documentos
  - Cantidad de editores(Documento unDocumento) -> retorna la cantidad total de editores del documento.
  - Cantidad total de editores - > retorna la cantidad total de editores de todos los documentos.
  - agregarDocumento(Documento unDocumento) -> Agrega un documento a la colección de documentos.
  - agregarEditor(Documento unDocumento, Editor unEditor) -> Agrega un editor al documento.
- a) Diseñe en UML una solución al enunciado, Identifique las clases involucradas, atributos, responsabilidades y relaciones.



**Ejercicio 10.**

En base a lo que encuentre en la bibliografía y en Internet responda:

- a) ¿Qué es JAVA?
- b) ¿Qué es la JRE y la JDK?
- c) ¿Es java 100% orientado a Objetos?
- d) ¿Qué es el Garbage Collector?
- e) ¿Para qué fue inicialmente diseñado JAVA?