**1. Planteamiento del problema**

El sistema desarrollado simula una plataforma de movilidad urbana compartida, donde los usuarios pueden alquilar diferentes tipos de vehículos (bicicletas, patinetes, motos grandes y motos pequeñas) distribuidos en una ciudad virtual representada por un mapa con bases y coordenadas. El objetivo es gestionar de manera eficiente la asignación, uso, mantenimiento y reparación de los vehículos, así como la administración de los usuarios y el personal técnico.

**Actores participantes**

* **Usuarios**: Personas que utilizan el sistema para alquilar vehículos. Pueden ser estándar o premium, con diferentes privilegios y restricciones.
* **Administradores**: Encargados de la gestión global del sistema, configuración inicial y supervisión.
* **Mecánicos**: Personal técnico responsable de la reparación de vehículos y bases averiadas.
* **Personal de mantenimiento**: Encargados del mantenimiento preventivo y correctivo de los vehículos.
* **Sistema**: Incluye la infraestructura de bases, el mapa de la ciudad, la gestión de viajes y la lógica de negocio.

**Relaciones entre actores**

* Los **usuarios** interactúan con el sistema para alquilar y devolver vehículos, consultar su saldo y viajes, y reportar incidencias.
* Los **administradores** configuran el sistema, gestionan usuarios y personal, y supervisan el estado general.
* Los **mecánicos** y **personal de mantenimiento** reciben asignaciones de vehículos o bases averiadas y actualizan su estado tras la reparación.
* El **sistema** coordina la disponibilidad de vehículos, la ocupación de bases, la asignación de tareas y la aplicación de reglas de negocio (tarifas, penalizaciones, privilegios premium, etc.).

**Funcionalidad a cumplir**

* Inicialización flexible del sistema (vacío, con semilla, manual).
* Registro y gestión de usuarios y personal.
* Distribución y gestión de vehículos en bases y mapa.
* Realización y registro de viajes.
* Gestión de incidencias, averías y reparaciones.
* Aplicación de tarifas, descuentos y penalizaciones.
* Visualización de informes y resúmenes del estado del sistema.

**2. Clases principales y relaciones**

**Jerarquía de personas**

* **Persona** (clase base): Atributos comunes como nombre, apellidos y DNI.
  + **Usuario** (abstracta): Añade historial de viajes, saldo y métodos de uso.
    - **Standard**: Usuario estándar, con restricciones normales.
    - **Premium**: Usuario premium, con privilegios adicionales (descuentos, acceso a vehículos con menos batería, etc.).
  + **Trabajador** (abstracta): Personal del sistema.
    - **Mecanico**: Puede ser asignado a vehículos o bases averiadas, gestiona reparaciones.
    - **Mantenimiento**: Encargado de tareas de mantenimiento preventivo/correctivo.
  + **Administrador**: Gestiona y supervisa el sistema.

**Jerarquía de vehículos**

* **Vehiculo** (abstracta): Atributos comunes como ID, tipo, batería, estado de avería, reservado, en uso.
  + **VehiculoDeBase**: Vehículo que se encuentra en una base (bicicletas, patinetes).
    - **Bicicleta**
    - **Patinete**
  + **VehiculoDeCoordenadas**: Vehículo que se encuentra en una posición libre del mapa (motos).
    - **MotoGrande**
    - **MotoPequena**

**Infraestructura**

* **Base**: Representa una estación física en el mapa, con capacidad limitada y lista de vehículos.
* **Mapa**: Representa la ciudad, gestiona las bases y las posiciones de los vehículos.
* **GestorBases**: Singleton que administra las bases, añade/elimina bases, asigna averías, genera informes.
* **GestorVehiculos**: Singleton que administra todos los vehículos, su ubicación y estado.
* **GestorPersonas**: Singleton que administra todas las personas del sistema.
* **GestorViajes**: Singleton que administra los viajes realizados y las tarifas.

**Relaciones de dependencia y colaboración**

* **GestorVehiculos** depende de **Mapa** y **Base** para ubicar vehículos.
* **GestorBases** depende de **Mapa** y **GestorVehiculos** para gestionar la infraestructura.
* **GestorPersonas** crea y gestiona instancias de las subclases de **Persona**.
* **GestorViajes** interactúa con **Usuario**, **Vehiculo** y **Base** para registrar viajes y aplicar reglas de negocio.
* **Mecanico** y **Mantenimiento** pueden tener listas de vehículos o bases asignadas para reparación/mantenimiento.

**3. Jerarquías de herencia y relaciones en BlueJ**

El diseño del sistema aprovecha la herencia para evitar duplicidad de código y facilitar la extensión:

* **Persona** → **Usuario**/**Trabajador** → (**Standard**, **Premium**, **Mecanico**, **Mantenimiento**)
* **Vehiculo** → **VehiculoDeBase**/**VehiculoDeCoordenadas** → (**Bicicleta**, **Patinete**, **MotoGrande**, **MotoPequena**)

Las relaciones de dependencia y composición se reflejan en los atributos y métodos de las clases gestoras y de infraestructura.