PROGRAMACIÓN ORIENTADA A OBJETOS
(Laboratorio de Prácticas)

Titulaciones de Grado en Ingeniería de Computadores, Ingeniería Informática y Sistemas de Información

Sesión 4
POO Básica en JAVA
(Soluciones a los ejercicios)

1. Ejercicio Lámpara (ejercicio obligatorio para II y SI, voluntario para IC)

Crear una clase Lámpara como representación de un sistema de iluminación. Nuestra lámpara es capaz de dar distintos niveles de iluminación. Para eso crearemos la clase con dos atributos y sus correspondientes métodos get y set:

- boolean encendida: que indicará si la luz está encendida o no.
- int intensidad: que indicara el nivel de luz (de 0 a 100 %).

Podemos cambiar la intensidad de la luz mediante el método setIntensidad, indicando la cantidad de luz que queremos recibir. También nos gustaría poder encender la luz indicando el voltaje. El voltaje será un número de tipo double con valores entre 1.5 y 12.5. Para eso crearemos el método: *public void setIntensidad(double voltaje)*. Si el parámetro voltaje es inferior a 1.5, la intensidad está al 0 %. Si el valor es mayor que 12.5, la intensidad está al 100%. Los otros valores del parámetro dependen del voltaje y se pueden calcular mediante una regla de 3.
Por último queremos conocer el estado actual de nuestra luz. Para realizar esto, sobrescribiremos el método toString() para que devuelva un mensaje de tipo: Luz: ON, Intensidad: 45%
Cuando creemos un objeto de tipo Lámpara, asignaremos a los atributos los valores por defecto:
encendida = false
intensidad = 0
Como podéis ver, en este ejercicio hemos sobrecargado el método setIntensidad con diferentes tipos de parámetros.
Codificar una clase con método main para probar que la sobrecarga funciona correctamente.

```
package lamparas;

public class Lampara {

    private boolean encendida;
    private int intensidad;

    public Lampara() {
        this.encendida = false;
        this.intensidad = 0;
    }

    public int getIntensidad() {
        return intensidad;
    }

    public void setIntensidad(int intensidad) {
        if (intensidad >= 0 && intensidad <=100) {
            this.intensidad = intensidad;
```

```
        } else {
            this.intensidad = 0;
        }
    }

    public void setIntensidad(double voltaje) {
        if (voltaje <= 1.5) {
            this.intensidad = 0;
        } else if (voltaje >= 12.5) {
            this.intensidad = 100;
        } else {
            this.intensidad = (int) (100/11.0 * (voltaje-1.5));
        }
    }

    public boolean isEncendida() {
        return encendida;
    }

    public void setEncendida(boolean encendida) {
        this.encendida = encendida;
    }

    @Override
    public String toString() {
        String luztxt = "";
        if (this.encendida) {
            luztxt = "ON";
        } else {
            luztxt = "OFF";
        }
        return "Luz: " + luztxt + ", , Intensidad: " + this.intensidad + "%";
    }
}
```

```
package lamparas;

public class PruebaLamparas {

    public static void main(String[] args) {
        Lampara l1 = new Lampara();
        Lampara l2 = new Lampara();
        l1.setEncendida(true);
        l1.setIntensidad(5.5);
        System.out.println("l1: " + l1.toString());
        l1.setIntensidad(50);
        System.out.println("l1: " + l1.toString());
        l2.setEncendida(true);
        l2.setIntensidad(1.5);
        System.out.println("l2: " + l2.toString());
    }
}
```

2. Ejercicio Nóminas

Se pretende modificar el proyecto anterior para que incluya la gestión de las nóminas de los empleados. Para realizarlo hay que crear una clase Nómina que se encargará de almacenar las nóminas de los empleados. Esta clase debe tener los siguientes atributos (tipo – nombre) con sus correspondientes métodos get y set (exceptuando el salarioNeto que solo tendrá método get):

1. Empresa – empresa.
2. Departamento – departamento.
3. Empleado – empleado.
4. GregorianCalendar – fecha.
5. double – salarioBruto.
6. double – retencion.
7. double – salarioNeto.

Realizar un constructor que reciba los siguientes parámetros: empresa, departamento, empleado y retención (representando un %). La fecha se extrae de la fecha actual del sistema. El salarioBruto se extrae del empleado mediante el método getSueldo(). En la instanciación de los objetos nómina se tendrá que realizar el cálculo del atributo salarioNeto mediante la resta al salario bruto de la retención. Crear también el método toString para que imprima la información de una nómina (se pueden utilizar los métodos toString de las clases asociadas para ir componiendo la cadena final).

Ya que a partir de la clase empleado se puede deducir la información del departamento y la empresa, realizar un nuevo constructor que solo reciba el empleado y la retención y extraer los otros datos del empleado.

Una vez realizada la clase Nómina crear una clase con un método main para probar la creación de varias nóminas de los empleados.

```
package empleados.nominas;

import empleados.Empresa;
import empleados.Departamento;
import empleados.Empleado;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;

public class Nomina {

    private Empresa empresa;
    private Departamento departamento;
    private Empleado empleado;
    private LocalDate fecha;
    private double salarioBruto;
    private double retencion;
    private double salarioNeto;

    public Nomina(Empresa empresa, Departamento departamento, Empleado empleado, double
retencion) {
        this.empresa = empresa;
        this.departamento = departamento;
        this.empleado = empleado;
        this.fecha = LocalDate.now();
        this.salarioBruto = empleado.getSueldo();
        this.retencion = retencion;
        this.salarioNeto = calculaSalarioNeto();
    }

    public Nomina(Empleado empleado, double retencion) {
        this.empleado = empleado;
        this.departamento = empleado.getDepartamento();
        this.empresa = empleado.getDepartamento().getEmpresa();
        this.fecha = LocalDate.now();
        this.salarioBruto = empleado.getSueldo();
        this.retencion = retencion;
        this.salarioNeto = calculaSalarioNeto();
    }

    private double calculaSalarioNeto() {
        return this.salarioBruto - (this.salarioBruto * (this.retencion / 100));
    }

    public Empresa getEmpresa() {
        return empresa;
    }

    public void setEmpresa(Empresa empresa) {
        this.empresa = empresa;
    }

    public Departamento getDepartamento() {
        return departamento;
    }

    public void setDepartamento(Departamento departamento) {
```

```
        this.departamento = departamento;
    }

    public Empleado getEmpleado() {
        return empleado;
    }

    public void setEmpleado(Empleado empleado) {
        this.empleado = empleado;
    }

    public LocalDate getFecha() {
        return fecha;
    }

    public void setFecha(LocalDate fecha) {
        this.fecha = fecha;
    }

    public String getFechaString() {
        DateTimeFormatter formatoCorto = DateTimeFormatter.ofPattern("dd/MM/yyyy");
        String hoystr = fecha.format(formatoCorto);
        return hoystr;
    }

    public double getSalarioBruto() {
        return salarioBruto;
    }

    public void setSalarioBruto(double salarioBruto) {
        this.salarioBruto = salarioBruto;
        this.calculaSalarioNeto();
    }

    public double getRetencion() {
        return retencion;
    }

    public void setRetencion(double retencion) {
        this.retencion = retencion;
        this.calculaSalarioNeto();
    }

    public double getSalarioNeto() {
        return salarioNeto;
    }

    @Override
    public String toString() {
        return "*** Nómina ***\n " + this.empleado + " \n " + this.empresa + " \n " +
this.getFechaString() + " \n " + this.salarioNeto;
    }
}
```

```
package empleados.nominas;

import empleados.Departamento;
import empleados.Empleado;
import empleados.Empresa;

public class PruebaNominas {

    public static void main(String[] args) {
        //1 - creamos la empresa
        Empresa e1 = new Empresa("Indra", "1234567");
        //2 - creamos el departamento
        Departamento d1 = new Departamento("Informática", "1", "Madrid", e1);
        //3 - creamos los empleados que asignamos al departamento
        Empleado emp1 = new Empleado("1234", "Pepe", 25, "soltero", 1500, "programador", d1);
        Empleado emp2 = new Empleado("4567", "Laura", 35, "casada", 2000, "analista", d1);
        //4 - creamos las nóminas de los empleados
        Nomina n1 = new Nomina(e1, d1, emp1, 20);
        Nomina n2 = new Nomina(e1, d1, emp2, 25);
```

```
        System.out.println(n1.toString());
        System.out.println(n2.toString());
    }

}
```

3. Ejercicio Taller

Se desea realizar una aplicación que simule un taller donde se reparan vehículos. El taller debe permitir el cálculo de la reparación de un vehículo en base al coste de las piezas utilizadas y las horas dedicadas a la reparación. La aplicación debe tener tres clases: Taller, Vehículo y Pieza:

- La clase Pieza representa una pieza del vehículo reparado y contendrá su nombre (String) y su precio (double).
- El Vehículo contendrá la matrícula (String), marca (String), modelo (String) y un ArrayList con las piezas que se han arreglado.
- La clase Taller tendrá un nombre (String), teléfono (String) y el precio hora (double). El taller debe tener un método repararVehiculo() que recibirá dos parámetros: un objeto de tipo vehículo y las horas dedicadas a la reparación; a partir del objeto vehículo sabremos las piezas utilizadas y su precio y con el coste de la hora debe ser capaz de devolver el importe total de la reparación.

Crear las clases y los métodos necesarios para poder implementar el método repararVehiculo() y una clase PruebaTaller con método main para poder probarlas.

```
package taller;

public class Pieza {

    private String nombre;
    private double precio;

    //constructor
    public Pieza(String nombre, double precio) {
        this.nombre = nombre;
        this.precio = precio;
    }

    public double getPrecio() {
        return precio;
    }

    public void setPrecio(double precio) {
        this.precio = precio;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    @Override
    public String toString() {
        return "Pieza{" + "nombre=" + nombre + ", precio=" + precio + '}';
    }
}
```

```
package taller;

import java.util.*;

public class Vehiculo {

    private String matricula;
```

```
    private String marca;
    private String modelo;
    private ArrayList<Pieza> piezas;

    //constructor
    public Vehiculo(String matricula, String marca, String modelo) {
        this.matricula = matricula;
        this.marca = marca;
        this.modelo = modelo;
        this.piezas = new ArrayList<>();
    }

    public ArrayList<Pieza> getPiezas() {
        return piezas;
    }

    public void añadirPieza(Pieza pieza) {
        this.piezas.add(pieza);
    }

    public String getModelo() {
        return modelo;
    }

    public void setModelo(String modelo) {
        this.modelo = modelo;
    }

    public String getMarca() {
        return marca;
    }

    public void setMarca(String marca) {
        this.marca = marca;
    }

    public String getMatricula() {
        return matricula;
    }

    public void setMatricula(String matricula) {
        this.matricula = matricula;
    }

    @Override
    public String toString() {
        return "Vehiculo{" + "matricula=" + matricula + ", marca=" + marca + ", modelo=" +
modelo + ", piezas=" + piezas + '}';
    }
}
```

```
package taller;

public class Taller {

    private String nombre;
    private String telefono;
    private double precioHora;

    //constructor
    public Taller(String nombre, String telefono, double precioHora) {
        this.nombre = nombre;
        this.telefono = telefono;
        this.precioHora = precioHora;
    }

    public double getPrecioHora() {
        return precioHora;
    }

    public void setPrecioHora(double precioHora) {
        this.precioHora = precioHora;
    }
```

```
    public String getTelefono() {
        return telefono;
    }

    public void setTelefono(String telefono) {
        this.telefono = telefono;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public double repararVehiculo(Vehiculo car, double horas) {
        double precio = 0;
        precio += horas * this.precioHora;
        for (int i = 0; i < car.getPiezas().size(); i++) {
            precio += car.getPiezas().get(i).getPrecio();
        }
        return precio;
    }

    @Override
    public String toString() {
        return "Taller{" + "nombre=" + nombre + ", telefono=" + telefono + ", precioHora=" +
precioHora + '}';
    }
}
```

```
package taller;

public class PruebaTaller {

    public static void main(String[] args) {
        Taller t1 = new Taller("T1", "918873642", 15.50);
        Taller t2 = new Taller("T2", "968147850", 20.30);
        Taller t3 = new Taller("T3", "932876642", 30.20);
        Pieza p1 = new Pieza("Correa de distribución", 500.0);
        Pieza p2 = new Pieza("Carburador", 400.0);
        Pieza p3 = new Pieza("Discos de freno", 150.0);
        Pieza p4 = new Pieza("Zapata", 20.0);
        Pieza p5 = new Pieza("Tubo de escape", 250.0);

        Vehiculo v1 = new Vehiculo("1773BTB", "Opel", "Corsa");
        Vehiculo v2 = new Vehiculo("9234CTC", "Ford", "Focus");
        Vehiculo v3 = new Vehiculo("4381CVB", "Mazda", "CX-5");
        Vehiculo v4 = new Vehiculo("4952GZV", "Nissan", "Juke");

        v1.añadirPieza(p4);
        v1.añadirPieza(p2);
        v2.añadirPieza(p1);
        v2.añadirPieza(p2);
        v2.añadirPieza(p3);
        v2.añadirPieza(p4);
        v3.añadirPieza(p3);
        v4.añadirPieza(p5);

        System.out.println("Coche 1, Taller 2, Reparación: " + t2.repararVehiculo(v1, 5.0));
        System.out.println("Coche 1, Taller 3, Reparación: " + t3.repararVehiculo(v1, 5.0));
        System.out.println("Coche 2, Taller 1, Reparación: " + t1.repararVehiculo(v2, 10.0));
        System.out.println("Coche 3, Taller 3, Reparación: " + t3.repararVehiculo(v3, 5.0));
        System.out.println("Coche 4, Taller 2, Reparación: " + t2.repararVehiculo(v4, 3.0));
    }
}
```

4. Ejercicio Aparcamiento (ejercicio obligatorio para II y SI, voluntario para IC)

Se pretende realizar una aplicación en Java que simule un aparcamiento donde se pueden estacionar vehículos por un tiempo limitado. Existen dos tipos de vehículos que pueden estacionar en el aparcamiento: automóviles y camiones. De todos los vehículos hay que guardar su matrícula, la fecha/hora de entrada en el aparcamiento y si se trata de un vehículo con abono, de los automóviles el tipo: turismo, todoterreno y furgoneta, y de los camiones el número de ejes.

La clase Vehículo tendrá un método abstracto llamado calcularImporte() que tendrán que implementar las clases hijas (polimorfismo) y que será el encargado de determinar los minutos que han transcurrido y mediante la siguiente fórmula calculará el importe del tiempo de aparcamiento:

- Automóvil:
  - turismo → minutos * 1.5 € / 60
  - todoterreno → minutos * 2.5 € / 60
  - furgoneta → minutos * 3.5 € / 60
- Camión:
  - Ejes ≤ 3 → minutos * 4.5 € / 60
  - Ejes > 3 → minutos * 6.5 € / 60

Si se trata de un vehículo con abono se le aplicará un descuento del 40% sobre el importe final.

Hay que desarrollar una clase Aparcamiento que se encargará de almacenar los vehículos que tiene estacionados, para hacerlo usará un ArrayList de vehículos. El aparcamiento tiene un número de plazas determinada que se establece a través de un atributo denominado capacidad (int). La clase contará con los siguientes métodos:

- intoducirVehiculo(Vehiculo v): Se encargará de añadir un nuevo vehículo al aparcamiento comprobando que no está dentro y que hay capacidad suficiente. También decrementará la capacidad del aparcamiento.
- sacarVehiculo(Vehiculo v) / sacarVehiculo(String matricula): Se encargará de sacar un vehículo del aparcamiento comprobando que está dentro y calculará cuanto tiene que cobrar haciendo uso del método calcularImporte() del vehículo correspondiente. También incrementará la capacidad del aparcamiento. Se realizarán dos versiones de este método utilizando sobrecarga, de esta forma se podrá sacar el vehículo con el objeto vehículo o mediante su matrícula.

Por último, para poder probar las clases desarrolladas crear una clase con método main para poder instanciar un conjunto de vehículos, introducirlos en el aparcamiento y sacarlos para ver los importes de estacionamiento.

```
package aparcamiento;

import java.time.LocalDateTime;
import java.util.Objects;

public abstract class Vehiculo {

    private String matricula;
    private LocalDateTime fechaEntrada;
    private boolean abono;

    public Vehiculo(String matricula, boolean abono) {
        this.matricula = matricula;
        this.abono = abono;
        this.fechaEntrada = LocalDateTime.now();
```

```
    }

    public Vehiculo(String matricula, LocalDateTime fechaEntrada, boolean abono) {
        this.matricula = matricula;
        this.fechaEntrada = fechaEntrada;
        this.abono = abono;
    }

    public String getMatricula() {
        return matricula;
    }

    public void setMatricula(String matricula) {
        this.matricula = matricula;
    }

    public LocalDateTime getFechaEntrada() {
        return fechaEntrada;
    }

    public void setFechaEntrada(LocalDateTime fechaEntrada) {
        this.fechaEntrada = fechaEntrada;
    }

    public boolean isAbono() {
        return abono;
    }

    public void setAbono(boolean abono) {
        this.abono = abono;
    }

    public abstract double calcularImporte();

    @Override
    public int hashCode() {
        int hash = 7;
        hash = 31 * hash + Objects.hashCode(this.matricula);
        return hash;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj) {
            return true;
        }
        if (obj == null) {
            return false;
        }
        if (getClass() != obj.getClass()) {
            return false;
        }
        final Vehiculo other = (Vehiculo) obj;
        if (!Objects.equals(this.matricula, other.matricula)) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "Vehiculo{" + "matricula=" + matricula + ", fechaEntrada=" + fechaEntrada + ",
abono=" + abono + '}';
    }

}
```

```
package aparcamiento;

import java.time.LocalDateTime;
import java.time.temporal.ChronoUnit;
```

```
public class Automovil extends Vehiculo {

    private String tipo;

    public Automovil(String matricula, boolean abono, String tipo) {
        super(matricula, abono);
        this.tipo = tipo;
    }

    public Automovil(String matricula, LocalDateTime fechaEntrada, boolean abono, String
tipo) {
        super(matricula, fechaEntrada, abono);
        this.tipo = tipo;
    }

    public String getTipo() {
        return tipo;
    }

    public void setTipo(String tipo) {
        this.tipo = tipo;
    }

    @Override
    public double calcularImporte() {
        LocalDateTime fechaSalida = LocalDateTime.now();
        long minutos = ChronoUnit.MINUTES.between(this.getFechaEntrada(), fechaSalida);
        double tasa = 0, total = 0;

        switch (tipo) {
            case "turismo":
                tasa = 1.5;
                break;
            case "todoterreno":
                tasa = 2.5;
                break;
            case "furgoneta":
                tasa = 3.5;
                break;
        }

        total = minutos * tasa / 60;

        if (this.isAbono()) {
            total -= (total * 0.4);
        }

        return total;
    }

    @Override
    public String toString() {
        return super.toString() + " # Automovil{" + "tipo=" + tipo + '}';
    }

}
```

```
package aparcamiento;

import java.time.LocalDateTime;
import java.time.temporal.ChronoUnit;

public class Camion extends Vehiculo {

    private int ejes;

    public Camion(String matricula, boolean abono, int ejes) {
        super(matricula, abono);
        this.ejes = ejes;
    }

    public Camion(String matricula, LocalDateTime fechaEntrada, boolean abono, int ejes) {
        super(matricula, fechaEntrada, abono);
        this.ejes = ejes;
```

```
    }

    public int getEjes() {
        return ejes;
    }

    public void setEjes(int ejes) {
        this.ejes = ejes;
    }

    @Override
    public double calcularImporte() {
        LocalDateTime fechaSalida = LocalDateTime.now();
        long minutos = ChronoUnit.MINUTES.between(this.getFechaEntrada(), fechaSalida);
        double tasa = 0, total = 0;

        if (ejes <= 3) {
            tasa = 4.5;
        } else {
            tasa = 6.5;
        }
        total = minutos * tasa / 60;

        if (this.isAbono()) {
            total -= (total*0.4);
        }

        return total;
    }

    @Override
    public String toString() {
        return super.toString() + " # Camion{" + "ejes=" + ejes + '}';
    }

}
```

```
package aparcamiento;

import java.util.ArrayList;

public class Aparcamiento {

    private static ArrayList<Vehiculo> vehiculos = new ArrayList<Vehiculo>();
    private final static int cantidad = 20;
    private static int numero = 0;

    public static ArrayList<Vehiculo> getVehiculos() {
        return vehiculos;
    }

    public static ArrayList<String> getMatriculasVehiculos() {
        ArrayList<String> matriculas = new ArrayList<>();
        for (Vehiculo v : vehiculos) {
            matriculas.add(v.getMatricula());
        }
        return matriculas;
    }

    public static String intoducirVehiculo(Vehiculo v) {
        if (vehiculos.contains(v)) {//si esta dentro
            return "El vehículo ya se encuentra en el aparcamiento";
        } else if (numero == cantidad) {
            return "El aparcamiento está lleno";
        } else {
            numero++;
            vehiculos.add(v);
            return "Vehículo aparcado: " + v.toString();
        }
    }

    public static String sacarVehiculo(Vehiculo v) {
        if (!Aparcamiento.getVehiculos().contains(v)) {//no esta dentro
            return "El vehículo no se encuentra en el aparcamiento";
```

```
        } else {
            double precio = v.calcularImporte();
            numero--;
            vehiculos.remove(v);
            return "Vehículo con matrícula " + v.getMatricula() + " retirado con precio: " +
precio + " €";
        }
    }

    public static String sacarVehiculo(String matricula) {
        boolean dentro = false;
        Vehiculo ve = null;
        for (Vehiculo v : vehiculos) {
            if (v.getMatricula().equals(matricula)) {
                ve = v;
                dentro = true;
            }
        }
        if (!dentro) {//no esta dentro
            return "El vehículo no se encuentra en el aparcamiento";
        } else {
            double precio = ve.calcularImporte();
            numero--;
            vehiculos.remove(ve);
            return "Vehículo con matrícula " + ve.getMatricula() + " retirado con precio: " +
precio + " €";
        }
    }

}
```

```
package aparcamiento;

public class PruebaAparcamiento1 {

    public static void main(String[] args) {
        Automovil a1 = new Automovil("1111-AAA", true, "turismo");
        Automovil a2 = new Automovil("2222-BBB", false, "todoterreno");
        Automovil a3 = new Automovil("3333-CCC", false, "furgoneta");
        Camion c1 = new Camion("4444-DDD", true, 4);
        Camion c2 = new Camion("5555-EEE", false, 3);
        Camion c3 = new Camion("6666-FFF", false, 5);
        System.out.println(Aparcamiento.intoducirVehiculo(a1));
        System.out.println(Aparcamiento.intoducirVehiculo(a2));
        System.out.println(Aparcamiento.intoducirVehiculo(a3));
        System.out.println(Aparcamiento.intoducirVehiculo(c1));
        System.out.println(Aparcamiento.intoducirVehiculo(c2));
        System.out.println(Aparcamiento.intoducirVehiculo(c3));
        System.out.println("Vehículos: "+ Aparcamiento.getMatriculasVehiculos().toString());

        System.out.println(Aparcamiento.sacarVehiculo(a1));
        System.out.println("Vehículos: "+ Aparcamiento.getMatriculasVehiculos().toString());
        System.out.println(Aparcamiento.sacarVehiculo(a2));
        System.out.println("Vehículos: "+ Aparcamiento.getMatriculasVehiculos().toString());
        System.out.println(Aparcamiento.sacarVehiculo(a3));
        System.out.println("Vehículos: "+ Aparcamiento.getMatriculasVehiculos().toString());
        System.out.println(Aparcamiento.sacarVehiculo(c1));
        System.out.println("Vehículos: "+ Aparcamiento.getMatriculasVehiculos().toString());
        System.out.println(Aparcamiento.sacarVehiculo(c2));
        System.out.println("Vehículos: "+ Aparcamiento.getMatriculasVehiculos().toString());
        System.out.println(Aparcamiento.sacarVehiculo(c3));
        System.out.println("Vehículos: "+ Aparcamiento.getMatriculasVehiculos().toString());

    }

}
```

```
package aparcamiento;

import java.time.LocalDateTime;
import java.time.Month;

public class PruebaAparcamiento2 {
```

```
    public static void main(String[] args) {
        LocalDateTime diaHora = LocalDateTime.of(2019, Month.FEBRUARY, 19, 10, 59, 59);
        Automovil a1 = new Automovil("1111-AAA", diaHora, true, "turismo");
        Automovil a2 = new Automovil("2222-BBB", diaHora, false, "todoterreno");
        Automovil a3 = new Automovil("3333-CCC", diaHora, false, "furgoneta");
        Camion c1 = new Camion("4444-DDD", diaHora, true, 4);
        Camion c2 = new Camion("5555-EEE", diaHora, false, 3);
        Camion c3 = new Camion("6666-FFF", diaHora, false, 5);
        System.out.println(Aparcamiento.intoducirVehiculo(a1));
        System.out.println(Aparcamiento.intoducirVehiculo(a2));
        System.out.println(Aparcamiento.intoducirVehiculo(a3));
        System.out.println(Aparcamiento.intoducirVehiculo(c1));
        System.out.println(Aparcamiento.intoducirVehiculo(c2));
        System.out.println(Aparcamiento.intoducirVehiculo(c3));
        System.out.println("Vehículos: "+ Aparcamiento.getMatriculasVehiculos().toString());

        System.out.println(Aparcamiento.sacarVehiculo(a1.getMatricula()));
        System.out.println("Vehículos: "+ Aparcamiento.getMatriculasVehiculos().toString());
        System.out.println(Aparcamiento.sacarVehiculo(a2.getMatricula()));
        System.out.println("Vehículos: "+ Aparcamiento.getMatriculasVehiculos().toString());
        System.out.println(Aparcamiento.sacarVehiculo(a3.getMatricula()));
        System.out.println("Vehículos: "+ Aparcamiento.getMatriculasVehiculos().toString());
        System.out.println(Aparcamiento.sacarVehiculo(c1.getMatricula()));
        System.out.println("Vehículos: "+ Aparcamiento.getMatriculasVehiculos().toString());
        System.out.println(Aparcamiento.sacarVehiculo(c2.getMatricula()));
        System.out.println("Vehículos: "+ Aparcamiento.getMatriculasVehiculos().toString());
        System.out.println(Aparcamiento.sacarVehiculo(c3.getMatricula()));
        System.out.println("Vehículos: "+ Aparcamiento.getMatriculasVehiculos().toString());

    }

}
```