Objetivo: Que el alumno logre:
- Interpretar diagramas de clases representados en UML
- Aprender a definir clases, con sus constructores y métodos
- Determinar la visibilidad de los elementos (pública, privada, etc.)
- Definir métodos de acceso a las variables miembro (get/set)
- Aprender a instanciar objetos, y que éstos interactúen, mediante envío de mensajes
- Adquirir buenas prácticas de desarrollo de software, como lo es la documentación del mismo (en los ejercicios entregados en el laboratorio se exigirá el uso de documentación utilizando javadoc)
- Ejercitar idioma inglés. Con el propósito de estimular la adquisición de competencias transversales en el proceso de formación profesional, según lo expresado en los objetivos del programa de la asignatura, se incluye **enunciado en inglés.**

Conceptos teóricos: Abstracción, encapsulamiento, ocultamiento, visibilidad, objeto, clase, doble encapsulamiento, objeto, clase, mensaje, método, protocolo, firma, comportamiento

Consigna: En cada ejercicio deberá
a) Crear el código JAVA correspondiente a los diagramas de Clase.
b) Aunque no se indique, debe implementar los Observadores y Mutadores (Accessors: getters y setters)
c) Importante: donde el diagrama de Clase no indique tipo de valor de retorno, debe ir tipo **VOID**.
d) **Agregar documentación utilizando la herramienta provista por java (javadoc)**
e) Crear una clase ejecutable, en la que se instancien varios objetos de las clases creadas y se utilicen los distintos métodos para verificar su correcto funcionamiento.
f) En los ejercicios. 1 y 2 los datos se ingresarán como constantes. Ejercicios 3 y 4 ingresar los datos como argumentos del método main(). Ejercicios restantes: ingresar los datos por teclado, utilizando la clase Scanner.

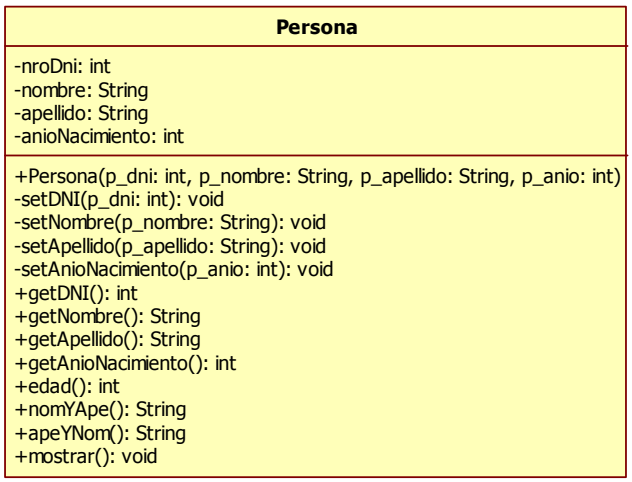
Ejercicios:

1. El diagrama de clase adjunto representa una abstracción del concepto "persona".
   Implemente en java la clase **Persona**. El método ***edad()*** devuelve la cantidad de años cumplidos a la fecha, considerando para el cálculo sólo la diferencia entre años. La salida impresa del método ***mostrar()*** debe ser la siguiente (los valores en negrita dependen del estado interno del objeto):

Nombre y Apellido: **Juan Perez**
DNI: **35123456** Edad: **22** años
Nota: Para trabajar con fechas, agregar:
import java.util.*;

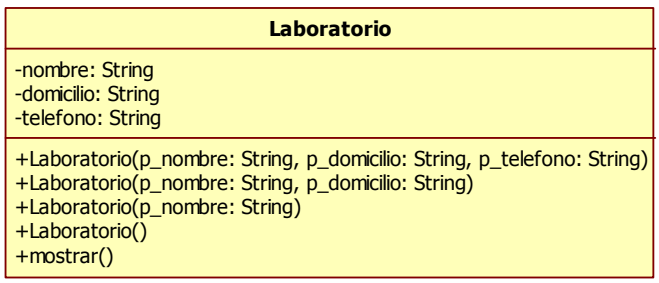
Para saber el año actual usar:
Calendar fechaHoy = new GregorianCalendar();
int anioHoy = fechaHoy.get(Calendar.YEAR);

Para más información sobre la clase Calendar buscar en la documentación de JAVA.
(http://docs.oracle.com/javase/6/docs/api/)



2. Una droguería que comercia con laboratorios que producen medicamentos modeló la clase **Laboratorio** representada en el diagrama de clases adjunto. Implemente en java la clase Laboratorio.
   La salida impresa del método ***mostrar()*** debe ser la siguiente (los valores en negrita dependen del estado interno del objeto):

Laboratorio: **Colgate S.A.**
Domicilio: **Scalabrini Ortiz 524** - Teléfono: **54-11 -4239-8447**



Nota: Observe que la clase tiene cuatro formas de instanciarse. (Constructores sobrecargados)

3. En la clase **Cliente,** el método ***agregaSaldo(p_importe)*** **agrega** *p_importe* al saldo actual (lo actualiza) y devuelve el nuevo saldo. El método ***nuevoSaldo(****p_importe****)*** **reemplaza** el saldo actual por *p_importe* y devuelve el nuevo saldo. La salida impresa del método ***mostrar()*** debe ser la siguiente: (los valores en negrita dependen del estado interno del objeto)

- Cliente -
Nombre y Apellido: **Juan Perez (24444333)**
Saldo: **$200.00**

Cliente
-nroDNI: int -apellido: String -nombre: String -saldo: double
+Cliente(p_dni: int, p_apellido: String, p_nombre: String, p_importe: double) +mostrar() +nuevoSaldo(p_importe: double): double +agregaSaldo(p_importe: double): double +apeYnom(): String +nomYape(): String

4. It is wanted to get the average of grade points of Object Oriented Programming students. For this, the abstract concept "pupil" is modeled. When an instance of Alumno is created, as grade points are not known yet, they are initialized in zero. The method ***promedio()*** returns the average of student's grade points. The method ***aprueba()*** returns true or false according to whether the student's grade points average is greater than 7.0 and if both grade points are equal or greater than 6.0. The method ***leyendaAprueba()*** returns a string APROBADO or DESAPROBADO according to the method ***aprueba()*** returns. The printed output of the method ***mostrar()*** must be: (bolded values show internal state of object)

Nombre y Apellido: **Juan Perez**
LU: **2020** Notas: **5.99 – 10.0**
Promedio: **7.995 - DESAPROBADO**

Nombre y Apellido: **María Gomez**
LU: **2051** Notas: **7.85 - 8.5**
Promedio**: 8.20 – APROBADO**

Alumno
-lu: int -nombre: String -apellido: String -nota1: double -nota2: double
+Alumno(p_lu: int, p_nombre: String, p_apellido: String) +setNota1(double p_nota): void +setNota2(double p_nota): void -aprueba(): boolean -leyendaAprueba(): String +promedio(): double +nomYApe(): String +apeYNom(): String +mostrar()

5. Una empresa desea administrar automáticamente la liquidación de sueldo de sus empleados. Para ello se modela la abstracción del concepto "empleado", teniendo en cuenta las características relevantes al problema, brindando el comportamiento adecuado.
Implemente en java la clase **Empleado** según lo indicado en el diagrama de clases.
El método ***antiguedad()*** devuelve la cantidad de años desde el ingreso a la empresa. El método ***descuento()*** corresponde al 2% del sueldo básico en concepto de obra social, mas $12 de seguro de vida. El método ***adicional()*** es una asignación que se realiza sobre el sueldo básico, en base a la antigüedad, según la siguiente tabla:

Empleado
-cuil: long -apellido: String -nombre: String -sueldoBasico: double -anioIngreso: int
+Empleado(p_cuil: long, p_apellido: String, p_nombre: String, p_importe: double, p_anio: int) +antiguedad(): int -descuento(): double -adicional(): double +sueldoNeto(): double +nomYApe(): String +apeYNom(): String +mostrar(): void +mostrarLinea(): String

Antigüedad	**Asignación con respecto al Sueldo Básico**
< 2	2%
>= 2 y <10	4%
>= 10	6%

El sueldo neto se calcula como la suma del sueldo básico más el adicional, menos el descuento.
La salida impresa del método ***mostrara()*** debe ser la siguiente (los valores en negrita dependen del estado interno del objeto):

Nombre y Apellido: **Juan Perez**
CUIL: **2035123456** Antigüedad: **22** años de servicio
Sueldo Neto: $ **3000.00**

El método ***mostrarLinea()*** retorna una cadena como la siguiente (los valores dependen del estado interno del objeto):

2035123456 Perez, Juan …… $ 3000.00

ATENCION!! Para el siguiente ejercicio repasar el concepto "Seudo-Variable" impartido en la teoría Tema 4.

6. Con el propósito de trabajar con figuras geométricas, se modela como colaborador, el concepto "punto", que participará más adelante en la construcción de figuras (por ejemplo, el centro de un círculo). Para ello se abstraen las características básicas de un punto, y se lo dota de comportamiento. Un punto tiene dos propiedades relevantes: la abscisa **X** y la ordenada **Y** en el plano. Para este objeto definimos dos constructores: uno **sin parámetros** que sitúa el punto en el origen (X=0; Y=0), y otro constructor explícito que recibe las coordenadas *X* e Y del punto concreto.
Implemente en java la clase **Punto** representada en el diagrama de clase.

El método ***desplazar(dx, dy)*** cambia la posición del punto desde (x, y) a (x+dx, y+dy).

El método ***distanciaA()*** debe calcular la distancia **desde él mismo** (el objeto que recibe el mensaje) hasta otro objeto Punto que es recibido como parámetro. Para esto usar Pitágoras: si p1(x1, y1) y p2(x2, y2), entonces la distancia entre p1 y p2 será: d= $\sqrt{(x2-x1)^2+(y2-y1)^2}$.

El método ***coordenadas()*** retorna una cadena con el formato: (**7.5**, **0.5)**

La salida impresa del método ***mostrar()*** debe ser:
Punto. X: **7.5**, Y: **0.5**
(los valores en negrita dependen del estado interno del objeto)