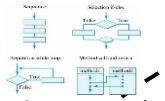
Introducción al Paradigma de Orientación a Objetos

Programación Estructurada Clásica

- Ejecución secuencial de instrucciones
- Uso de **subrutinas** o procedimientos
- Transferencia incondicional del flujo (**GOTO**)
- Problemas: código espagueti, dificultad de modificación y depuración
- Crisis del software (1965-1968)

Programación Estructurada Moderna

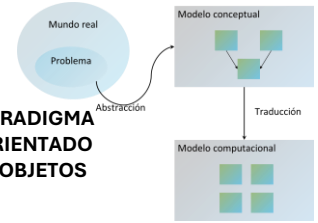


- Teorema de **Böhm-Jacopini**
- Uso de estructuras de control: secuencia, selección e iteración
- Definición de tareas, descomposición y estructuración de datos
- Modularidad y agrupación de funciones

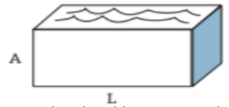
```
import math

def calcularAreaCirculo(radio):
    return math.pi * (radio**2)
```

CAMBIO EN EL ENFOQUE

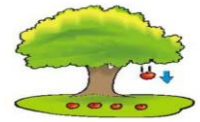
PARADIGMA PROCEDIMENTAL



PARADIGMA ORIENTADO A OBJETOS



Se resuelve el problema pensando en **funciones** o **procedimientos**
V = largo * ancho * alto



Ejecución del programa se realiza por **gravedad**.

Inicios de la Programación Orientada a Objetos (POO)

- Modula-67 (**primer lenguaje** orientado a objetos)
- **Smalltalk** (desarrollo en XEROX PARC, herencia)
- Java (popularización en los años 80 y 90, Sun Microsystems — Construir objetos complejos a partir de objetos más simples -> **Herencia**
- POO como técnica dominante

Técnica o **estilo de programación** que utiliza **objetos** como bloques esenciales de construcción — se **centra en los datos datos**
un **método** para **implementar programas** que son organizados en **colecciones de objetos** **colaborativos**

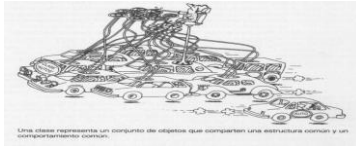
BENEFICIOS
- **Reutilización** de código
- **Facilidad** de modelado
- **Independencia** entre componentes

Beneficios de POO

- Facilita el mantenimiento y la modificación del código
- Reutilización de código
- Independencia entre componentes

Conceptos Fundamentales de POO

Clases — se **estructuran** internamente los objetos (**atributos**)
- Es un **modelo de comportamiento**
- las **clases** son a sus **objetos** lo que los **tipos primitivos** son a sus **variables**



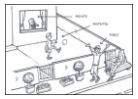
Objetos — Es un concepto o **entidad** con límites bien definidos y con significado para un problema planteado

Abstracción
- Proceso de modelado mediante la eliminación de detalles irrelevantes
- Proceso de modelado mediante la eliminación de detalles irrelevantes
- Proceso de modelado mediante la eliminación de detalles irrelevantes

Alto nivel
Bajo nivel

Encapsulamiento
- **Ocultamiento** de la implementación (estado y comportamiento)
- **Ocultamiento** de la implementación (estado y comportamiento)
- Modificadores de acceso (público, privado)

Cada miembro tiene **visibilidad**
- **Público**
- **Privado**
- **Otros**

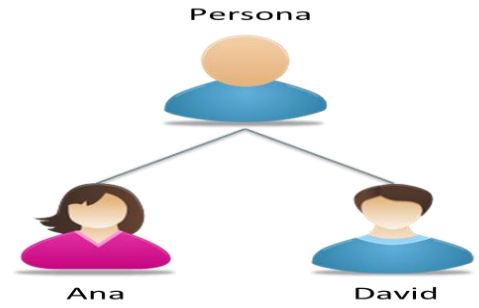


Modularidad
- División de una aplicación en partes más pequeñas
- Espacios de nombres (namespace)

Organizar aplicaciones a nivel interno.
El paquete crea un **espacio de nombres**

Herencia
- Reutilización de código mediante subclases y superclases
- Generalización y especialización

Mensajes — Comunicación entre objetos a través de métodos
- El objeto destinatario del mensaje
- El nombre del mensaje, que describe la tarea a realizar (el nombre del método)
- La información que el objeto origen envía al destinatario, necesaria para que éste pueda completar su tarea (los parámetros del método)



Todos los objetos son instancias de alguna clase

Estado:
valor del volumen
valor del tono
AM/FM
apagado/encendido
frecuencia seleccionada
CD introducido

Comportamiento:
Subir/bajar volumen
Cambiar tono
Cambiar banda
Encender/Apagar
Selecc. frecuencia
Introducir CD