

# Cheat-Sheet. Parcial 1. PD00

• Java  
• Ruby

Antonio Xavier Rodríguez Romero

## Apuntes Teoría

### Construcción objetos

Persona p = new Persona("Juan");  
Persona p2 = p;  $\Rightarrow$  Juan  $\in$  p2  
Persona ps;  $\Rightarrow$  NO  
p = Persona new("Juan")

### Atributos / Métodos

De instancia  $\Rightarrow$  private int edad;  $\Rightarrow$  en initialize()  
@edad def edad(); return @edad end

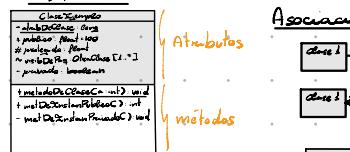
De clase  $\Rightarrow$  static private int numPersonas = 0  
static int getNumPersonas() {return numPersonas;}  
@numPersonas = 0 | def self.numPersonas  
fuerza de initialize() end @numPersonas

$\Rightarrow$  @atributo-de-clase  $\Rightarrow$  Depende del contexto

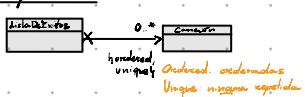
### Constructores de clase (Clase Reserva)

Persona C() {  
 nombre: "";  
 edad: 0; // declaran, inicializan solo  
}  
Persona C(int edad, String nombre) {  
 this.nombre = nombre;  
 this.edad = edad;  
}

### Diagramas



### Propiedades



### Herencia

Relaciones de herencia "el-un"  $\Rightarrow$  DripizConGoma es un dripiz

Un DripizConGoma es un dripiz

Un dripizConGoma se puede usar para todo lo que se usa un dripiz

```

class Persona {
    attr_reader :nombre
    def initialize(n)
        @nombre = n
    end
}
class Empleado < Persona
end
class Director < Empleado
end

```

class Persona {
 public String nombre()
 ...
}

public String nombre();
 ...
}

return ("Nombre como persona");
 ...
}

class Profesor entiendo Persona {
 public String hablar();
 ...
}

return ("Habla como un profesor");
 ...
}

Cambia la implementación  
del método de Persona

• Java  
• Ruby

## Sintaxis

System.out.println("texto" + dato);  
puts "texto", dato / print "texto", dato

Constantes  $\Rightarrow$  final int CONSTANTE = 1  
CONSTANTE = 1  
Arrays  $\Rightarrow$  import java.util.ArrayList;  
ArrayList<int> vector = new ArrayList();  
vector = [1]  $\Rightarrow$  Vector vacío  
Constructores  $\Rightarrow$  Como C++, permite sobrecarga  
def initialize  $\Rightarrow$  No sobrecarga  
end  $\Rightarrow$  def self.newCopy()  
objeto = Clase new end  
objeto = Clase.newCopy(Clase)

### Declaración de atributos

<private/public>(<static>)(<final>) <clase> <nombre> = ...  
Atrib. de instancia  $\Rightarrow$  En initialize() (@nombre  
clase  $\Rightarrow$  fuerza de initialize())

### Lectura/Escritura Atributos

.get Atributo()  $\Rightarrow$  atributo  
.set Atributo(atributo)  $\Rightarrow$  atributo = Clase.nuevo  
attr\_accessor/writer/accessory: atributo

### Protección

private int a; privado  
public float b; público  
double c; paquete  
def void DeBaque  
end  
private  
def visiblizada  
end  
public

### Paquetes externos

import <paquetes>

ClaseDeBaque obj

Modulo::Clase

include <paquetes>  $\Rightarrow$  Como si se copiara explícitamente

### Cabeceras

casos de otros paquetes  
package <nom\_paquete>;  
import ...;  $\Rightarrow$  atributos  
public class <nombre> h;  $\Rightarrow$  tiene main  
static public void main (String[] args) h

#encoding utf-8  
require <lib-ruby>  
require\_relative 'archivo.rb'  
module <nombre>  $\Rightarrow$  Para cosa de otro  
class <nombre> módulo NombreModule::  
def self main  
end  
end  $\Rightarrow$  nombre del archivo ejecutando  
if \_\_FILE\_\_ == "nombre.archivo.rb"  
<nombre>.main  
end

### Tipos enumerados

<Cabecera>  
module TipoEnum  
TIPO1 = :tipo1  
TIPO2 = :tipo2  
ó  
public enum WeaponType h  
LASER200;  
MESSING0;  
FLASHAC400;  $\Rightarrow$  sub  
private final float power;  
WeaponType float p0;  
this.power = p;  
public String toString() h  
return...  
end  
LASER = Type.new(C.0)  
MESSING = Type.new(C.0)  
end

## Diagrama de Comunicación

<G> - Global  
<A> - Asociación  
<P> - Parámetro  
<l> - Local  
<s> - Self