- Laboratorio Lab3 -

Herencia en Java

1. Objetivo

En esta práctica incorporaremos un elemento de POO muy importante y que ya hemos estudiado en clase de teoría: la **herencia**. En esta primera parte de la práctica programaremos en Java la jerarquía de clases mostrada por el diagrama UML de la Figura 1.

2. Trabajo del estudiante en esta práctica

1. Implementa en Java la jerarquía de clases descrita en el diagrama UML todo ello en un paquete llamado **PaqComercio**.

2. Asígnales a las clases y a los atributos de cada clase los modificadores de acceso adecuados.

3. Como las clases *del diagrama*, igual que otras clases de Java, son subclases de *Object*, sobrescribe los métodos siguientes para todas las clases:
- equals()
- toString()

4. Para comprobar tu código, se recomienda que uses una o más Java clases principales. Estas clases las situarás en otro paquete distinto al anterior llamado **PaqPruebas**. Para poder usar las clases públicas del paquete **PaqComercio** tendrás que escribir esto:

import PaqComercio.*;

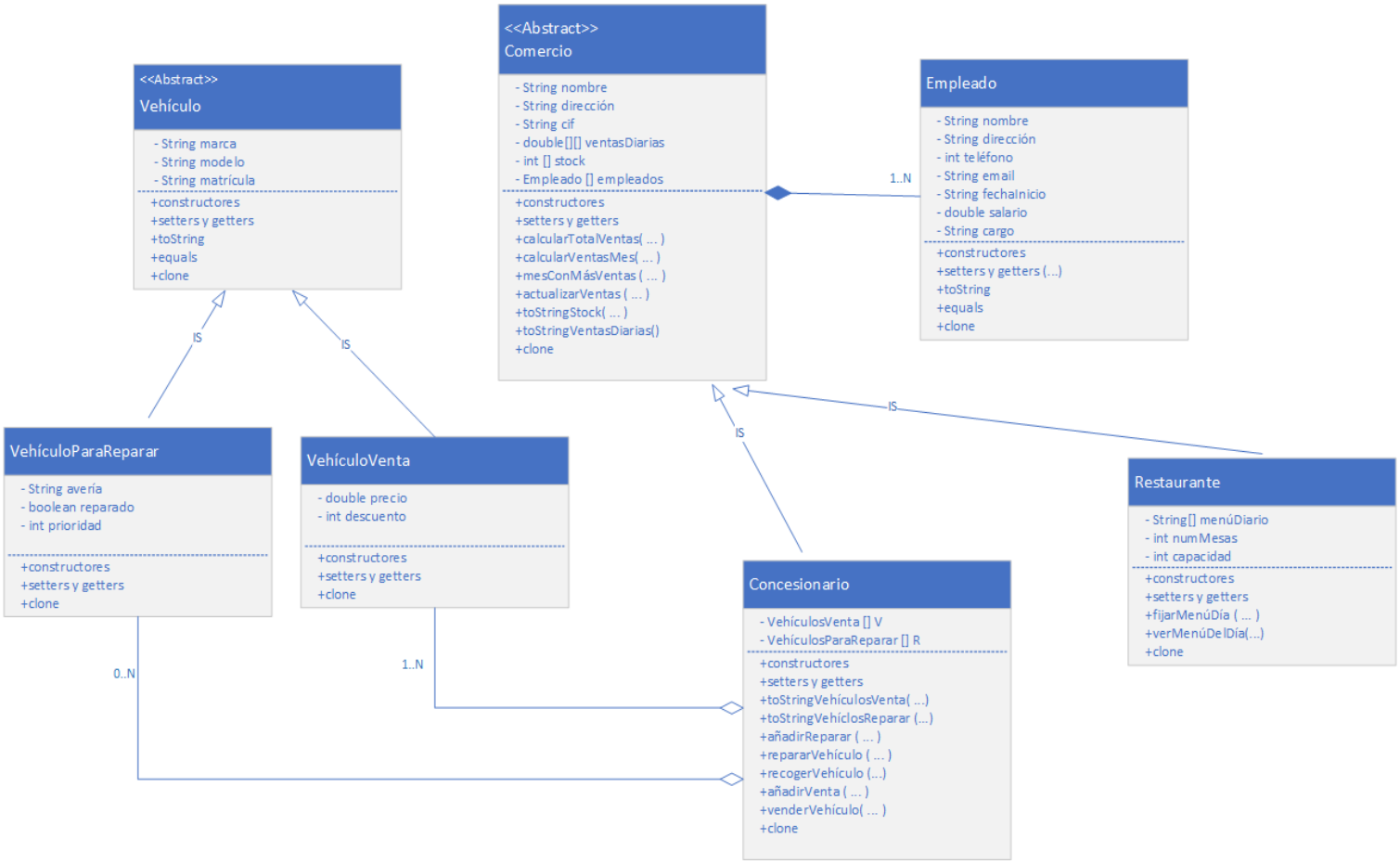

FIGURA 1. DIAGRAMA DE CLASES (UML)

Descripción de los atributos y métodos más relevantes:

- Clase Comercio
  - ventasDiarias: matriz de 12 x 31 donde en cada casilla se almacena el importe total de las ventas realizadas para cada día del mes.
  - stock : en cada casilla se almacena el stock para cada uno de los artículos del comercio.
  - *calcularTotalVentas*: devuelve la suma de todas las ventas realizadas por el comercio a partir de la matriz *ventasDiarias*.
  - *calcularVentasMes:* dado un mes pasado como argumento, devuelve el total de ventas realizado en dicho mes.
  - *mesConMásVentas:* devuelve el mes en el que más ventas se han realizado.

  - *actualizarVentas:* a partir del día y el mes en curso, se actualizará la casilla correspondiente de la matriz *ventasDiarias* con una cantidad que se le pasa al método como argumento.
  - *duplicar:* hace una *deep copy* del *Comercio.*

- Clase Concesionario:
  - *añadirReparar:* añade un vehículo al vector *VehículosParaReparar* (ten en cuenta que los vehículos para reparar tienen una prioridad asignada y que el vector donde se almacenan **tiene que estar ordenado** en base a esa prioridad).
  - *repararVehículo:* dada una posición del vector, pone a true el atributo reparado del vehículo correspondiente.
  - *recogerVehículo:* dada una matrícula, busca en el vector de *VehículosParaReparar* si hay un vehículo con esa matrícula y si está reparado, lo devuelve y lo elimina.
  - *añadirVenta:* añade un vehículo al vector de vehículos para vender.
  - *venderVehículo:* dada una posición del vector elimina el vehículo que ocupa dicha posición.

- Clase Restaurante:
  - *fijarMenúDelDía*: a partir de un String que contiene el menú del día y a partir del día de la semana, guarda en el vector menúDiario el menú en la posición correspondiente.
  - *verMenúDelDía*: devuelve un String con el menú de un día.

- Clase VehículoParaReparar:
  - *prioridad:* valor comprendido entre 1 y 3 donde el valor 1 se corresponde con los vehículos que tienen mayor prioridad a la hora de ser reparados.