1.-¿Qué ventajas observas de la separación del código de las clases que acceden a la BBDD de las clases que implementan la lógica de negocio, en este caso, solo la clase principal?

Las ventajas de la separación de código son muchas, pero una de las principales que hay que mencionar es la detección de errores y lógicamente la corrección de estos de manera mucho más fácil.

El separar las conexiones a la base de datos, el controlador de acceso a los datos (DAO) como he dicho no solo facilitan la detección de errores, sino que ademas permite la reutilización de código de manera muy sencilla.

Otra de las ventajas que tiene seria una posible escalabilidad, cierto es que este programa es una demostración de una serie de método, pero realmente si la clase principal fuera una clase más compleja que mostrara los datos de forma mas visual, estaríamos implementando una especie de Modelo – Vista y en este caso faltaría el Controlador y esto es uno de los modelos de programación más utilizados actualmente, precisamente por los motivos que estos diciendo, detección de errores, escalabilidad, múltiples representaciones de los datos, etc

Como ejemplo de que dividendo es mucho mejor es y si se cambia de gestor de base de datos? Pues la solución es bien sencilla tan solo se tendría que cambiar la clase conexión y el resto del programa seguiría funcionando sin ningún tipo de modificación.

2.-¿Se te ocurre alguna propuesta para no tener que crear el objeto Connection en la clase principal?.

Un método más correcto para el Objeto conexión seria crear lo que se llama un Pool de conexión, donde la conexión en si es un atributo estático de la clase ConnectionBD. De tal manera que una vez iniciado por cualquiera de los métodos de las clases DAO esta conexión de mantiene abierta, pero dentro del propio Objeto de connectionBD y el resto de los método DAO pueden acceder siempre a la conexión. Esta tipo de clses se hacen basándose en el Patrón Singleton, cierto que tiene más cosas como los constructures privados, que el constructor se llama desde un getter estático, etc pero eso es entrar más en profudidad de patrones y demás.

Por sintetizar la explicación se trata de una clase que tiene abierta la conexión a la base de datos, y cuando un método necesita esa conexión a la base de datos en vez de abrirla, se la pide al pool.

Descripción de la mejoras añadidas

CompruebaPropietarios este método es para comprobar si existe un propietario en la base de datos por medio de su DNI

```java
/**
 * Método que me devuelve cuantos propietarios hay con el mismo DNI
 * @param dni
 * @param conexion
 * @return
 */
public static boolean compruebaPropietarios(String dni, Connection conexion){
    boolean existe = true;
    String sentenciaSQL = "SELECT * FROM propietarios where dni_prop= ?";
    PreparedStatement sentencia=null;
    ResultSet rs = null;

    try {
        sentencia = conexion.prepareStatement(sentenciaSQL);
        sentencia.setString(1, dni);
        rs = sentencia.executeQuery();

        if (rs.next()==false) existe=false;

    } catch (SQLException ex) {
        System.out.println(ex.getMessage());
    }
    return existe;
}
```

ExisteVehiculo Método que nos dice si un vehículo con una matricula determinada existe en la base de datos.

```java
/**
 * Metetodo que comprueba si existe un vehiculo por medio de su matricula
 * @param matricula matricula del vehiculo que se quiere saber si existe
 * @param conexion conexión con la base de datos
 * @return true si existe
 */
private static boolean existeVehiculo(String matricula, Connection conexion){
    PreparedStatement sentencia = null;
    ResultSet rs = null;
    boolean existe= true;

    String sentenciaSQL = "SELECT * FROM vehiculos v WHERE v.mat_veh=?";
    try {
        sentencia = conexion.prepareStatement(sentenciaSQL);
        sentencia.setString(1, matricula);
        rs = sentencia.executeQuery();

        if (rs.next()==false) existe=false;

    } catch (SQLException ex) {
        Logger.getLogger(VehiculosDAO.class.getName()).log(Level.SEVERE, null, ex);
    }
    return existe;
}
```

ExistePropietario Este método nos dice si existe un propietario en base a su ID de la base de datos

```java
/**
 * Método que compruba si un propietario dado por ID existe en la tabla propietarios
 * @param idpropietario id de propietario que se quiere saber si existe
 * @param conexion conexión con la base de datos
 * @return true si existe
 */
private static boolean existePropietario(int idpropietario, Connection conexion){
    PreparedStatement sentencia = null;
    ResultSet rs = null;
    boolean existe= true;

    String sentenciaSQL = "SELECT * FROM propietarios p WHERE p.id_prop=?";
    try {
        sentencia = conexion.prepareStatement(sentenciaSQL);
        sentencia.setInt(1, idpropietario);
        rs = sentencia.executeQuery();

        if (rs.next()==false) existe=false;

    } catch (SQLException ex) {
        Logger.getLogger(VehiculosDAO.class.getName()).log(Level.SEVERE, null, ex);
    }
    return existe;
}
```

Vehiculo es una clase que replica la estructura de la tabla vehículos y es para facilitar el traspaso de datos entre la aplicación y la base de datos.

```java
public class Vehiculo {
    private String matricula;
    private String marca;
    private int km;
    private float precio;
    private String descripcion;
    private int id_prop;

    public Vehiculo(String matricula, String marca, int km, float precio, String descripcion, int id_prop) {
        this.matricula = matricula;
        this.marca = marca;
        this.km = km;
        this.precio = precio;
        this.descripcion = descripcion;
        this.id_prop = id_prop;
    }
```

Propietario es una clase que replica la estructura de la tabla propietarios y es para facilitar el traspaso de datos entre la aplicación y la base de datos.

```java
/**
 * Claee Propietario
 * @author JRBlanco
 */
public class Propietario {
    private int id;
    private String nombre;
    private String dni;

    public Propietario(int id, String nombre, String dni) {
        this.id = id;
        this.nombre = nombre;
        this.dni = dni;
    }
```