

# Introducción al paradigma orientado a objetos

Jorge I. Meza

[jimezam@autonoma.edu.co](mailto:jimezam@autonoma.edu.co)

# Contenidos

- Paradigmas de programación
- Nuestro camino
- Concepto de modularidad
- Clase y objeto
- Elementos de una clase
- Reglas de nombrado
  - Generales
  - Clases
  - Atributos / variables
  - Constantes
  - Métodos
- Representación de las clases
  - Diagrama de clases
  - Lenguaje de programación Java

# Paradigmas de programación

- Un paradigma de programación es un enfoque o estilo particular de solucionar los problemas.
- Esto impacta la manera de estructurar y organizar el código de un programa.
- La *modularidad* en el código de la solución es muy deseable.

# Paradigmas de programación

- Paradigma Imperativo
  - Realizar tareas en secuencias de instrucciones.
  - FORTRAN y COBOL.
- **Paradigma procedimental**
  - Es una extensión natural del paradigma imperativo, descomponiendo el programa en procedimientos o funciones independientes y reutilizables.
  - ALGOL y C.

Curso Lógica de programación

# Paradigmas de programación

- Paradigma Funcional
  - Se basa en el concepto de funciones matemáticas y evita cambiar el estado y los datos mutables.
  - Lisp
- Paradigma Lógico
  - Se basa en la lógica formal y la inferencia, describiendo qué se debe hacer en lugar de cómo hacerlo.
  - Prolog

# Paradigmas de programación

Cursos FPOO /  
POO

- **Paradigma Orientado a Objetos**
  - Se basa en el modelamiento del mundo real, organizando el código en objetos que encapsulan datos y comportamientos.
  - Simula, SmallTalk, C++, Java
- **Paradigma Declarativo**
  - Mezcla elementos de aspectos de programación lógica y funcional. Se basa en especificar qué se quiere y no cómo se debe lograrlo.
  - SQL

# Paradigmas de programación

- Paradigma Basado en Eventos
  - Muy utilizado en la implementación de respuestas en interfaces de usuario.

# Nuestro camino

- Paradigma **Imperativo**
  - Todo el código en el bloque `main` 🙈
- Paradigma **procedimental**
  - Código organizado entre funciones y procedimientos 👍
- Paradigma **Orientado a Objetos**
  - Código en clases creando objetos con encapsulamiento, abstracción, herencia y polimorfismo 💪💪

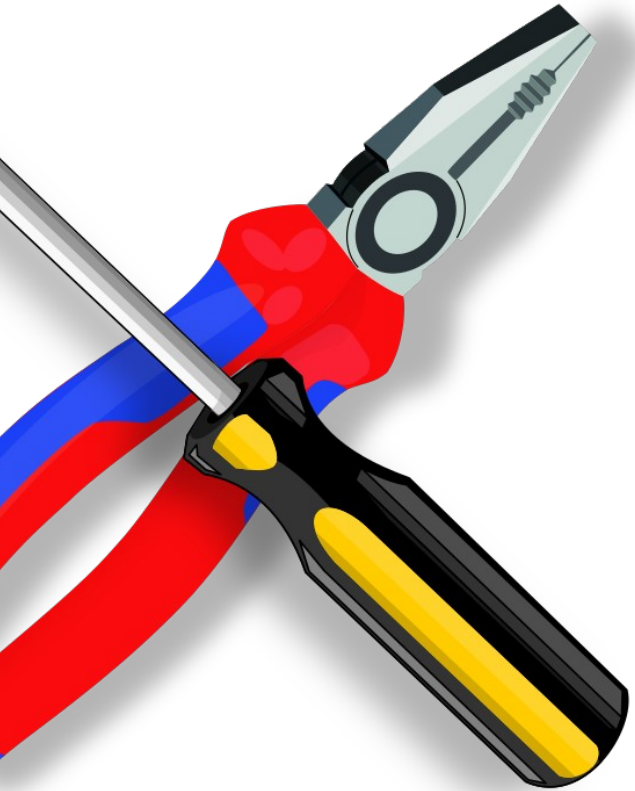
Modularidad



# Actualmente paradigma imperativo



# Actualmente paradigma procedimental



Datos  
(estado)

- Variables
- Constantes
- Arreglos y matrices

Funcionalidad  
(acciones)

- Funciones
- Procedimientos

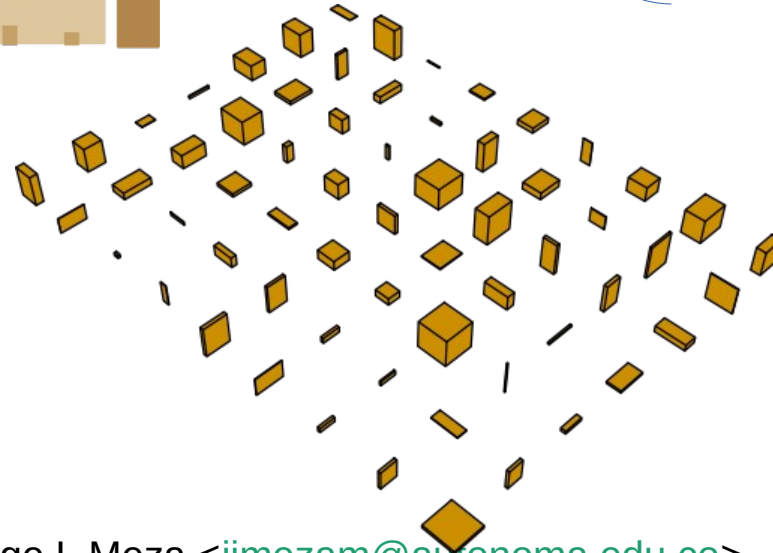
# Próximamente paradigma orientado a objetos



Entidades

Características

Acciones



Más  
Modular

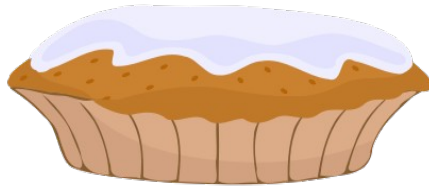
# Ventajas de la modularidad

- **Facilita el mantenimiento**
- **Permite la reutilización del código**
- Mejora la comprensión y legibilidad del código
- Facilita la colaboración entre desarrolladores
- Facilita las pruebas
- Hace que el código sea más flexible y escalable
- Reduce la complejidad de la solución

# Conceptos clave

## Simple Cake Recipe

225g (8 oz) self-raising  
flour.  
225g (8 oz) soft butter  
(i.e. room temperature).  
225g (8 oz) caster sugar.  
4 eggs.  
1 teaspoon baking powder.



Mix the ingredients well in a large bowl using an electric  
whisk.  
Halve the mixture and pour into 2 non-stick 18cm (7 inch)  
cake tins.  
Cook till golden brown (15-25 minutes) in a preheated oven  
at 180 degrees C (gas mark 4).  
Cool on a wire rack before serving, add jam between  
the two halves and optionally top with butter cream.

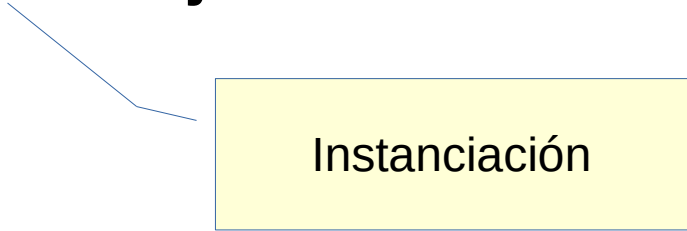


Receta → Torta

# Conceptos clave

Receta → Torta

Clase → Objeto



Instanciación

# Conceptos clave

- La clase es una "plantilla" que describe una entidad y permite crear objetos.
- La clase determina el "contenido" de los objetos creados con ella.
- Los objetos son entidades computacionales "vivas" con las cuales se puede interactuar de acuerdo con lo definido por su clase.

# Conceptos clave

Contexto del problema  
(Mundo real)



Análisis

Contexto de la solución

Diseño

Entidades

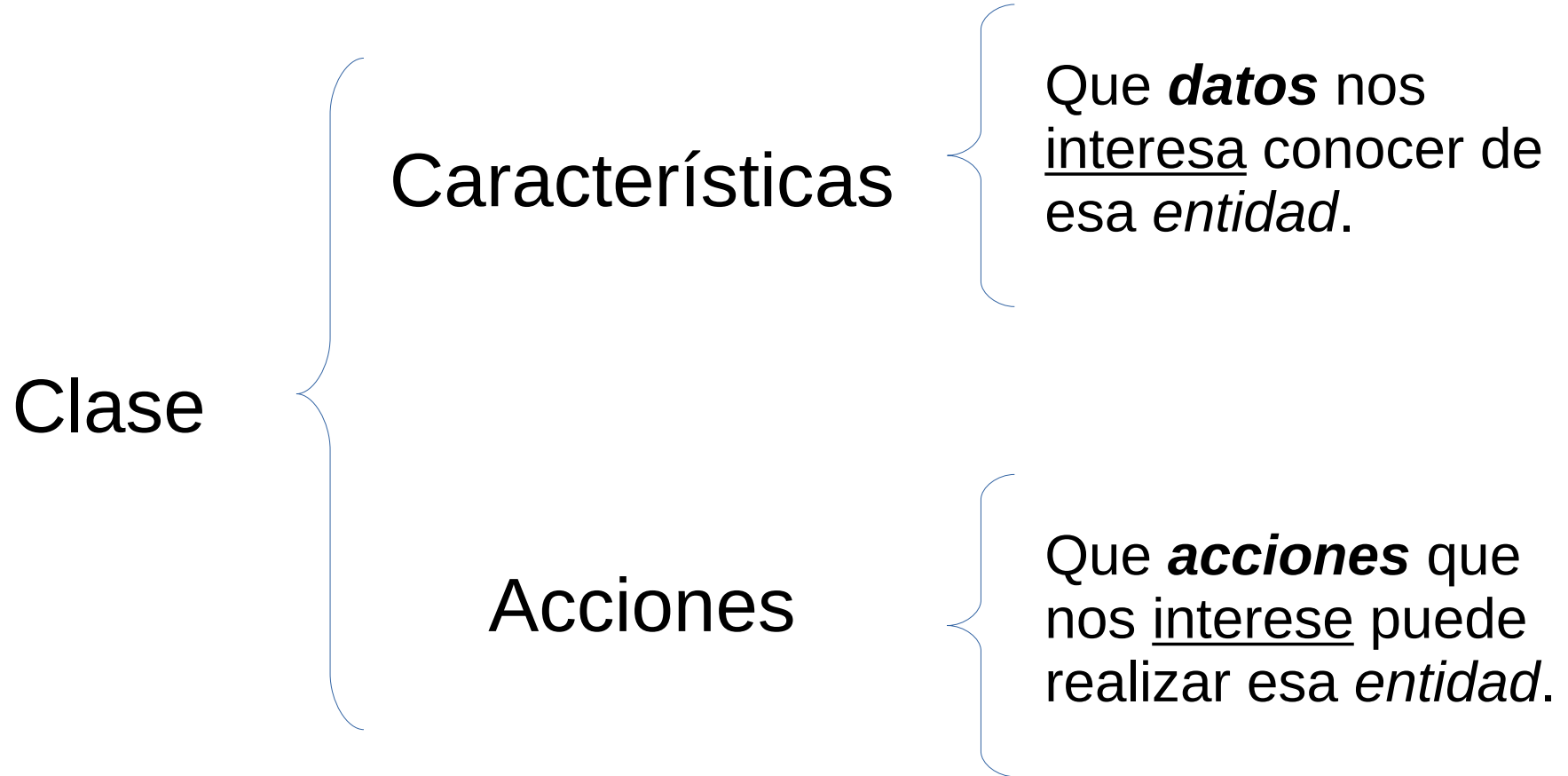
Implementación

Clases

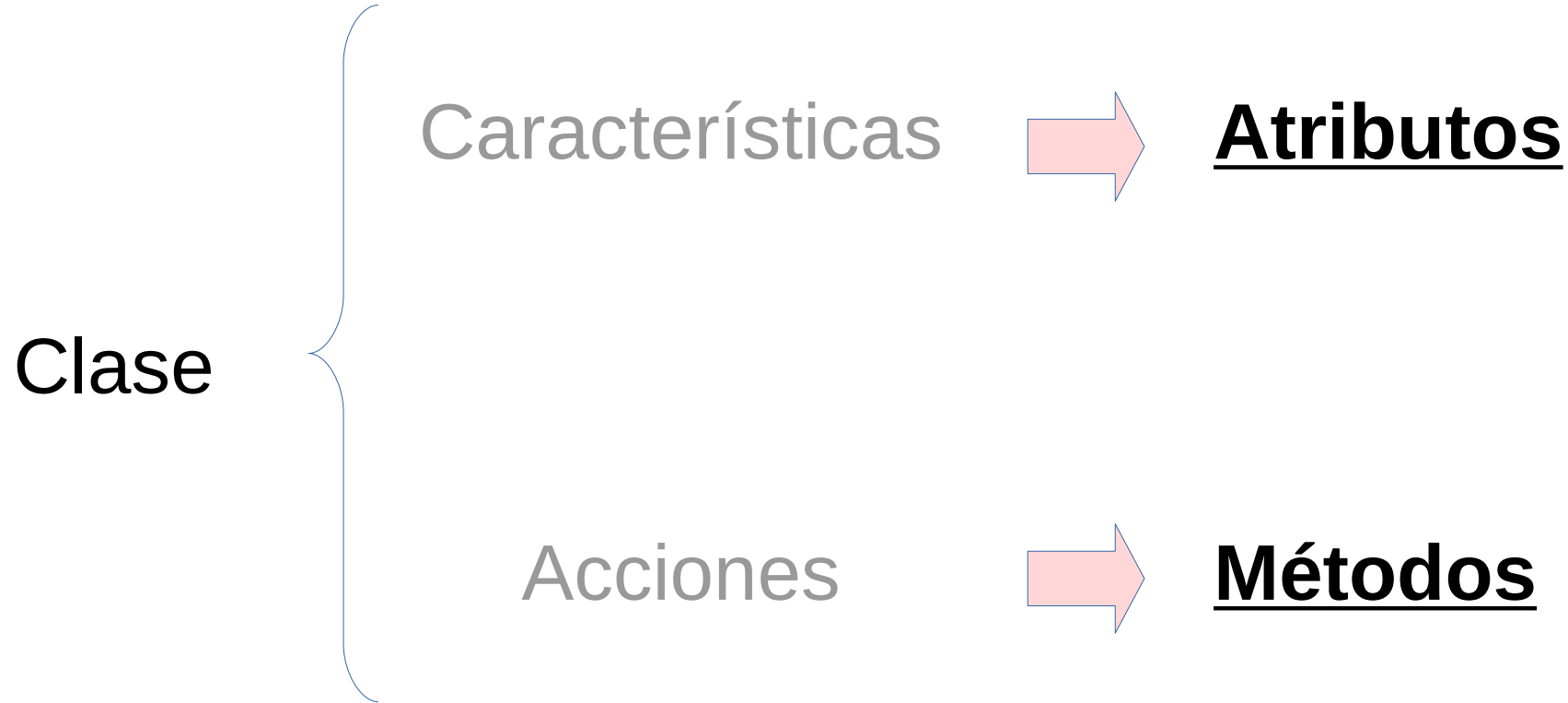
Objetos



# Elementos de una clase



# Elementos de una clase



# Ejemplo de una clase

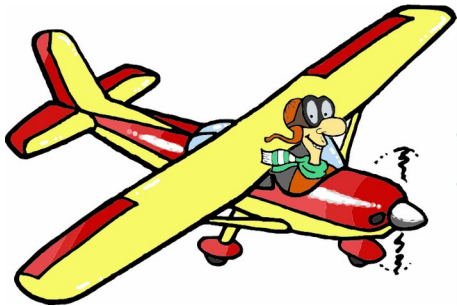
Clase

Atributos

- marca
- modelo
- color
- sabor
- identificación
- velocidad
- altitud
- estado

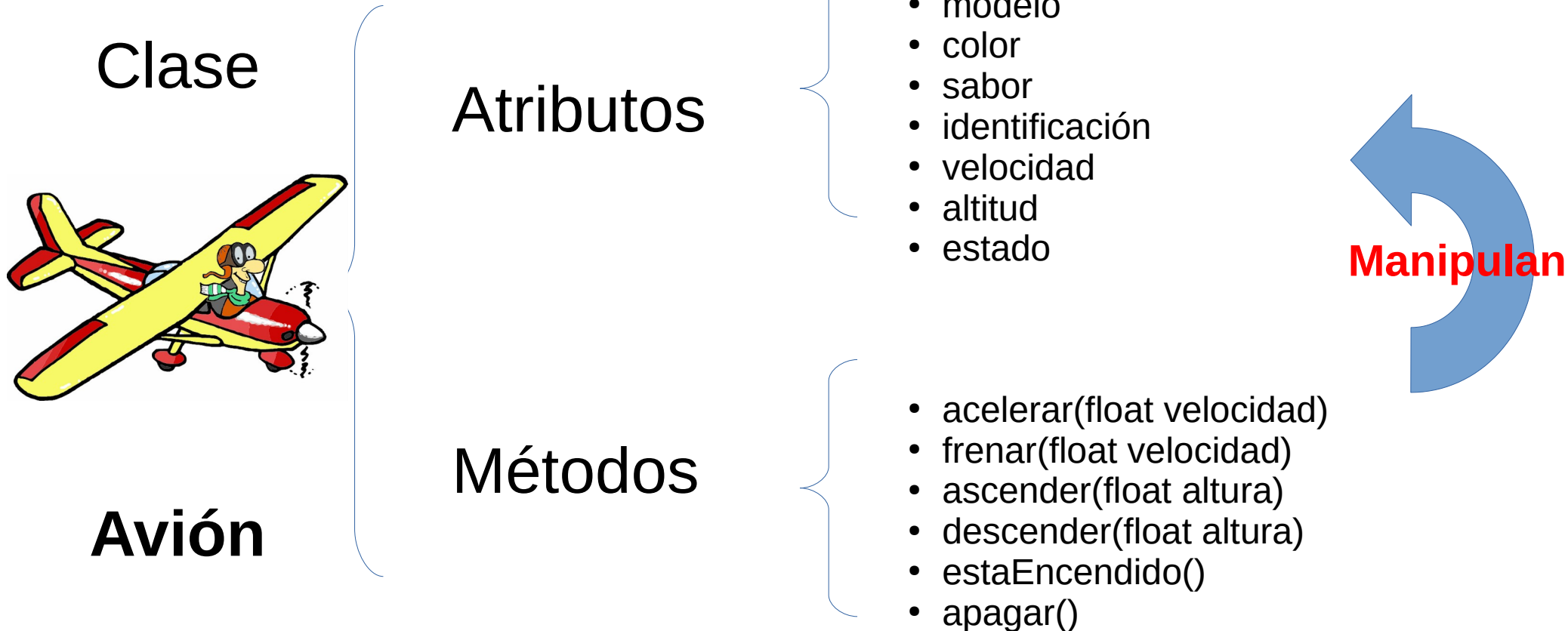
Métodos

- acelerar(float velocidad)
- frenar(float velocidad)
- ascender(float altura)
- descender(float altura)
- estaEncendido()



Avión

# Ejemplo de una clase



# Consideraciones importantes

- Cada clase "encapsula" todo lo referente a esa *entidad* que se está modelando.
- Los atributos y métodos se eligen de acuerdo con lo requerido por la solución propuesta (abstracción).
- Tanto atributos como métodos son "*reflexivos*".
- Los métodos modifican el estado del objeto, es decir, sus atributos.

# Reglas generales de nombrado



- El nombre del identificador debe ser claro y conciso y dar una idea precisa de su significado.

# Reglas generales de nombrado

- El carácter Inicial debe ser una letra (mayúscula o minúscula), un signo de dólar (\$) o un guion bajo (\_).
- No puede empezar con un dígito.
- Los demás caracteres pueden ser letras, dígitos, signos de dólar (\$) o guiones bajos (\_).
- No se deben utilizar *palabras reservadas* del lenguaje como identificadores.
- Son sensibles a mayúsculas y minúsculas.

# Ejemplos identificadores

## Válidos

```
int edad;  
double _salario;  
String $nombre;  
float valorTotal2;  
char caracterEspecial;  
boolean isTrue;
```

## Inválidos

```
int 2edad;  
String nombre#;  
float valor total;  
char char;  
boolean true;
```



# Acerca de identificadores de clases

- Las clases modelan *entidades*, así que deben ser **sustantivos** y generalmente en **singular**.
- La primera letra debe ser **mayúscula**.
- Si es nombre compuesto, utilizar **Camel Case**.
- Ejemplos:
  - `Persona`
  - `Camión`
  - `PaqueteUrgente`

# Acerca de identificadores de atributos o variables

- Los atributos modelan *características*, así que deben ser **sustantivos**. Pueden ser en **singular** o **plural** de acuerdo con su naturaleza.
- La primera letra debe ser **minúsculas**.
- Si es nombre compuesto, utilizar **Camel Case**.
- Ejemplos:
  - nombre
  - direcciónPersonal
  - nombresJugadores

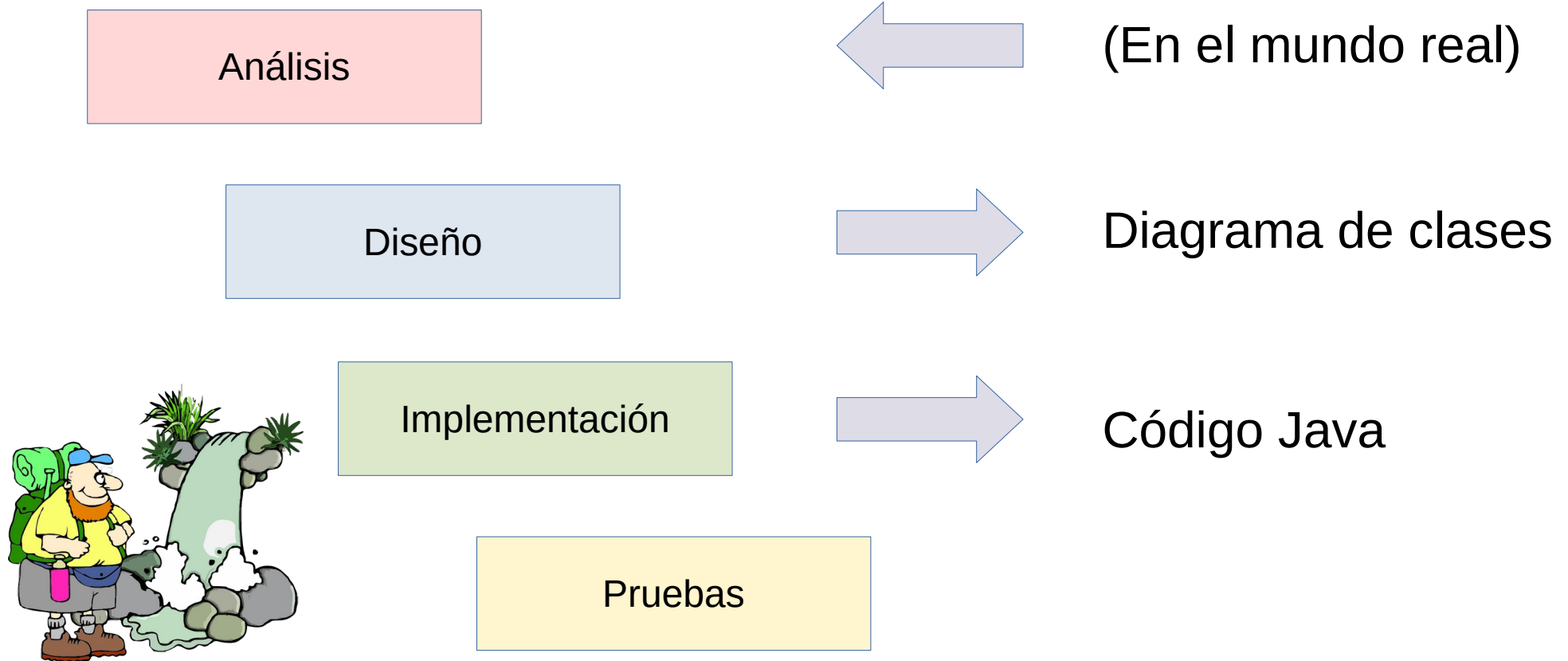
# Acerca de identificadores de constantes

- Las constantes modelan *características* o *valores*, así que deben ser **sustantivos**. Pueden ser en **singular** o **plural** de acuerdo con su naturaleza.
- Deben escribirse completamente en **mayúsculas**.
- Si es nombre compuesto, utilizar **snake\_case**.
- Ejemplos:
  - GRAVEDAD
  - IMPUESTO\_RENTA
  - VELOCIDAD\_MÁXIMA

# Acerca de identificadores de métodos

- Los métodos modelan *acciones*, así que deben ser **verbos** y estar conjugados en **infinitivo** (\*ar, \*er, \*ir).
- La primera letra debe ser **minúsculas**.
- Si es nombre compuesto, utilizar **Camel Case**.
- Ejemplos:
  - cargar
  - validarUsuario
  - mostrarDatos

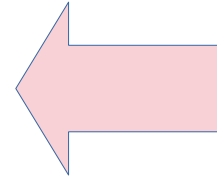
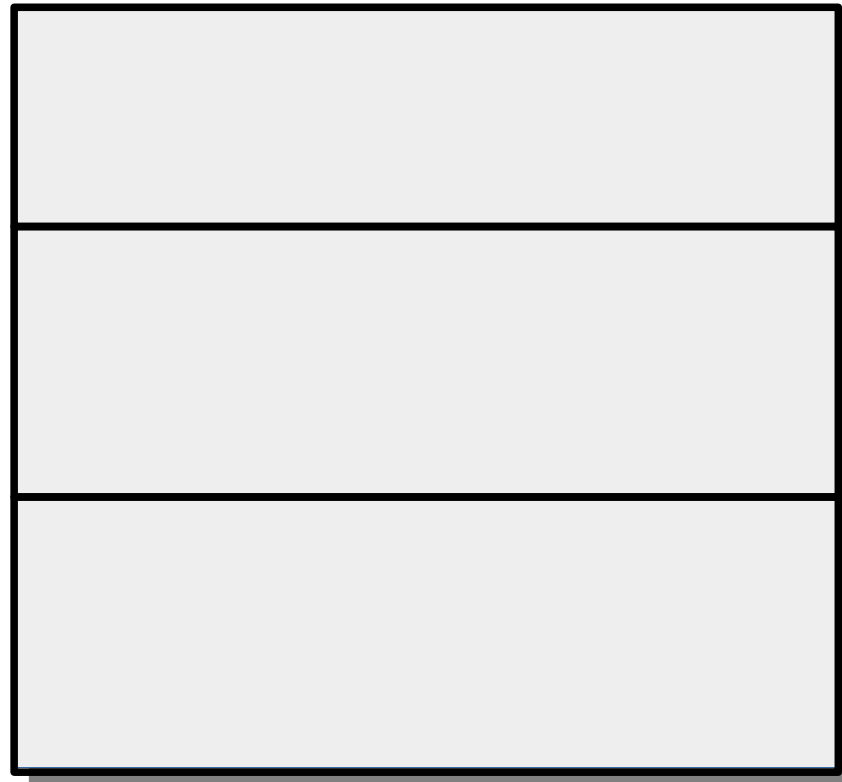
# Representación de las clases



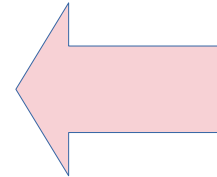
# Representación de las clases

- Durante la **etapa de diseño**, las clases se representan gráficamente en el **diagrama de clases**.
- Mientras que en la **etapa de implementación** las clases se representan en **código fuente**, en nuestro caso, utilizando **Java**.

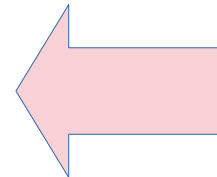
# Clases en el diagrama de clases



Nombre



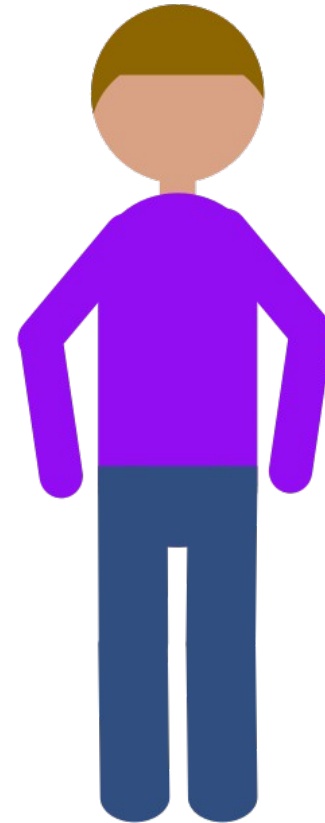
Atributos



Métodos

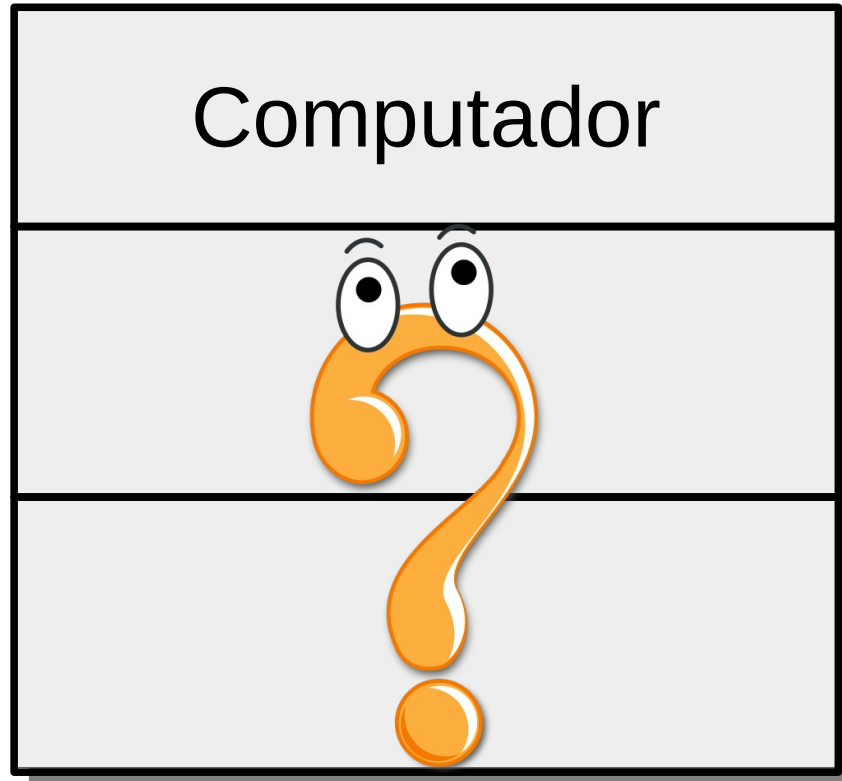
# Ejemplo: clase Persona

Persona
nombres : String apelliidos : String cédula : String edad : int
dormir() : void respirar(veces : int) : void saltar(veces : int, altura : float) : boolean comer(alimento : Hamburguesa) : void estáDespierto() : boolean





# Ejemplo: clase Computador



# Ejemplo: clase Computador

Computador
estado : boolean marca : String velocidadCPU : double cantidadRAM : int tipo : byte
encender() : boolean esPortatil() : boolean ejecutar(instrucción : String) : String guardarValor(nombre : String, valor : int) : boolean recuperarValor(nombre : String) : int



# Clases en código Java

- Generalmente se escriben a través del IDE. Netbeans por ejemplo.
- **Cada clase debe escribirse en un archivo independiente y el nombre del archivo debe ser el mismo nombre de la clase, con la extensión ".java".**

# Clase base en código Java

```
// Nombre.java
```

```
class Nombre {
```

```
    // Atributos
```

```
    // Métodos
```

```
}
```

# Ejemplo: clase Persona

```
// Persona.java
```

```
class Persona {  
    String nombres;  
    String apellidos;  
    String cedula;  
    int edad;
```

```
// ...
```

```
    void dormir() {  
    }  
    void respirar(int veces) {  
    }  
    boolean saltar  
        (int veces, float altura) {  
    }  
    void comer  
        (Hamburguesa alimento) {  
    }  
    boolean estáDespierto() {  
    }  
}
```

# Ejemplo: clase Computador

```
// Computador.java
```

```
class Computador {
```

```
    boolean estado;
```

```
    String marca;
```

```
    double velocidadCPU;
```

```
    int cantidadRAM;
```

```
    byte tipo;
```

```
    // ...
```

```
    boolean encender() {  
    }
```

```
    boolean esPortatil() {  
    }
```

```
    String ejecutar(String instrucción) {  
    }
```

```
    boolean guardarValor(String nombre,  
int valor) {  
    }
```

```
    int recuperarValor(String nombre) {  
    }
```

```
}
```



¿Alguna pregunta?