CHANTRAINE VINCENT
02/04/24

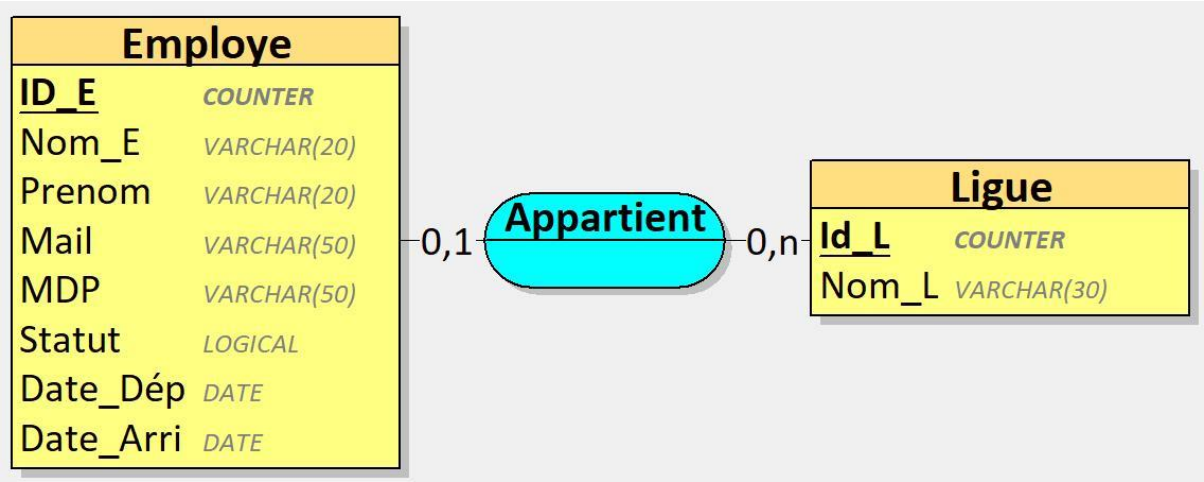
AP2-Personnel

Nous avons avec mes camarades de groupe, apporté des modifications à une application en java, des nouvelles fonctionnalités comme :

- La gestion des ligues (insertion, modification, suppression)
- La gestion des employés (insertion, modification, suppression)
- La gestion de l'administrateur (en ligne de commande/interface du terminal)
- La gestion des dates, mot de passes, tests unitaires
- La création la base de données correspondante et du lien entre les deux
- La modélisation de l'interface graphique avec des maquettes.

1- Création de la base de données (mcd, script, admin) :

- Le mcd :



AP2-Personnel

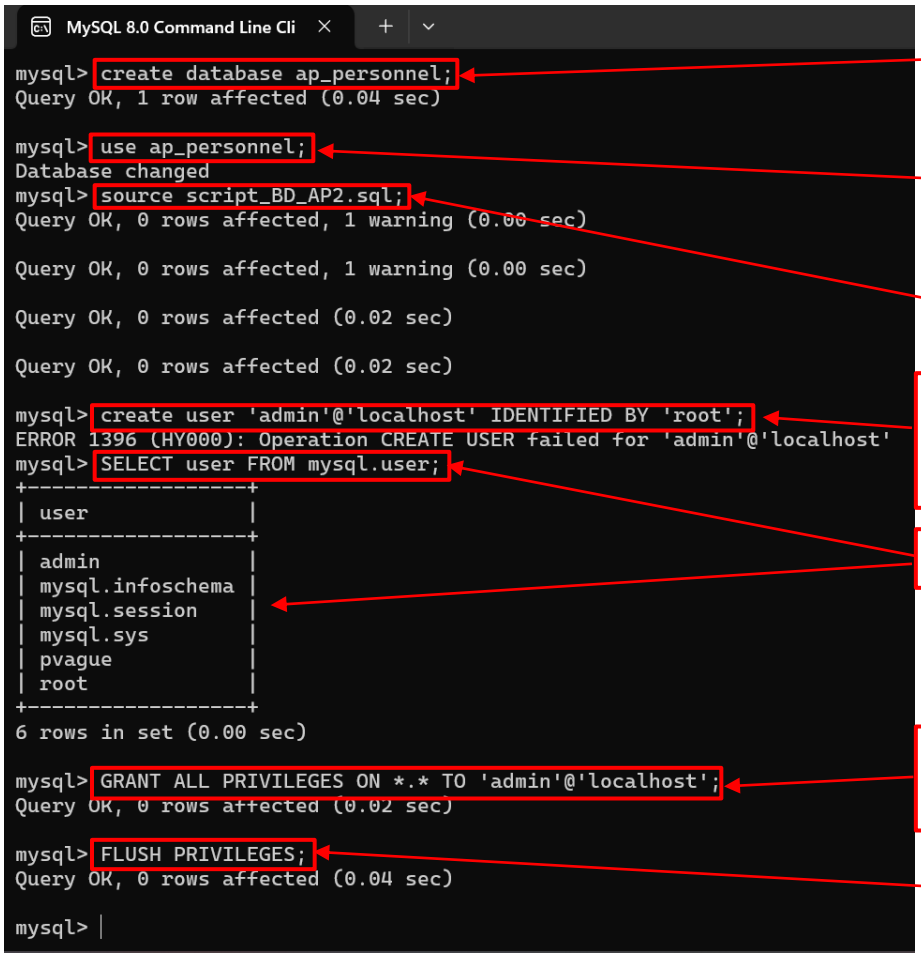
- Le script :

Placer le fichier du script dans le dossier bin de mysql (C : > Programmes > MySQL Server 8.0 > mysql > bin)

```
/*-----------------------Tables-----------------------*/
drop table if exists Ligue;
drop table if exists Employer;

CREATE TABLE Ligue(
   Id_L SMALLINT NOT NULL AUTO_INCREMENT,
   Nom_L VARCHAR(30),
   PRIMARY KEY(Id_L)
)ENGINE = INNODB;

CREATE TABLE Employe(
   ID_E SMALLINT NOT NULL AUTO_INCREMENT,
   Nom_E VARCHAR(20),
   Prenom VARCHAR(20),
   Mail VARCHAR(50),
   MDP VARCHAR(50),
   Statut VARCHAR(5),
   Date_Dep Date,
   Date_Arri Date,
   Id_L SMALLINT NULL,
   PRIMARY KEY(ID_E),
   FOREIGN KEY(Id_L) REFERENCES Ligue(Id_L)
)ENGINE = INNODB;

/*-------------------Test Enregistrement---------------*/

/*Delete from Ligue;
INSERT INTO Ligue (Id_L, Nom_L)
VALUES
   (1,"?");
select * from Ligue;

Delete from Employer;
INSERT INTO Employer (ID_E, Nom_E, Prenom, Mail, MDP, Status,
Date_Dep, Date_Arri, Id_L)
VALUES
   (1,"?","?","?","?","?","2000-00-00","2000-00-00",1);
select * from Employer;*/
```

AP2-Personnel

- La création :



Nous allons créer la database

Puis la sélectionner

Et lancer le script (placé dans bin)

Créer l'administrateur (déjà fait dans mon cas d'où l'erreur)

Afficher les utilisateurs

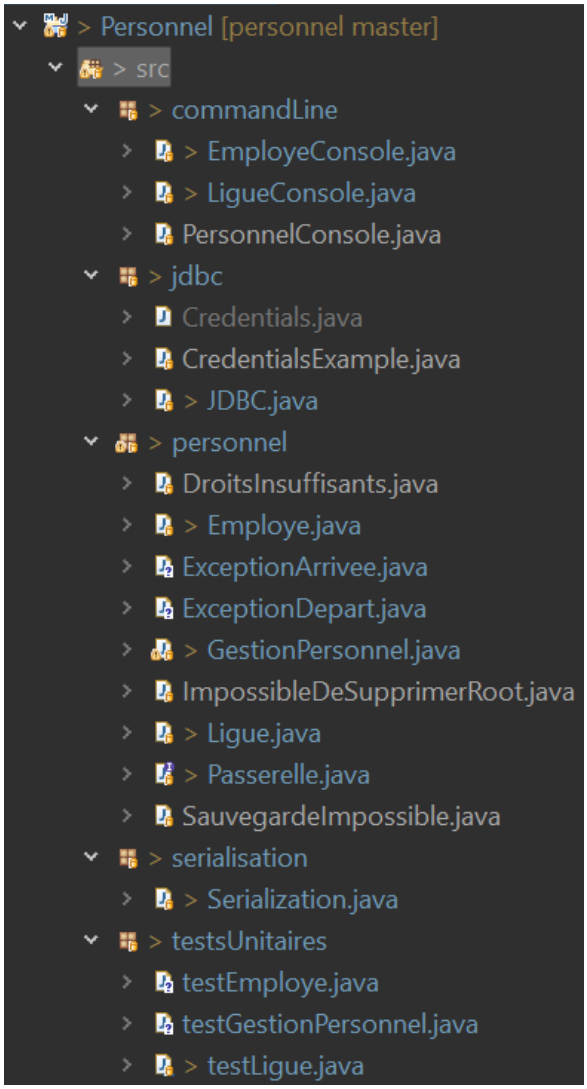
Attribuer tous les droits à l'administrateur

Activer ces droits

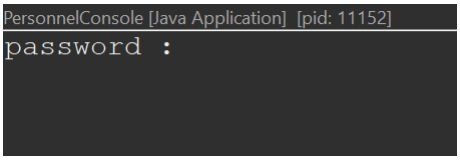
AP2-Personnel

2- L'application JAVA :

Arbre des différentes classes :

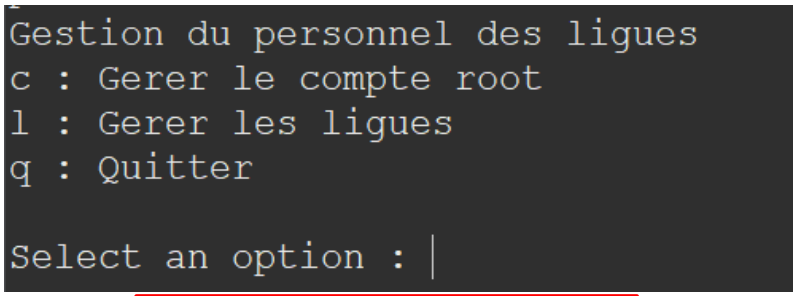


Vu du terminal :




```
public boolean checkPassword(String password)
{
    //System.out.println(this.password); //pour afficher le mdp
    return this.password.equals(password);
}
```

Pour pouvoir lancer l'application il a fallu trouvé le mdp pour cela on à dû faire un affichage du mdp dans la méthode checkPassword de la classe Employe après quoi le mdp s'afficher dans le terminal il n'y avait plus cas le saisir



Affichage après saisi du mot de passe

AP2-Personnel

CommandeLine

1) Dans EmployeConsole :

```
Option editerEmploye(Employe employe)
{
        Menu menu = new Menu("Gerer le compte " + employe.getNom(), "c");
        menu.add(afficher(employe));
        menu.add(changerNom(employe));
        menu.add(changerPrenom(employe));
        menu.add(changerMail(employe));
        menu.add(changerPassword(employe));
        menu.add(changerStatut(employe));
        menu.addBack("q");
        return menu;
}

private Option changerNom(final Employe employe)
{
    return new Option("Changer le nom", "n",
            () -> {employe.setNom(getString("Nouveau nom : "));}
        );
}

private Option changerPrenom(final Employe employe)
{
    return new Option("Changer le prenom", "p", () -> {employe.setPrenom(getString("Nouveau prenom : "));});
}

private Option changerMail(final Employe employe)
{
    return new Option("Changer le mail", "e", () -> {employe.setMail(getString("Nouveau mail : "));});
}

private Option changerPassword(final Employe employe)
{
    return new Option("Changer le password", "x", () -> {employe.setPassword(getString("Nouveau password : "));});
}

private Option changerStatut(final Employe employe)
{
    return new Option("Changer le statut", "s", () -> {employe.setStatut(getString("Nouveau statut : "));});
}
```

Ajout de changerStatut

2) Dans LigueConsole :

```
private Menu editerLigue(Ligue ligue)
{
    Menu menu = new Menu("Editer " + ligue.getNom());
    menu.add(afficher(ligue));
    menu.add(gererEmployes(ligue));
    menu.add(changerAdministrateur(ligue));
    menu.add(changerNom(ligue));
    menu.add(supprimer(ligue));
    menu.addBack("q");
    return menu;
}


private List<Employe> changerAdministrateur(final Ligue ligue)
{
    return new List<>("Changer l'administrateur", "c",
            () -> new ArrayList<>(ligue.getEmployes()),
            (index, element) -> {ligue.setAdministrateur(element);}
            );
}
```

Ajout de changerAdministrateur

Déjà précréer

AP2-Personnel

Serialisation

3) Dans serialization :

```
@Override
public int insert(Ligue ligue) throws SauvegardeImpossible
{return -1;}

public void delete(Ligue ligue) throws SauvegardeImpossible
{}

public void update(Ligue ligue) throws SauvegardeImpossible
{}

public int insert(Employe employe) throws SauvegardeImpossible
{return 0;}

public void delete(Employe employe) throws SauvegardeImpossible
{}

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

Ajout des 5 autres méthodes

testsUnitaires

4) Dans testLigue :

```
@Test
void getNom() throws SauvegardeImpossible
{
    Ligue ligue = gestionPersonnel.addLigue("Fléchettes");
    assertEquals("Fléchettes", ligue.getNom());
}

@Test
void setNom() throws SauvegardeImpossible
{
    Ligue ligue = gestionPersonnel.addLigue("Fléchettes");
    ligue.setNom("Billard");
    assertEquals("Billard", ligue.getNom());
}

@Test
void getAdministrateur() throws SauvegardeImpossible
{
    Ligue ligue = gestionPersonnel.addLigue("Fléchettes");
    Employe employe = ligue.addEmploye("Bouchard", "Gerard", "g.bouchard@gmail.com", "azerty");
    ligue.setAdministrateur(employe);
    assertEquals("root", employe.getNom());
}
```

Ajout de test unitaire

AP2-Personnel

5) Dans testEmploye (créer) :

```java
package testsUnitaires;

import static org.junit.jupiter.api.Assertions.*;

public class testEmploye {

    GestionPersonnel gestionPersonnel = GestionPersonnel.getGestionPersonnel();

    @Test
    void createEmploye() throws SauvegardeImpossible
    {
        Ligue ligue = gestionPersonnel.addLigue("Fléchettes");
        Employe employe = ligue.addEmploye("Bouchard", "Gérard", "g.bouchard@gmail.com", "azerty");
        assertEquals("Bouchard", employe.getNom());
        assertEquals("Gérard", employe.getPrenom());
        assertEquals("g.bouchard@gmail.com", employe.getMail());
        assertEquals(1, employe.checkPassword("azerty"));
    }

    @Test
    void NomEmploye() throws SauvegardeImpossible
    {
        Ligue ligue = gestionPersonnel.addLigue("Fléchettes");
        Employe employe = ligue.addEmploye("Bouchard", "Gérard", "g.bouchard@gmail.com", "azerty");
        assertEquals("Bouchard", employe.getNom());
    }

    @Test
    void PrenomEmploye() throws SauvegardeImpossible
    {
        Ligue ligue = gestionPersonnel.addLigue("Fléchettes");
        Employe employe = ligue.addEmploye("Bouchard", "Gérard", "g.bouchard@gmail.com", "azerty");
        assertEquals("Gérard", employe.getPrenom());
    }

    @Test
    void MailEmploye() throws SauvegardeImpossible
    {
        Ligue ligue = gestionPersonnel.addLigue("Fléchettes");
        Employe employe = ligue.addEmploye("Bouchard", "Gérard", "g.bouchard@gmail.com", "azerty");
        assertEquals("g.bouchard@gmail.com", employe.getMail());
    }
}
```

Ajout de test unitaire

6) Dans testGestionPersonnel (créer) :

```java
package testsUnitaires;

import static org.junit.jupiter.api.Assertions.*;

public class testGestionPersonnel {

    GestionPersonnel gestionPersonnel = GestionPersonnel.getGestionPersonnel();

    @Test
    void getLigue() throws SauvegardeImpossible
    {
        Ligue ligue = gestionPersonnel.addLigue("Fléchettes");
        //Employe employe = ligue.addEmploye("root", "", "g.bouchard@gmail.com", "azerty");
        assertEquals("root", ligue.getAdministrateur());
    }

    @Test
    void addLigue() throws SauvegardeImpossible
    {
        Ligue ligue = gestionPersonnel.addLigue("Flechettes");
        assertEquals("Flechettes", ligue.getNom());
    }

    @Test
    void addEmploye() throws SauvegardeImpossible
    {
        Ligue ligue = gestionPersonnel.addLigue("Fléchettes");
        Employe employe = ligue.addEmploye("Bouchard", "Gérard", "g.bouchard@gmail.com", "azerty");
        assertEquals(employe, ligue.getEmployes().first());
    }
}
```

Ajout de test unitaire

JDBC

AP2-Personnel

7) Dans Credentials :

```
package jdbc;

public class Credentials
{
    private static String driver = "mysql";
    private static String driverClassName = "com.mysql.cj.jdbc.Driver";
    private static String host = "localhost";
    private static String port = "3306";
    private static String database = "ap_personnel";
    private static String user = "admin";
    private static String password = "root";

    static String getUrl()
    {
        return "jdbc:" + driver + "://" + host + ":" + port + "/" + database +"?zeroDateTimeBehavior=CONVERT_TO_NULL&serverTimezone=UTC";
    }
```

Lien vers la base de données

Sinon erreur de fuseau horaire

8) Dans jdbc :

```
//Ligue

@Override
public int insert(Ligue ligue) throws SauvegardeImpossible
{
    try
    {
        PreparedStatement instruction;
        instruction = connection.prepareStatement("insert into ligue (nom_L) values(?)", Statement.RETURN_GENERATED_KEYS);
        instruction.setString(1, ligue.getNom());
        instruction.executeUpdate();
        ResultSet Id_L = instruction.getGeneratedKeys();
        Id_L.next();
        return Id_L.getInt(1);
    }
    catch (SQLException exception)
    {
        exception.printStackTrace();
        throw new SauvegardeImpossible(exception);
    }
}
```

Requête et communication avec la BDD

Pour correspondre à la BDD

```
@Override
public void delete(Ligue ligue) throws SauvegardeImpossible
{
    try
    {
        PreparedStatement instruction;
        instruction = connection.prepareStatement("delete from ligue where Nom_L = (?)", Statement.RETURN_GENERATED_KEYS);
        instruction.setString(1, ligue.getNom());
        instruction.executeUpdate();
    }
    catch (SQLException exception)
    {
        exception.printStackTrace();
        throw new SauvegardeImpossible(exception);
    }
}
```

```
@Override
public void update(Ligue ligue) throws SauvegardeImpossible
{
    try
    {
        PreparedStatement instruction;
        instruction = connection.prepareStatement("update ligue set nom_L=(?) where id_L = (?)", Statement.RETURN_GENERATED_KEYS);
        instruction.setString(1, ligue.getNom());
        instruction.setInt(2, ligue.getID());
        instruction.executeUpdate();
    }
    catch (SQLException exception)
    {
        exception.printStackTrace();
        throw new SauvegardeImpossible(exception);
    }
}
```

Insertion, suppression, modification des ligues

AP2-Personnel

Si c'est un admin ou non

```
//Employe

@Override
public int insert(Employe employe) throws SauvegardeImpossible
{
    try
    {
        Date dateArriveeSQL = Date.valueOf(employe.getDateArrivee());
        Date dateDepartSQL = Date.valueOf(employe.getDateDepart());
        PreparedStatement instruction;
        if (employe.getLigue()!=null) {
            instruction = connection.prepareStatement("insert into employe (nom_E, prenom, mail, mdp, Date_Dep, Date_Arri, Id_L) values(?, ?, ?, ?, ?, ?, ?)", Statement.RETURN_GENERATED_KEYS);
            instruction.setString(1, employe.getNom());
            instruction.setString(2, employe.getPrenom());
            instruction.setString(3, employe.getMail());
            instruction.setString(4, employe.getPassword());
            instruction.setDate(5, dateArriveeSQL);
            instruction.setDate(6, dateDepartSQL);
            instruction.setInt(7, employe.getLigue().getID());
            }
        else{
            instruction = connection.prepareStatement("insert into employe (nom_E, prenom, mail, mdp, statut, Date_Dep, Date_Arri) values(?, ?, ?, ?, ?, ?, ?)", Statement.RETURN_GENERATED_KEYS);
            instruction.setString(1, employe.getNom());
            instruction.setString(2, employe.getPrenom());
            instruction.setString(3, employe.getMail());
            instruction.setString(4, employe.getPassword());
            instruction.setString(5, employe.getStatut());
            instruction.setDate(6, dateArriveeSQL);
            instruction.setDate(7, dateDepartSQL);
        }
        instruction.executeUpdate();
        ResultSet Id_E = instruction.getGeneratedKeys();
        Id_E.next();
        return Id_E.getInt(1);

    }
    catch (SQLException exception)
    {
        exception.printStackTrace();
        throw new SauvegardeImpossible(exception);
    }
}
```

Insertion, suppression, modification des employés

```
@Override
public void delete(Employe employe) throws SauvegardeImpossible
{
    try
    {
        PreparedStatement instruction;
        instruction = connection.prepareStatement("delete from employe where id_E=  ?", Statement.RETURN_GENERATED_KEYS);
        instruction.setInt(1, employe.getID());
        instruction.executeUpdate();
    }
    catch (SQLException exception)
    {
        exception.printStackTrace();
        throw new SauvegardeImpossible(exception);
    }
}
```

```
@Override
public void update(Employe employe) throws SauvegardeImpossible
{
    try
    {
        Date dateArriveeSQL = Date.valueOf(employe.getDateArrivee());
        Date dateDepartSQL = Date.valueOf(employe.getDateDepart());
        PreparedStatement instruction;
        instruction = connection.prepareStatement("update employe set nom_E=(?), prenom=(?), mail=(?), mdp=(?), statut=(?), date_dep=(?), date_arri=(?) where id_E = (?)", Statement.RETURN_GENERATED_KEYS);
        instruction.setString(1, employe.getNom());
        instruction.setString(2, employe.getPrenom());
        instruction.setString(3, employe.getMail());
        instruction.setString(4, employe.getPassword());
        instruction.setString(5, employe.getStatut());
        instruction.setDate(6, dateArriveeSQL);
        instruction.setDate(7, dateDepartSQL);
        instruction.setInt(8, employe.getID());
        instruction.executeUpdate();
    }
    catch (SQLException exception)
    {
        exception.printStackTrace();
        throw new SauvegardeImpossible(exception);
    }
}
```

Personnel

9) Dans passerelle :

```
package personnel;

public interface Passerelle
{
    public GestionPersonnel getGestionPersonnel();
    public void sauvegarderGestionPersonnel(GestionPersonnel gestionPersonnel)  throws SauvegardeImpossible;
    public int insert(Ligue ligue) throws SauvegardeImpossible;
    public void delete(Ligue ligue) throws SauvegardeImpossible;
    public void update(Ligue ligue) throws SauvegardeImpossible;
    public int insert(Employe employe) throws SauvegardeImpossible;
    public void delete(Employe employe) throws SauvegardeImpossible;
    public void update(Employe employe) throws SauvegardeImpossible;
}
```

Lien des méthodes de jdbc avec gestionPersonnel

AP2-Personnel

10) Dans ExceptionDepart (créer) :

```
package personnel;

public class ExceptionDepart extends Exception{
        /**
     *
     */
    private static final long serialVersionUID = 1L;
        public ExceptionDepart()
        {
            System.out.println("Exception ExceptionDepart has been raised...");
        }
        @Override
        public String toString()
        {
          return "La date de départ ne peut pas etre avant la date d'arrivée ";
        }
}
```

Gestion des dates de départ

11) Dans ExceptionArrivee (créer) :

```
package personnel;

public class ExceptionArrivee extends Exception {
        /**
     *
     */
    private static final long serialVersionUID = 1L;
        public ExceptionArrivee()
        {
            System.out.println("Exception ExceptionArrivee has been raised...");
        }
        @Override
        public String toString()
        {
          return "La date d'arrivée ne peut pas etre avant la date de départ ";
        }
}
```

Gestion des dates d'arrivé

12) Dans GestionPersonnel :

```
public class GestionPersonnel implements Serializable
{
    private static final long serialVersionUID = -105283113987886425L;
    private static GestionPersonnel gestionPersonnel = null;
    private SortedSet<Ligue> ligues;
    private Employe root = new Employe(this, null, "root", "", "", "Admin", "toor", null, null);
    public final static int SERIALIZATION = 1, JDBC = 2,
            TYPE_PASSERELLE = JDBC;
    private static Passerelle passerelle = TYPE_PASSERELLE == JDBC ? new jdbc.JDBC() : new serialisation.Serialization();
```

Paramètre de l'administrateur et changement par JDBC

AP2-Personnel

```
int insert(Ligue ligue) throws SauvegardeImpossible
{
    return passerelle.insert(ligue);
}

void update(Ligue ligue) throws SauvegardeImpossible
{
    passerelle.update(ligue);
}

void delete(Ligue ligue) throws SauvegardeImpossible
{
    passerelle.delete(ligue);
}

int insert(Employe employe) throws SauvegardeImpossible
{
    return passerelle.insert(employe);
}

void update(Employe employe) throws SauvegardeImpossible
{
    passerelle.update(employe);
}

void delete(Employe employe) throws SauvegardeImpossible
{
    passerelle.delete(employe);
}
```

Ajout des 5 autres méthodes

12) Dans Ligue :

```
import java.time.LocalDate;
```

Importer les dates

```
public class Ligue implements Serializable, Comparable<Ligue>
{
    private static final long serialVersionUID = 1L;
    private int id = -1;
    private String nom, statut;
    private SortedSet<Employe> employes;
    private Employe administrateur;
    private GestionPersonnel gestionPersonnel;
    private LocalDate dateArrivee = LocalDate.of(0000, 01, 01);
    private LocalDate dateDepart = LocalDate.of(0000, 01, 01);
```

Ajout de statut et des dates

```
public int getID() {
    return id;
}
```

Ajout du getter de l'id

```
public void setNom(String nom)
{
    this.nom = nom;
    try
    {
        gestionPersonnel.update(this);
    }
    catch(SauvegardeImpossible e)
    {
        e.printStackTrace();
    }
}
```

Ajout des try/catch

AP2-Personnel

```
public Employe addEmploye(String nom, String prenom, String mail, String password)
{
    Employe employe = new Employe(this.gestionPersonnel, this, nom, prenom, mail, password, statut, dateArrivee, dateDepart);
    employes.add(employe);
    return employe;
}
```

Ajout de statut et des dates

```
public void remove()
{
    gestionPersonnel.remove(this);
    try {
        gestionPersonnel.delete(this);
    }
    catch(SauvegardeImpossible e) {
        e.printStackTrace();
    }
}
```

Ajout des try/catch

13) Dans Employe :

```
public class Employe implements Serializable, Comparable<Employe>
{
    private static final long serialVersionUID = 4795721718037994734L;
    private String nom, prenom, password, mail ,statut;
    private Ligue ligue;
    private GestionPersonnel gestionPersonnel;
    private LocalDate dateArrivee = LocalDate.of(0000, 01, 01);
    private LocalDate dateDepart = LocalDate.of(0000, 01, 01);
    private int id;

    Employe(GestionPersonnel gestionPersonnel, Ligue ligue, String nom, String prenom, String mail, String statut, String password, LocalDate dateArrivee, LocalDate dateDepart)
    {
        this.gestionPersonnel = gestionPersonnel;
        this.nom = nom;
        this.prenom = prenom;
        this.password = password;
        this.mail = mail;
        this.statut = statut;
        this.ligue = ligue;
        //this.dateArrivee = dateArrivee;
        //this.dateDepart = dateDepart;

        try {
        this.id = gestionPersonnel.insert(this);
        }
        catch(SauvegardeImpossible e)
        {
            e.printStackTrace();
        }
    }
```

Ajout du statut et des dates + try/catch

```
public GestionPersonnel getGestionPersonnel()
{
    return gestionPersonnel;
}
```

Getter gestionPersonnel

```
public int getID() {
    return id;
}
```

Getter de l'id

```
public void setNom(String nom)
{
    this.nom = nom;
    try {
        gestionPersonnel.update(this);
    }
    catch (SauvegardeImpossible e)
    {
        e.printStackTrace();
    }
}
```

Try/catch du setter de nom

```
public void setPrenom(String prenom)
{
    this.prenom = prenom;
    try {
        gestionPersonnel.update(this);
    }
    catch (SauvegardeImpossible e)
    {
        e.printStackTrace();
    }
}
```

Try/catch du setter de prénom

AP2-Personnel

```
public void setMail(String mail)
{
    this.mail = mail;
    try {
        gestionPersonnel.update(this);
    }
    catch (SauvegardeImpossible e)
    {
        e.printStackTrace();
    }
}
```

Try/catch du setter de mail

```
public String getStatut()
{
    return statut;
}

public void setStatut(String statut)
{
    this.statut = statut;
    try {
        gestionPersonnel.update(this);
    }
    catch (SauvegardeImpossible e)
    {
        e.printStackTrace();
    }
}
```

Getter de statut

Setter de statut

```
public String getPassword()
{
    return password;
}

/**
 * Change le password de l'employÃ©.
 * @param password le nouveau password de l'employÃ©.
 */

public void setPassword(String password)
{
    this.password= password;
    try {
        gestionPersonnel.update(this);
    }
    catch (SauvegardeImpossible e)
    {
        e.printStackTrace();
    }
}
```

Getter de password

Try/catch du setter de password

```
public void remove()
{
    Employe root = gestionPersonnel.getRoot();
    if (this != root)
    {
        if (estAdmin(getLigue()))
            getLigue().setAdministrateur(root);
        getLigue().remove(this);
    }
    else
        throw new ImpossibleDeSupprimerRoot();
    try
    {
        gestionPersonnel.delete(this);
    }
    catch(SauvegardeImpossible e)
    {
        e.printStackTrace();
    }
}
```

Else et Try/catch de remove()

AP2-Personnel

```java
public LocalDate getDateArrivee() {
    return dateArrivee;
}
public LocalDate getDateDepart() {
    return dateDepart;
}
```

Getter des dates

```java
public void setDateArrivee(LocalDate dateArrivee) throws ExceptionArrivee{
    if( (dateDepart != null) && (dateArrivee.isBefore(dateDepart) ) )
    {
        throw new ExceptionArrivee();
    }
    this.dateArrivee = dateArrivee;
    try {
        gestionPersonnel.update(this);
    }
    catch (SauvegardeImpossible e)
    {
        e.printStackTrace();
    }



public void setDateDepart(LocalDate dateDepart) throws ExceptionDepart {
    if( (dateArrivee != null) && (dateDepart.isAfter(dateArrivee) ) )
    {
        throw new ExceptionDepart();
    }
    this.dateDepart = dateDepart;
    try {
        gestionPersonnel.update(this);
    }
    catch (SauvegardeImpossible e)
    {
        e.printStackTrace();
    }
}
```

Setter des dates
+ try/catch

AP2-Personnel

3- Interface graphique (Maquette) :