Atributos y Métodos de Instancia y de Clase

Programación y Diseño Orientado a Objetos (PDOO)

Tabla de Contenidos

Atributos y Métodos de Instancia

Están asociados a un **objeto individual**. Cada objeto (instancia) tiene su propia copia y espacio en memoria para estos elementos.

Atributos de Instancia (Estado)

Característica	Descripción
Definición	Variables que definen el estado particular de cada objeto
Memoria	Cada instancia tiene su propia zona de memoria, aunque compartan el mismo nombre

Ejemplo en Java

```
// Cada instancia 'ana' y 'pepe' tendrá su propia variable 'nombre'
public class Persona {
    private String nombre; // Atributo de instancia
    // ... constructor ...
}
/* Uso:
Persona ana = new Persona("Ana");    // nombre: "Ana"
Persona pepe = new Persona("José");  // nombre: "José"
*/
```

Métodos de Instancia (Comportamiento)

Característica	Descripción
Definición	Métodos que operan sobre los atributos de instancia de su propio objeto
Acceso	Tienen acceso a los atributos de instancia (lectura y escritura)

Ejemplo en Java

```
class Persona {
    private String nombre;
    // Método de instancia
    String saludar() {
        // Accede al atributo de instancia 'nombre'
        return "Hola, me llamo " + nombre;
    }
    void cambiaNombre(String otroNombre) {
        nombre = otroNombre; // Modifica el atributo de instancia
    }
}
```

Atributos y Métodos de Clase (Estáticos)

Están asociados a la **propia clase**, no a una instancia específica. Existen de forma única y son globales para todas las instancias.

Atributos de Clase (Estáticos)

Característica	Descripción
Definición	Almacenan información común y única para todas las instancias
Memoria	Existe una sola copia en una zona de memoria asociada a la clase
Uso Típico	Contadores de instancias o constantes

Ejemplo en Java (Atributo `static`)

```
public class Persona {
    // Atributo de clase (static)
    static private int numPersonas = 0;
    // Constructor
    Persona (String unNombre) {
        // ...
        numPersonas++; // Se incrementa para TODAS las instancias
    }
}
```

Ejemplo en Ruby (Atributo `@@`)

```
class Persona
    # Atributo de clase (se comparte con subclases, posible peligro)
    @@num_personas = 0
    def initialize (un_nombre)
        # ...
        @@num_personas += 1
    end
end
```

Métodos de Clase (Estáticos)

Característica	Descripción
Definición	Métodos asociados a la clase. Se llaman directamente usando el nombre de la clase
Acceso	Pueden acceder/actualizar atributos de clase
Restricción	⚠ No pueden acceder directamente a atributos/métodos de instancia sin especificar una instancia concreta

Ejemplo en Java (Método `static`)

```
public class Persona {
    static private int numPersonas = 0;
    // Método de clase (static)
    static int getNumPersonas () {
        return numPersonas; // Accede al atributo de clase
    }
}
```

Ejemplo en Ruby (Método `self.`)

```
class Persona
    @@num_personas = 0
    # Método de clase (def self.nombre_metodo)
    def self.num_personas
        @@num_personas
    end
end
```

Pseudovariables

Palabras reservadas que referencian al **propio objeto** o a la **clase** dentro del código.

Lenguaje	Pseudovariable	Referencia	Uso Típico
Java	`this`	La instancia actual	Distinguir atributos de parámetros (`this.nombre = nombre`) o llamar a otros constructores
Ruby	`self`	La instancia actual o la clase	Dentro de métodos de instancia → objeto. Dentro de métodos de clase → clase

Ejemplo en Java (`this`)

```
class Persona {
    private String nombre;
    // 'this.nombre' se refiere al atributo de la instancia
    // 'nombre' se refiere al parámetro recibido
    Persona (String nombre) {
        this.nombre = nombre;
    }
    // Llamada a otro constructor de la misma clase
    Persona () {
        this("Anónimo");
    }
}
```

Ejemplo en Ruby (`self`)

```
class Persona
    @@MAYORIA_EDAD = 18
    # self.mayoria_edad se refiere al método de la CLASE
    def self.mayoria_edad
        return @@MAYORIA_EDAD
    end
    def to_s
        # self.nombre se refiere al método de la INSTANCIA
        return "Me llamo " + self.nombre
    end
end
```

Visibilidad y Especificadores de Acceso

Define los niveles de acceso a atributos y métodos.

> **Regla de Oro**: Asignar la visibilidad **más restrictiva posible**.

Reglas de Diseño

Elemento	Visibilidad Recomendada	Razón
Atributos	✅ Privados	Proteger el encapsulamiento
Acceso Externo	✅ Consultores/Modificadores públicos	Control de lectura/escritura con validación
Encapsulamiento	✅ Crítico	Los atributos solo deben modificarse por métodos propios del objeto

Acceso Privado (Diferencias Clave)

Característica	Java	Ruby
Atributos	Pueden ser privados	Siempre son privados (`@` o `@@`)
Acceso entre instancias	✅ Permitido. Una instancia de `A` puede acceder a los privados de otra instancia de `A`	❌ No permitido. Encapsulamiento más estricto; acceso privado solo para la instancia actual (`self`)
Especificador	`private`	`private` (métodos de instancia) `private_class_method` (métodos de clase)

🔄 Resumen Comparativo

Tipo	Java	Ruby	Alcance
Atributo de Instancia	`private String nombre;`	`@nombre`	Por objeto
Atributo de Clase	`static int count;`	`@@count`	Por clase
Método de Instancia	`public void metodo()`	`def metodo`	Por objeto
Método de Clase	`static void metodo()`	`def self.metodo`	Por clase
Pseudovariable	`this`	`self`	Contexto actual

Adam Bourbahh Romero ~ PDOO