

Acceso a Datos.

Tarea para AD03.

Se trata de hacer una aplicación en Java que acceda a una base de datos Oracle de una aerolínea. Consiste en una conexión a las tablas VUELOS y PASAJEROS de una BD Oracle que se implementara con el fichero que se adjunta. (Se puede hacer con otras bases de datos como MySQL, SQLite o PostgreSQL)

Consta de los siguientes clases MAIN:

1. Mostrar y pedir información de la base de datos en general.
2. Mostrar la información de la tabla pasajeros.
3. Ver la información de los pasajeros de un vuelo, pasando el código de vuelo como parámetro.
4. Insertar un vuelo cuyos valores se pasan como parámetros.
5. Borrar el vuelo que se metió anteriormente en el que se pasa por parámetro su número de vuelo.
6. Modificar los vuelos de fumadores a no fumadores.

Elabora un **proyecto NetBeans** el cual será enviado para evaluar, que resuelva los puntos anteriormente mencionados.

Para este ejercicio, he entendido que debe generarse en el método **main** el código necesario para completar las acciones solicitadas en el enunciado, de forma similar a como se representa en el temario.

Por ello, y dado que tampoco es el propósito de al menos esta parte de la asignatura el manejo de interfaces gráficas, he decidido trabajar sobre una opción de consola, lo más compacta y eficiente posible, sin entrar a la división de la tarea por clases que enturbiaran el aprendizaje de lo expuesto en el tema.

A partir de ahí, lo primero que he hecho, es acudir al enlace facilitado en el tema para descargar el driver para **mysql**, necesario para nuestra conexión, en la carpeta creada para el proyecto.

MySQL Connectors

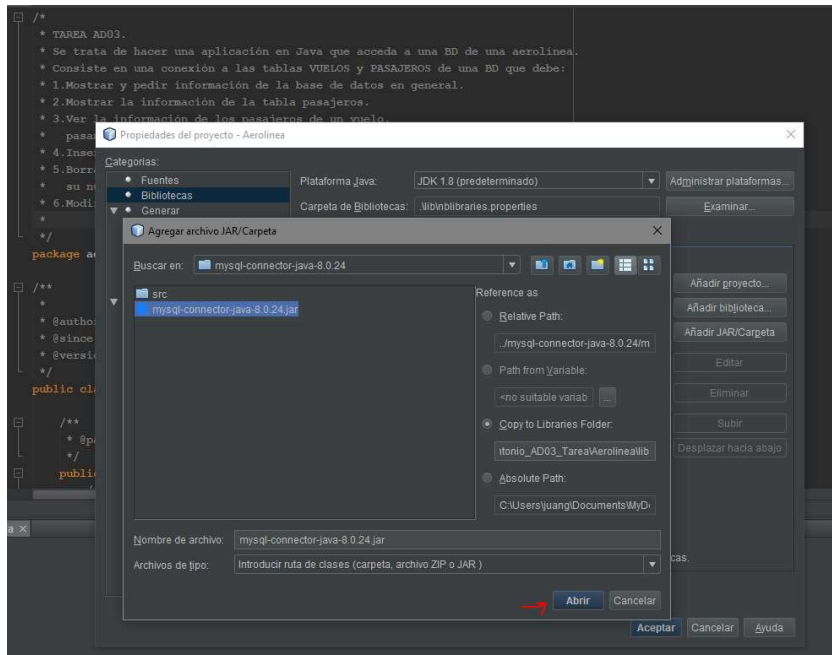
MySQL provides standards-based drivers for JDBC, ODBC, and .Net enabling developers to embed MySQL directly into their applications.

Developed by MySQL	
ADO.NET Driver for MySQL (Connector/NET)	Download
ODBC Driver for MySQL (Connector/ODBC)	Download
JDBC Driver for MySQL (Connector/J)	Download
Node.js Driver for MySQL (Connector/Node.js)	Download
Python Driver for MySQL (Connector/Python)	Download
C++ Driver for MySQL (Connector/C++)	Download
C Driver for MySQL (Connector/C)	Download
C API for MySQL (mysqlclient)	Download

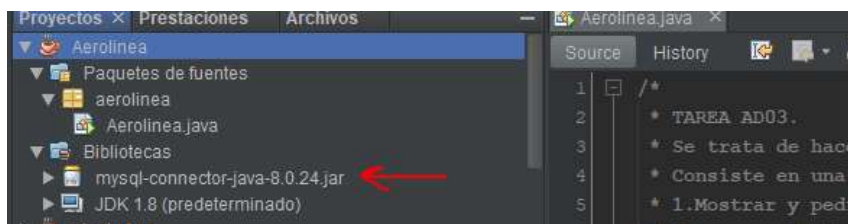
Dentro tendremos el archivo **jar** necesario



Creamos nuestro proyecto en NetBeans para añadirle el **driver** de conexión, desde **Propiedades del proyecto>Añadir JAR/Carpeta**.



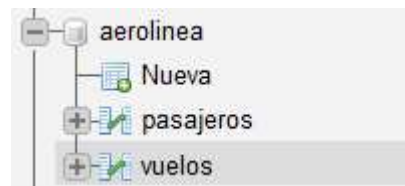
Tras aceptar, ya lo tenemos añadido a nuestro directorio y se puede empezar con el desarrollo.



Como en el ejercicio del tema anterior, he intentado documentar lo mejor posible el código y por ende las capturas demostrativas, con el fin de que pueda entenderse cada paso y toma de decisión de la mejor manera posible.

Debo advertir que en mi caso, para que me aceptara en **mySql** la importación del fichero facilitado con la base de datos, tuve que comentar las dos primeras líneas de borrado de tablas y modificar los tipos de datos **VARCHAR2** por **VARCHAR** y **NUMBER** por **INTEGER**.

```
java.sql,-- CREACIÓN DE LAS TABLAS VUELOS Y PASAJEROS:
java.util,-- DROP TABLE VUELOS CASCADE CONSTRAINTS;
java.util,-- DROP TABLE PASAJEROS CASCADE CONSTRAINTS;
-----
CREATE TABLE VUELOS
(
  COD_VUELO VARCHAR2(10) PRIMARY KEY,
  HORA_SALIDA VARCHAR2(15),
  DESTINO VARCHAR2(15),
  PROCEDENCIA VARCHAR2(15),
  PLAZAS_FUMADOR NUMBER(3),
  PLAZAS_NO_FUMADOR NUMBER(3),
  PLAZAS_TURISTA NUMBER(3),
  PLAZAS_PRIMERA NUMBER(3)
);
-----
CREATE TABLE PASAJEROS
(
  NUM NUMBER(7),
  COD_VUELO VARCHAR2(10), -- 'TU': TURISTA, 'PR': PRIMERA
  TIPO_PLAZA VARCHAR2(2), -- 'SI' o 'NO'
  FUMADOR VARCHAR2(2), -- 'SI' o 'NO'
  CONSTRAINT PK_PASAJEROS PRIMARY KEY(NUM, COD_VUELO),
  CONSTRAINT FK_PASAJEROS FOREIGN KEY(COD_VUELO) REFERENCES VUELOS
);
```



Asimismo, si se quiere replicar en otro equipo el funcionamiento del programa, comentar que en mi caso, mi usuario **root** en **mySql** no mantiene contraseña alguna, por lo que ese campo

`pass`, debe modificarse para conectar de forma correcta (por ejemplo, con Oracle, muchos hemos usado `system`).

Vamos a ir viendo los pasos en el código.

Aerolinea.java

```
/*
 * TAREA AD03.
 * Se trata de hacer una aplicación en Java que acceda a una BD de una
aerolínea.
 * Consiste en una conexión a las tablas VUELOS y PASAJEROS de una BD
que debe:
 * 1.Mostrar y pedir información de la base de datos en general.
 * 2.Mostrar la información de la tabla pasajeros.
 * 3.Ver la información de los pasajeros de un vuelo,
 *   pasando el código de vuelo como parámetro.
 * 4.Insertar un vuelo cuyos valores se pasan como parámetros.
 * 5.Borrar el vuelo que se metió anteriormente en el que se pasa por
parámetro
 *   su número de vuelo.
 * 6.Modificar los vuelos de fumadores a no fumadores.
 *
 */
package aerolinea;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Scanner;

/**
 *
 * @author juang <juangmuelas@gmail.com>
 * @since 16/12/2021
 * @version 1
 */
public class Aerolinea {

    /**
     * @param args the command line arguments
     * @param String url, cadena restringida con la conexión a la BD
     * @param String user, cadena restringida con el usuario a utilizar
     * @param String pass, cadena restringida con la password a utilizar
     * que en este caso está vacía, y debe completarse con la que tenga
     * cada uno en su conexión (ej.común: system)
     */

    static final String url = "jdbc:mysql://localhost:3306/aerolinea";
    static final String user="root";
    static final String pass="";

    /**
     * Creación de objeto de la clase Scanner para recibir datos
     * por teclado.
     */
    static Scanner teclado = new Scanner(System.in);
```

Lo primero es declarar e inicializar las variables que como decía anteriormente, voy a usar para la conexión con el driver de `mysql`.

También, ya que se va a utilizar como forma de hacer más dinámico el ejercicio, se declara un objeto de la clase `Scanner`, para la recogida de datos por teclado.

A partir de ahí, iniciamos los procesos dentro del método `main`.

Como se puede observar, se crean las variables que van a recoger los distintos objetos de la clase `java.sql`, como la conexión, los `ResultSet` que se usaran para mostrar los datos o los `Statement` que se usarán para las consultas.

```
public static void main(String[] args) {
    /**
     * @param miConexion, variable inicializada a null de un objeto
     * Connection
     * @param dbmd, variable inicializada a null de un objeto
     * DatabaseMetaData
     * @param miStatement, variable inicializada a null de un objeto
     * Statement
     * @param miResultSet, variable inicializada a null de un objeto
     * ResultSet
     * @param result, variable inicializada a null de un objeto ResultSet
     * @param resultColumns, variable inicializada a null de un objeto
     * ResultSet
     * @param pasajerosVuelo, variable inicializada a null de un objeto
     * PreparedStatement
     * @param insertVuelo, variable inicializada a null de un objeto
     * PreparedStatement
     * @param dropVuelo, variable inicializada a null de un objeto
     * PreparedStatement
     */

    Connection miConexion=null;
    java.sql.DatabaseMetaData dbmd =null;
    Statement miStatement = null;
    ResultSet miResultSet = null;
    ResultSet result = null;
    ResultSet miResultSetVuelo = null;
    PreparedStatement pasajerosVuelo = null;
    PreparedStatement insertVuelo = null;
    PreparedStatement dropVuelo = null;

    try {
        /**
         * Creamos la conexión con los datos de las variables declaradas
         * anteriormente. Si cambiamos de conexión, solo se debe modificar
         * el valor de esas variables
         */

        miConexion = DriverManager.getConnection(url, user, pass);
        System.out.println("Conectado a BD...");
        //crear el objeto statement para la consulta
        miStatement = miConexion.createStatement();
```

Lo siguiente, es envolver todo en un try-catch que recoja los posibles errores generales de la conexión, y hacer la llamada al driver con el método `getConnection()` que hace la conexión con nuestra base de datos a través de los parámetros facilitados.

Conectados, declaro de forma ordenada las distintas cadenas con las `query` a utilizar durante el ejercicio, para así ir pasándolas según se requieran, así como los parámetros para lectura de teclado.

```

/**
 * Declaramos los parámetros para las consultas:
 * @param String infoPasajeros, recibe la cadena que se usará
 * como query para la consulta de pasajeros.
 * @param String infoVuelo, recibe la cadena que se usará
 * como query para la consulta de pasajeros por un vuelo concreto.
 * @param String anadirVuelo, recibe la cadena que se usará
 * como query para insertar un nuevo vuelo en la tabla vuelos.
 * @param String borrarVuelo, recibe la cadena que se usará
 * como query para eliminar un vuelo de la tabla vuelos.
 * @param String modificarFumador, recibe la cadena que se usará
 * como query modificar el parámetro de SI a NO en la
 * tabla pasajeros.
 */

String infoPasajeros ="SELECT * FROM PASAJEROS";
String infoVuelo="SELECT * FROM PASAJEROS WHERE
                COD_VUELO = ? ";
String anadirVuelo="INSERT INTO vuelos VALUES
                (?, ?, ?, ?, ?, ?, ?, ?) ";
String borrarVuelo="DELETE FROM vuelos WHERE
                COD_VUELO = ? ";
String modificarFumador="UPDATE pasajeros SET FUMADOR='NO'
                WHERE FUMADOR='SI'";

/**
 * Declaramos los parámetros a usar en la recogida de datos.
 * @param String tipo, cadena que recogerá el tipo de plaza:
 * (turista o primera)
 * @param String vuelo, cadena que recogerá el código del vuelo.
 * @param String hora, cadena que recogerá fecha y hora.
 * @param String tipo, cadena que recogerá el origen.
 * @param String tipo, cadena que recogerá el destino.
 * @param integer fumador, entero que recoge las plazas de
 * fumador.
 * @param integer no_fumador, entero que recoge las plazas de
 * NO fumador.
 * @param integer fumador, entero que recoge las plazas de
 * turista.
 * @param integer fumador, entero que recoge las plazas de
 * primera.
 * @param boolean seguir, se inciliza a true para poder seguir
 * iterando mientras se desee con las distintas opciones.
 * @param char opcion, caracter numérico para las opciones de
 * menu.
 * @param boolean erroneo inicializado a true, que nos permite
 * mantenernos en el menú ante opciones erróneas.
 */

String tipo="";
String vuelo="";
String hora="";
String origen="";
String destino="";
int fumador,no_fumador,turista,primera;
boolean seguir = true;

```

El último booleano lo utilizaremos para manejar el menú de opciones, que encerramos dentro de un bucle **do-while**, que nos mantendrá en el menú mientras ese valor no cambie.

Dentro, hay un `switch` que recoge los valores de las distintas opciones, según la elección se ejecutará cada una de las peticiones de la tarea.

```
/**
 * Se envuelve en un bucle do-while el menú de opciones
 */
do {
    System.out.println("====BASE DATOS AEROLINEA====");
    System.out.println("1. Información general BD");
    System.out.println("2. Información pasajeros");
    System.out.println("3. Datos pasajeros por vuelo");
    System.out.println("4. Añadir vuelo");
    System.out.println("5. Borrar vuelo");
    System.out.println("6. Modificar fumadores por no fumadores");
    System.out.println("7. Salir");
    System.out.println("=====");

    char opcion = '7';
    boolean erroneo = true;
    do {
        System.out.print("Teclea opcion: ");
        opcion = teclado.next().charAt(0);
        if (opcion >= '1' && opcion <= '7')
            erroneo = false;
    } while (erroneo);

    /**
     * Con la opción recogida dentro de los valores correctos,
     * recorreremos con un switch los distintos elementos del
     * menú.
     */

    switch (opcion) {
        case '1':
            System.out.println("Buscando datos DB AEROLINEA...");

            try {
                /**
                 * Se crea la conexión para los metadatos
                 * y con el método getTables, recorreremos
                 * el ResultSet, que recoge los datos de
                 * nuestras BD
                 */
                dbmd = miConexion.getMetaData();
                result = dbmd.getTables(null,
                    "aerolinea", null, null);
                while (result.next()){
                    String nombrebd =
                        result.getString(1); // columna 1
                    String esquema =
                        result.getString(2); // columna 2
                    String tabla =
                        result.getString(3); // columna 3
                    String tipotabla =
                        result.getString(4); // columna 4
```

En la primera opción debe recoger mediante los métodos `getString()`, los valores de las distintas tablas obtenidas mediante `getTable()`, un método de la clase `DatabaseMetadata`.

Por una cuestión de limpieza para la tarea, he filtrado la salida.

```

/**
 * Por ser un ejemplo y dar una salida más limpia
 * vamos a mostrar sólo las tablas de aerolinea
 */

if (result.getString(1).matches("aerolinea"))
    System.out.printf("Tipo de tabla: %s,
        Nombre BD: %s, Esquema: %s, Nombre Tabla: %s %n"
        , tipotabla, nombrebd, esquema, tabla);
}
} catch (SQLException e) {
    System.out.println(e.getMessage());
    System.out.println(e.getErrorCode());
    System.out.println(e.getSQLState());
}
break;

```

Podemos observar por la consola, los datos generados, y que nos devuelve correctamente al menú para seguir interactuando.

```

Conectado a BD....
=====BASE DATOS AEROLINEA=====
1. Información general BD
2. Información pasajeros
3. Datos pasajeros por vuelo
4. Añadir vuelo
5. Borrar vuelo
6. Modificar fumadores por no fumadores
7. Salir
=====
Teclea opcion: 1
Buscando datos DB AEROLINEA...
Tipo de tabla: TABLE ,Nombre BD: aerolinea, Esquema: null, Nombre Tabla: pasajeros
Tipo de tabla: TABLE ,Nombre BD: aerolinea, Esquema: null, Nombre Tabla: vuelos
=====BASE DATOS AEROLINEA=====
1. Información general BD

```

Para la segunda petición, se recorre la [query](#) mediante el método `executeQuery()` con un bucle `while` que recogerá datos mientras se sigan recibiendo.

En este punto, he intentado crear una interfaz de consola más amigable mediante dos detalles: el uso de secuencias de escape [ANSI](#) para reflejar por consola en distintos colores los datos, y con modificando la impresión de los valores "TU" y "PR" por "TURISTA" y "PRIMERA".

```

case '2':
/**
 * Ejecutamos la query y recorremos el ResultSet
 */
miResultSet =
miStatement.executeQuery(infoPasajeros);

while(miResultSet.next()){
/**
 * Hacemos una interfaz más amigable
 * modificando el dato mostrado por
 * consola según sea turista (TU)
 * o primera (PR).
 */
if(miResultSet.getString("TIPO_PLAZA").matches("TU"))
{
    tipo = "\u001B[36mTURISTA";
}
else{
    tipo = "\u001B[32mPRIMERA";
}
}

```

```

System.out.println("\u001B[36mNUM: "
    + miResultSet.getString(1)
    + "\t \u001B[36mCOD_VUELO: "
    + miResultSet.getString(2)
    + "\t TIPO_PLAZA: " + tipo
    + "\t \u001B[36mFUMADOR: "
    + miResultSet.getString(4));
} //fin while miResultSet
break;

```

En la ejecución se puede ver como se muestran los datos según lo comentado:

Tecllea opcion: 2

NUM: 123	COD_VUELO: IB-SP-4567	TIPO_PLAZA: TURISTA	FUMADOR: SI
NUM: 124	COD_VUELO: IB-SP-4567	TIPO_PLAZA: PRIMERA	FUMADOR: SI
NUM: 125	COD_VUELO: IB-SP-4567	TIPO_PLAZA: PRIMERA	FUMADOR: NO
NUM: 126	COD_VUELO: IB-BA-46DC	TIPO_PLAZA: TURISTA	FUMADOR: SI
NUM: 127	COD_VUELO: IB-BA-46DC	TIPO_PLAZA: PRIMERA	FUMADOR: SI
NUM: 128	COD_VUELO: FR-DC-4667	TIPO_PLAZA: TURISTA	FUMADOR: SI
NUM: 129	COD_VUELO: FR-DC-4667	TIPO_PLAZA: TURISTA	FUMADOR: NO
NUM: 130	COD_VUELO: AV-DC9-233	TIPO_PLAZA: TURISTA	FUMADOR: SI
NUM: 131	COD_VUELO: AV-DC9-233	TIPO_PLAZA: TURISTA	FUMADOR: SI
NUM: 132	COD_VUELO: AV-DC9-233	TIPO_PLAZA: PRIMERA	FUMADOR: NO
NUM: 133	COD_VUELO: IB-D5-347	TIPO_PLAZA: PRIMERA	FUMADOR: SI
NUM: 134	COD_VUELO: IB-D5-347	TIPO_PLAZA: PRIMERA	FUMADOR: SI
NUM: 135	COD_VUELO: IB-D5-347	TIPO_PLAZA: TURISTA	FUMADOR: SI
NUM: 136	COD_VUELO: IB-D5-347	TIPO_PLAZA: TURISTA	FUMADOR: NO

El punto tercero, trabaja de forma similar, pero tiene ciertos detalles a los que comenzar a dar atención.

El primero es recordar forzar la limpieza de caché mediante el método `nextLine()` que evite errores en ejecución al pedir los datos.

Otro detalle que recordar es que, si bien anteriormente, en el `ResultSet` se le pasaba la `query`, en este caso el método `executeQuery()`, recibe esta desde el `Statement`, por lo que añadir ese valor, provocaría un error `NullPointerException` en tiempo de ejecución.

```

case '3':
/**
 * Pasamos a nuestra conexión la query por un objeto
 * PreparedStatement, que recibirá los datos a través de
 * peticiones por teclado.
 * Para evitar errores de caché en NetBeans, se
 * limpia en tiempo de ejecución mediante un
 * nextLine().
 */
pasajerosVuelo =
miConexion.prepareStatement(infoVuelo);
teclado.nextLine();
System.out.println("INDICAR CODIGO VUELO:");
vuelo = teclado.nextLine();
/**
 * Tras recoger datos, se muestra en consola para
 * el seguimiento y se añade el dato a la query
 */
System.out.println("DATOS DE PASAJEROS DEL VUELO: "
    + vuelo);
pasajerosVuelo.setString(1, vuelo);
/**
 * Al inicializar el Statement, ya hemos asociado
 * la consulta, por lo que el método
 * executeQuery(), debe
 * ir sin otro parámetro para evitar errores
 * NullPointerException.
 */

```



```

miResultSetVuelo =
pasajerosVuelo.executeQuery();
System.out.println(miResultSetVuelo);
while(miResultSetVuelo.next()){
    //hacemos más amigable la salida
    if(miResultSetVuelo.getString("TIPO_PLAZA")
        .matches("TU")){
        tipo = "\u001B[36mTURISTA";
    }else{
        tipo = "\u001B[32mPRIMERA";
    }
    System.out.println("VUELO "
        + miResultSetVuelo.getString(2));
    System.out.println("\u001B[36mNUM: "
        + miResultSetVuelo.getString(1)
        + "\t TIPO_PLAZA: " + tipo
        + "\t \u001B[36mFUMADOR: "
        + miResultSetVuelo.getString(4));
} //fin while infovuelo
break;

```

Aprovechando la salida por consola del punto anterior, podemos coger uno de los vuelos y ver como se ejecuta este punto:

```

=====
Teclea opcion: 3
INDICAR CODIGO VUELO:
SP-DC-438
DATOS DE PASAJEROS DEL VUELO: SP-DC-438
com.mysql.cj.jdbc.result.ResultSetImpl@7823a2f9
VUELO SP-DC-438
NUM: 137      TIPO_PLAZA: TURISTA      FUMADOR: NO
VUELO SP-DC-438
NUM: 138      TIPO_PLAZA: TURISTA      FUMADOR: NO
VUELO SP-DC-438
NUM: 139      TIPO_PLAZA: PRIMERA      FUMADOR: NO
VUELO SP-DC-438
NUM: 140      TIPO_PLAZA: PRIMERA      FUMADOR: NO
=====BASE DATOS AEROLINEA=====
1. Información general BD

```

Al no haber separado en distintos ficheros y clases el ejercicio, y para no volver más complejo el código, en esta parte hemos confiado el manejo de errores al `try-catch` que lo envuelve, por lo que si se introduce una cadena errónea, ya sea por no coincidir con ninguna de la tabla o por superar las restricciones del propio campo, la `SQLException`, nos va a devolver al menú.

```

=====
Teclea opcion: 3
INDICAR CODIGO VUELO:
weqwer
DATOS DE PASAJEROS DEL VUELO: weqwer
com.mysql.cj.jdbc.result.ResultSetImpl@5034c75a
=====BASE DATOS AEROLINEA=====

```

El punto cuarto debe facilitar añadir un vuelo a nuestra tabla. El funcionamiento es muy similar al punto anterior, debiendo tomar las precauciones apuntadas antes y cuidando la recogida de los datos, pues ahora no sólo recibe cadenas de texto mediante `setString()`, sino también valores numéricos mediante `setInt()`.

```

case '4':
/**
 * Pasamos a nuestra conexión la query por un objeto
 * PreparedStatement, que recibirá los datos a través de
 * peticiones por teclado.
 * Para evitar errores de caché en NetBeans, se
 * limpia en tiempo de ejecución mediante un
 * nextLine().
 */
insertVuelo =
miConexion.prepareStatement(anadirVuelo);
teclado.nextLine();
System.out.println("INDICAR CODIGO VUELO:");
vuelo = teclado.nextLine();
System.out.println("INDICAR HORA SALIDA (dd/mm/yy-
                    hh:mm):");
hora = teclado.nextLine();
System.out.println("INDICAR DESTINO:");
destino = teclado.nextLine();
System.out.println("INDICAR ORIGEN:");
origen = teclado.nextLine();
System.out.println("INDICAR N° PLAZAS FUMADOR:");
fumador = teclado.nextInt();
System.out.println("INDICAR N° PLAZAS NO FUMADOR:");
no_fumador = teclado.nextInt();
System.out.println("INDICAR N° PLAZAS TURISTA:");
turista = teclado.nextInt();
System.out.println("INDICAR N° PLAZAS PRIMERA:");
primera = teclado.nextInt();

/**
 * Tras recoger datos, se muestran en consola para
 * su seguimiento y se añaden los datos a la query
 */
System.out.println("AÑADIENDO DATOS DE PASAJEROS
                    NUEVO VUELO: " + vuelo);
System.out.println("COD VUELO: " + vuelo
                    + " HORA SALIDA: " + hora
                    + " DESTINO: " + destino
                    + " ORIGEN: " + origen
                    + " FUMADOR: " + fumador
                    + " NO FUMADOR: " + no_fumador
                    + " PLAZA TURISTA: " + turista
                    + " PLAZA PRIMERA: " + primera
                    );

//forzamos la limpieza de caché antes de añadir datos
teclado.nextLine();
insertVuelo.setString(1, vuelo);
insertVuelo.setString(2, hora);
insertVuelo.setString(3, destino);
insertVuelo.setString(4, origen);
insertVuelo.setInt(5, fumador);
insertVuelo.setInt(6, no_fumador);
insertVuelo.setInt(7, turista);
insertVuelo.setInt(8, primera);

/**
 * Al inicializar el Statement, ya hemos asociado
 * la consulta, por lo que el método
 * executeUpdate()
 * debe ir sin otro parámetro para evitar errores
 * NullPointerException.
 */

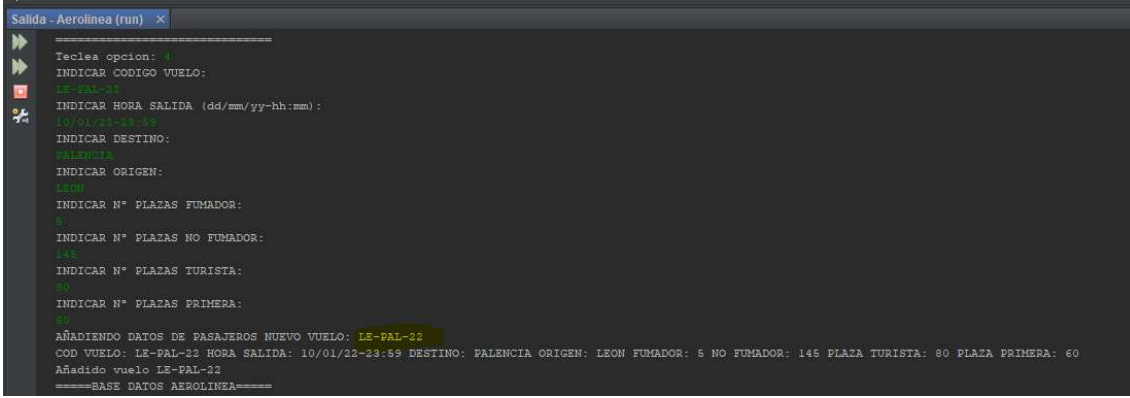
```

```
insertVuelo.executeUpdate();

System.out.println("Añadido vuelo " + vuelo);

break;
```

Al ejecutarlo, podemos comprobar tanto por consola como en la propia BD la inserción de la nueva fila.



```
Salida - Aerolinea (run) X
=====
Teclea opcion: 4
INDICAR CODIGO VUELO:
LE-PAL-22
INDICAR HORA SALIDA (dd/mm/yy-hh:mm):
10/01/22-23:59
INDICAR DESTINO:
PALENCIA
INDICAR ORIGEN:
LEON
INDICAR N° PLAZAS FUMADOR:
5
INDICAR N° PLAZAS NO FUMADOR:
145
INDICAR N° PLAZAS TURISTA:
80
INDICAR N° PLAZAS PRIMERA:
60
AÑADIENDO DATOS DE PASAJEROS NUEVO VUELO: LE-PAL-22
COD VUELO: LE-PAL-22 HORA SALIDA: 10/01/22-23:59 DESTINO: PALENCIA ORIGEN: LEON FUMADOR: 5 NO FUMADOR: 145 PLAZA TURISTA: 80 PLAZA PRIMERA: 60
Añadido vuelo LE-PAL-22
=====BASE DATOS AEROLINEA=====
```

IB-D5-347	13:3	ZARAGOZA	PARIS	100	200	210	90
IB-SP-4567	27/03/99-10:3	PARIS	MADRID	100	100	160	40
LE-PAL-22	10/01/22-23:59	PALENCIA	LEON	5	145	80	60
SP-DC-438	30/03/99-09:2	MOSCÚ	SEVILLA	90	100	160	30

Vamos a eliminar en el siguiente apartado el vuelo.

La estructura para eliminar es prácticamente igual que para añadir, pues la diferencia se enmarca en la [query](#) llamada para esta instrucción.

```
case '5':
    System.out.println("Borrando vuelo...");
    dropVuelo =
    miConexion.prepareStatement(borrarVuelo);
    teclado.nextLine();
    System.out.println("INDICAR CODIGO VUELO A ELIMINAR:");
    vuelo = teclado.nextLine();
    dropVuelo.setString(1, vuelo);
    dropVuelo.executeUpdate();
    System.out.println("Borrado vuelo " +vuelo);

    break;
```

Al igual que en el punto 4, se puede comprobar la ejecución por consola o refrescando nuestra tabla.



```
=====
Teclea opcion: 5
Borrando vuelo...
INDICAR CODIGO VUELO A ELIMINAR:
LE-PAL-22
Borrado vuelo LE-PAL-22
=====BASE DATOS AEROLINEA=====
1. Información general BD
```

IB-D5-347	13:3	ZARAGOZA	PARIS				
IB-SP-4567	27/03/99-10:3	PARIS	MADRID				
SP-DC-438	30/03/99-09:2	MOSCÚ	SEVILLA				

Para los elementos que están marcados: [Editar](#) [Copiar](#) [Borrar](#)

El último punto, pide modificar un campo, para que los pasajeros con plaza de fumador pasen a ser de no fumador.

Al ser una petición global, sólo debemos invocar a la [query](#) creada.

```

        case '6':
            System.out.println(
                miStatement.executeUpdate(modificarFumador)
            );
            break;

```

Nuevamente, se comprueba la propagación de la petición en la tabla:

NUM: 137	COD_VUELO: SP-DC-438	TIPO_PLAZA: TURISTA	FUMADOR: NO
NUM: 138	COD_VUELO: SP-DC-438	TIPO_PLAZA: TURISTA	FUMADOR: NO
NUM: 139	COD_VUELO: SP-DC-438	TIPO_PLAZA: PRIMERA	FUMADOR: NO
NUM: 140	COD_VUELO: SP-DC-438	TIPO_PLAZA: PRIMERA	FUMADOR: SI
NUM: 141	COD_VUELO: FR-DC7-247	TIPO_PLAZA: PRIMERA	FUMADOR: NO
NUM: 142	COD_VUELO: FR-DC7-247	TIPO_PLAZA: TURISTA	FUMADOR: NO
NUM: 143	COD_VUELO: FR-DC7-247	TIPO_PLAZA: TURISTA	FUMADOR: NO


```

=====BASE DATOS AEROLINEA=====
1. Información general BD
2. Información pasajeros
3. Datos pasajeros por vuelo
4. Añadir vuelo
5. Borrar vuelo
6. Modificar fumadores por no fumadores
7. Salir

=====
Teclea opcion: 6
11
=====BASE DATOS AEROLINEA=====
1. Información general BD
2. Información pasajeros
3. Datos pasajeros por vuelo
4. Añadir vuelo
5. Borrar vuelo
6. Modificar fumadores por no fumadores
7. Salir

=====
Teclea opcion: 2
NUM: 123      COD_VUELO: IB-SP-4567  TIPO_PLAZA: TURISTA  FUMADOR: NO
NUM: 124      COD_VUELO: IB-SP-4567  TIPO_PLAZA: PRIMERA  FUMADOR: NO
NUM: 125      COD_VUELO: IB-SP-4567  TIPO_PLAZA: PRIMERA  FUMADOR: NO

```

NUM	COD_VUELO	TIPO_PLAZA	FUMADOR
123	IB-SP-4567	TU	NO
124	IB-SP-4567	PR	NO
125	IB-SP-4567	PR	NO
126	IB-BA-46DC	TU	NO
127	IB-BA-46DC	PR	NO
128	FR-DC-4667	TU	NO
129	FR-DC-4667	TU	NO
130	AV-DC9-233	TU	NO
131	AV-DC9-233	TU	NO
132	AV-DC9-233	PR	NO
133	IB-D5-347	PR	NO

Para finalizar, damos la opción de salir de nuestro menú cambiando el valor del booleano y cerramos objetos, liberando recursos.

```

        case '7':
            //Para salir del menú.
            seguir = false;
        } //fin switch opciones
    } while (seguir); //fin bucle dowhile
} catch (SQLException e) {
    System.out.println("ERROR DE CONEXION!");
    e.printStackTrace();
} finally{
    //si hay conexión abierta, se cierra
    if(miConexion != null){
        try {
            //cerramos los objetos de la conexión y liberamos recursos
            miResultSet.close();
            miStatement.close();
            miResultSetVuelo.close();
            insertVuelo.close();
            dropVuelo.close();
            miConexion.close();
            System.out.println("CERRANDO CONEXIÓN");
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
} // fin bloque try-catch-finally
} // fin main
} //fin clase Aerolinea

```