Documentation itération3

Insertion root

Méthode insert (Employe..) Permettant d'insérer un employé

```
5   5           public GestionPersonnel getGestionPersonnel();
6   6           public void sauvegarderGestionPersonnel(GestionPersonnel gestionPersonnel)  throws SauvegardeImpossible;
7   7           public int insert(Ligue ligue) throws SauvegardeImpossible;
    8   +       public int insert(Employe employe) throws SauvegardeImpossible;
8   9       }
```

Ajout de la variable id dans Employe (ligne 18 jusqu'à ligne 74)

Implémenter cette méthode dans la classe JDBC

Ajout insert(Employe employe) pour transmettre l'ordre d'insertion à la passerelle dans GestionPersonnel.

```
    106 +       int insert(Employe employe) throws SauvegardeImpossible
    107 +       {
    108 +       return passerelle.insert(employe);
    109 +       }
```

Créez une méthode d'instance addRoot(...) dans la classe GestionPersonnel permettant de créer le root à partir de son nom et de son mot de passe, puis de l'affecter à la variable d'instance root de la classe GestionPersonnel.

```
28
29          public void addRoot(String nom, String motDePasse) {
30                  root = new Employe(this, null, nom, "", "", motDePasse, null, null);
31          }
```

Modification ligue

Même procéder que la modification root, les codes se suivent peut etre pas tous, à vérifier.

Modification root

Dans la Passerelle ajout d'une méthode update (Employe employe) permettant de modifier un employé

Implémentation dans la classe JDBC

```
public interface Passerelle {
    public GestionPersonnel getGestionPersonnel();

    public void sauvegarderGestionPersonnel(GestionPersonnel gestionPersonnel) throws SauvegardeImpossible;

    public int insert(Ligue ligue) throws SauvegardeImpossible;

    public void update(Employe employe) throws SauvegardeImpossible;
}
```

Création d'une méthode update(Employe employe) dans la classe GestionPersonnel transmettant l'ordre de modification à la passerelle

```
64  +
65  +       public void update(Employe employe) throws SauvegardeImpossible {
66  +               passerelle.update(employe);
67  +       }
```

Appel à gestionPersonnel.update(employe) insérer dans chaque setter de la classe Employé.

Insertions employées

Insérer un employé dans la base de données (JDBC)

```
90  + }
91  +       @Override
92  + public int insert(Employe employe) throws SauvegardeImpossible
93  + {
94  +     try
95  +     {
96  +         PreparedStatement instruction;
97  +         instruction = connection.prepareStatement(
98  +             "INSERT INTO employe (nom, prenom, mail, password, idLigue) VALUES (?, ?, ?, ?, ?)",
99  +             Statement.RETURN_GENERATED_KEYS
100 +         );
101 +
102 +         instruction.setString(1, employe.getNom());
103 +         instruction.setString(2, employe.getPrenom());
104 +         instruction.setString(3, employe.getMail());
105 +         instruction.setString(4, employe.checkPassword(employe.getMail()) ? employe.getMail() : employe.getMail()); // À modifier si besoin
106 +         instruction.setInt(5, employe.getLigue() != null ? employe.getLigue().getId() : null);
107 +
108 +         instruction.executeUpdate();
109 +         ResultSet id = instruction.getGeneratedKeys();
110 +         id.next();
111 +         return id.getInt(1);
112 +     }
113 +     catch (SQLException exception)
114 +     {
115 +         exception.printStackTrace();
116 +         throw new SauvegardeImpossible(exception);
117 +     }
118 + }
```

Suppressions employées (non)

Suppression ligue

Adapter le code pour que l'on puisse supprimer une ligue de la base de données. (Aussi supp dans GestionPersonnel)

Ligue supp dans passerelle

```
7   7           public int insert(Ligue ligue) throws SauvegardeImpossible;
8   8           public int insert(Employe employe) throws SauvegardeImpossible;
9   9           public void update(Ligue ligue) throws SauvegardeImpossible;
    10  +       public void remove (Ligue ligue) throws SauvegardeImpossible;
10  11      }
```

Ligue supp dans ligue.java

```
142     -       public void remove()
143     -       {
144     -               gestionPersonnel.remove(this);
145     -       }
146     -
    142 +       public void remove() {
    143 +   if (this.administrateur != null) {
    144 +       Employe root = gestionPersonnel.getRoot();
    145 +       if (this.administrateur != root) {
    146 +           setAdministrateur(root);
    147 +       }
    148 +   }
    149 +   for (Employe employe : employes) {
    150 +       employe.setLigue(null);
    151 +   }
    152 +   for (Employe employe : employes) {
    153 +       gestionPersonnel.removeEmploye(employe);
    154 +   }
    155 +   gestionPersonnel.remove(this);
```

Ligue supp dans employé

```
178 +         public void setLigue(Ligue ligue) {
179 +     this.ligue = ligue;
180 +     try {
181 +         gestionPersonnel.getPasserelle().updateLigue(this);
182 +     } catch (SauvegardeImpossible e) {
183 +         System.out.println("Erreur lors de la mise à jour de la ligue de l'employé : " + e.getMessage());
184 +     }
185 + }
```

Modifications employées (probleme)

Adaptation du code pour que l'on puisse modifier les données d'un employé déjà présent dans la base de données.

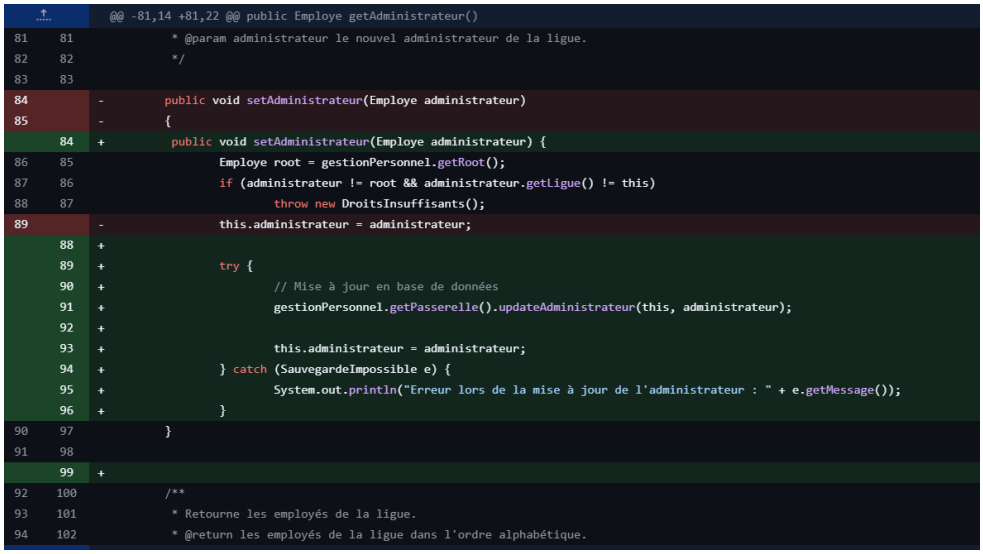
Lecture administrateur

Ecriture administrateur

Adaptation du code pour enregistrer dans la base de données un changement d'administrateur

```
200 + // changement d'admin dans la BDD
201 + public void updateAdministrateur(Ligue ligue, Employe nouvelAdmin) throws SauvegardeImpossible {
202 +         try {
203 +                 // Étape 1 : Désactiver l'ancien administrateur de la ligue
204 +                 String resetAdminSQL = "UPDATE UTILISATEUR SET idType = 3 WHERE numLigue = ? AND idType = 2";
205 +                 PreparedStatement resetAdminStmt = connection.prepareStatement(resetAdminSQL);
206 +                 resetAdminStmt.setInt(1, ligue.getId());
207 +                 resetAdminStmt.executeUpdate();
208 +
209 +                 // Étape 2 : Nommer le nouvel administrateur
210 +                 String updateAdminSQL = "UPDATE UTILISATEUR SET idType = 2 WHERE idUtilisateur = ?";
211 +                 PreparedStatement updateAdminStmt = connection.prepareStatement(updateAdminSQL);
212 +                 updateAdminStmt.setInt(1, nouvelAdmin.getId());
213 +                 updateAdminStmt.executeUpdate();
214 +
215 +                 // Étape 3 : Mettre à jour l'objet en mémoire
216 +                 ligue.setAdministrateur(nouvelAdmin);
217 +         }
218 +         catch (SQLException e) {
219 +                 throw new SauvegardeImpossible(e);
220 +         }
221 + }
222 + }
223 +
```

Puis maj dans la BD



```
@@ -81,14 +81,22 @@ public Employe getAdministrateur()
81  81           * @param administrateur le nouvel administrateur de la ligue.
82  82           */
83  83
84      -        public void setAdministrateur(Employe administrateur)
85      -        {
    84  +        public void setAdministrateur(Employe administrateur) {
86  85                   Employe root = gestionPersonnel.getRoot();
87  86                   if (administrateur != root && administrateur.getLigue() != this)
88  87                           throw new DroitsInsuffisants();
89      -                this.administrateur = administrateur;
    88  +
    89  +                try {
    90  +                        // Mise à jour en base de données
    91  +                        gestionPersonnel.getPasserelle().updateAdministrateur(this, administrateur);
    92  +
    93  +                        this.administrateur = administrateur;
    94  +                } catch (SauvegardeImpossible e) {
    95  +                        System.out.println("Erreur lors de la mise à jour de l'administrateur : " + e.getMessage());
    96  +                }
90  97           }
91  98
    99  +
92  100          /**
93  101           * Retourne les employés de la ligue.
94  102           * @return les employés de la ligue dans l'ordre alphabétique.
```