TEMA-3.2.pdf

pabloo22

Programación

1º Grado en Ingeniería Informática

Escuela Politécnica Superior
Universidad de Alcalá

Universidad de Alcalá
Departamamento Ciencias de la Computación

Programación Orientada a Objetos

Tema 3:
Lenguaje POO Java

Tema 3-2: Ejemplo POO en Java

Contenidos

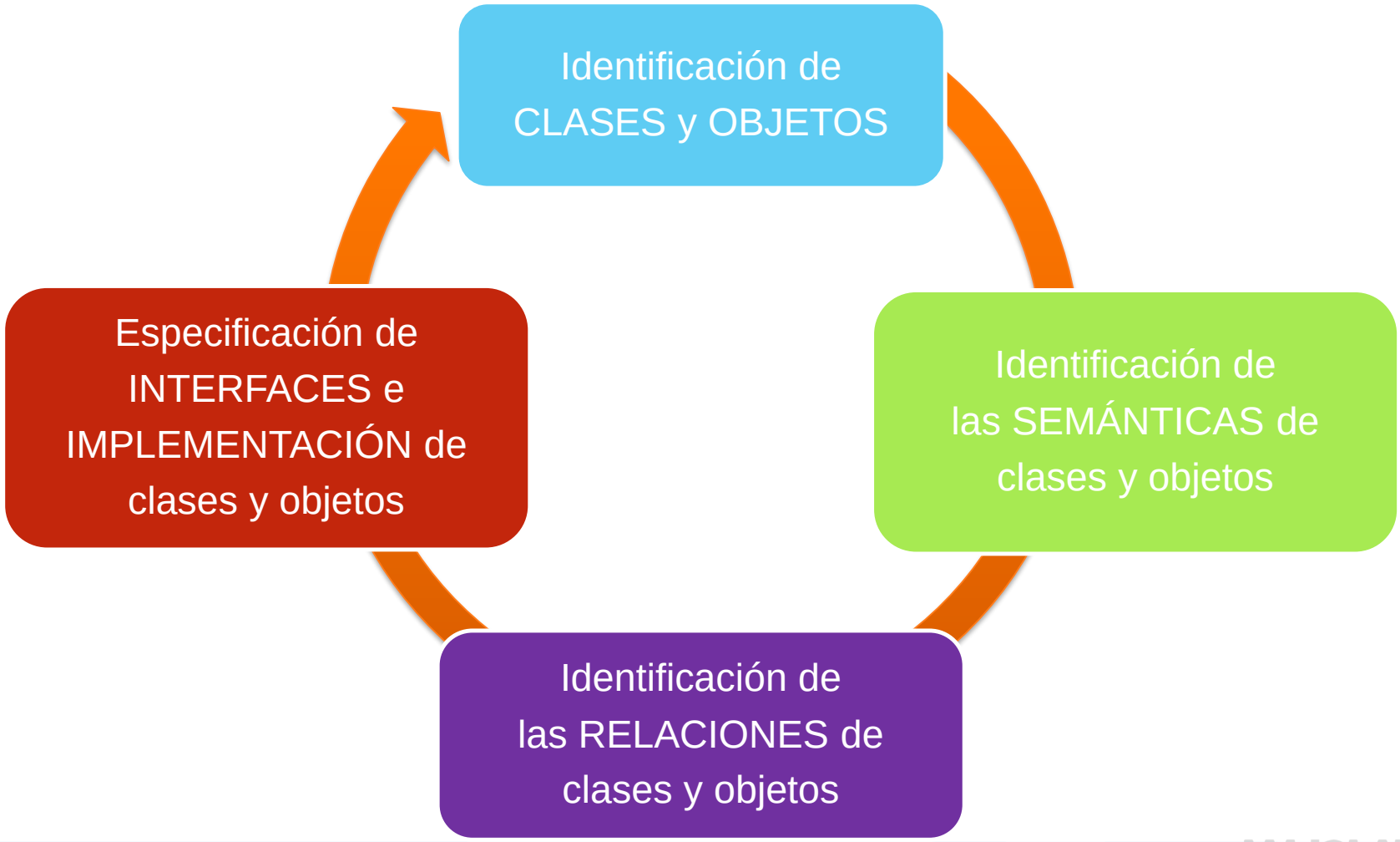
Tema 3-2: Ejemplo POO en Java

1. DISEÑO DE CLASES
2. EJEMPLO POO PEAJE
3. EJEMPLO POO CLASES CON HERENCIA
4. EJEMPLO POO CLASES ABSTRACTAS + HERENCIA + POLIMORFISMO

DISEÑO DE CLASES

DESCRIPCIÓN DEL PROBLEMA.	Describe el sistema que debemos modelar. Sería imposible ofrecer una solución satisfactoria sin un total conocimiento del problema.
ENCONTRAR LOS OBJETOS PRINCIPALES.	Buscar los elementos más importantes del modelo. Recomendación: Hacer una lista de todos los nombres que aparecen en la descripción del problema y elegir aquellos que nos parezcan más importantes.
DETERMINAR EL COMPORTAMIENTO DESEADO PARA CADA UNO DE LOS OBJETOS PRINCIPALES.	Este paso producirá el conjunto de métodos necesarios en las clases a las que esos objetos pertenecen. Si necesitamos varias clases, las trataremos una a una.
DETERMINAR LA INTERFAZ.	Estableceremos el prototipo de cada método: sus argumentos y tipo de retorno. Recomendación: Escribir un código de ejemplo que utilice el objeto y que muestre cómo deberían ocurrir las invocaciones de una forma natural.
DEFINIR LOS ATRIBUTOS E IMPLEMENTAR LOS MÉTODOS.	Concluiremos con una ***justificación*** de la solución propuesta.
DETERMINAR LAS RELACIONES CON OTRAS CLASES.	Analizar las relaciones entre clases para resolver el problema planteado.

DISEÑO DE CLASES



EJEMPLO POO PEAJE

DESCRIPCIÓN DEL PROBLEMA.

Definición del problema

1. El organismo encargado de las autopistas de un determinado país está instalando un sistema de Cobro de Peajes en una de sus carreteras más importantes.

2. Los camiones que llegan a una cabina de peaje deben pagar 5€ por eje y 2€ por cada tonelada de peso.

3. Cada cabina tiene un identificador único y tiene un agente encargado de su funcionamiento. Cada agente se identificará mediante su DNI y su nombre.

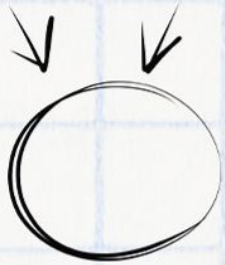
4. Una pantalla en la cabina de peaje muestra la cantidad del total de recibos cobrados y la cantidad de camiones que han pagado desde la última recaudación.

Imagínate aprobando el examen

Necesitas tiempo y concentración

Planes	PLAN TURBO	PLAN PRO	PLAN PRO+
Descargas sin publi al mes	10	40	80
Elimina el video entre descargas	✓	✓	✓
Descarga carpetas	✗	✓	✓
Descarga archivos grandes	✗	✓	✓
Visualiza apuntes online sin publi	✗	✓	✓
Elimina toda la publi web	✗	✗	✓
Precios Anual	0,99 € / mes	3,99 € / mes	7,99 € / mes

Ahora que puedes conseguirlo, ¿Qué nota vas a sacar?



WUOLAH

Oferta limitada en el tiempo. Los planes están sujetos a los términos y condiciones que puedes consultar en **wuolah.com/terms**

EJEMPLO POO PEAJE

DESCRIPCIÓN DEL PROBLEMA.

Un Escenario de Ejemplo

Para ayudar a diseñar el sistema, imaginemos como funcionaría un sistema de cobro de peaje de esta naturaleza:

1. Un agente de peajes espera en la cabina de peaje, la cual posee una pantalla donde se muestra información. Cuando llega un camión, el agente comprueba sus datos.
2. La información del camión y el importe del peaje se muestra en la pantalla del ordenador, por ejemplo:

   ```
   Camion{ejes = 3, pesoTotal = 10000} → Peaje: 35€
   ```

3. Al pulsar el botón junto a la pantalla, se muestran los totales para esa cabina de peaje:

   ```
   Totales desde la última recogida → Peaje: 79€ - Camiones: 2
   ```

4. Cuando se realiza la recaudación se muestra el siguiente mensaje y se ponen todas las cantidades a cero:

   ```
   **** Ejecutando recaudación ****
   Totales desde la última recogida → Peaje: 550€ - Camiones: 5
   ```

ENCONTRAR LOS OBJETOS PRINCIPALES.

Encontrar los objetos principales

- Podemos encontrar los objetos del problema buscando los nombres más significativos dentro de las frases que forman la definición del problema.

  En nuestro caso son : camiones, eje, peso, cabina, identificador, agente, dni, nombre y recibo.

- De ellos, las cabinas, los camiones y los agentes parecen ser los más importantes. El resto de elementos son propiedades secundarias.
- De esta forma tomamos los objetos principales y diseñamos las clases relacionadas:

```
public class Camion {
}
public class CabinaPeaje {
}
public class Agente {
}
```

EJEMPLO POO PEAJE

DETERMINAR EL COMPORTAMIENTO DESEADO PARA CADA UNO DE LOS OBJETOS PRINCIPALES.

Clase Camion

1. Determinar el comportamiento deseado

a) El peaje depende del número de ejes y el peso del camión. Las cabinas necesitan estos datos de cada camión.

b) Nuestra clase camión debería ofrecer métodos que nos dieran esta información.

c) También tendrá un constructor que nos permitirá crear camiones.

Métodos:

- **`Camion`** `(constructor)`
- **`toString`**
- **`getEjes`**
- **`getPesoTotal`**

EJEMPLO POO PEAJE

DETERMINAR LA INTERFAZ.

Clase Camion

2. Definición de la interfaz

a) La interfaz de la clase camión la componen las signaturas de su conjunto de métodos. Determina cómo otras clases se comunican con un objeto **`Camion`**.

b) Una buena forma de definir una interfaz es realizar un trozo de código que use un objeto de la clase que se está diseñando.

Por ejemplo:

```
Camion camion1 = new Camion(3, 10000)
```

c) Representaría un camión con 3 ejes y 10.000 Kg. de peso.

d) Preguntar cuantos ejes y peso tiene un camión es fácil:

```
camion1.getEjes()        camion1.getPesoTotal()
```

e) Por lo tanto podremos definir el interfaz de la clase como sigue:

```
public class Camion {
    //Métodos
    public Camion(int ejes, int pesoTotal) {...}
    public int getEjes() {...}
    public int getPesoTotal() {...}
    public String toString() {...}
    //Atributos...
}
```

EJEMPLO POO PEAJE

DEFINIR LOS ATRIBUTOS

Clase Camion

3. Definir los atributos

a) Los valores de los ejes y el peso deben ir asociado a cada camión por lo tanto:

```
public class Camion {
   //Atributos
   private int ejes;
   private int pesoTotal;
   //Métodos
   public Camion(int ejes, int pesoTotal) {...}
   public int getEjes() {...}
   public int getPesoTotal() {...}
   public String toString() {...}
}
```

EJEMPLO POO PEAJE

IMPLEMENTAR LOS MÉTODOS

Clase Camion

4. Implementar los métodos

```
public class Camion {
    //Atributos
    private int ejes;
    private int pesoTotal;

    //Métodos
    public Camion(int ejes, int pesoTotal) {
        this.ejes = ejes;
        this.pesoTotal = pesoTotal;
    }
    public int getEjes() {
        return this.ejes;
    }
    public int getPesoTotal() {
        return this.pesoTotal;
    }
    @Override
    public String toString() {
        return "Camion{" + "ejes = " + ejes + ", pesoTotal = " + pesoTotal + '}';
    }
}
```

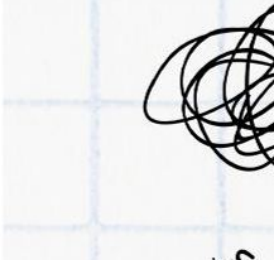
EJEMPLO POO PEAJE

DETERMINAR LAS RELACIONES CON OTRAS CLASES.

Clase Agente

El objetivo de la clase Agente es representar a los trabajadores de las Cabinas de Peaje. Necesitamos almacenar sus datos para identificarlo correctamente.

```
public class Agente {
    //Atributos
    private String dni;
    private String nombre;
    //Métodos
    public Agente(String dni, String nombre) {
        this.dni = dni;    this.nombre = nombre;
    }
    public String getDni() {
        return dni;
    }
    public String getNombre() {
        return nombre;
    }
    @Override
    public String toString() {
        return "Agente{" + "dni = " + dni + ", nombre = " + nombre + '}';
    }
}
```



Clase CabinaPeaje

1. Determinar el comportamiento deseado

a) El principal comportamiento deseado para la clase **`CabinaPeaje`** es el cálculo del importe del peaje. Debe contar el total de recibos y de camiones que ha procesado. Se tiene que asociar con el agente que trabaja en esa cabina.

Métodos:

- **`CabinaPeaje`** `(constructor)`
- **`getAgente`**
- **`setAgente`**
- **`calculaPeaje`**
- **`muestraDatos`**
- **`recaudar`**

EJEMPLO POO PEAJE

Clase CabinaPeaje

2. Definición de la interfaz:

a) Para crear una nueva cabina deberemos utilizar el siguiente código:

```
Agente agente1 = new Agente("12345678Z", "Juan García");
CabinaPeaje cabina = new CabinaPeaje("C1", agente1);
```

b) Cada vez que se calcule un peaje:

```
cabina.calculaPeaje(camion1)
```

c) Para mostrar los datos del total recaudado hasta el momento:

```
cabina.muestraDatos()
```

d) Para realizar la recaudación:

```
cabina.recaudar()
```

e) Por lo tanto la interfaz de la clase sería la siguiente:

```
public class CabinaPeaje {
    //Métodos
    public CabinaPeaje(String id, Agente agente) {...}
    public Agente getAgente() {...}
    public void setAgente(Agente agente) {...}
    public void calculaPeaje(Camion camion) {...}
    public void recaudar() {...}
    public void muestraDatos() {...}
    //Atributos
    ...
}
```

EJEMPLO POO PEAJE

Clase CabinaPeaje

3. Definir los atributos

a) La cabina con un determinado identificador debe mantener el importe total de todos los recibos de peaje cobrados y el número de camiones desde la última recaudación. Tiene que saber que agente trabaja en esa cabina.

```
public class CabinaPeaje {
    //Atributos
    private String id;      //identificador
    private int total;      //total recaudado
    private int camiones;   //cantidad de camiones que pagan peaje
    private Agente agente; //el agente que trabaja en la cabina
}
```

EJEMPLO POO PEAJE

Clase CabinaPeaje

```
public class CabinaPeaje {

    //Atributos
    private String id;      //identificador de la cabina
    private int total;      //total recaudado en la cabina
    private int camiones;   //cantidad de camiones que pagan peaje en la cabina
    private Agente agente; //el agente que trabaja en la cabina

    //Métodos
    public CabinaPeaje(String id, Agente agente) {
        this.id = id;
        this.agente = agente;
        this.total = 0;
        this.camiones = 0;
    }

    public Agente getAgente() {
        return agente;
    }

    public void setAgente(Agente agente) {
        this.agente = agente;
    }
```

EJEMPLO POO PEAJE

Clase CabinaPeaje

```
    public void muestraDatos() {
        System.out.println("Cabina con id: " + this.id + " # " +
                           agente.toString());
        System.out.println("Totales desde la última recogida - Peaje: " +
                           this.total + "€ - Camiones: " + this.camiones);
    }

    public void recaudar() {
        System.out.println("**** Ejecutando recaudación ****");
        muestraDatos();
        this.total = 0;
        this.camiones = 0;
    }

    public void calculaPeaje(Camion camion) {
        int ejes = camion.getEjes();
        int pesoTotal = camion.getPesoTotal();
        int peaje = 5 * ejes + 2 * (pesoTotal / 1000);
        System.out.println(camion.toString() + " - Peaje: " + peaje + "€");
        this.camiones++;
        this.total += peaje;
    }
}
```

EJEMPLO POO PEAJE

La prueba de las clases

```
public class PruebaCabinaPeaje {
    public static void main(String args[]) {
        //Crea los agentes
        Agente agente1 = new Agente("12345678Z", "Juan García");
        Agente agente2 = new Agente("23456789D", "María Perez");
        //Crea la cabina
        CabinaPeaje cabina1 = new CabinaPeaje("C1", agente1);
        //Crea camiones
        Camion camion1 = new Camion(3, 10000);
        Camion camion2 = new Camion(4, 12500);
        //Cobra peajes
        cabina1.calculaPeaje(camion1);
        cabina1.muestraDatos();
        //Cambiamos al agente
        cabina1.setAgente(agente2);
        cabina1.calculaPeaje(camion2);
        cabina1.muestraDatos();
        //Recauda
        cabina1.recaudar();
        cabina1.muestraDatos();
    }
}
```

```
>java PruebaCabinaPeaje
Camion{ejes=3, pesoTotal=10000} - Peaje: 35€
Cabina con id: C1 # Agente{dni=12345678Z, nombre=Juan García}
Totales desde la última recogida - Peaje: 35€ - Camiones: 1
Camion{ejes=4, pesoTotal=12500} - Peaje: 44€
Cabina con id: C1 # Agente{dni=23456789D, nombre=María Perez}
Totales desde la última recogida - Peaje: 79€ - Camiones: 2
**** Ejecutando recaudación ****
Cabina con id: C1 # Agente{dni=23456789D, nombre=María Perez}
Totales desde la última recogida - Peaje: 79€ - Camiones: 2
Cabina con id: C1 # Agente{dni=23456789D, nombre=María Perez}
Totales desde la última recogida - Peaje: 0€ - Camiones: 0
```

CLASES CON HERENCIA

Cambio de Definición del problema (cambios respecto a la especificación de requisitos inicial)

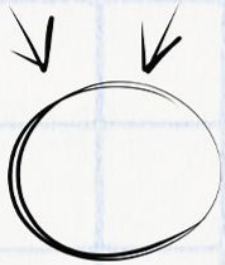
- El organismo encargado de las autopistas de un determinado país está instalando un sistema de cobro de peajes en una de sus carreteras más importantes.
- **Se debe cobrar peaje a dos tipos distintos de vehículos, los camiones y los autobuses de la siguiente forma:**
  - Los camiones que llegan a una cabina de peaje deben pagar 5€ por eje y 2€ por cada tonelada de peso y los autobuses deben pagar 1€ por pasajero y 5€ por cada tonelada de peso.
- Cada cabina tiene un identificador único y tiene un agente encargado de su funcionamiento. Cada agente se identificará mediante su DNI y su nombre.
- Una pantalla en la cabina de peaje muestra la cantidad del total de recibos cobrados y la cantidad de vehículos que han pagado desde la última recaudación.
- Cada vez que se realice un peaje **debe mostrar su importe así como la matrícula del vehículo y su peso para cualquier tipo de vehículo**. Si se trata de un camión debe mostrar además de esa información el **número de ejes** y si se trata de un autobús el **número de pasajeros**.
- También se quiere saber cuantos vehículos han pagado peaje en el total de las cabinas.

Imagínate aprobando el examen

Necesitas tiempo y concentración

Planes	PLAN TURBO	PLAN PRO	PLAN PRO+
Descargas sin publi al mes	10	40	80
Elimina el video entre descargas	✓	✓	✓
Descarga carpetas	✗	✓	✓
Descarga archivos grandes	✗	✓	✓
Visualiza apuntes online sin publi	✗	✓	✓
Elimina toda la publi web	✗	✗	✓
Precios Anual	0,99 € / mes	3,99 € / mes	7,99 € / mes

Ahora que puedes conseguirlo, ¿Qué nota vas a sacar?



WUOLAH

Oferta limitada en el tiempo. Los planes están sujetos a los términos y condiciones que puedes consultar en **wuolah.com/terms**

Encontrar los objetos principales

- Nombres significativos: vehículos, camiones, autobuses, matrícula, peso, eje, pasajeros, cabina, identificador, agente, dni, nombre y recibo.
- De ellos, las cabinas, los agentes, los vehículos, los camiones y los autobuses parecen ser los más importantes. Los demás son propiedades de los objetos que definamos.
- De las clases principales nos damos cuenta que hay una relación muy estrecha entre vehículos, camiones y autobuses, ya que comparten propiedades y métodos comunes → **Mecanismo de la Herencia**.

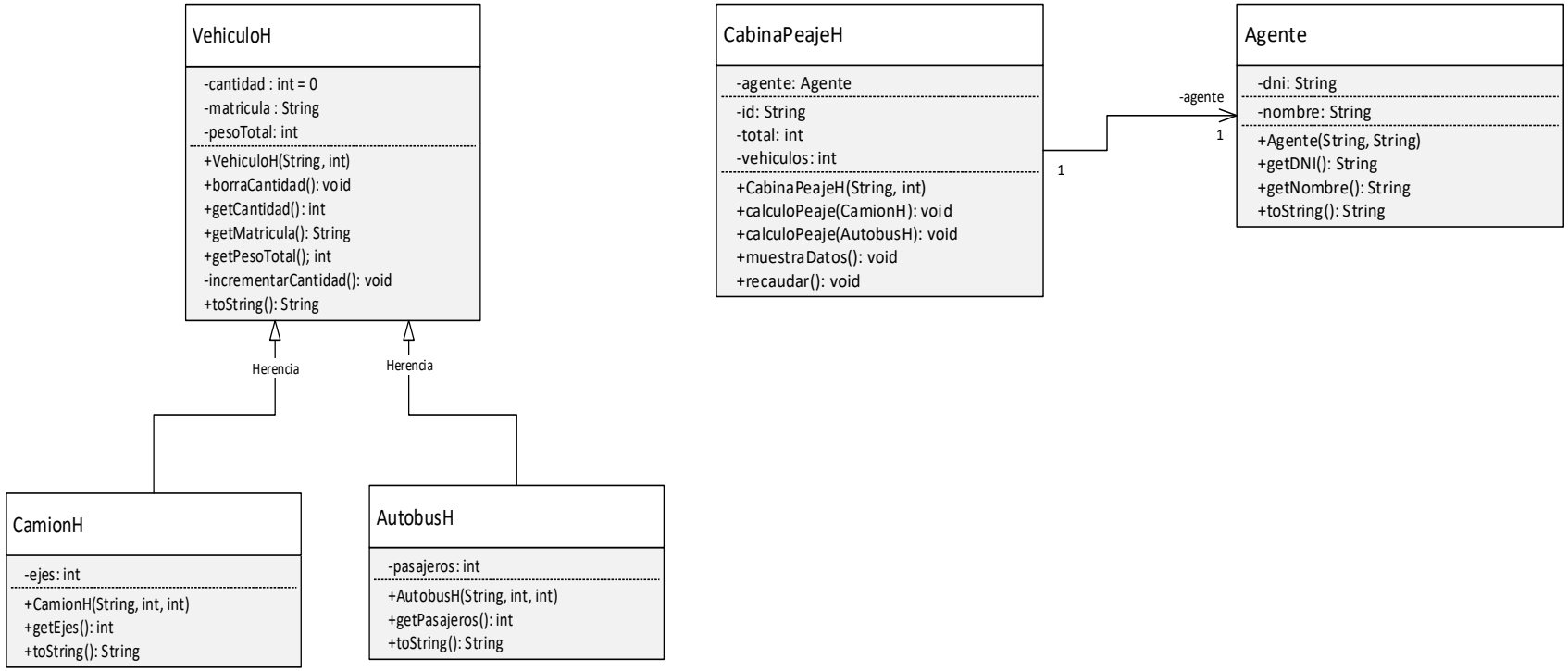
Encontrar los objetos principales

De esta forma tomamos los vehículos, los camiones, los autobuses, los agentes y las cabinas como objetos principales y diseñamos las clases relacionadas:

```
public class VehiculoH {
}
public class CamionH extends VehiculoH {
}
public class AutobusH extends VehiculoH {
}
public class Agente {
}
public class CabinaPeaje {

}
```

Encontrar los objetos principales

Diagrama de clases UML



CLASES CON HERENCIA

Clase VehiculoH:

```
public class VehiculoH
{
      //Atributos
      private String matricula;
      private int pesoTotal;
      //Cantidad de vehículos
      //que han pasado por el peaje
      private static int cantidad = 0;
      //Métodos
      public VehiculoH(String matricula,
                       int pesoTotal) {
            this.matricula = matricula;
            this.pesoTotal = pesoTotal;
            VehiculoH.incrementaCantidad();
      }

      private static void incrementaCantidad() {
            VehiculoH.cantidad++;
      }

      public String getMatricula() {
            return this.matricula;
      }

      public int getPesoTotal() {
            return this.pesoTotal;
      }

      public static int getCantidad() {
            return VehiculoH.cantidad;
      }
      public static void borraCantidad() {
            VehiculoH.cantidad = 0;
      }

      @Override
      public String toString() {
            return "Vehículo con matrícula: "
            +this.matricula+" Peso total: "
            +this.pesoTotal;
      }
}
```

Nota: Utilizamos una variable estática para saber el número de instancias de objetos de tipo VehiculoH.

CLASES CON HERENCIA

Clase CamionH

```
class CamionH extends VehiculoH {
     //Atributos
     private int ejes;
     //Métodos
     public CamionH(String matricula, int pesoTotal, int ejes) {
          super(matricula, pesoTotal);
          this.ejes = ejes;
     }
     public int getEjes() {
          return this.ejes;
     }
     public String toString() {
          String s = super.toString();
          return s + " # Camión - Ejes: " + this.ejes;
     }
}
```

CLASES CON HERENCIA

Clase AutobusH

```
class AutobusH extends VehiculoH {
    //Atributos
    private int pasajeros;
    //Métodos
    public AutobusH(String matricula, int pesoTotal, int pasajeros) {
        super(matricula, pesoTotal);
        this.pasajeros = pasajeros;
    }
    public int getPasajeros() {
        return this.pasajeros;
    }
    public String toString() {
        String s = super.toString();
        return s + " # Autobús - Pasajeros: " + this.pasajeros;
    }
}
```

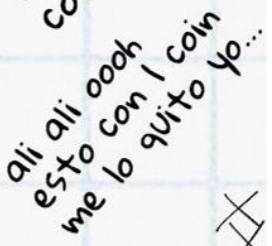
CLASES CON HERENCIA

Clase CabinaPeajeH

```
public class CabinaPeajeH {

    //Atributos
    private String id;      //identificador de la cabina
    private int total;      //total recaudado en la cabina
    private int vehiculos; //cantidad de vehículos que han pasado por esa cabina
    private Agente agente; //el agente que trabaja en la cabina

    //Métodos
    public CabinaPeajeH(String id, Agente agente) {
        this.id = id;
        this.agente = agente;
        this.total = 0;
        this.vehiculos = 0;
    }
    public void muestraDatos() {
        System.out.println("Cabina con id: " + this.id + " # " + agente.toString());
        System.out.println("Totales desde la última recogida - Peaje: " + this.total +
                           "€ - Vehículos: " + this.vehiculos);
    }
    public void recaudar() {
        System.out.println("**** Ejecutando recaudación ****");
        muestraDatos();
        this.total = 0;
        this.vehiculos = 0;
    }
```

CLASES CON HERENCIA

```
        //SOBRECARGA DE MÉTODOS
        public void calculaPeaje(CamionH camion)
        {
             int ejes = camion.getEjes();
             int pesoTotal = camion.getPesoTotal();
             int peaje = 5 * ejes + 2 *(pesoTotal/1000);
             System.out.println(camion.toString() + " - Peaje: " + peaje + "€" );
             this.vehiculos++;
             this.total += peaje;
        }

        public void calculaPeaje(AutobusH bus)
        {
             int pasajeros = bus.getPasajeros();
             int pesoTotal = bus.getPesoTotal();
             int peaje = 1 * pasajeros + 5 * (pesoTotal/1000);
             System.out.println(bus.toString() + " - Peaje: " + peaje + "€" );
             this.vehiculos++;
             this.total += peaje;
        }

}
```



CLASES CON HERENCIA

```
public class PruebaCabinaPeajeH {
    public static void main(String args[]) {
        //Crea el agente
        Agente agente1 = new Agente("12345678Z", "Juan García");
        Agente agente2 = new Agente("23456789D", "María Perez");
        //Crea las cabinas
        CabinaPeajeH cabina1 = new CabinaPeajeH("C1", agente1);
        CabinaPeajeH cabina2 = new CabinaPeajeH("C2", agente2);
        //Crea camiones
        CamionH camion1 = new CamionH("1234-BCD", 10000, 3);
        CamionH camion2 = new CamionH("2345-CDF", 15500, 4);
        //Crea autobuses
        AutobusH bus1 = new AutobusH("5678-CFG", 7000, 35);
        AutobusH bus2 = new AutobusH("6789-DTD", 8000, 50);
        //Cobra peajes
        cabina1.calculaPeaje(camion1);
        cabina1.calculaPeaje(bus1);
        cabina1.muestraDatos();
        cabina2.calculaPeaje(camion2);
        cabina2.calculaPeaje(bus2);
        cabina2.muestraDatos();
        //Recauda
        cabina1.recaudar();
        cabina1.muestraDatos();
        cabina2.recaudar();
        cabina2.muestraDatos();
        System.out.println("Total vehículos de todas las cabinas: " + VehiculoH.getCantidad());
        VehiculoH.borraCantidad();
    }
}
```

CLASES CON HERENCIA

```
>java PruebaCabinaPeajeH
Vehículo con matrícula: 1234-BCD Peso total: 10000 # Camión - Ejes: 3 - Peaje: 35€
Vehículo con matrícula: 5678-CFG Peso total: 7000 # Autobús - Pasajeros: 35 - Peaje: 70€
Cabina con id: C1 # Agente{dni=12345678Z, nombre=Juan García}
Totales desde la última recogida - Peaje: 105€ - Vehículos: 2
Vehículo con matrícula: 2345-CDF Peso total: 15500 # Camión - Ejes: 4 - Peaje: 50€
Vehículo con matrícula: 6789-DTD Peso total: 8000 # Autobús - Pasajeros: 50 - Peaje: 90€
Cabina con id: C2 # Agente{dni=23456789D, nombre=María Perez}
Totales desde la última recogida - Peaje: 140€ - Vehículos: 2
**** Ejecutando recaudación ****
Cabina con id: C1 # Agente{dni=12345678Z, nombre=Juan García}
Totales desde la última recogida - Peaje: 105€ - Vehículos: 2
Cabina con id: C1 # Agente{dni=12345678Z, nombre=Juan García}
Totales desde la última recogida - Peaje: 0€ - Vehículos: 0
**** Ejecutando recaudación ****
Cabina con id: C2 # Agente{dni=23456789D, nombre=María Perez}
Totales desde la última recogida - Peaje: 140€ - Vehículos: 2
Cabina con id: C2 # Agente{dni=23456789D, nombre=María Perez}
Totales desde la última recogida - Peaje: 0€ - Vehículos: 0
Total vehículos de todas las cabinas: 4
```

CLASES ABSTRACTAS + HERENCIA + POLIMORFISMO

Cambio en el diseño de la aplicación

En el ejemplo anterior hemos creado una jerarquía de clases para los distintos tipos de vehículos y la clase **`CabinaPeajeH`** se encargaba de calcular el peaje de los distintos tipos de vehículos a través de la sobrecarga de métodos.

- Sin embargo se puede mejorar el diseño anterior haciendo uso de las clases **ABSTRACTAS** y el **POLIMORFISMO**.

- Si nos damos cuenta la clase **`CabinaPeajeH`** necesita capturar los datos de los vehículos para hacer el cálculo del peaje.

  Ya que los vehículos tienen los datos para calcular su peaje, **¿No sería más lógico preguntarle al vehículo cuanto nos tiene que pagar?**

CLASES ABSTRACTAS + HERENCIA + POLIMORFISMO

Cambio en el diseño de la aplicación

Para solucionar el planteamiento anterior hay que realizar dos cambios en el diseño de las clases:

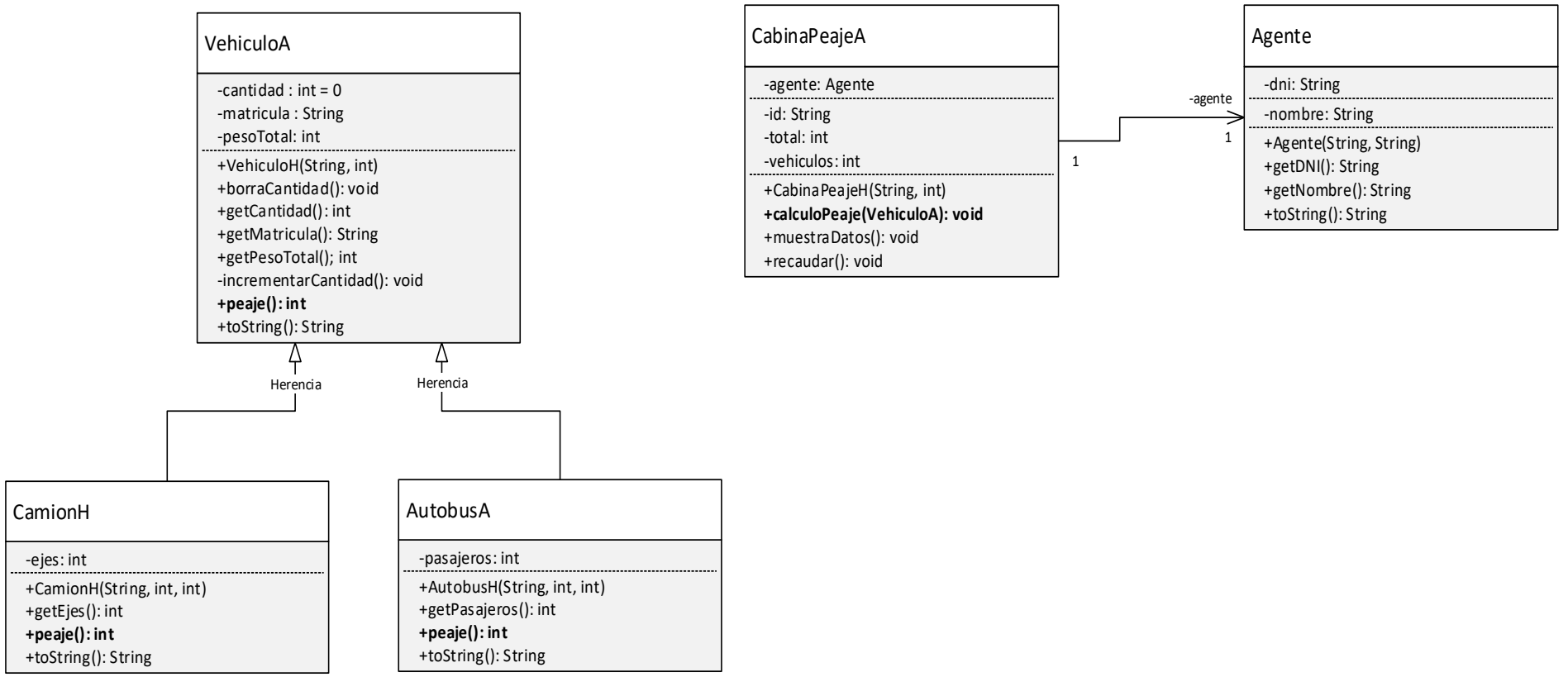
1. Modificar la clase **`VehiculoH`** a clase ABSTRACTA de forma que defina el método abstracto peaje para que las clases hijas lo implementen.
2. Modificar la clase **`CabinaPeajeH`** de forma que implemente un método POLIMÓRFICO para cobrar el peaje a los distintos tipos de vehículos.

```
public abstract class VehiculoA {
      public abstract int peaje();
}
public class CamionA extends VehiculoA {
      public int peaje() { ... }
}
public class AutobusA extends VehiculoA {
      public int peaje() { ... }
}
public class CabinaPeaje {
      public void calculaPeaje(VehiculoA va) {
                  int peaje = va.peaje(); ...
      }
}
```

CLASES ABSTRACTAS +HERENCIA + POLIMORFISMO

Clases del problema

Diagrama de clases UML :



CLASES ABSTRACTAS + HERENCIA + POLIMORFISMO

```java
public abstract class VehiculoA {
	//Atributos
	private String matricula;
	private int pesoTotal;
	//Cantidad de vehículos que han pasado por
	//el peaje
	private static int cantidad = 0;
	//Métodos
	public VehiculoA(String matricula,
			int pesoTotal) {
		this.matricula = matricula;
		this.pesoTotal = pesoTotal;
		VehiculoA.incrementaCantidad();
	}
	private static void incrementaCantidad() {
		VehiculoA.cantidad++;
	}
	public String getMatricula() {
		return this.matricula;
	}
	public int getPesoTotal() {
		return this.pesoTotal;
	}
	public static int getCantidad() {
		return VehiculoA.cantidad;
	}
	public static void borraCantidad() {
		VehiculoA.cantidad = 0;
	}
	public String toString() {
		return "Vehículo con matrícula: "
		+ this.matricula + " Peso total: "
		+ this.pesoTotal;
	}
	//Método Abstracto
	public abstract int peaje();
}
```

```java
public  class CamionA extends VehiculoA {
    private int ejes;
    public CamionA(String matricula, int pesoTotal, int ejes){
            super(matricula, pesoTotal);
            this.ejes=ejes;
    }
    public int getEjes() { return this.ejes; }
    public String toString() {String s = super.toString();
            return s + " # Camión - Ejes: " + this.ejes;
    }
    @Override
    public int peaje() {
       return 5 * this.ejes + 2 *(super.getPesoTotal()/1000);
    }
}
```

```java
public class AutobusA extends VehiculoA {
    private int pasajeros;
    public AutobusA(String matricula, int pesoTotal,
                    int pasajeros) {
            super(matricula, pesoTotal);
            this.pasajeros = pasajeros;
    }
    public int getPasajeros() { return this.pasajeros; }
    public String toString() {
            String s = super.toString();
            return s+" # Autobús - Pasajeros: "+ this.pasajeros;
    }
    @Override
    public int peaje() {
       return 1 *this.pasajeros + 5*(super.getPesoTotal()/1000);
    }
}
```

CLASES ABSTRACTAS + HERENCIA + POLIMORFISMO

```java
public class CabinaPeajeA {
    //Atributos
    private String id;      //identificador de la cabina
    private int total;      //total recaudado en la cabina
    private int vehiculos; //cantidad de vehículos que han pasado por esa cabina
    private Agente agente; //el agente que trabaja en la cabina
    //Métodos
    public CabinaPeajeA(String id, Agente agente) {
        this.id = id;
        this.agente = agente;
        this.total = 0;
        this.vehiculos = 0;
    }
    public void muestraDatos() {
        System.out.println("Cabina con id: " + this.id + " # " + agente.toString());
        System.out.println("Totales desde la última recogida - Peaje: " + this.total +
                           "€ - Vehículos: " + this.vehiculos);
    }
    public void recaudar() {
        System.out.println("**** Ejecutando recaudación ****");
        muestraDatos();
        this.total = 0;
        this.vehiculos = 0;
    }
    public void calculaPeaje(VehiculoA va) {
        int peaje = va.peaje();  //polimorfismo
        System.out.println(va.toString() + " - Peaje: " + peaje + "€");
        this.vehiculos++;
        this.total += peaje;
    }
}
```

CLASES ABSTRACTAS +HERENCIA + POLIMORFISMO

La prueba de las clases

```
public class PruebaCabinaPeajeA {
    public static void main(String args[]) {
        //Crea el agente
        Agente agente1 = new Agente("12345678Z", "Juan García");
        Agente agente2 = new Agente("23456789D", "María Perez");
        //Crea las cabinas
        CabinaPeajeA cabina1 = new CabinaPeajeA("C1", agente1);
        CabinaPeajeA cabina2 = new CabinaPeajeA("C2", agente2);
        //Crea distintos vehículos
        VehiculoA v1 = new CamionA("1234-BCD", 10000, 3);
        VehiculoA v2 = new CamionA("2345-CDF", 15500, 4);
        VehiculoA v3 = new AutobusA("5678-CFG", 7000, 35);
        VehiculoA v4 = new AutobusA("6789-DTD", 8000, 50);
        //Cobra peajes
        cabina1.calculaPeaje(v1);
        cabina1.calculaPeaje(v3);
        cabina1.muestraDatos();
        cabina2.calculaPeaje(v2);
        cabina2.calculaPeaje(v4);
        cabina2.muestraDatos();
        //Recauda
        cabina1.recaudar();
        cabina1.muestraDatos();
        cabina2.recaudar();
        cabina2.muestraDatos();
        System.out.println("Total vehículos de todas las cabinas: " +VehiculoA.getCantidad());
        VehiculoA.borraCantidad();
    }
}
```

CLASES ABSTRACTAS +HERENCIA + POLIMORFISMO

```
>java.exe PruebaCabinaPeajeA
Vehículo con matrícula: 1234-BCD Peso total: 10000 # Camión - Ejes: 3 - Peaje: 35€
Vehículo con matrícula: 5678-CFG Peso total: 7000 # Autobús - Pasajeros: 35 - Peaje: 70€
Cabina con id: C1 # Agente{dni=12345678Z, nombre=Juan García}
Totales desde la última recogida - Peaje: 105€ - Vehículos: 2
Vehículo con matrícula: 2345-CDF Peso total: 15500 # Camión - Ejes: 4 - Peaje: 50€
Vehículo con matrícula: 6789-DTD Peso total: 8000 # Autobús - Pasajeros: 50 - Peaje: 90€
Cabina con id: C2 # Agente{dni=23456789D, nombre=María Perez}
Totales desde la última recogida - Peaje: 140€ - Vehículos: 2
**** Ejecutando recaudación ****
Cabina con id: C1 # Agente{dni=12345678Z, nombre=Juan García}
Totales desde la última recogida - Peaje: 105€ - Vehículos: 2
Cabina con id: C1 # Agente{dni=12345678Z, nombre=Juan García}
Totales desde la última recogida - Peaje: 0€ - Vehículos: 0
**** Ejecutando recaudación ****
Cabina con id: C2 # Agente{dni=23456789D, nombre=María Perez}
Totales desde la última recogida - Peaje: 140€ - Vehículos: 2
Cabina con id: C2 # Agente{dni=23456789D, nombre=María Perez}
Totales desde la última recogida - Peaje: 0€ - Vehículos: 0
Total vehículos de todas las cabinas: 4
```