

# PRACTICA 09Y10

Rodríguez Alvarez María José

November 2025

## 1 Marco Teórico

### 1.1 Abstracción

La abstracción consiste en identificar las características esenciales de un objeto y omitir detalles que no son relevantes para el problema que se desea resolver. Esto nos permite diseñar clases que representan conceptos generales y establecer una base común para clases más específicas.

### 1.2 Encapsulamiento

El encapsulamiento es el mecanismo que protege los datos internos de un objeto, restringiendo su acceso mediante modificadores de visibilidad. En Dart, el prefijo “\_” indica que un atributo es privado a nivel de archivo. El uso de getters y setters permite validar datos y mantener la integridad de los valores asignados.

### 1.3 Herencia

La herencia permite crear nuevas clases basadas en una clase ya existente, reutilizando sus atributos y métodos. Gracias a este principio, las clases derivadas pueden extender o redefinir comportamientos. En este proyecto, `Auto`, `Moto` y `Camion` heredan de la clase abstracta `Vehiculo`.

### 1.4 Polimorfismo

El polimorfismo permite que distintas clases implementen métodos con el mismo nombre pero con comportamientos diferentes. Esto facilita procesar objetos heterogéneos de manera uniforme. En esta práctica, cada tipo de vehículo implementa su propia versión de los métodos `calcularServicio()` y `generarReporteServicio()`.

### 1.5 Interfaces

Una interfaz define un conjunto de métodos que una clase debe implementar sin especificar su comportamiento. En Dart, las interfaces ayudan a establecer

contratos claros dentro del diseño del software. La interfaz `ServicioTaller` asegura que todos los vehículos posean los métodos necesarios para calcular costos y generar reportes.

## 1.6 Manejo de Excepciones

El manejo de excepciones permite identificar y gestionar errores durante la ejecución del programa. Dart ofrece mecanismos como `throw`, `try`, `catch` y `finally` para controlar situaciones inesperadas. En esta práctica, se utilizan para validar datos del usuario y evitar fallos en el flujo del programa.

## 1.7 UML (Lenguaje de Modelado Unificado)

El Lenguaje de Modelado Unificado (UML) es un estándar utilizado para representar visualmente la estructura y el comportamiento de un sistema. Los diagramas de clases muestran relaciones jerárquicas y atributos, mientras que los diagramas de secuencia representan la interacción dinámica entre objetos. Su uso facilita la documentación y comprensión del diseño del software.