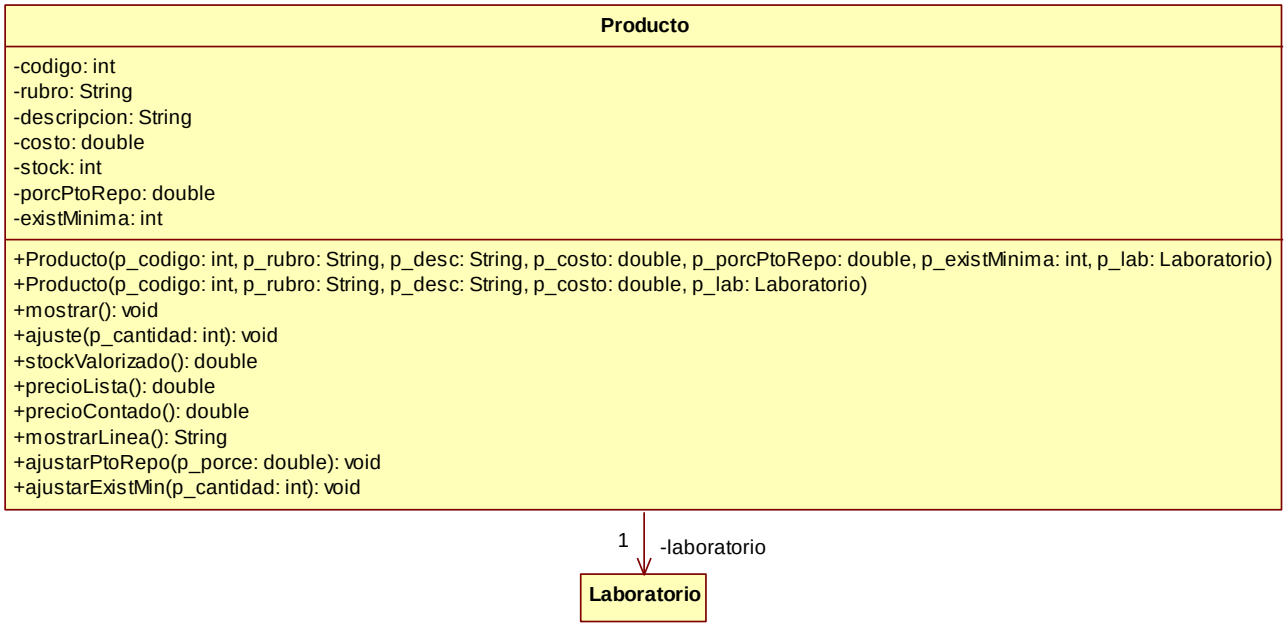
Objetivos:
- Interpretar diagramas de clases representados en UML
- Aprender a declarar clases con colaboradores
- Comprender el concepto de "conocimiento" entre objetos
- Aprender a instanciar objetos complejos (contienen a otros objetos)
- Comprender la utilidad del encapsulamiento de estructura y métodos: si se modifica la estructura de una clase, la interfaz pública no varía

Conceptos teóricos: Conocimiento entre objetos, colaboradores, ventajas del ocultamiento de información

Consignas: En cada ejercicio deberá
a) Crear el código JAVA correspondiente a los diagramas de Clase.
b) Aunque no se indique explícitamente, debe implementar los constructores y accessors
c) Agregar documentación utilizando la herramienta provista por java (javadoc)
d) Crear una clase ejecutable, en la que se instancien varios objetos de las clases creadas y se utilicen los distintos métodos para verificar su funcionamiento.
e) Ejercitar el ingreso de datos como constantes, como argumentos del método main(), y utilizando la clase Scanner, variando en los distintos ejercicios.

Nota: Para la resolución de los ejercicios se requiere la comprensión del concepto de **conocimiento entre objetos**. Repasar "Formas de conocimiento" dado en la teoría Tema 4.

Ejercicios:
1. Los productos que comercializa una droguería fueron modelados según se representa en el diagrama de clase.



Implemente en java la clase **Producto** (reutilice la clase Laboratorio definida previamente). Al dar de alta un producto, que se compra por primera vez, el stock deberá iniciarse en 0. Éste se incrementará en futuras cargas de compras realizadas a los laboratorios. El método ***ajuste(p_cantidad)*** permite modificar el stock (agregando o quitando), por ejemplo en un inventario, o en el caso de destrucción de productos vencidos, etc. El método ***precioLista()*** devuelve el valor que resulta de agregar un 12% al precio de costo. El método ***precioContado()*** devuelve el valor que representa el precio por pago al contado del producto, que se calcula restando un 5% al precio de lista. El método ***stockValorizado()*** debe devolver el resultado de multiplicar el stock por el precio de costo, más una rentabilidad de 12%.
Por diversos factores (demanda estacional, demanda de mercado, etc.), es política de la empresa que el porcentaje de punto de reposición y la existencia mínima de cada producto sea variable. Estos pueden ser modificados mediante los métodos ***ajustarPtoRepo()*** y ***ajustarExistMin()***, que permiten asignar un nuevo valor a dichos atributos.
La salida impresa del método ***mostrar()*** es la siguiente (los valores en negrita dependen del estado interno del objeto):

Laboratorio: **Colgate S.A.**
Domicilio**: Scalabrini Ortiz 5524** - Teléfono**: 54-11 -4239-8447**

Rubro: **Perfumería**
Descripción: **Jabón Deluxe**
Precio Costo: **5.25**
Stock: **500** - Stock Valorizado: $**2940**

El método ***mostrarLinea()*** devuelve un String formado por la concatenación de la descripción del producto, el precio de lista y el precio de contado, según el siguiente formato:

Jabón Deluxe 11.20 10.64

Implemente una clase ejecutable **GestionStock** en la que se instancie 1 producto, se le asigne un stock de 500, que surgieron de una promoción de un laboratorio, y luego se muestre por pantalla sus datos, incluido el stock valorizado. Luego simule una baja de 200 productos, que se destruyeron por estar mal estibados, y muestre nuevamente el estado actual. Por último, ante la consulta de un cliente, muestre el precio de lista y el precio al contado.

Punto
-x: double -y: double
+Punto() +Punto(p_x: double, p_y: double) -setX(p_x: double) -setY(p_y: double) +getX(): double +getY(): double +distanciaA(p_ptoDistante: Punto): double +desplazar(p_dx: double, p_dy: double): void +mostrar() +coordenadas(): String

2. Se desea agregar comportamiento a la clase Punto desarrollada previamente, que será necesario al reutilizar dicha clase en la creación de figuras.
   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 ello usar Pitágoras: si p1(x1, y1) y p2(x2, y2), entonces la distancia entre p1 y p2 será: d= $\sqrt{(x2-x1)^2+(y2-y1)^2}$.

3. Se desea trabajar con un generador de figuras geométricas, y se inicia con el modelado de círculos, considerando para ello las características que lo definen, tales como el *centro* (objeto de clase *Punto*) y el *radio*. El constructor sin parámetros crea un círculo con radio 0, cuyo origen está situado en el punto (0, 0). El constructor explícito crea un círculo con centro y radio que se pasarán al constructor como parámetros. Se desea poder desplazar el círculo, sin cambiar sus dimensiones, trasladando su centro a otra posición. Para ello se deberá desplazar el objeto colaborador *Punto* que representa al *centro.* Además, se desea calcular el área y el perímetro e implementar un método que permita mostrar todas las características del círculo, con el siguiente formato:

Circulo
-radio: double -centro: Punto
+Circulo(double p_radio, Punto p_centro) +Circulo() -setCentro(Punto p_centro): void -setRadio(double p_radio): void +getCentro(): Punto +getRadio(): double +desplazar(double p_dx, double p_dy): void +caracteristicas(): void +perimetro(): double +superficie(): double +distanciaA(Circulo otroCirculo): double +elMayor(Circulo otroCirculo): Circulo

   ****** Circulo ******
   Centro: **(2.0, 3.5)** - Radio: **20.0**
   Superficie: **1256.64** - Perímetro: **125.66**

   También se desea implementar un método que permita calcular la distancia entre dos círculos, que estará representada por la distancia entre sus centros, y otro método que indique, entre dos círculos, cual es el mayor, en función del tamaño de su superficie.

   Por último, implementar una clase ejecutable **CreaFigura** donde deberá:
   - crear un círculo centrado en el punto (0, 0) y cuyo radio surja de un número aleatorio
   - desplazar el círculo a otro punto situado 240 unidades hacia la izquierda y 230 hacia abajo
   - mostrar las características del círculo creado
   - crear otro círculo centrado en el punto (5.2, 0.5) y cuyo radio sea un nuevo número aleatorio.
   - mostrar las características del mayor de los dos círculos
   - mostrar la distancia entre los dos círculo

   Nota: para obtener un numero aleatorio (importar el paquete java.util.Random)

```
Random unNumero = new Random();

double radio = unNumero.nextDouble() * 100.0;
```

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

Rectangulo
-origen: Punto -ancho: double -alto: double
+Rectangulo(Punto p_origen, double p_ancho, double p_alto) +Rectangulo(double p_ancho, double p_alto) -setAncho(double p_ancho): void -setAlto(double p_alto): void +getAncho(): double +getAlto(): double +desplazar(double p_dx, double p_dy): void +caracteristicas(): void +perimetro(): double +superficie(): double +distanciaA(Rectangulo otroRectangulo): double +elMayor(Rectangulo otroRectangulo): Rectangulo

4. El generador de figuras geométricas también debe permitir definir rectángulos. La clase **Rectangulo** tiene como datos miembro: el *origen* (esquina inferior izquierda) que es un objeto de la clase **Punto** y las dimensiones *ancho* y *alto*. El constructor que recibe sólo las dimensiones (ancho-alto) crea un rectángulo cuyo origen está situado en el punto (0, 0). El otro constructor crea un rectángulo situado en un

determinado punto ***p*** y con las dimensiones que recibe como parámetros. Se desea poder desplazar el rectángulo, trasladando su origen a otra posición, sin cambiar sus dimensiones. Para ello se deberá desplazar el objeto colaborador *Punto* que representa al origen. Además, se desea calcular el área y el perímetro e implementar un método que permita mostrar todas las características del rectángulo, con el siguiente formato:

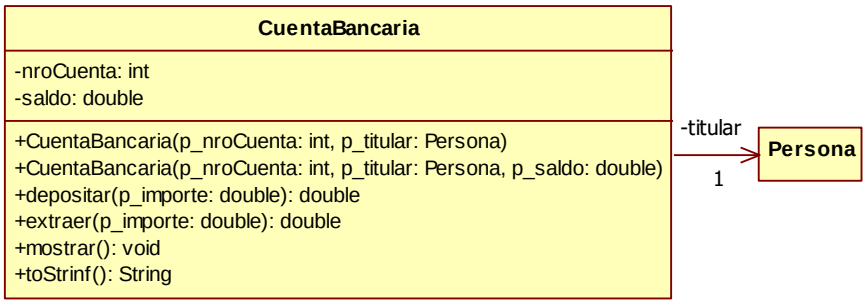
****** *Rectangulo* ******
Origen: **(0.0, 0.0) -** Alto: **200.0 -** Ancho: **100.0**
Superficie: **20000.0 -** Perímetro: **600.0**

Implementar además un método que permita calcular la distancia entre dos rectángulos, que estará representada por la distancia entre sus orígenes, y otro método que indique, entre dos rectángulos, cual es el mayor, en función del tamaño de su superficie.

En la clase *CreaFigura* creada previamente agregar:

- crear un rectángulo situado en el punto (0, 0) y cuyas dimensiones sean generadas a partir de números aleatorios
- desplazar el rectángulo a otro punto situado 40 unidades hacia la derecha y 20 hacia abajo
- mostrar las características del rectángulo creado
- crear otro rectángulo con origen en el punto (7.4, 4.5) y cuyas dimensiones sean nuevos números aleatorios.
- mostrar las características del mayor de los dos rectángulos
- mostrar la distancia entre los dos rectángulos

5. A bank needs to model the concept of a bank account, which has to allow deposits and withdrawals. The method ***depositar(p_importe)*** increases the current balance with the amount passed as a parameter. The method ***extraer(p_importe)*** decreases the current balance. Both methods return the resulting balance after the operation. The model is represented in the following class diagram:



The exit screen of the method ***mostrar()*** must be as follows (the values in bold depend on the internal state of the object):

- Cuenta Bancaria -
Titular**: Juan Perez** (**34** años)
Saldo**: 155.25**

The method toString(), returns a string, composed of the concatenation of the data (número, titular, saldo), tabulated. For example:

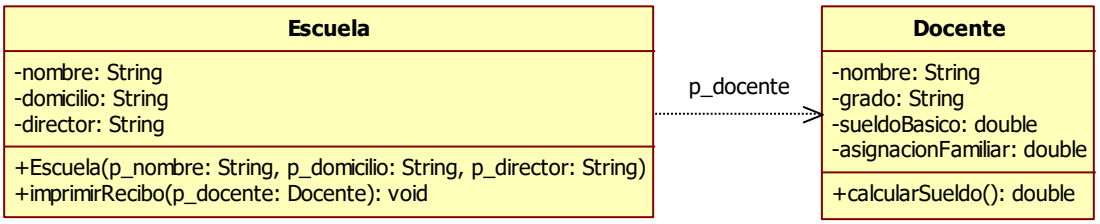
1249 Juan Perez 1500.0

6. Una escuela necesita emitir el recibo de un docente, cada vez que éste lo solicite. El sueldo del docente se calcula mediante la suma de sueldo básico y la asignación familiar. El recibo que emite la escuela tiene el siguiente formato:

Escuela**: Manuel Belgrano** Domicilio**: Irigoyen 1580** Director**: Leopoldo Juez**

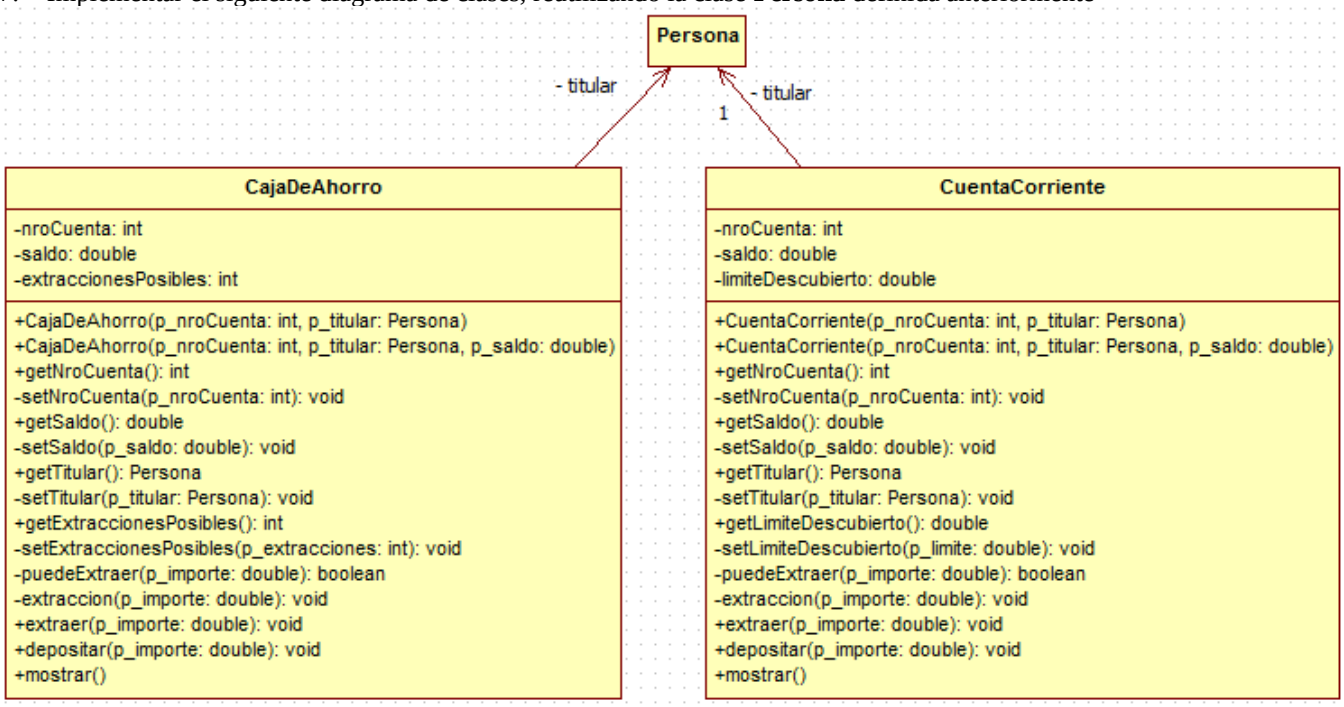
Docente: **Elisa Sánchez**.
Sueldo: ……. $ **2.890**.
Sueldo Básico……. $ **1.600**
Asignación familiar…… $ **1.290**

El diagrama de clases que modela estos requerimientos es el siguiente:



Implemente una clase ejecutable *Secretaria* donde instancie una escuela, un docente, y emita un recibo.

7. Implementar el siguiente diagrama de clases, reutilizando la clase **Persona** definida anteriormente



Para la implementación debe tener en cuenta lo siguiente:

7.1. Clase **CuentaCorriente**: Si la cuenta se instancia sin saldo, este tomará el valor 0. En todos los casos, el límite de descubierto por defecto es $500. El método ***depositar(p_importe)*** sólo agrega el importe al saldo actual. El método ***extraer(p_importe)*** coordina la operación, de acuerdo a si se cumplen las condiciones de extracción, caso contrario informará el motivo por el cual no se pudo extraer. El método ***puedeExtraer(p_importe)*** devuelve *true* si el importe a retirar no supera el saldo más el límite de descubierto autorizado. El método ***extraccion(p_importe)*** es el que realiza efectivamente la extracción. La salida impresa del método ***mostrar()*** debe ser la siguiente: (los valores en negrita dependen del estado interno del objeto)

- Cuenta Corriente –

Nro. Cuenta: **1735** - Saldo: **1500.0**
Titular**: Juan Perez**
Descubierto**: 500.0**

Si no puede extraer el mensaje debe ser: **El importe de extraccion sobrepasa el límite de descubierto!**

7.2. Clase **CajaDeAhorro**: Si la cuenta se instancia sin saldo, este tomará el valor 0. En todos los casos, la cantidad máxima de extracciones es 10. El método ***depositar(p_importe)*** sólo agrega el importe al saldo actual. El método ***extraer(p_importe)*** coordina la operación, de acuerdo a si se cumplen las condiciones de extracción, si no pudiera informará el motivo por el cual no se pudo extraer. El método ***puedeExtraer(p_importe)*** devuelve *true* si el importe a retirar no supera el saldo disponible y si aún no usó todas las extracciones posibles. El método ***extraccion(p_importe)*** es el que realiza efectivamente la extracción y descuenta 1 al número de extracciones posibles. La salida impresa del método ***mostrar()*** debe ser la siguiente: (los valores en negrita dependen del estado interno del objeto)

7.3. Caja de Ahorro –

Nro. Cuenta: **2135** - Saldo: **155.25**
Titular**: Juán Pérez**
Extracciones posibles**: 10**

Si no puede extraer el mensaje debe ser:
No tiene habilitadas mas extracciones! (si el número de extracciones posibles es cero)
No puede extraer mas que el saldo! (si el importe es mayor al saldo)

Implementar una clase ejecutable **Banco**, donde instancie 1 caja de ahorro y 1 cuenta corriente, ambas para un mismo titular. Luego realice depósitos y extracciones, con los importes necesarios para verificar todas las opciones.

8. Un hospital público brinda como servicio hacia sus pacientes, autoridades judiciales, obras sociales, etc., enviarles los datos filiatorios, por correo postal, al domicilio declarado donde vive. Previo al envío, el operador verifica por pantalla los datos del paciente. Para realizar esta tarea, implemente en java el siguiente diagrama de clases, donde se observa un doble conocimiento de la clase **Paciente** con Localidad. El método ***mostrar()*** de Localidad retorna la siguiente cadena (los valores en negrita dependen del estado interno del objeto):

Localidad: **Monte Caseros** Provincia: **Corrientes**

El método ***mostrarDatosPantalla()***, muestra al operador la información requerida con el siguiente formato:

Paciente: **Juan Manuel Ortigoza** Historia Clínica: **57869** Domicilio: **Bulevar 3 de Abril**
Localidad: **Monte Caseros** Provincia: **Corrientes**

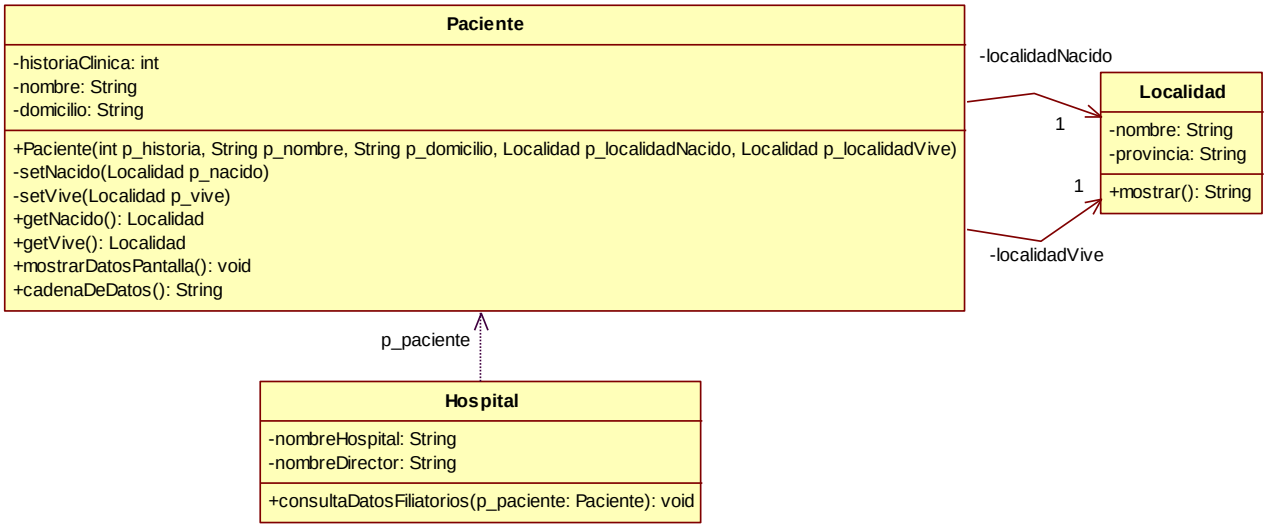
El método ***cadenaDeDatos()*** retorna un String con los mismos datos, pero con el siguiente formato

Juan Manuel Ortigoza …… 57869 ..… Bulevar 3 de Abril - Monte Caseros – Corrientes

El método ***consultaDatosFiliatorios()*** tiene una salida con el siguiente formato

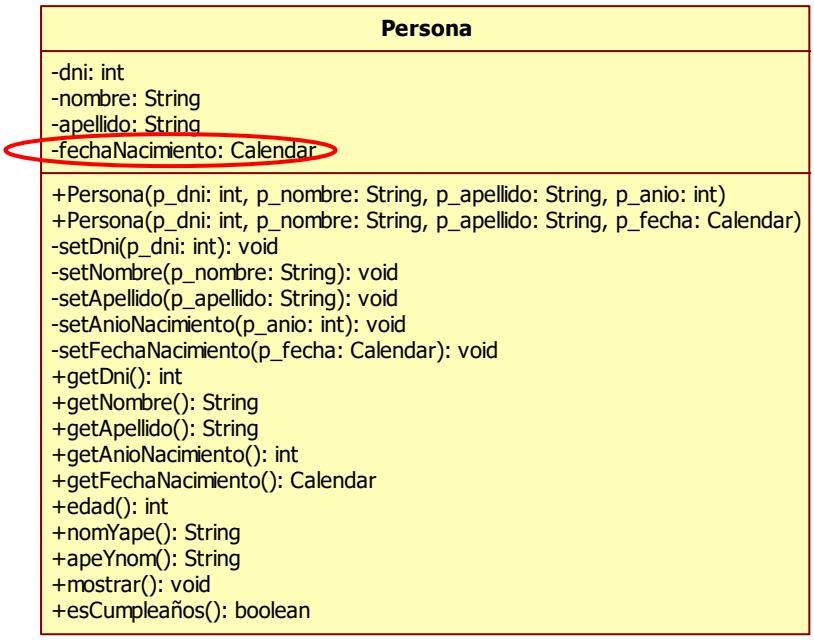
Hospital**: Garrahan** Director**: Leonardo Ruiz**

Paciente: **Juan Manuel Ortigoza** Historia Clínica: **578669** Domicilio: **Bulevar 3 de Abril**
Localidad: **Monte Caseros** Provincia: **Corrientes**



Implementar una clase ejecutable GestionHospital en la que se instancie un hospital, un paciente, con sus correspondientes localidades, y muestre por pantalla los datos filiatorios, utilizando los métodos apropiados.

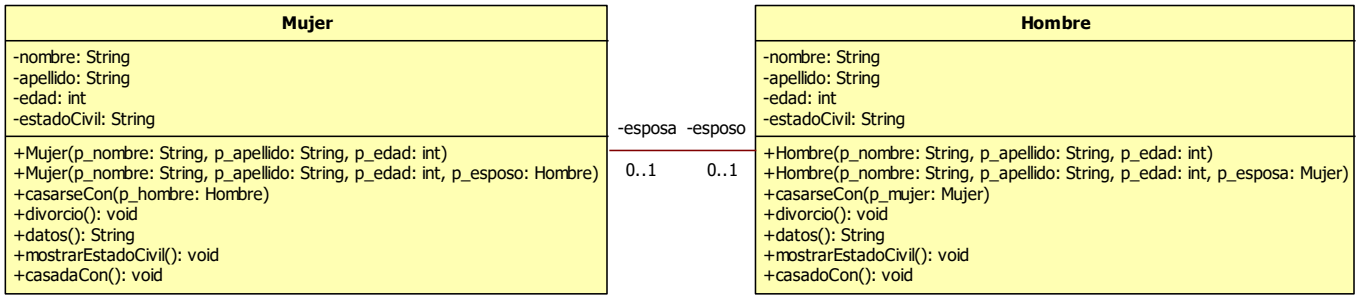
9. La entidad bancaria que administra las cuentas bancarias definidas previamente desea mandar una tarjeta de felicitaciones el día del cumpleaños de sus clientes (instancias de la clase Persona). Para ello necesita disponer de la fecha de nacimiento completa. Ante este nuevo requerimiento, es necesario modificar la estructura de los objetos persona: en lugar del atributo **año de nacimiento,** de tipo entero, debe tener el atributo **fecha de nacimiento,** de tipo Calendar (Atención: el atributo Calendar **REEMPLAZA** al atributo int). Sin embargo, como otros objetos ya están utilizando objetos persona, es imprescindible, mantener el protocolo, lo cual implica modificar los métodos correspondientes de modo que continúen entregando el mismo servicio.



Int anioNacimiento →
Calendar fechaNacimiento

El método ***esCumpleaños()*** comprueba y retorna true o false según sea o no el cumpleaños de la persona en el día que se ejecuta el programa.
En la clase Banco creada previamente, agregue el envío del mensaje de cumpleaños, si cumple la condición.

10. El Registro Nacional de las Personas desea automatizar la emisión de certificados de matrimonio y divorcio, para lo cual necesita administrar la información de hombres, mujeres y la relación entre ellos. El diagrama de clases que modela este requerimiento es el siguiente, donde se observa que ambas clases se conocen entre sí:



En ambas clases:

El atributo **estadoCivil**, cuando se crea el objeto, toma por valor **Soltero/a**, al casarse **Casado/a** y al divorciarse **Divorciado/a**. Este cambio debe ser realizado desde los métodos correspondientes.

El método ***divorcio()***, además**,** coloca *null* en el atributo esposo/esposa, mientras que el método ***casarseCon()*** asigna el cónyuge indicado.

El método ***datos()*** retorna: los datos del objeto (nombre, apellido y edad) ej:

María Gómez de 28 años (ejemplo Mujer)

El método ***mostrarEstadoCivil ()*** tiene la siguiente salida impresa:

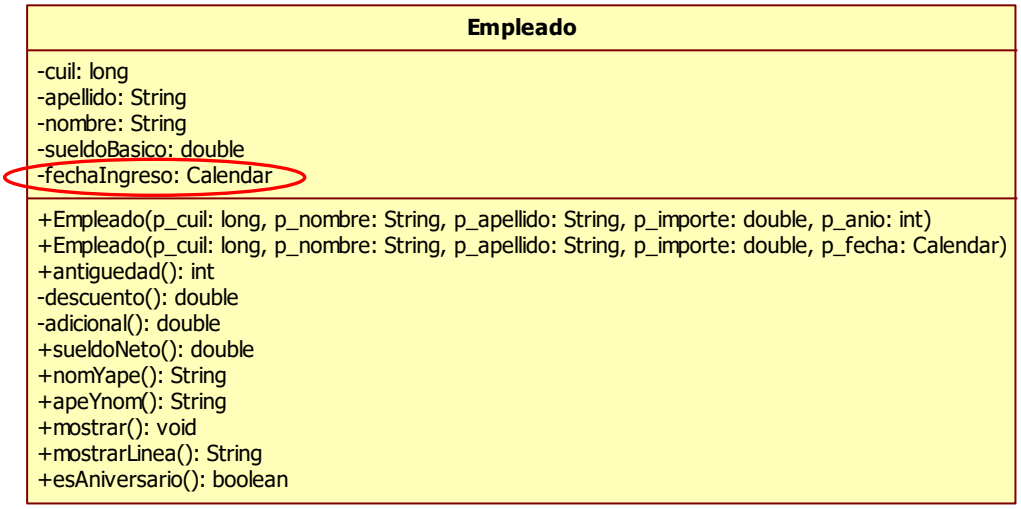
María Gómez de 28 años - Casada

El método ***casadaCon()/casadoCon(),*** tiene la siguiente salida impresa:

Pedro Leyes de 29 años está casado con María Gómez de 28 años (ejemplo Hombre)

Implemente en Java el diagrama de clases, y luego una clase ejecutable **RegistroCivil** que emita el certificado de matrimonio.

11. La empresa "Iberá Servicios" desea permitir a sus empleados (instancias de la clase Empleado) retirarse 1 hora más temprano cada vez que se cumpla 1 año más de su ingreso. Necesita por lo tanto almacenar la fecha de ingreso completa. En la abstracción inicial, la fecha de ingreso completa no era una variable relevante al problema original. Al presentarse un nuevo requerimiento, es necesario modificar la estructura de los objetos empleado: en lugar del atributo año de ingreso, de tipo entero, debe tener el atributo fecha de ingreso, de tipo Calendar. (Atención: el atributo Calendar **REEMPLAZA** al atributo int). No obstante, seguramente existen otros objetos que ya están utilizando objetos empleados, por lo tanto es imprescindible mantener el protocolo, lo cual implica modificar los métodos correspondientes de modo que continúen entregando el mismo servicio.



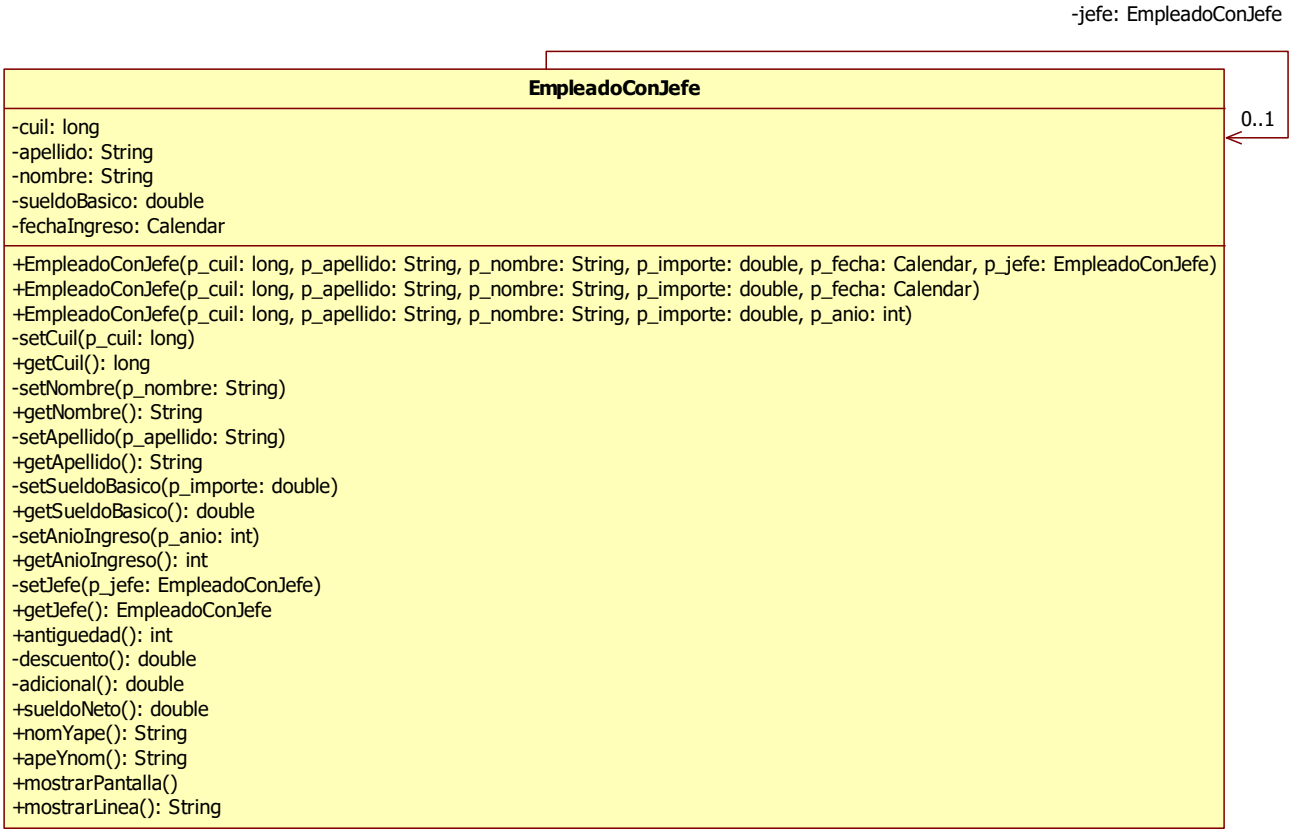
El método ***esAniversario()*** comprueba y retorna true o false según sea o no el día en que la persona cumple un año más en la empresa.

Realice los cambios necesarios en la clase **Empleado**, y luego implemente una clase ejecutable **Empresa**, donde se instancie un empleado y se imprima un permiso de salida, si corresponde.

12. A fin de mejorar la gestión de RRHH, la empresa "Iberá Servicios" decide implementar una estructura de jerarquía, asignando a cada empleado un jefe a quien debe responder, salvo el Gerente General que no tiene jefe. Al momento de visualizar los datos de los empleados (método mostrar()), se desea reflejar el nombre del jefe al que el empleado responde, si lo tiene, o la leyenda "GERENTE GENERAL", en caso contrario:

    Nombre y Apellido: **Juan Perez**
    CUIL: **20351234385** Antigüedad: **22** años de servicio
    Sueldo Neto: $ **3000.00**
    Responde a: **Bulgheri, Gregorio / GERENTE GENERAL**

    El siguiente diagrama de clase representa la relación entre el empleado y su jefe.



Finalmente, en la clase ejecutable, cuando corresponda, emitir un permiso de salida. Deberá estar firmado al pié por el jefe.