Sesión 2: Atributos y métodos

Hay tipos de atributos:

- Instancia
- De clase
- Finales

Atributos de instancia

Los **atributos de instancia** son particulares de cada objeto creado por esa clase. Por lo tanto son para datos únicos y propias para cada objeto.

RUBY

En **Ruby** se identifica con **@**.

Ejemplo en Ruby, donde el constructor recibe un parámetro nombre y lo almacena en el atributo @nombre.

```ruby
class Persona
    def initialize(nombre)
        @name = name
    end
end
```

Y ese @nombre ahora se puede usar desde cualquier método

```ruby
class Persona
    def initialize(nombre)
        @name = name
    end

    def saludar(otro_nombre)
        "Hola " + otro_nombre + " me llamo " + @nombre
    end
end
```

El atributo que empieza por @ **solo puede ser leído y modificado dentro de Persona** ya que es **privado**.

JAVA

En Java para empezar hay 4 tipos de atributos:

- Pública (+) --> public
- Protegida (#) --> protected
- Privada (-) --> private
- De paquete (~) --> package

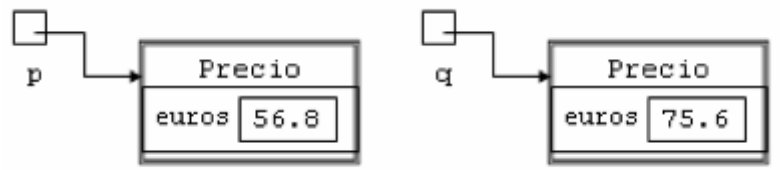
Una vez entendido esto. Pasamos a las variables de instancia. En java se declaran sin nada en especial.

```
public class Precio {
    public String euros;
}
```

Esto significa que se está reserva una dirección de memoria para euros de esa clase. Por ejemplo:

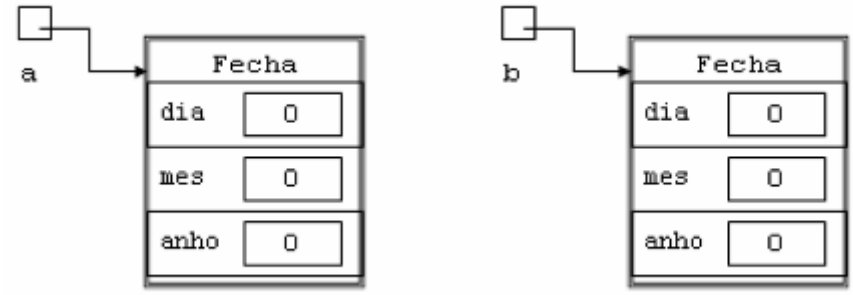
```
Precio p = new Precio();
p.pone(56.8);
Precio q = new Precio();
q.pone(75.6);
```

Esto se representa así en el sistema:



Y otro ejemplo:

```
public class Fecha {
    // Declaracion de atributos o variables miembro
    private int dia;
    private int mes;
    private int anho;
}
```



Atributos de clase

Los **atributos de clase** implican que en una **única zona de memoria se guarda todas las instancias de la clase** y no una copia por objeto. Esto significa que estos atributos son compartidos por todas las instancias de esa clase.

Los atributos de clase **no son visibles fuera de la clase** pero SI en **subclases**.

Un ejemplo es contar el número de personas que hay en una clase de bachillerato. En este caso utilizarías un atributo de clase para que vaya aumentando sobre un número común. O por ejemplo, si queremos poner un valor por defecto, para esto también se usaría. Otro ejemplo sería el número de billetes de la lotería.

RUBY

En el caso de Ruby, se utiliza **@@**.

En el siguiente ejemplo vamos a inicializar un contador a 0 y vamos a ir aumentando el valor en el **initialize**.

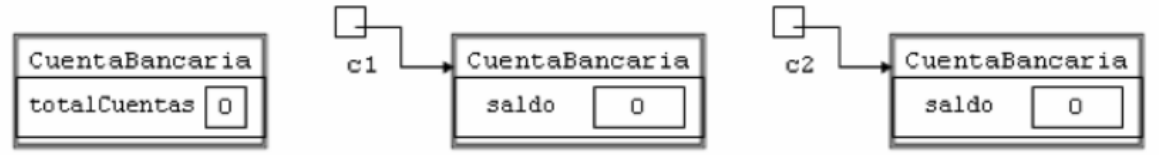
```ruby
class Persona
    @@contador = 0 #Inicializo el contador de personas a 0

    def initialize
        @@contador += 1 # Aumento el contador porque estoy creando una persona
    end
end
```

JAVA

Los atributos de clase en java se representan mediante la palabra **static**.

```java
public class CuentaBancaria{
    public static int totalCuentas = 0;
}
```

Y por muchas cuentas que creemos, empiezan con el valor igualado a 0.



Atributos de instancia de clase

Se representa también con un @. Estos atributos no son accesibles al ámbito de la instancia y no comparten con las subclases.

¡CUIDADO! Hay que tener cuidado donde se colocan las cosas, todo lo que esté antes del initialize es de clase o de instancia de clase.

Ruby: Confusión entre atributos

```ruby
class Clase
  @@variable = "De clase"
  @variable = "De instancia de la clase"

  def initialize
    @variable = "De instancia"
  end

  def muestraValores
    puts @@variable
    puts @variable
  end

  def self.muestraValores
    puts @@variable
    puts @variable
  end
end

objeto = Clase.new
objeto.muestraValores
Clase.muestraValores
```

En Java no hay problema:

Java: Atributos y métodos de clase y de instancia, variables locales

```java
public class Persona {

  private static final int MAYORIAEDAD=18; //Atributo de clase
  private LocalDateTime fechaNacimiento;   //Atributo de instancia

  Persona(LocalDateTime fecha) {
      fechaNacimiento=fecha;
  }

  public boolean mayorDeEdad() { // Método de instancia
    LocalDateTime ahora= LocalDateTime.now(); //Llamada a método de clase
    // "ahora" es una variable local

    //Años completos transcurridos
    long edad=ChronoUnit.YEARS.between(fechaNacimiento,ahora);

    return (edad>=MAYORIAEDAD);
  }
}
```

ATRIBUTOS CONSTANTES O FINALES

Este tipo de atributo se utiliza para contener atributos constantes, de esta manera, el atributo no se puede modificar que no se van a modificar y suelen ir con los **atributos de clase** (static final).

En java:

```java
public class Circulo {
    private static final double PI = 3.141592;
}
```

MÉTODOS DE INSTANCIA

Son **funciones o métodos** definidos en una clase y que estarán asociados a los objetos de dicha clase. Desde estos métodos podemos acceder a los atributos.

RUBY

Como podemos ver, para poder acceder desde otras instancias (recordemos que son otros objetos de esa misma clase), tenemos que crear métodos.

```ruby
class Persona
    def initialize(nombre)
        @nombre = nombre
    end

    def saludar(otro_nombre)
        "Hola " + otro_nombre + " me llamo " + @nombre
    end

    def name
        @nombre
    end

    def cambiar_nombre(otro_nombre)
        @nombre = otro_nombre
    end
end
```

Ejecutable

```ruby
p1 = Persona.new("Maria")
puts p1.name # Imprime Maria

p1.cambiar_nombre("Paula")
puts p1.name #Imprime Paula
```

Atajos

Atajos para crear métodos de consulta y modificación de un atributo de instancia.

- **attr_accessor :n** --> leer y escribir
- **attr_reader :n** --> leer
- **attr_writer :n** --> escribir

```ruby
class Persona
    attr_accessor :nombre, :genero #Puedes leer y escribir
    attr_reader :edad # Puedes solo leer
end
```

En este caso, para escribir en la edad habría que crear un método a parte.

JAVA

Un método en java se declara con parámetros o sin ellos.

```
class Ejemplo{
    void imprimir(){ }
    int sumar (int a, int b){
        return a + b;
    }
}
```

El ejecutable:

```
Ejemplo ejemplo = new Ejemplo();
ejemplo.imprimir();
int resultado1 = ejemplo.sumar(4,6); // llamamos al método suma con argumentos 4
y 6
int resultado2 = ejemplo.sumar(3,3); // llamamos al método suma con argumentos 3
y 3
```

MÉTODOS DE CLASE

Los **métodos de clase** son procedimientos asociados a la propia clase pero no depende de ninguna instancia.

Hay situaciones donde podemos necesitar llamar a un método sin tener que definir un objeto de la clase, por ejemplo, queremos saber el valor del contador de personas de la clase Persona, para ello no hace falta crear un objeto.

RUBY

En Ruby se utiliza **self.nombre_metodo** para crear un método de clase.

```
class Persona
    @@num_personas = 0
    def self.num_personas
        @@num_personas
    end

    def initialize (nombre_nuevo)
        @nombre = nombre_nuevo
        @@num_personas += 1
    end

end
```

JAVA

En java se utiliza **static**.

```
class Persona {
    static private int numPersonas = 0;
    static int getNumPersonas(){
        return numPersonas;
    }
    private String nombre;
    Persona(String nombreNuevo){
        nombre = nombreNuevo;
        numPersonas++;
    }
}
```

VISIBILIDAD

En **Java**:

- Desde una instancia a otra instancia en la misma clase
- Desde el ámbito de clase a una instancia de esa clase
- Desde el ámbito de la instancia a la clase de que es esa instancia.

En **Ruby**:

- Todos lo anterior F
- Todos los atributos son privados