

# Orientación a Objetos I

## 2025

Explicación de práctica #4  
correspondiente a los ejercicios de la  
semana del 22 de Septiembre




FACULTAD DE INFORMATICA



UNIVERSIDAD  
NACIONAL  
DE LA PLATA

# Actividades de la semana anterior

- Seguimos con los ejercicios:
  - **Ejercicio 7: Figuras y cuerpos** 

- Ejercicio 8: Genealogía salvaje

Programación Orientada a Objetos 1 - Facultad de Informática, UNLP

- Ejercicio 9: Red de Alumbrado

## Cuadernillo Semestral de Actividades

Actualizado: 8 de septiembre de 2025



FACULTAD DE INFORMATICA



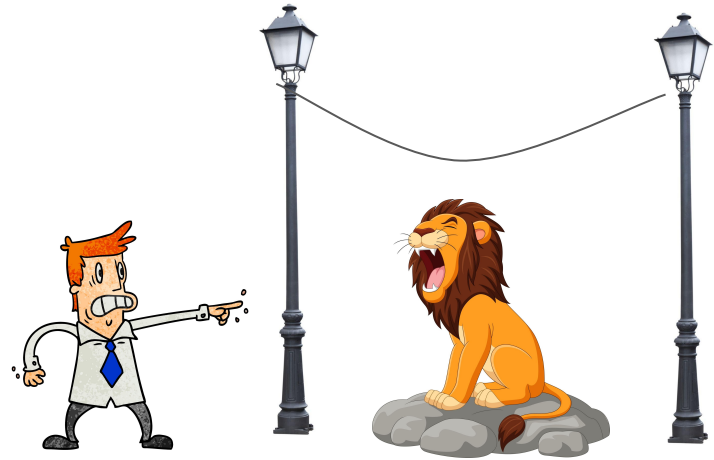
UNIVERSIDAD  
NACIONAL  
DE LA PLATA

# Actividades esperadas para esta semana

- Ejercicio 8: Genealogía Salvaje
- Ejercicio 9: Red de Alumbrado

## Esta semana:

- Ejercicio 10: Method lookup con Empleados
- Ejercicio 11: Cuenta con ganchos
- Ejercicio 12: Job scheduler



FACULTAD DE INFORMATICA



UNIVERSIDAD  
NACIONAL  
DE LA PLATA

# Ejercicio 8: Genealogía Salvaje

- **Ejercicio 8: Genealogía Salvaje**
- Ejercicio 9: Red de Alumbrado

## Ejercicio 8: Genealogía salvaje

En una reserva de vida salvaje (como la estación de cría ECAS, en el camino Centenario), los cuidadores quieren llevar registro detallado de los animales que cuidan y sus familias. Para ello nos han pedido ayuda. Debemos:

### Tareas:

#### a) Complete el diseño e implemente

Modelar una solución en objetos e implementar la clase Mamífero (como subclase de Object). El siguiente diagrama de clases (incompleto) nos da una idea de los mensajes que un mamífero entiende.

Proponga una solución para el método tieneComoAncestroA(...) y deje la implementación para el final y discuta su solución con el ayudante.



FACULTAD DE INFORMATICA



UNIVERSIDAD  
NACIONAL  
DE LA PLATA

# Ejercicio 8: Genealogía Salvaje



## Clase Mamifero

¿Atributos?

¿Relaciones?

¿Variables de instancia?



FACULTAD DE INFORMATICA



UNIVERSIDAD  
NACIONAL  
DE LA PLATA

# Ejercicio 8: Genealogía Salvaje

Siguiendo los ejemplos de ejercicios anteriores, ejecute las pruebas automatizadas provistas.

En este caso, se trata de una clase, **MamiferoTest**, que debe agregar dentro del paquete tests.

En esta clase se trabaja con la familia mostrada en la siguiente figura.



FACULTAD DE INFORMÁTICA



UNIVERSIDAD  
NACIONAL  
DE LA PLATA

# Ejercicio 9: Red de Alumbrado

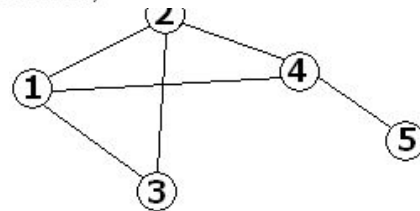
- Ejercicio 8: Genealogía Salvaje
- **Ejercicio 9: Red de Alumbrado**

## Ejercicio 9: Red de Alumbrado

Imagine una red de alumbrado donde cada farola está conectada a una o varias vecinas formando un [grafo conexo](#)<sup>2</sup>. Cada una de las farolas tiene un interruptor. Es suficiente con encender o apagar una farola cualquiera para que se enciendan o apaguen todas las demás. Sin embargo, si se intenta apagar una farola apagada (o si se intenta encender una farola encendida) no habrá ningún efecto, ya que no se propagará esta acción hacia las vecinas.

La funcionalidad a proveer permite:

1. crear farolas (inicialmente están apagadas)
2. conectar farolas a tantas vecinas como uno quiera (las conexiones son bi-direccionales)
3. encender una farola (y obtener el efecto antes descrito)
4. apagar una farola (y obtener el efecto antes descrito)



FACULTAD DE INFORMATICA



UNIVERSIDAD  
NACIONAL  
DE LA PLATA

# Ejercicio 9: Red de Alumbrado

```
public class Farola {  
  
    // Variables de instancia??  
  
    /*  
  
    Crea la relación de vecinos entre las farolas. La relación de vecinos entre las farolas es recíproca, es  
    decir el receptor del mensaje será vecino de otraFarola, al igual que otraFarola también se convertirá en  
    vecina del receptor del mensaje  
  
    */  
  
    public void pairWithNeighbor(Farola otraFarola) {  
        ...  
    }  
  
    /*  
  
    * Si la farola no está encendida, la enciende y propaga la acción.  
    */  
  
    public void turnOn() {  
        ...  
    }  
}
```



FACULTAD DE INFORMATICA



UNIVERSIDAD  
NACIONAL  
DE LA PLATA

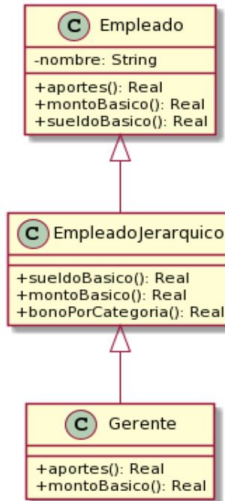


# Ejercicio 10: Method lookup con Empleados

Sea la jerarquía de `Empleado` como muestra la figura de la izquierda, cuya implementación de referencia se incluye en la tabla de la derecha.

Ejercicios de la semana:

- **Ejercicio 10: Method lookup con Empleados**
- Ejercicio 11: Cuenta con ganchos
- Ejercicio 12: Job scheduler



Empleado	EmpleadoJerarquico	Gerente
<pre>public double montoBasico() {     return 35000; }</pre>	<pre>public double sueldoBasico() {     return super.sueldoBasico()+ this.bonoPorCategoria(); }</pre>	<pre>public double aportes() {     return this.montoBasico() * 0.05d; }</pre>
<pre>public double aportes(){     return 13500; }</pre>	<pre>public double montoBasico() {     return 45000; }</pre>	<pre>public double montoBasico() {     return 57000; }</pre>
<pre>public double sueldoBasico() {     return this.montoBasico() + this.aportes(); }</pre>	<pre>public double bonoPorCategoria() {     return 8000; }</pre>	



FACULTAD DE INFORMATICA



UNIVERSIDAD  
NACIONAL  
DE LA PLATA

# Ejercicio 10: Method lookup con Empleados

Analice cada uno de los siguientes fragmentos de código y resuelva las tareas indicadas abajo:

```
Gerente alan = new Gerente("Alan Turing");  
double aportesDeAlan = alan.aportes();
```

```
Gerente alan = new Gerente("Alan Turing");  
double sueldoBasicoDeAlan = alan.sueldoBasico();
```

## Tareas:

1. Liste todos los métodos, indicando nombre y clase, que son ejecutados como resultado del envío del último mensaje de cada fragmento de código (por ejemplo, (1) método +aportes de la clase Empleado, (2) ...)
2. ¿Qué valores tendrán las variables aportesDeAlan y sueldoBasicoDeAlan luego de ejecutar cada fragmento de código?



FACULTAD DE INFORMATICA



UNIVERSIDAD  
NACIONAL  
DE LA PLATA

# Ejercicio 11: Cuenta con ganchos

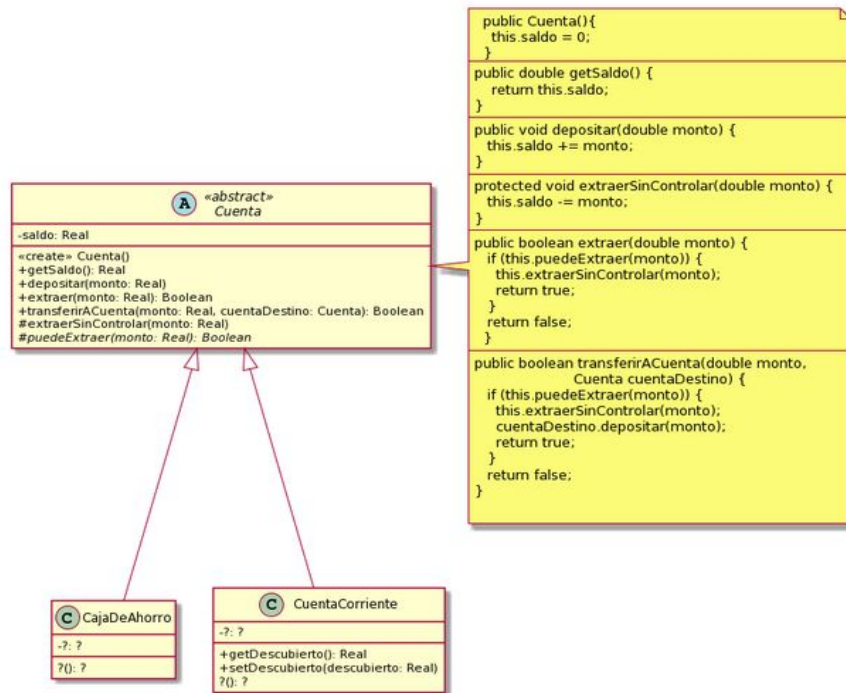
Observe con detenimiento el diseño que se muestra en el siguiente diagrama. La clase cuenta es *abstracta*. El método `puedeExtraer()` es abstracto. Las clases `CajaDeAhorro` y `CuentaCorriente` son concretas y están incompletas.

## Ejercicios de la semana:

- Ejercicio 10: Method lookup con Empleados
- **Ejercicio 11: Cuenta con ganchos**
- Ejercicio 12: Job scheduler



FACULTAD DE INFORM



# Ejercicio 11: Cuenta con ganchos

## Ejercicios de la semana:

- Ejercicio 10: Method lookup con Empleados
- **Ejercicio 11: Cuenta con ganchos**
- Ejercicio 12: Job scheduler

**Tarea A:** Complete la implementación de las clases CajaDeAhorro y CuentaCorriente para que se puedan efectuar depósitos, extracciones y transferencias teniendo en cuenta los siguientes criterios.

- 1) Las **cajas de ahorro** solo pueden extraer y transferir cuando cuentan con fondos suficientes.
- 2) Las extracciones, los depósitos y las transferencias desde **cajas de ahorro** tienen un costo adicional de 2% del monto en cuestión (téngalo en cuenta antes de permitir una extracción o transferencia desde caja de ahorro).
- 3) Las **cuentas corrientes** pueden extraer aún cuando el saldo de la cuenta sea insuficiente. Sin embargo, no deben superar cierto límite por debajo del saldo. Dicho límite se conoce como límite de descubierto (algo así como el máximo saldo negativo permitido). Ese límite es diferente para cada cuenta (lo negocia el cliente con la gente del banco).
- 4) Cuando se abre una **cuenta corriente**, su límite descubierto es 0 (no olvide definir el constructor por default).



FACULTAD DE INFORMATICA



UNIVERSIDAD  
NACIONAL  
DE LA PLATA

# Ejercicio 11: Cuenta con ganchos

Ejercicios de la semana:

- Ejercicio 10: Method lookup con Empleados
- **Ejercicio 11: Cuenta con ganchos**
- Ejercicio 12: Job scheduler

	Caja Ahorro \$ 400	Cuenta Corriente \$100 límite: \$500
depositar(200) ?	596 ?	300 ?
extraer(\$100) ?	298 ?	0 ?
extraer(\$400) ?	x ?	-300 ?



FACULTAD DE INFORMATICA



UNIVERSIDAD  
NACIONAL  
DE LA PLATA

# Ejercicio 11: Cuenta con ganchos

Ejercicios de la semana:

- Ejercicio 10: Method lookup con Empleados
- **Ejercicio 11: Cuenta con ganchos**
- Ejercicio 12: Job scheduler

	Caja Ahorro \$ 400	Cuenta Corriente \$100 límite: \$500
depositar(200) ?	✓ Saldo \$596	✓ Saldo \$300
extraer(\$100) ?	✓ Saldo \$298	✓ Saldo \$0
extraer(\$400) ?	✗	✓ Saldo \$-300



FACULTAD DE INFORMATICA



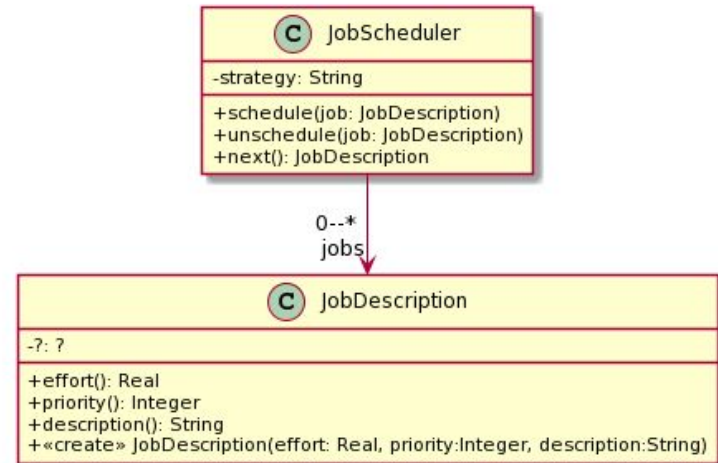
UNIVERSIDAD  
NACIONAL  
DE LA PLATA

# Ejercicio 12: Job Scheduler

## Ejercicios de la semana:

- Ejercicio 10: Method lookup con Empleados
- Ejercicio 11: Cuenta con ganchos
- **Ejercicio 12: Job scheduler**

El JobScheduler es un objeto cuya responsabilidad es determinar qué trabajo debe resolverse a continuación. El siguiente diseño ayuda a entender cómo funciona la implementación actual del JobScheduler.



- El mensaje `schedule(job: JobDescription)` recibe un job (trabajo) y lo agrega al final de la colección de trabajos pendientes.
- El mensaje `next()` determina cuál es el siguiente trabajo de la colección que debe ser atendido, lo retorna, y lo quita de la colección.



FACULT



# Foros de consulta

## Cómo preguntar en el foro

Breve guía para poder sacar el mejor provecho al foro y a la convivencia a través de las preguntas y respuestas.

### Cómo preguntar en el foro

Antes de Preguntar: Busca una respuesta por tus propios medios

Elegí el foro específico

Elegí un título apropiado para la pregunta

No envíes una solución para que la corrijan

Describir qué estás intentando hacer

Describir el problema y lo que has intentado para resolverlo

Escribir claro

No solicites respuestas a tu correo

Si no entendés la respuesta

Terminá con una breve nota de conclusión.

Evitá el "Me sumo al pedido"



Foro: Ejercicio 10 - Method lookup con Empleados



Foro: Ejercicio 11 - Cuentas con ganchos



Foro: Ejercicio 12 - Job Scheduler



FACULTAD DE INFORMATICA



UNIVERSIDAD  
NACIONAL  
DE LA PLATA