**Nombre Alumno/s**: Fernandez Alejandro Facundo **DNI:** 40047821

Gomez Angela Rebeca **DNI:** 40488634

**Nombre Profesor:** Aldo Mettini, Andrea Airaldi

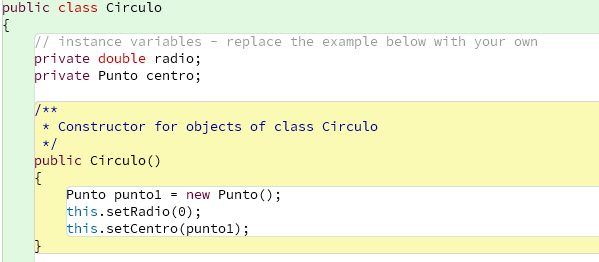
**Grupo Laboratorio:** 3 **TP:** 3 **Fecha de entrega:** 08/09/2025

Conceptos incorporados e implementados para la resolución de los ejercicios de la serie N°3:

Para este trabajo práctico se interpretaron diagramas de clases en UML y se implementaron clases con colaboradores y distintos tipos de relaciones entre objetos, aplicando los siguientes conceptos:

**Colaboradores:** son aquellos objetos que reciben mensajes, responden de forma activa y ejecutan métodos de su protocolo.

Ejemplo: la clase Circulo colabora con Punto, ya que necesita un centro para poder funcionar:

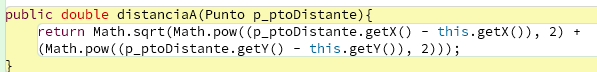


**Conocimiento entre objetos:** un objeto debe conocer a otro para poder enviarle un mensaje, en otras palabras, nombrarlo ya que existe una ligadura entre un objeto y su nombre.

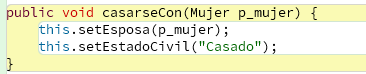
Esto se da a través de:

* Variables de instancia, quienes definen una relación entre un objeto y sus atributos y son permanentes mientras exista el objeto.
* Parámetros, que se emplean al nombrar objetos que el objeto receptor (colaborador) necesita para cumplir una petición y existen solo durante la ejecución de un método.
* Variables temporales, utilizadas dentro de un método y también existen solo durante la ejecución de un método, pero cuando finaliza la activación del método éstas dejan de existir.
* Seudo-variable *this*, que permite autoreferenciar al propio objeto. Todos los objetos tienen la referencia implícita *this.*

Ejemplo de uso de parámetro:



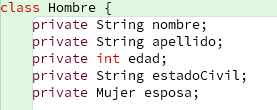
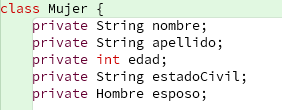
Ejemplo de this:



**Relaciones en UML**: las relaciones entre las clases indican cómo se comunican los objetos de esas clases entre sí. Existen cuatro tipos de relaciones, pero en esta guía solo empleamos dos: de Asociación y de Dependencia.

* Asociación: una clase contiene a otra (ej. Producto tiene un Laboratorio). La asociación puede ser bidireccional (se pueden recorrer en ambos sentidos y generalmente las navegaciones son en esta modalidad) o unidireccional (en ocasiones es necesario restringir la navegación en un solo sentido).
* Dependencia: una clase utiliza a otra para cumplir con un servicio (ej. Escuela depende de Docente para emitir un recibo).

Ejemplo de asociación bidireccional:

**Ventajas del encapsulamiento y ocultamiento de información:**

Permite modificar la estructura interna de una clase sin afectar el comportamiento esperado por otros objetos.

Ejemplo:

