PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

PROGRAMACIÓN 3
Examen 2
(Primer semestre 2025)

Indicaciones generales:

- **Duración:** 3 horas.
- **Materiales o equipos a utilizar:**
  La sección teórica se realizará sin material de apoyo.
  En la sección práctica podrá utilizar material de apoyo: diapositivas, ejemplos de clase, grabaciones de clase, código fuente y documentación oficial de Microsoft u Oracle.
- No está permitido el uso de ningún material o equipo electrónico adicional al indicado.
- **La presentación, la ortografía y la gramática influirán en la calificación.**

Puntaje total: 20 puntos

PARTE TEÓRICA: (4 puntos)
Preguntas Teóricas
Responda en PAIDEIA las dos preguntas teóricas.

PARTE PRÁCTICA: (16 puntos)
Pregunta 1 (04 puntos) – Obligatoria (JAVA – con Netbeans 25 – JDK 21)
Desarrollar un servicio **RESTful** para gestionar información de conductores y sus vehículos, implementando los métodos necesarios para registrar, consultar y validar esta información.
Este servicio será utilizado por la **Superintendencia Nacional de los Registros Públicos (SUNARP)** para diversos fines relacionados con la verificación de la propiedad vehicular y asociación con conductores.
Métodos a implementar

Método	Endpoint	Descripción
GET	/conductores	Listar todos los conductores registrados. Permitir filtrar por tipo de licencia mediante un query parameter opcional, por ejemplo: **/conductores?tipoLicencia=A1.**
GET	/conductores/{num_licencia}/vehiculos	Listar los vehículos registrados a nombre de un conductor identificado por su número de licencia.
GET	/vehiculos/{placa}/conductor	Listar los conductores asociados al vehículo identificado por su número de placa.

POST	/conductores	Registrar un nuevo conductor junto con **su vehículo** asociado en el mismo registro. Asumir que el conductor no está registrado en la BD.
		Ejemplo de body:
		{
		"numeroLicencia": "B98765432",
		"idTipoLicencia": 2,
		"nombres": "María Fernanda",
		"apellidoPaterno": "Vera",
		"apellidoMaterno": "López",
		"vehiculo":
		{
		"placa": "FHZ123",
		"marca": "Kia",
		"modelo": "Rio",
		"anio": 2022,
		"fechaAdquisicion": "2023-01-01"
		}
		}

Consideraciones

- Los estudiantes deben definir **únicamente las clases necesarias** para cumplir con los requerimientos.
- Se debe respetar la **arquitectura de software** y los patrones de diseño vistos en clase.
- La funcionalidad de los servicios REST debe ser probada utilizando **Postman**.
- Se proporciona un **script SQL** con la creación de tablas e inserción de datos de prueba.
- Se entrega un proyecto base en Maven con una solución inicial para el desarrollo.
    - **Puede modificar las clases del dominio (model) si considera necesario.**

Pregunta 2 (04 puntos) – Obligatoria (JAVA – con Netbeans 25 – JDK 21 y C# – con VS2022 – .NET Framework 4.8.1)

Se le ha solicitado implementar el formulario web que se visualiza en la Figura 01. Este formulario se ha diseñado con el propósito de que pueda registrar los datos de un conductor en conjunto con sus vehículos adquiridos. Los vehículos ya se encuentran registrados en la base de datos. Al hacer clic en la lupa, aparecerá la lista de vehículos de la base de datos en un modal. Posteriormente el usuario seleccionará el vehículo y se traerán los datos de placa, marca, modelo y año al formulario principal. Luego, el usuario ingresará la fecha de adquisición del vehículo seleccionado y dará clic en el botón +. Este vehículo aparecerá como parte de una lista dentro de un *gridview* y será tratado como un detalle. Se pondrá asignar más vehículos al conductor repitiendo este procedimiento. Una vez que se han asignado todos los vehículos, el usuario procederá a dar clic en el botón "Guardar" y se guardarán tanto los datos del conductor como las asignaciones de los vehículos.

Importante: Se ha solicitado que debe implementar esta funcionalidad utilizando servicios **SOAP** bajo una arquitectura cliente (C#) – servidor (JAVA). La capa de **DBManager, Domain, Persistance** y **Business** ya están implementadas y han sido adjuntadas al proyecto web mediante JARs. En este sentido se le solicita utilizar las siguientes clases y métodos para el implementar el correcto funcionamiento del formulario:

- **Clase:** `TipoLicenciaBO` – **Método:** `public ArrayList<TipoLicencia> listarTodos();` que lista todos los tipos de licencia desde la base de datos. Se utilizará para poblar el dropdownlist.
- **Clase:** `VehiculoBO` – **Método:** `public ArrayList<Vehiculo> listarPorPlaca(String placa);` que lista todos los vehículos desde la base de datos. Se utilizará para el modal que permite seleccionar los vehículos a asignar al conductor.
- **Clase:** `ConductorBO` – **Método:** `public int insertar(Conductor conductor, ArrayList<VehiculoConductor> vehiculosConductores)` que permite registrar los datos de un conductor y además, la lista de vehículos asignados al mismo en el formato de una lista de objetos de tipo VehiculoConductor. Se utilizará para guardar toda la información ingresada en el formulario.

Descargue los proyectos de C# y de JAVA y complete lo necesario en ambos proyectos web (C# y JAVA) para alcanzar la funcionalidad solicitada. En el proyecto web de JAVA debe colocar sus datos de conexión en el archivo **db.properties** del paquete **pe.edu.pucp.transitsoft.config**.

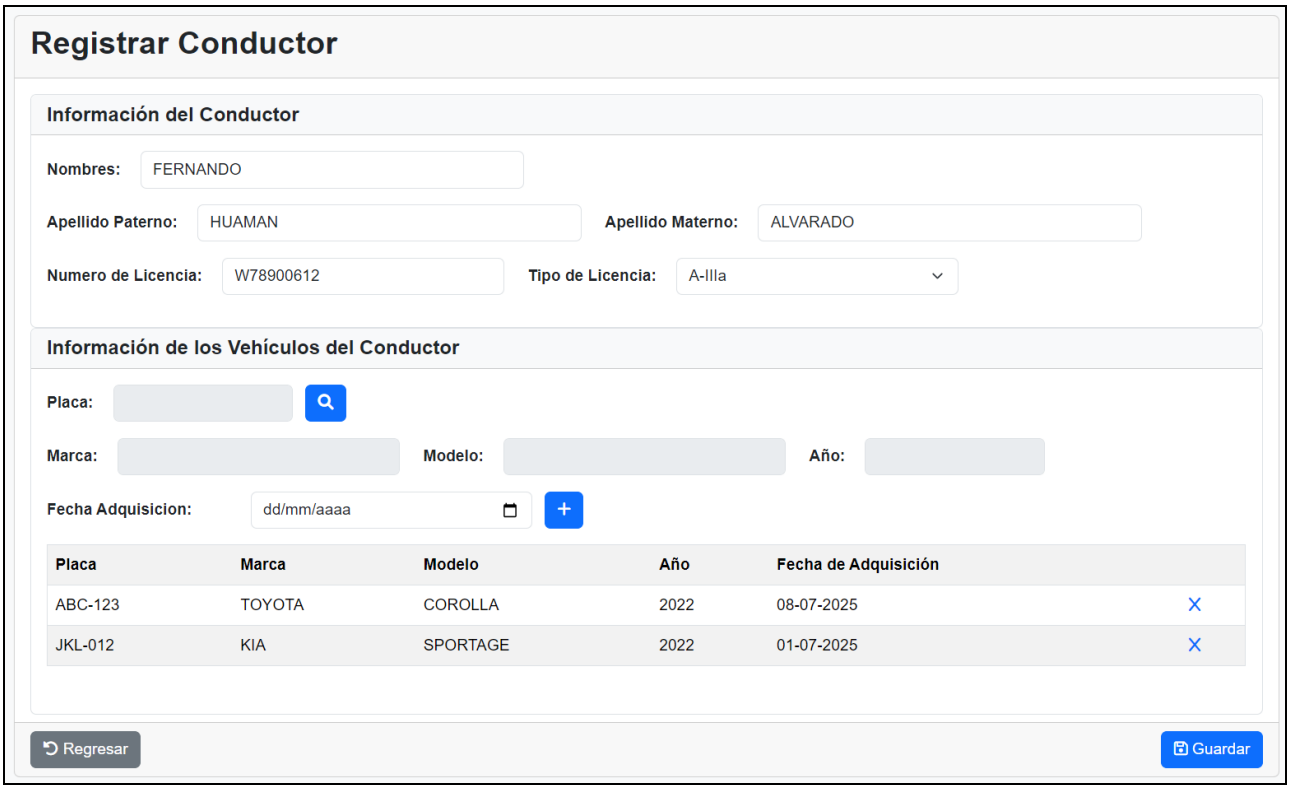

Fig. 01. Formulario de Registro de Conductor.

Pregunta 3 (04 puntos) – **Obligatoria (JAVA – con Netbeans 25 – JDK 21)**

Se les proporciona el archivo creacion_esquema_sqlserver_final.sql, el cual contiene las instrucciones necesarias para crear la base de datos correspondiente al caso descrito anteriormente. Además, cuentan con el archivo insert_sqlserver_final.sql, que incluye sentencias INSERT para poblar las tablas con datos de ejemplo.

Se solicita implementar un método que permita **obtener un conductor por su ID** (ex2_conductor). Este método debe tener la capacidad de **retornar, opcionalmente, los datos completos del tipo de licencia** asociado al conductor, en función de un parámetro específico que se reciba como entrada.

Es importante que el método sea implementado de forma **óptima y eficiente**, evitando consultas innecesarias a la base de datos y reduciendo la repetición de código al mínimo.

Nota: Se les proporciona la capa de dominio en el archivo SoftEx2Model.zip.

Pregunta 4 (04 puntos) – **Obligatoria (JAVA – con Netbeans 25 – JDK 21)**

La Autoridad de Transporte Urbano ha puesto en marcha un plan piloto que requiere el desarrollo de un programa informático para automatizar el registro de infracciones por **exceso de velocidad (M20)** en una de las principales avenidas de la ciudad. Como parte de esta iniciativa, se ha instalado un conjunto de **cuatro cámaras** en un punto de alto flujo vehicular de dicha avenida. Todas las cámaras tienen un campo de visión que les permite detectar a la totalidad de los vehículos que transitan por el área monitoreada, por lo que el sistema debe asegurar que **ningún vehículo** sea registrado **más de una vez** por distintas cámaras.
El programa cuenta con un **registro central del flujo vehicular**, alimentado por un generador de tráfico. Este registro es **compartido** entre el generador y todas las cámaras, lo que hace necesario que cualquier operación sobre el mismo se realice de manera **sincronizada**, garantizando la **exclusión mutua** y previniendo **condiciones de carrera**.
Las cámaras generarán **capturas** de los vehículos en el flujo vehicular, una captura representa la velocidad, fecha, hora y la placa del vehículo cuando fue registrado por una cámara. Cuando una cámara detecta que un **vehículo supera el límite de velocidad** establecido de 80 km/h, se debe **registrar una infracción**. Para ello, el sistema dispone de un **registro centralizado de infracciones**, también compartido por todas las cámaras, cuyas operaciones deben igualmente ser **sincronizadas** de forma adecuada para asegurar la **exclusión mutua**.

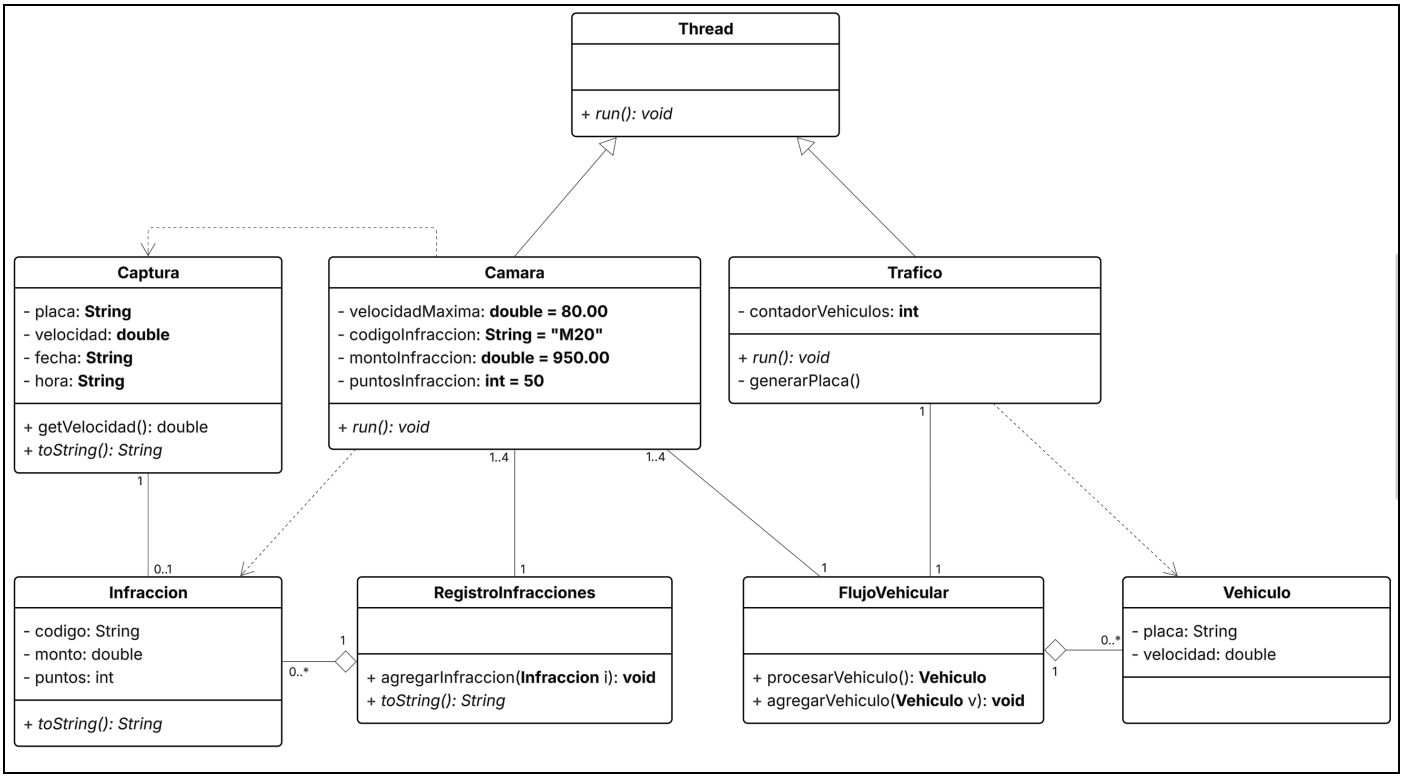

Fig. 02. Diagrama de clases

Al iniciar el programa, se activarán las cuatro cámaras, las cuales permanecerán en **espera** hasta que se detecte **flujo vehicular**. Una vez que todo el flujo haya sido procesado, las cámaras **volverán** a un estado de **espera** hasta que se genere **nuevo tráfico**. Es decir que un vehículo una vez procesado por una cámara es **removido del flujo**

vehicular. Esta lógica sigue el patrón de diseño **Productor/Consumidor**, donde el generador de tráfico actúa como productor y las cámaras como consumidores.
Finalmente, al concluir la ejecución del programa, se debe mostrar en pantalla un reporte detallado con todas las infracciones registradas.

Se entrega:

1. Un proyecto Maven con una solución inicial.
2. La clase principal y el método main implementado.
3. Las clases Trafico (generador), Vehiculo, Captura, Infraccion implementadas.
4. Implementación inicial de la clase Camara.

Se solicita:

1. Implementar las clases FlujoVehicular y RegistroInfracciones. (1 punto)
2. Implementar la clase Camara, con las siguientes responsabilidades: (2 puntos)
    a. Procesar el flujo vehicular.
    b. Detectar y registrar nuevas infracciones de tránsito.
3. Mostrar en pantalla un reporte con todas las infracciones cometidas. (1 punto)

Profesores del curso:

Dr. Freddy Paz
Dr. Andrés Melgar
Dr. Heider Sánchez
Dr. Eric Huiza

San Miguel, 11 de Julio de 2025