AIR-AFPA

Activité-type 1 : Développer une application client/serveur.

Situation 4 Coder en Sql.

El Moussaoui Salim

Sommaire :

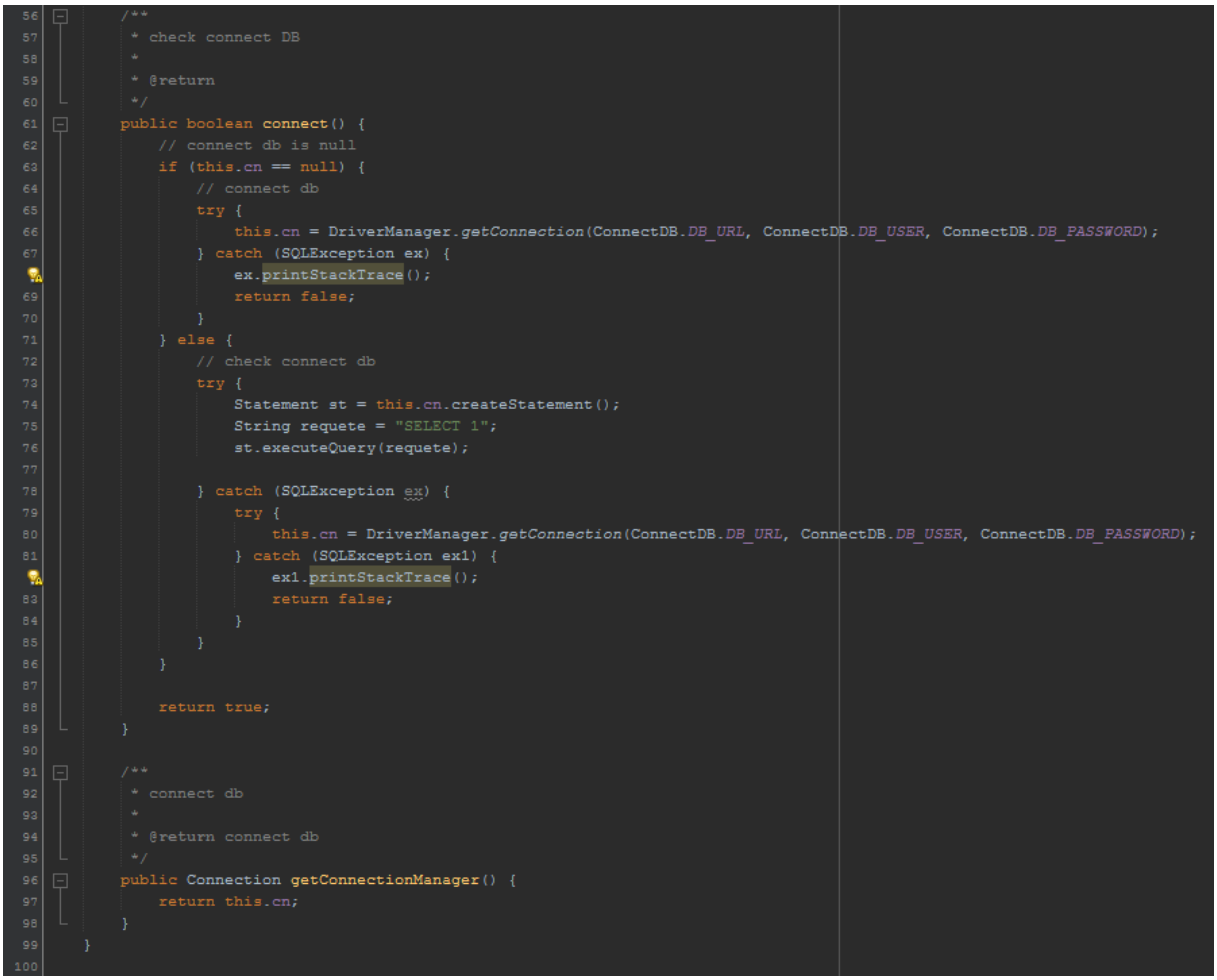
1. Introduction

AIR-AFPA est une filiale de la société AFPA TRAVEL France créée en décembre 2012. Spécialisée dans le marché low-cost du transport de passagers, elle a pour but de se faire une place sur ce marché. En effet, d'ici 2020 le nombre de vol low-cost devrait s'accroître de 50 %. Jusqu'en mi-2013, la filiale avait recours aux logiciels de sa maison-mère, mais ses besoins changeants elle souhaite développer son propre système d'information, c'est dans cette optique que le service des vols demande à présent la réalisation d'une application de gestion des vols.

2. Productions

2.1 Connexion la base de donnée avec la Classe ConnectDB (Singleton)

```java
/**
 *
 * @author Salim El Moussaoui <salim.elmoussaoui.afpa2017@gmail.com>
 */
public class ConnectDB {

    // Assign values for connection db mysql
    private static final String DB_URL = "jdbc:mysql://localhost:3306/airafpa";
    private static final String DB_JDBC_DRIVER = "com.mysql.jdbc.Driver";
    private static final String DB_USER = "afpa";
    private static final String DB_PASSWORD = "afpa";

    private Connection cn = null;

    private ConnectDB() {
        // import jdbc mysql
        try {
            Class.forName(ConnectDB.DB_JDBC_DRIVER);

        } catch (ClassNotFoundException ex) {
            ex.printStackTrace();
            System.exit(1);
        }

    }

    /**
     * Singleton for connexion DB
     *
     * @return
     */
    public static ConnectDB getInstance() {
        return ConnexionBdDHolder.INSTANCE;
    }

    /**
     * Assign Instance class ConnectDB
     */
    private static class ConnexionBdDHolder {

        private static final ConnectDB INSTANCE = new ConnectDB();
    }

```



```java
    /**
     * check connect DB
     *
     * @return
     */
    public boolean connect() {
        // connect db is null
        if (this.cn == null) {
            // connect db
            try {
                this.cn = DriverManager.getConnection(ConnectDB.DB_URL, ConnectDB.DB_USER, ConnectDB.DB_PASSWORD);
            } catch (SQLException ex) {
                ex.printStackTrace();
                return false;
            }
        } else {
            // check connect db
            try {
                Statement st = this.cn.createStatement();
                String requete = "SELECT 1";
                st.executeQuery(requete);

            } catch (SQLException ex) {
                try {
                    this.cn = DriverManager.getConnection(ConnectDB.DB_URL, ConnectDB.DB_USER, ConnectDB.DB_PASSWORD);
                } catch (SQLException ex1) {
                    ex1.printStackTrace();
                    return false;
                }
            }
        }

        return true;
    }

    /**
     * connect db
     *
     * @return connect db
     */
    public Connection getConnectionManager() {
        return this.cn;
    }
}

```

La classe ConnectDB permet de connecter a la base de donnée mysql par un jdbc de java. Il est instancié qu'une seule fois pour tout la durée de l'utilisation donc une utilise la pattern Singleton pour cela.

2.2 Model d'une table en java : Flight (Model)

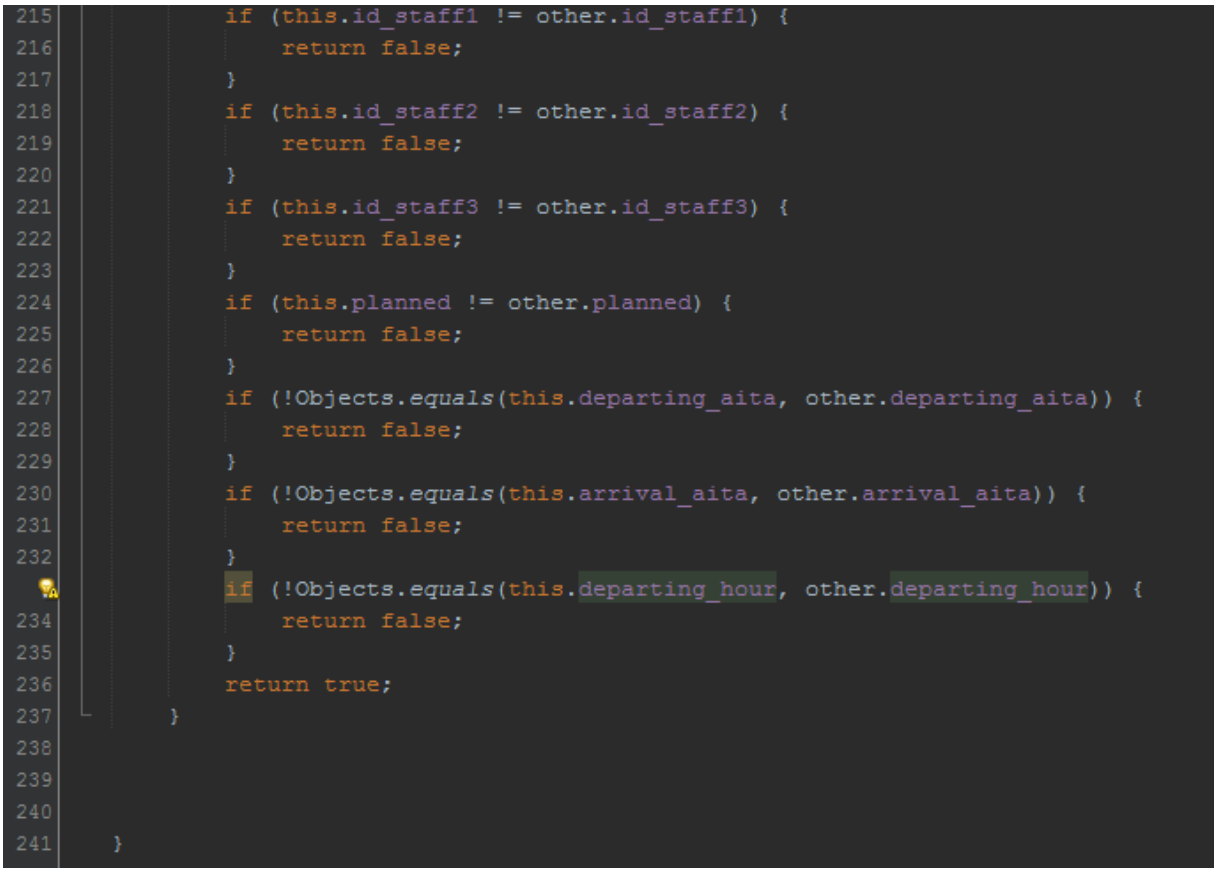
```java
 *
 * @author Salim El Moussaoui <salim.elmoussaoui.afpa2017@gmail.com>
 */
public class Flight {
    // name columns in table flights
    private long id;
    private String departing_aita;
    private String arrival_aita;
    private String departing_hour;
    private int duration;
    private double price;
    private long id_pilot;
    private long id_copilot;
    private long id_staff1;
    private long id_staff2;
    private long id_staff3;
    private boolean planned;

    // this construct create flight empty
    public Flight() {
        this.id = 0;
        this.departing_aita = "";
        this.arrival_aita = "";
        this.departing_hour = "";
        this.duration = 0;
        this.price = 0;
        this.id_pilot = 0;
        this.id_copilot = 0;
        this.id_staff1 = 0;
        this.id_staff2 = 0;
        this.id_staff3 = 0;
        this.planned = false;
    }
```

```java
    // this construct create flight full
    public Flight(
            long id, String departing_aita, String arrival_aita,
            String departing_hour, int duration, double price, long id_pilot,
            long id_copilot, long id_staff1, long id_staff2, long id_staff3,
            boolean planned) {
        this.id = id;
        this.departing_aita = departing_aita;
        this.arrival_aita = arrival_aita;
        this.departing_hour = departing_hour;
        this.duration = duration;
        this.price = price;
        this.id_pilot = id_pilot;
        this.id_copilot = id_copilot;
        this.id_staff1 = id_staff1;
        this.id_staff2 = id_staff2;
        this.id_staff3 = id_staff3;
        this.planned = planned;
    }

    public long getId() {
        return id;
    }

    public String getDeparting_aita() {
        return departing_aita;
    }

    public String getArrival_aita() {
        return arrival_aita;
    }

    public String getDeparting_hour() {
        return departing_hour;
    }

    public int getDuration() {
        return duration;
    }

    public double getPrice() {
        return price;
    }

```

```
88      public long getId_pilot() {
89          return id_pilot;
90      }
91
92      public long getId_copilot() {
93          return id_copilot;
94      }
95
96      public long getId_staff1() {
97          return id_staff1;
98      }
99
100     public long getId_staff2() {
101         return id_staff2;
102     }
103
104     public long getId_staff3() {
105         return id_staff3;
106     }
107
108     public boolean getPlanned() {
109         return planned;
110     }
111
112     public void setId(long id) {
113         this.id = id;
114     }
115     public void setDeparting_aita(String departing_aita) {
116         this.departing_aita = departing_aita;
117     }
118
119     public void setArrival_aita(String arrival_aita) {
120         this.arrival_aita = arrival_aita;
121     }
122
123     public void setDeparting_hour(String departing_hour) {
124         this.departing_hour = departing_hour;
125     }
126
127     public void setDuration(int duration) {
128         this.duration = duration;
129     }
130
131     public void setPrice(double price) {
132         this.price = price;
133     }
```

```
135     public void setId_pilot(long id_pilot) {
136         this.id_pilot = id_pilot;
137     }
138
139     public void setId_copilot(long id_copilot) {
140         this.id_copilot = id_copilot;
141     }
142
143     public void setId_staff1(long id_staff1) {
144         this.id_staff1 = id_staff1;
145     }
146
147     public void setId_staff2(long id_staff2) {
148         this.id_staff2 = id_staff2;
149     }
150
151     public void setId_staff3(long id_staff3) {
152         this.id_staff3 = id_staff3;
153     }
154
155     public void setPlanned(boolean planned) {
156         this.planned = planned;
157     }
158
159     @Override
    public String toString() {
161         return "Flight{" + "id=" + id + ", departing_aita=" + departing_aita
162                 + ", arrival_aita=" + arrival_aita + ", departing_hour=" +
163                 departing_hour + ", duration=" + duration + ", price=" + price
164                 + ", id_pilot=" + id_pilot + ", id_copilot=" + id_copilot
165                 + ", id_staff1=" + id_staff1 + ", id_staff2=" + id_staff2
166                 + ", id_staff3=" + id_staff3 + ", planned=" + planned + '}';
167     }
```

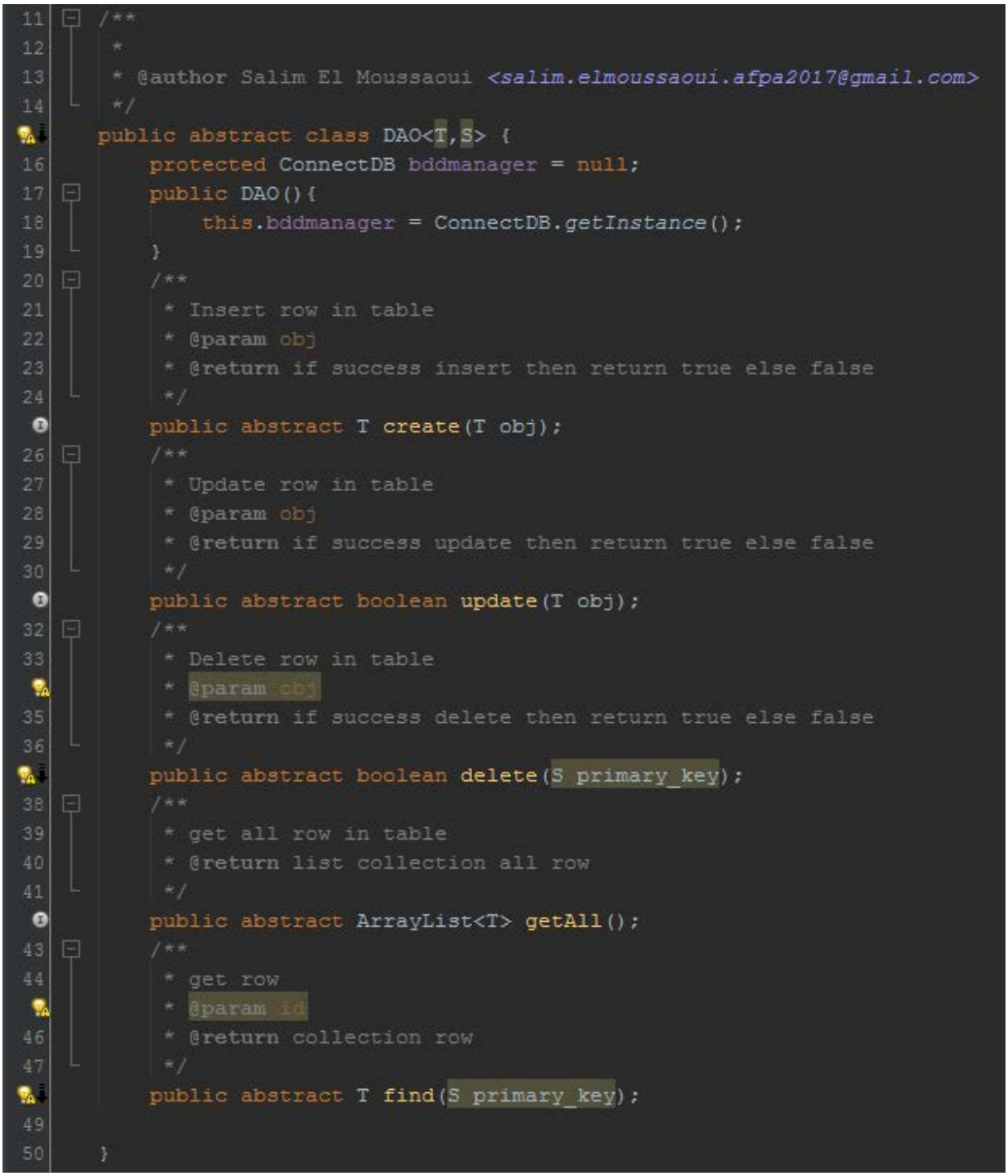
```java
    @Override
    public int hashCode() {
        int hash = 7;
        hash = 29 * hash + (int) (this.id ^ (this.id >>> 32));
        hash = 29 * hash + Objects.hashCode(this.departing_aita);
        hash = 29 * hash + Objects.hashCode(this.arrival_aita);
        hash = 29 * hash + Objects.hashCode(this.departing_hour);
        hash = 29 * hash + this.duration;
        hash = 29 * hash + (int) (Double.doubleToLongBits(this.price) ^
                                  (Double.doubleToLongBits(this.price) >>> 32));
        hash = 29 * hash + (int) (this.id_pilot ^ (this.id_pilot >>> 32));
        hash = 29 * hash + (int) (this.id_copilot ^ (this.id_copilot >>> 32));
        hash = 29 * hash + (int) (this.id_staff1 ^ (this.id_staff1 >>> 32));
        hash = 29 * hash + (int) (this.id_staff2 ^ (this.id_staff2 >>> 32));
        hash = 29 * hash + (int) (this.id_staff3 ^ (this.id_staff3 >>> 32));
        hash = 29 * hash + (this.planned ? 1 : 0);
        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 Flight other = (Flight) obj;
        if (this.id != other.id) {
            return false;
        }
        if (this.duration != other.duration) {
            return false;
        }
        if (Double.doubleToLongBits(this.price) != Double.doubleToLongBits(other.price)) {
            return false;
        }
        if (this.id_pilot != other.id_pilot) {
            return false;
        }
        if (this.id_copilot != other.id_copilot) {
            return false;
        }
```



```
215             if (this.id_staff1 != other.id_staff1) {
216                 return false;
217             }
218             if (this.id_staff2 != other.id_staff2) {
219                 return false;
220             }
221             if (this.id_staff3 != other.id_staff3) {
222                 return false;
223             }
224             if (this.planned != other.planned) {
225                 return false;
226             }
227             if (!Objects.equals(this.departing_aita, other.departing_aita)) {
228                 return false;
229             }
230             if (!Objects.equals(this.arrival_aita, other.arrival_aita)) {
231                 return false;
232             }
233             if (!Objects.equals(this.departing_hour, other.departing_hour)) {
234                 return false;
235             }
236             return true;
237         }
238
239
240
241     }
```

La classe Fligh est un model de la table flights dans la base de donnée, il ya tous les champs avec leur type. On a aussi les assesseurs et mutateurs qui permettrons de l'utiliser à l'extérieur.

2.3 La classe DAO



```
/**
 *
 * @author Salim El Moussaoui <salim.elmoussaoui.afpa2017@gmail.com>
 */
public abstract class DAO<T,S> {
    protected ConnectDB bddmanager = null;
    public DAO(){
        this.bddmanager = ConnectDB.getInstance();
    }
    /**
     * Insert row in table
     * @param obj
     * @return if success insert then return true else false
     */
    public abstract T create(T obj);
    /**
     * Update row in table
     * @param obj
     * @return if success update then return true else false
     */
    public abstract boolean update(T obj);
    /**
     * Delete row in table
     * @param obj
     * @return if success delete then return true else false
     */
    public abstract boolean delete(S primary_key);
    /**
     * get all row in table
     * @return list collection all row
     */
    public abstract ArrayList<T> getAll();
    /**
     * get row
     * @param id
     * @return collection row
     */
    public abstract T find(S primary_key);

}
```

Le DOA est un design pattern permet de faire la distinction entre les données auxquelles qu'on souhaite accéder et de la manière de les stockées. On utilise la méthode CRUD :
C (Créate) : on crée un vol
R (Read) : on sélection un ou plusieurs vol
U (Update) : on met a jour le vol
D (Delete) : on supprimer le vole
Lorsque qu'on crée un model DAO et qu'on étend la classe DAO , la classe fille doit obligatoirement les méthodes mère car la Classe DAO est abstract, il y a des généricités <T> pour le model et <S> pour le type de clé primaire du model.

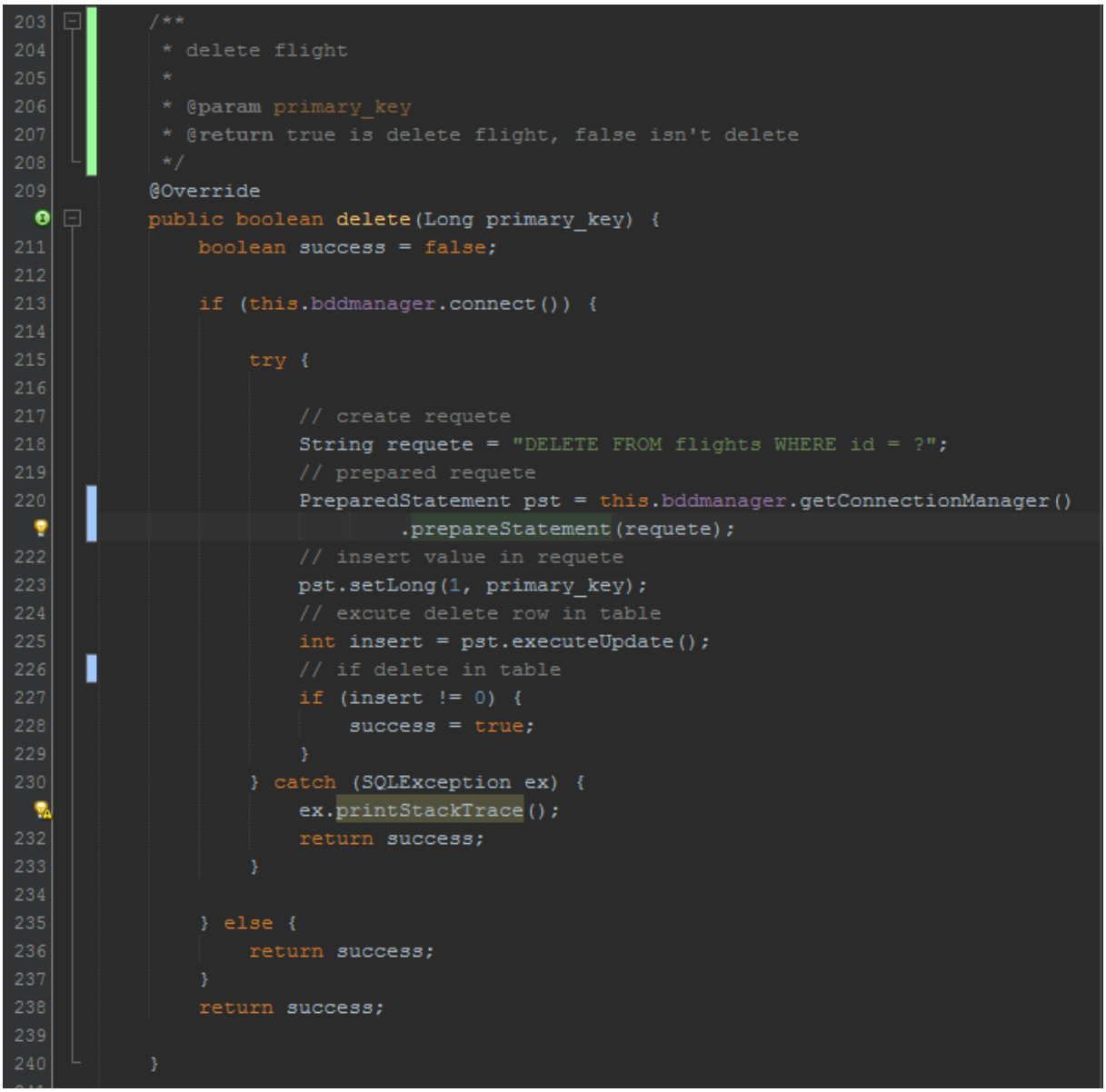
2.4 la classe FlightDAO

```java
 * @author Salim El Moussaoui <salim.elmoussaoui.afpa2017@gmail.com>
 */
public class FlightDAO extends DAO<Flight, Long> {

    public FlightDAO() {
        super();
    }
    /**
     *
     * Create flight
     *
     * @param flight
     * @return flight object
     */
    @Override
    public Flight create(Flight flight) {

        Flight flightCreate = new Flight();
        if (this.bddmanager.connect()) {

            try {

                // create requete
                String requete = "INSERT INTO flights ( "
                        + "departing_aita,\n"
                        + " arrival_aita,\n"
                        + " departing_hour,\n"
                        + " duration,\n"
                        + " price,\n"
                        + " id_pilot,\n"
                        + " id_copilot,\n"
                        + " id_staff1,\n"
                        + " id_staff2,\n"
                        + " id_staff3,\n"
                        + " planned"
                        + ") VALUES (?,?,?,?,?,?,?,?,?,?,?)";
                // prepared requete and get return generated key
                PreparedStatement pst = this.bddmanager.getConnectionManager()
                   .prepareStatement(requete, Statement.RETURN_GENERATED_KEYS);
                // insert value in requete
                pst.setString(1, flight.getDeparting_aita());
                pst.setString(2, flight.getArrival_aita());
                pst.setString(3, flight.getDeparting_hour());
                pst.setInt(4, flight.getDuration());
                pst.setDouble(5, flight.getPrice());
```

```
                // if it's null
                 if (flight.getId_pilot() == 0) {
                    pst.setNull(6, Types.BIGINT);
                } else {
                    pst.setLong(6, flight.getId_pilot());
                }
                if (Long.valueOf(flight.getId_copilot()) == 0) {

                    pst.setNull(7, Types.BIGINT);
                } else {
                    pst.setLong(7, flight.getId_copilot());
                }
                if (flight.getId_staff1() == 0) {
                    pst.setNull(8, Types.BIGINT);
                } else {
                    pst.setLong(8, flight.getId_staff1());
                }

                if (flight.getId_staff2() == 0) {
                    pst.setNull(9, Types.BIGINT);
                } else {
                    pst.setLong(9, flight.getId_staff2());
                }
                if (flight.getId_staff3() == 0) {
                    pst.setNull(10, Types.BIGINT);
                } else {
                    pst.setLong(10, flight.getId_staff3());
                }
                pst.setBoolean(11, flight.getPlanned());

                // excute insert row in table
                int insert = pst.executeUpdate();
                // if insert in table
                if (insert != 0) {
                    // get generate key
                    ResultSet id_increment = pst.getGeneratedKeys();
                    // if it's generate key
                    if (id_increment.next()) {
                        // assign key in fligh object
                        flight.setId(id_increment.getInt(1));
                        // assign object fligh in object return
                        flightCreate = flight;
                    }

                }
            } catch (SQLException ex) {
                ex.printStackTrace();
                return flightCreate;
            }

        } else {
            return flightCreate;
        }

        return flightCreate;
    }
```

```java
    /**
     * Update flight
     *
     * @param flight
     * @return true is update flight, false isn't update
     */
    @Override
    public boolean update(Flight flight) {
        boolean success = false;

        if (this.bddmanager.connect()) {

            try {

                // create requete
                String requete = "Update flights set departing_aita = ?,\n"
                        + " arrival_aita = ?,\n"
                        + " departing_hour = ?,\n"
                        + " duration = ?,\n"
                        + " price = ?,\n"
                        + " id_pilot = ?,\n"
                        + " id_copilot = ?,\n"
                        + " id_staff1 = ?,\n"
                        + " id_staff2 = ?,\n"
                        + " id_staff3 = ?,\n"
                        + " planned = ?"
                        + " WHERE id = ?";
                // prepared requete
                PreparedStatement pst = this.bddmanager
                        .getConnectionManager().prepareStatement(requete);
                // insert value in requete
                pst.setString(1, flight.getDeparting_aita());
                pst.setString(2, flight.getArrival_aita());
                pst.setString(3, flight.getDeparting_hour());
                pst.setInt(4, flight.getDuration());
                pst.setDouble(5, flight.getPrice());
                // if it's null
                if (flight.getId_pilot() == 0) {
                    pst.setNull(6, Types.BIGINT);

                } else {
                    pst.setLong(6, flight.getId_pilot());
                }
                if (Long.valueOf(flight.getId_copilot()) == 0) {

                    pst.setNull(7, Types.BIGINT);
                } else {
```

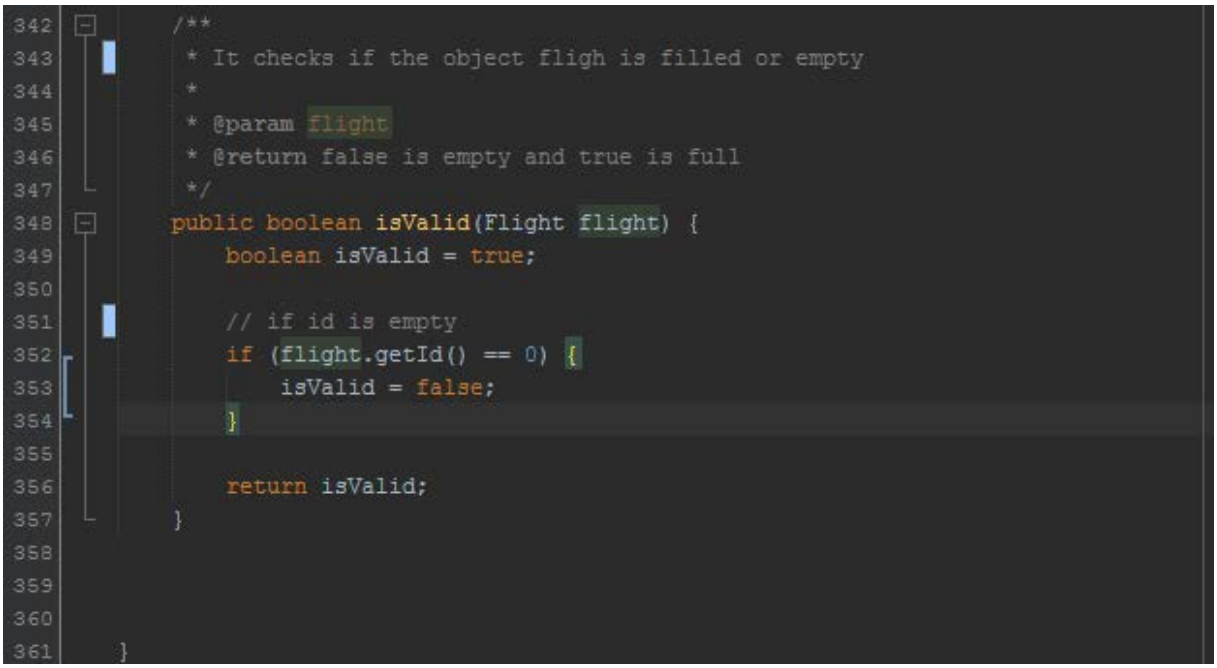
```
                    pst.setLong(7, flight.getId_copilot());
                }
                if (flight.getId_staff1() == 0) {
                    pst.setNull(8, Types.BIGINT);
                } else {
                    pst.setLong(8, flight.getId_staff1());
                }

                if (flight.getId_staff2() == 0) {
                    pst.setNull(9, Types.BIGINT);
                } else {
                    pst.setLong(9, flight.getId_staff2());
                }
                if (flight.getId_staff3() == 0) {
                    pst.setNull(10, Types.BIGINT);
                } else {
                    pst.setLong(10, flight.getId_staff3());
                }
                pst.setBoolean(11, flight.getPlanned());
                pst.setLong(12, flight.getId());
                // excute update row in table
                int insert = pst.executeUpdate();
                // if insert in table
                if (insert != 0) {
                    success = true;
                }
            } catch (SQLException ex) {
                ex.printStackTrace();
                return success;
            }

        } else {
            return success;
        }
        return success;
    }
```



```java
    /**
     * delete flight
     *
     * @param primary_key
     * @return true is delete flight, false isn't delete
     */
    @Override
    public boolean delete(Long primary_key) {
        boolean success = false;

        if (this.bddmanager.connect()) {

            try {

                // create requete
                String requete = "DELETE FROM flights WHERE id = ?";
                // prepared requete
                PreparedStatement pst = this.bddmanager.getConnectionManager()
                        .prepareStatement(requete);
                // insert value in requete
                pst.setLong(1, primary_key);
                // excute delete row in table
                int insert = pst.executeUpdate();
                // if delete in table
                if (insert != 0) {
                    success = true;
                }
            } catch (SQLException ex) {
                ex.printStackTrace();
                return success;
            }

        } else {
            return success;
        }
        return success;

    }
```

```java
    /**
     * get all flights
     *
     * @return all flights
     */
    @Override
    public ArrayList getAll() {
         // create array list airport empty
        ArrayList<Flight> listFlight = new ArrayList<>();
        if (this.bddmanager.connect()) {

            try {
                 // create statement
                Statement st = this.bddmanager
                        .getConnectionManager()
                        .createStatement();
                 // create requete
                String requete = "SELECT * FROM flights";
                // excute requete
                ResultSet rs = st.executeQuery(requete);
                // insert all airports in array object airport
                while (rs.next()) {
                    Flight el = new Flight(
                            rs.getInt("id"),
                            rs.getString("departing_aita"),
                            rs.getString("Arrival_aita"),
                            rs.getString("departing_hour"),
                            rs.getInt("duration"),
                            rs.getDouble("price"),
                            rs.getInt("id_pilot"),
                            rs.getInt("id_copilot"),
                            rs.getInt("id_staff1"),
                            rs.getInt("id_staff2"),
                            rs.getInt("id_staff3"),
                            rs.getBoolean("planned")
                    );
                    listFlight.add(el);

                }

            } catch (SQLException ex) {
                ex.printStackTrace();
                return listFlight;
            }

        } else {
            return listFlight;
        }

        return listFlight;
    }
```

```java
/**
 * find flight
 *
 * @param primary_key
 * @return flight
 */
@Override
public Flight find(Long primary_key) {
    // create array airport empty
    Flight flight = new Flight();
    //check if connect db
    if (this.bddmanager.connect()) {

        try {
            // create statement for find
            Statement st = this.bddmanager.getConnectionManager()
                    .createStatement();
            // create requete add primary key
            String requete = "SELECT * FROM flights WHERE id = " + primary_key;
             // excute requete
            ResultSet rs = st.executeQuery(requete);
              // if result is full
            if (rs.next()) {
                // insert airports in object
                flight.setId(rs.getLong("id"));
                flight.setDeparting_aita(rs.getString("departing_aita"));
                flight.setArrival_aita(rs.getString("Arrival_aita"));
                flight.setDeparting_hour(rs.getString("departing_hour"));
                flight.setDuration(rs.getInt("duration"));
                flight.setPrice(rs.getDouble("price"));
                flight.setId_pilot(rs.getLong("id_pilot"));
                flight.setId_copilot(rs.getLong("id_copilot"));
                flight.setId_staff1(rs.getLong("id_staff1"));
                flight.setId_staff2(rs.getLong("id_staff2"));
                flight.setId_staff3(rs.getLong("id_staff3"));
                flight.setPlanned(rs.getBoolean("planned"));
            }

        } catch (SQLException ex) {
            ex.printStackTrace();
            return flight;
        }

    } else {
        return flight;
    }

    return flight;
}
```



```
342     /**
343      * It checks if the object fligh is filled or empty
344      *
345      * @param flight
346      * @return false is empty and true is full
347      */
348     public boolean isValid(Flight flight) {
349         boolean isValid = true;
350
351         // if id is empty
352         if (flight.getId() == 0) {
353             isValid = false;
354         }
355
356         return isValid;
357     }
358
359
360
361 }
```

La Classe FlightDAO comme son nom l'indique on étend la classe DAO en utilisant le model Flight comme généricité <T> et le la clé primaire de type Long car coté db est un type BIGINT.

Le CRUD il y est, on juste ajouter une méthode isValid pour vérifier si le vol est valide on retourne TRUE ou FALSE en vérifiant ID si il est égal a 0 ca veux dire que le vol n'a pas été enregistré dans la base de donnée.

Test Unitaire de la Classe FlightDAO

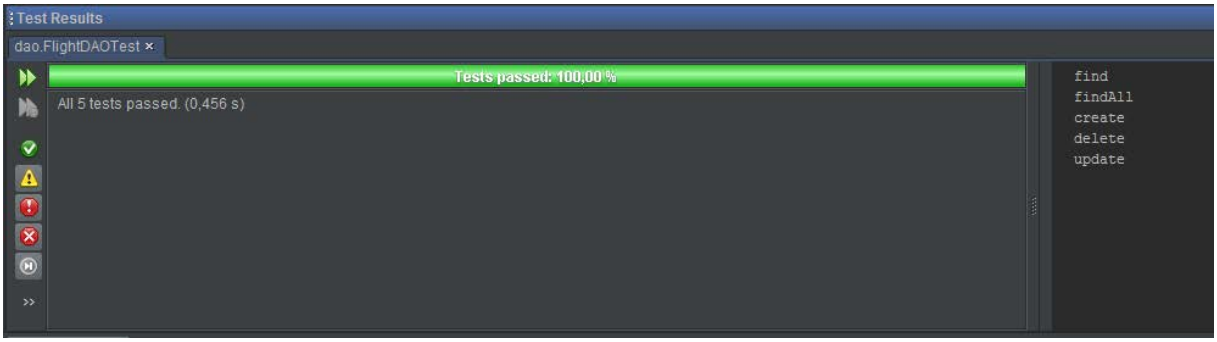
```java
/**
 *
 * @author Salim El Moussaoui <salim.elmoussaoui.afpa2017@gmail.com>
 */
public class FlightDAOTest {

    /**
     * Test of create method, of class FlightDAO.
     */
    @Test
    public void testCreate() {
        System.out.println("create");
        // create object flight
        Flight flight = new Flight(0, "MRS", "FNJ",
                "2017-03-20 05:30:00.0", 400, 275.75, 1, 2, 3, 4, 5, true);
        FlightDAO flightDAO = new FlightDAO();
        // find flight  create
        Flight expResult = flightDAO.find(flight.getId());

        // if find aita is empty or null
        if (!flightDAO.isValid(expResult)) {
            // insert flight in table
            Flight result = flightDAO.create(flight);
            // find flight
            expResult = flightDAO.find(flight.getId());
            // delete flight test
            flightDAO.delete(result.getId());
            assertEquals(expResult, result);

        }

    }
```

```java
    /**
     * Test of update method, of class FlightDAO.
     */
    @Test
    public void testUpdate() {
        System.out.println("update");
        // create object flight
        Flight flightInsert = new Flight(0, "FNJ", "LAS", "2017-03-20 05:30:00",
                400, 478.75, 0, 0, 0, 0, 0, false);
        // instatied flightDAO
        FlightDAO flightDAO = new FlightDAO();
        // find flight create
        Flight findFlight = flightDAO.find(flightInsert.getId());
        // if find flight is empty
        if (!flightDAO.isValid(findFlight)) {
            // Insert flight in table
            Flight resultFlight = flightDAO.create(flightInsert);
            // find flight
            findFlight = flightDAO.find(resultFlight.getId());
        }
            // modify flight
            findFlight.setDeparting_aita("MRS");
            findFlight.setArrival_aita("FNJ");
            findFlight.setDeparting_hour("2017-03-21 05:30:00.0");
            findFlight.setDuration(400);
            findFlight.setPrice(278.75);
            findFlight.setId_pilot(1);
            findFlight.setId_copilot(0);
            findFlight.setId_staff1(0);
            findFlight.setId_staff2(0);
            findFlight.setId_staff3(5);
            findFlight.setPlanned(true);

        // update flight
        boolean result = flightDAO.update(findFlight);
        boolean expResult = true;
        // delete flight test
        flightDAO.delete(findFlight.getId());
        assertEquals(expResult, result);
    }
```

```java
    /**
     * Test of delete method, of class FlightDAO.
     */
    @Test
    public void testDelete() {
        System.out.println("delete");
        FlightDAO flightDAO = new FlightDAO();

        // create object flight
        Flight flightInsert = new Flight(0, "TLS", "MRS", "2017-03-20 05:30:00",
                250, 175.75, 1, 2, 3, 4, 5, true);
        // find flight  create
        Flight findFlight = flightDAO.find(flightInsert.getId());

        // if find flight is empty
        if (!flightDAO.isValid(findFlight)) {
            // Insert flight in table
            Flight resultFlight = flightDAO.create(flightInsert);
            // find flight
            findFlight = flightDAO.find(resultFlight.getId());
            // delete flight
            boolean result = flightDAO.delete(findFlight.getId());
            boolean expResult = true;
            assertEquals(expResult, result);
        }
    }
```

```java
    /**
     * Test of findAll method, of class FlightDAO.
     */
    @Test
    public void testGetdAll() {
        System.out.println("findAll");
        FlightDAO flightDAO = new FlightDAO();
        ArrayList<Flight> arrayFlight = flightDAO.getAll();
        String expResult = "";
        String result = "";
        for (int i = 0; i < arrayFlight.size(); i++) {
            expResult += flightDAO.find(arrayFlight.get(i).getId());
            result += arrayFlight.get(i);
        }
        assertEquals(expResult, result);

    }
```

```java
    /**
     * Test of find method, of class FlightDAO.
     */
    @Test
    public void testFind() {
        System.out.println("find");
        FlightDAO flightDAO = new FlightDAO();
        // assign primary key
        long primary_key = 5;
        // assign result exemple
        String expResult = "Flight{id=5, departing_aita=DXB, arrival_aita=ICN, "
                + "departing_hour=2017-05-01 08:00:00.0, duration=420, "
                + "price=358.67, id_pilot=1, id_copilot=0, id_staff1=4, "
                + "id_staff2=0, id_staff3=0, planned=false}";

        // find flight
        String result = flightDAO.find(primary_key).toString();

        assertEquals(expResult, result);
    }
}
```



On fait un test unitaire sur la classe FlightDAO avec JUNIT, on test tout les méthodes.