

## Practica de Semana 04

### ***Enunciado del Caso Práctico: Sistema de Reservas de Transporte***

Una empresa de movilidad inteligente está desarrollando un Sistema de Reservas de Transporte para gestionar y procesar solicitudes de viaje realizadas por los usuarios. El sistema debe ser capaz de manejar distintos tipos de transporte, entre ellos:

- 🚕 Taxi tradicional
- 🚗 Vehículo privado con chofer
- 🚌 Autobús interurbano

Todos los tipos de transporte comparten información básica como el origen, destino, fecha y hora del viaje, pero cada uno tiene datos y comportamientos particulares que los distinguen (por ejemplo, un taxi requiere número de licencia del conductor, un vehículo privado requiere matrícula del automóvil, y un autobús requiere número de ruta y empresa operadora).

### ***Requisitos Técnicos***

#### **Herencia:**

Se debe definir una clase base o abstracta Reserva que contenga los atributos y comportamientos comunes de cualquier tipo de transporte.

#### **Interfaces:**

Algunos tipos de transporte permiten confirmación inmediata al usuario (por ejemplo, taxi y vehículo privado), mientras que otros requieren validación manual (autobús). Define una interfaz Confirmable con un método `enviarConfirmacion()` que implementarán las clases correspondientes.

#### **Polimorfismo:**

El sistema debe procesar las reservas mediante una función que reciba un objeto de tipo Reserva (o su interfaz), y que invoque los métodos correspondientes usando polimorfismo.

No se debe utilizar `instanceof`, condicionales por tipo, ni castings. Cada tipo de reserva debe definir su comportamiento específico sobrescribiendo métodos.

#### **Validación de requisitos:**

El sistema debe verificar que cada reserva cumpla con requisitos mínimos antes de confirmarse:

- Taxi: licencia válida del conductor.

- Vehículo privado: matrícula registrada en el sistema.
- Autobús: ruta existente y empresa autorizada.

### **Restricción:**

Para mantener la simplicidad del sistema, no se deben usar colecciones como listas, arreglos ni estructuras de datos. Cada reserva se procesará de manera individual, una a la vez.

### **Evaluación del Caso Práctico**

<b>Criterio Requisito</b>	<b>/ Puntaje Máximo</b>	<b>Pts Obtenidos</b>	<b>Observaciones</b>
<b>Herencia</b>	20		Se definió correctamente la clase base Reserva, pero faltó incluir un atributo común (hora de llegada).
<b>Interfaces</b>	15		La interfaz Confirmable se implementó en Taxi y Vehículo privado, pero no en Autobús.
<b>Polimorfismo</b>	20		Se procesan las reservas con métodos sobrescritos, sin instanceof ni castings. Excelente aplicación.
<b>Validación de requisitos</b>	20		Taxi y Vehículo privado validan correctamente; Autobús no valida empresa operadora.
<b>Restricción de colecciones</b>	10		No se usaron listas ni arreglos, cada reserva se procesa individualmente.
<b>Notificaciones</b>	15		Taxi envía confirmación, Vehículo privado lo hace parcialmente, Autobús no implementa notificación.

---

**Total: / 100 puntos**