CÓMO INSTANCIAR
RUBY: **JAVA:**
obj=Clase.new(params) Clase obj=new Clase(params)

ATRIBUTOS DE INSTANCIA
RUBY: @atributo_ de_instancia = valor. Se declaran en el método initialize.
JAVA: (visibilidad) (tipo) deinstancia = defecto; Se declaran al crear la clase.
ATRIBUTOS DE CLASE
RUBY: @@declase = valor. Se declaran en la clase(fuera de métodos instanc)
@instanciadeclase=valor. No se comparten con las subclases.
¡Cuidado! Al acceder en instancia tengo que usar getters y setters.
Java: (final) static (tipo) DECLASE = valor. Se declaran al crear la clase.

PSEUDOVARIABLES: this en Java, self en Ruby.
VARIABLES LOCALES: Sin @.
CONSTANTES: final en Java y en mayúscula en Ruby, sin @. No se usará.

MÉTODOS:
RUBY: No se pueden sobrecargar.
Permite params por defecto.
private pone priv todos los de instancia que tenga después. Tb private :metodo
private_class_method :metodo funciona para métodos de clase.
Para llamar desde amb instancia a amb clase, si método de clase private NO.
Si es public,
self.class.metodo
Una insta. no puede usar métodos de clase, pero sí a un método que lo haga.
Una instancia no puede llamar a métodos de otra instancia privados.
Algo.metodo es error si el método es privado.

JAVA: No permite params por defecto.
Para llamar desde amb instancia a amb clase, sin problema.
Una insta. puede usar métodos de clase públicos (no recomendable).

CONSTRUCTORES:
RUBY:
El constructor de ruby llama al método especial llamado initialize. Se ocupa de la **creación** e inicialización de atributos de instancia. Para tener 2:

```
class Punto
    def initialize(x, y, z)
        @x=x
        @y=y
        @z=z
    end
    def self.new_2(x,y)
        new (x,y,0) #Llama a initialize con dos parámetros
    end
    # Queremos quitar el normal. Hacemos el new privado

    #Otra opción
    def initialize (x,y,*z) #*z es un arrray con el resto
        #en funcion de z.size haces
    end
end
```

JAVA: es necesario que this esté en la 1ª línea.

JAVA:
Mismo nombre que la clase. Se puede reutilizar un constructor desde otro constructor con this. NO LLEVA PRIVATE NI otros especificadores de acceso.

CONSULTORES Y MODIFICADORES
JAVA: getAtributo{return atributo},void setAtributo(par){atributo = par}
RUBY: attr_reader :atributoinstancia #consultor
attr_writer :atributoinstancia #modif attr_accesor :atributoinstancia #ambos
Para atributos de clase o de instancia de la clase tengo que hacerlos yo.
Los modificadores son def atributo=param, y solo llevan un parámetro.

AGRUPACION:
JAVA: Package. No existen subpaquetes, y se usa import Paquete.Clase
RUBY: Module. Hay módulos dentro de módulos en Ruby. require_relative(archivo propio) o require (del lenguaje). Cuidado con los extras, ¡bucles!

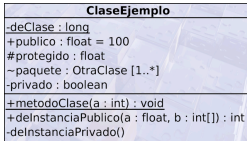
HERENCIA: *es-un*.
RUBY: class Hija < Padre
JAVA: class Hija extends Padre

REDEFINICIÓN (O SOBREESCRITURA) DE MÉTODOS (VS SOBRECARGA):
JAVA: @Override en redefinir.
No se pueden redefinir final ni privados.
Se puede cambiar la cabecera para dar *más* accesibilidad o devolver *subclase* de la que estaba.
RUBY: Aunque cambies los parámetros, ya has redefinido. Mismo nombre es perder el del padre.

Super para no repetir código.
JAVA: Se puede llamar super.CualquierMétodo. Solo se recomienda llamar al actual.
RUBY: super siempre llamará al actual. Si no le pasas params, se usarán los del método del hijo.

CONSTRUCTORES:
JAVA: Obliga a que super esté en 1ª línea.
En el hijo, por defecto se llamará al sin parámetros del padre. Si no tiene, dará error.

UML:
Asociaciones que generan atributos de referencia:
Clase asociación -------- (contrato)
agregación ----<> Se pone en el TODO. Ahí cualquier card.
composición ---- Se pone en el TODO. Ahí cardinalidad 1
Dependencia ----→ No genera atributo de referencia.
Herencia ----|> triangulo en padre.



Cardinalidad.1 por defecto. n*… n a muchos.
La cardinal en MI extremo significa cuántas instancias como yo están relacionadas con UNA del extremo opuesto.
Paquetes: Carpetas, y si están fuera con \oplus.

Diagramas de Secuencia
Condiciones debajo del nombre entre corchetes:
- Alt para if/else
- Opt para if
- Loop

Línea de vida mientras vive. Recursividad es línea doble.