```java
import java.util.*;

/**
 * Vehiculo abstracto. Superclase de las clases de vehiculos
 * reales
 */
public abstract class Vehiculo {

  private String matricula;
  private Calendar horaEntrada;

  public Vehiculo(String matricula) {
    this.matricula = matricula;
  }

  public String getMatricula() {
    return matricula;
  }

  public Calendar getHoraEntrada() {
    return horaEntrada;
  }

  public void comienzaMes() {
  }

  public void finEstancia() {
  }

  public void comienzaEstancia() {
    this.horaEntrada=Calendar.getInstance();
  }

}
```

```java
import java.util.*;
/**
 * Vehiculo no residente. Debe pagar la estancia al salir.
 */
public class VehiculoNoResidente extends Vehiculo {

  private static final double precioMinuto = 0.02;

  private double pagoEstancia=0.0;

  public VehiculoNoResidente(String matrícula) {
    super(matrícula);
  }

  @Override
  public void finEstancia() {
    // calcula la cantidad a pagar
```

```
      pagoEstancia =
        difEnMinutos(getHoraEntrada(),Calendar.getInstance())
        * precioMinuto;
    }

    public double pagoEstancia() {
      return pagoEstancia;
    }

}
```

```
import java.util.Calendar;
import java.util.LinkedList;
/**
 * Vehículo oficial. Lleva la lista de las estancias
 * en el aparcamiento realizadas en el mes en curso
 */
public class VehiculoOficial extends Vehiculo {

  // lista de las estancias en el mes en curso
  private LinkedList<Estancia> estancias =
                          new LinkedList<Estancia>();

  public VehiculoOficial(String matrícula) {
    super(matrícula);
  }

  public LinkedList<Estancia> getEstancias() {
    return estancias;
  }

  @Override
  public void comienzaMes() {
    estancias.clear(); // borra la lista de estancias
  }

  @Override
  public void finEstancia() {
    // añade la estancia a la lista
    estancias.add(
        new Estancia(getHoraEntrada(),Calendar.getInstance()));
  }

}
```

```java
import java.util.Calendar;
/**
 * Vehículo de residente. Lleva la cuenta del tiempo
 * de estancia acumulado en el mes en curso
 */
public class VehiculoResidente extends Vehiculo {

  private static final double precioMinuto = 0.002;

  // tiempo de estancia acumulado en el mes en curso
  private int tiempoAcumulado = 0;

  public VehiculoResidente(String matrícula) {
    super(matrícula);
  }

  public int getTiempoAcumulado() {
    return tiempoAcumulado;
  }

  @Override
  public void comienzaMes() {
    // pone a 0 el tiempo acumulado
    tiempoAcumulado=0;
  }

  @Override
  public void finEstancia() {
    // incrementa el tiempo acumulado en la duración de
    // la estancia que finaliza en este instante
    tiempoAcumulado +=
      difEnMinutos(getHoraEntrada(),Calendar.getInstance());
  }

  public double pagoMes() {
    return tiempoAcumulado*precioMinuto;
  }
}
```

```java
import java.util.*;
import java.io.*;

/**
 *  Gestiona el aparcamiento
 */
public class Aparcamiento {

   public static class VehiculoYaRegistrado extends Exception {}
   public static class VehiculoYaAparcado extends Exception {}
   public static class VehiculoNoAparcado extends Exception {}

   // Registro de vehículos oficiales
   private LinkedList<VehiculoOficial> oficiales =
     new LinkedList<VehiculoOficial>();


// Registro de vehículos de residentes
private LinkedList<VehiculoResidente> residentes =
  new LinkedList<VehiculoResidente>();

// Vehículos que se encuentran en un momento dado en el
// aparcamiento. Pueden ser tanto vehículos de residentes,
// como oficiales como de no residentes
private LinkedList<Vehiculo> aparcados =
   new LinkedList<Vehiculo>();

/**
 * Busca el coche con la matrícula indicada en la
 * lista de vehículos oficiales
 * @param matrícula matrícula del vehículo buscado
 * @return el vehículo o null en caso de que no lo encuentre
 */
private VehiculoOficial buscaOficial(String matrícula) {
  for(VehiculoOficial v:oficiales) {
    if (v.getMatrícula().equals(matrícula))
      return v;
  }
  return null;
}
```

```java
/**
 * Busca el coche con la matrícula indicada en la
 * lista de vehículos de residentes
 * @param matrícula matrícula del vehículo buscado
 * @return el vehículo o null en caso de que no lo encuentre
 */
private VehiculoResidente buscaResidente(String matrícula) {
  for(VehiculoResidente v:residentes) {
    if (v.getMatrícula().equals(matrícula))
      return v;
  }
  return null;
}

/**
 * Busca el coche con la matrícula indicada en la
 * lista de vehículos aparcados
 * @param matrícula matrícula del vehículo buscado
 * @return el vehículo o null en caso de que no lo encuentre
 */
private Vehiculo buscaAparcado(String matrícula) {
  for(Vehiculo v:aparcados) {
    if (v.getMatrícula().equals(matrícula))
      return v;
  }
  return null;
}

/**
 * Añade el coche con la matrícula indicada a la lista de
```

```
 * vehículos oficiales
 * @param matrícula matrícula del nuevo coche oficial
 * @throws VehiculoYaRegistrado ya existe un coche con
 * esa matrícula en la lista
 */
public void registraOficial(String matrícula)
                  throws VehiculoYaRegistrado {
  VehiculoOficial v = buscaOficial(matrícula);
  if (v!=null)
    throw new VehiculoYaRegistrado();

  v = new VehiculoOficial(matrícula);
  oficiales.add(v);
}

/**
 * Añade el coche con la matrícula indicada a la lista de
 * vehículos de residentes
 * @param matrícula matrícula del nuevo coche de residente
 * @throws VehiculoYaRegistrado ya existe un coche con
 * esa matrícula en la lista
 */
public void registraResidente(String matrícula)
                           throws VehiculoYaRegistrado {
  VehiculoResidente v = buscaResidente(matrícula);
  if (v!=null)
    throw new VehiculoYaRegistrado();

  v = new VehiculoResidente(matrícula);
  residentes.add(v);
}
```

```java
/**
 * Un vehiculo entra al aparcamiento
 * @param matrícula matrícula del coche que entra
 * @throws VehiculoYaAparcado ya existe un coche con esa
 * matrícula dentro del aparcamiento
 */
public void entra(String matrícula)
                                throws VehiculoYaAparcado {
  Vehiculo v = buscaAparcado(matrícula);
  if (v!=null) {
    // error: ya existe un coche dentro del aparcamiento
    // con esa matrícula
    throw new VehiculoYaAparcado();
  }

  // vemos si es un vehículo de residente
  v = buscaResidente(matrícula);
  if (v==null) {
    // no es un vehículo de residente, vemos si es oficial
    v = buscaOficial(matrícula);
    if (v==null) {
      // tampoco es oficial, luego es de no residente.
```

```
        // Crea un vehículo de no residente
        v = new VehiculoNoResidente(matrícula);
      }
    }

    // sea del tipo que sea, llamamos al método correspondiente
    // a comenzar la estancia y le añadimos a la lista de
    // vehículos aparcados
    v.comienzaEstancia();
    aparcados.add(v);
  }

  /**
   * Un vehículo sale del aparcamiento
   * @param matrícula matrícula del vehículo que sale
   * @return el vehículo que sale para que, si es necesario,
   * puedan consultarse sus datos (pago, estancias, ..)
   * @throws VehiculoNoAparcado si la matrícula no corresponde
   * a ningún vehículo aparcado
   */
  public Vehiculo sale(String matrícula)
                                  throws VehiculoNoAparcado {
    Vehiculo v = buscaAparcado(matrícula);
    if (v==null) {
      // error: el vehículo debería estar en el aparcamiento!!
      throw new VehiculoNoAparcado();
    }

    // finaliza la estancia y se elimina de la lista de aparcados
    v.finEstancia();
    aparcados.remove(v);

    return v;
  }
```

```java
/**
 * Pone a 0 los registros de todos los vehiculos
 */
public void comienzaMes() {
  for(VehiculoResidente v: residentes)
    v.comienzaMes();
  for(VehiculoOficial v: oficiales)
    v.comienzaMes();
}

/**
 * Genera un informe con los pagos que deben realizar los
 * residentes
 * @param nomFich fichero en el que se escribe el informe
 * @throws IOException error de E/S
 */
public void generaInformePagosResidentes(String nomFich)
                                          throws IOException {
  PrintWriter sal = null;


    try {
      sal = new PrintWriter(new FileWriter(nomFich));

      sal.println("Matrícula    Tiempo estacionado (min.)"
                + "    Cantidad a pagar");
      for(VehiculoResidente v: residentes) {
        sal.printf("%-20s %7d %20.2f%n", v.getMatricula(),
            v.getTiempoAcumulado(), v.pagoMes());
      }
    } finally {
      if (sal!=null)
        sal.close();
    }
  }
}
```

```java
import java.util.*;
/**
 * Estancia de un vehiculo oficial
 */
public class Estancia {

  private Calendar horaEntrada;
  private Calendar horaSalida;

  public Estancia(Calendar horaEntrada, Calendar horaSalida) {
    super();
    this.horaEntrada = horaEntrada;
    this.horaSalida = horaSalida;
  }

  public Calendar getHoraEntrada() {
    return horaEntrada;
  }

  public Calendar getHoraSalida() {
    return horaSalida;
  }
}
```