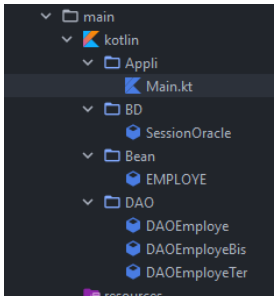
Florian Tran Grp3
Base de donnée SQL TD4

On commence par créer le projet, on implémente tout les fichiers demandé(pour la SessionOracle,Package Appli et Bean)
Dans le package appli on créer le constructeur Employe, avec toutes les méthodes GET, SET et toString :

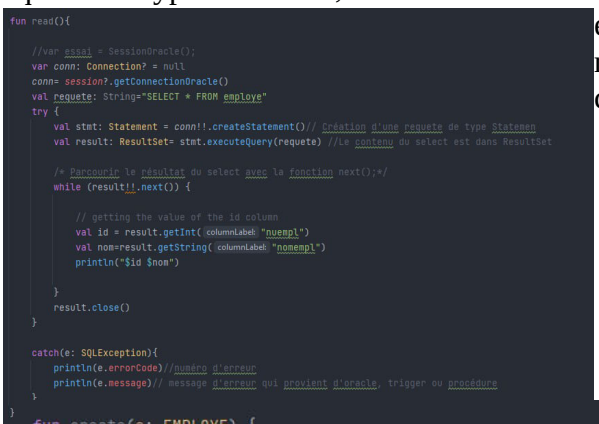
```
package Bean;

class EMPLOYE(nuempl :Int, nomempl: String,hebdo: Int,affect: Int, salaire: Int) {
    var nuempl: Int
    var nomempl: String
    var hebdo: Int
    var affect: Int
    var salaire: Int


    init {
        this.nuempl = nuempl
        this.nomempl = nomempl
        this.hebdo = hebdo
        this.affect = affect
        this.salaire = salaire
    }

    fun get_nuempl(): Int {
        return this.nuempl
    }
```

```
    fun set_salaire(a: Int) {
        this.salaire = a
    }

//====================ToString=======================//

    override fun toString(): String {
        return "nuempl: "+ this.nomempl + "nomempl: " +
                this.nuempl + "hebdo: " + this.hebdo +
                "affect: " + this.affect + "salaire: " +
                this.salaire
    }
}
```

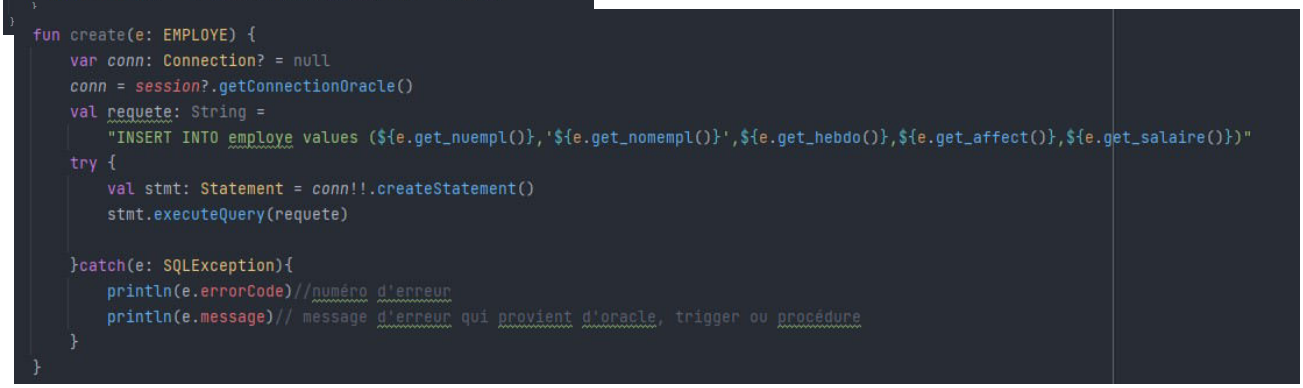
Arborescence :



Ensuite on va créer nos 3 différents package DAO (employe,employeBis,employeTer)
On commence par le DAOEmploye :
On créer nos 4 méthodes Read,Update,Create,Delete : Tout d'abords on prends en paramètre E un Employe qu'on à créer précédemment, on se connecte à la session Oracle ensuite on créer nos requêtes de type Statement, et ensuite on l'exécute alors on parcours le résultat et on get les valeurs et on fini par notre catch avec le numéro d'erreur. On répète cela avec les différentes méthodes en changeant les requêtes sql.



```
fun read(){

    //var essai = SessionOracle();
    var conn: Connection? = null
    conn= session?.getConnectionOracle()
    val requete: String="SELECT * FROM employe"
    try {
        val stmt: Statement = conn!!.createStatement()// Création d'une requete de type Statemen
        val result: ResultSet= stmt.executeQuery(requete) //Le contenu du select est dans ResultSet

        /* Parcourir le résultat du select avec la fonction next();*/
        while (result!!.next()) {

            // getting the value of the id column
            val id = result.getInt( columnLabel: "nuempl")
            val nom=result.getString( columnLabel: "nomempl")
            println("$id $nom")

        }
        result.close()
    }

    catch(e: SQLException){
        println(e.errorCode)//numéro d'erreur
        println(e.message)// message d'erreur qui provient d'oracle, trigger ou procédure
    }
}
```



```
fun create(e: EMPLOYE) {
    var conn: Connection? = null
    conn = session?.getConnectionOracle()
    val requete: String =
        "INSERT INTO employe values (${e.get_nuempl()},'${e.get_nomempl()}',${e.get_hebdo()},${e.get_affect()},${e.get_salaire()})"
    try {
        val stmt: Statement = conn!!.createStatement()
        stmt.executeQuery(requete)

    }catch(e: SQLException){
        println(e.errorCode)//numéro d'erreur
        println(e.message)// message d'erreur qui provient d'oracle, trigger ou procédure
    }
}
```

```
fun update(e: EMPLOYE) {
    var conn: Connection? = null
    conn = session?.getConnectionOracle()
    val requete: String =
        "UPDATE employe SET nomempl = '${e.get_nomempl()}', hebdo = ${e.get_hebdo()}, affect = ${e.get_affect()}, salaire = ${e.get_salaire()} where nuempl = ${e.get_nuempl()} "
    try {
        val stmt: Statement = conn!!.createStatement()
        stmt.executeQuery(requete)

    }catch(e: SQLException){
        println(e.errorCode)//numéro d'erreur
        println(e.message)// message d'erreur qui provient d'oracle, trigger ou procédure
    }
}
```

```
fun delete(e: EMPLOYE) {
    var conn: Connection? = null
    conn = session?.getConnectionOracle()
    val requete: String =
        "DELETE FROM employe WHERE nuempl = ${e.get_nuempl()}"
    try {
        val stmt: Statement = conn!!.createStatement()
        stmt.executeQuery(requete)

    }catch(e: SQLException){
        println(e.errorCode)//numéro d'erreur
        println(e.message)// message d'erreur qui provient d'oracle, trigger ou procédure
    }

}
```

Pour tester on se dirige vers le main :
On se connecte à notre base de donnée avec en paramètre nos identifiants, on créer une variable avec notre DAOEmploye avec en paramètre notre session. On créer un employé grâce au constructeur
et ensuite on test nos différentes méthodes.

```
fun main(args: Array<String>) {
    var essai = SessionOracle( username: "i2c08a",  password: "i2c08a");

    essai.getConnectionOracle()

    var dd = DAOEmploye(essai)
    var e = EMPLOYE( nuempl: 88,  nomempl: "matthis",  hebdo: 36,  affect: 5,  salaire: 1200)

    dd.read()
    dd.create(e)
    dd.read()
    e.set_nomempl("mathis")
    dd.update(e)
    dd.read()
    dd.delete(e)
    dd.read()
}
```

Ex de résultat :

```
28 marie
30 edith
62 marcelle
68 casimir
71 gedeon
19 bruno
65 simone
67 bertrand
73 germaine
88 mathis
69 matthis
20 marcel
23 claude
37 michele
```

Pour le DAOEmplyeBis on remplace les Statement par des PreparedStatetment et CreateStatetment dans les méthodes ensuite les paramètres des requêtes sql sont

remplacé par de ‘?’ ex : Insert into employe values(?, ?, ?, ?,?) et chaque paramètres sont initialisés avec des méthodes de type :
smtp.setInt(position du paramètre,valeur) . On oublie pas d’enlever la variable requête dans l’execute.

```
fun create(e: EMPLOYE) {
    var conn: Connection? = null
    conn = session?.getConnectionOracle()
    val requete: String =
        "INSERT INTO employe values(?,?,?,?,?)"
    try {
        val stmt: PreparedStatement = conn!!.prepareStatement(requete)
        stmt.setInt( parameterIndex: 1,e.get_nuempl())
        stmt.setString( parameterIndex: 2,e.get_nomempl())
        stmt.setInt( parameterIndex: 3,e.get_hebdo())
        stmt.setInt( parameterIndex: 4,e.get_affect())
        stmt.setInt( parameterIndex: 5,e.get_salaire())
        stmt.executeUpdate()

    }catch(e: SQLException){
        println(e.errorCode)//numéro d'erreur
        println(e.message)// message d'erreur qui provient d'oracle, trigger ou procédure
    }
}
```

```
fun delete(e: EMPLOYE) {
    var conn: Connection? = null
    conn = session?.getConnectionOracle()
    val requete: String =
        "DELETE FROM employe WHERE nuempl = ?"
    try {
        val stmt: PreparedStatement = conn!!.prepareStatement(requete)
        stmt.setInt( parameterIndex: 1,e.get_nuempl())
        stmt.executeUpdate()

    }catch(e: SQLException){
        println(e.errorCode)//numéro d'erreur
        println(e.message)// message d'erreur qui provient d'oracle, trigger ou procédure
    }

}
```

```
fun update(e: EMPLOYE) {
    var conn: Connection? = null
    conn = session?.getConnectionOracle()
    val requete: String =
        "UPDATE employe SET nomempl=?,hebdo=?,affect=?,salaire=?,WHERE=nuempl=?"
    try {
        val stmt: PreparedStatement = conn!!.prepareStatement(requete)
        stmt.setString( parameterIndex: 1,e.get_nomempl())
        stmt.setInt( parameterIndex: 2,e.get_hebdo())
        stmt.setInt( parameterIndex: 3,e.get_affect())
        stmt.setInt( parameterIndex: 4,e.get_salaire())
        stmt.setInt( parameterIndex: 5,e.get_nuempl())
        stmt.executeQuery()

    }catch(e: SQLException){
        println(e.errorCode)//numéro d'erreur
        println(e.message)// message d'erreur qui provient d'oracle, trigger ou procédure
    }
}
```

Pour tester on reprends le même main sauf qu’on change la variable de DAOEmploye à DAOEmployeBis.

Pour DAOEmployeTer qui fait appel aux procédures stockés ,on va utiliser les calableStatement à la place de preparedStatement et prepareCall à la place de prepare statement. Les paramètre de la requêtes sont identiques. On change juste la requête qui va être de la forme suivante :
Nom_Package.Nom_Procedure(?,?,?….)
Dans notre cas on appelle MAJ.CREER_EMPLOYE(?, ?, ?, ?,?) et LECTURE.LISTE_EMPLOYE(?)

```
fun create(e: EMPLOYE) {
    var conn: Connection? = null
    conn = session?.getConnectionOracle()
    val requete: String =
        "CALL MAJ.CREER_EMPLOYE(?,?,?,?,?)"
    try {
        val stmt: CallableStatement = conn!!.prepareCall(requete)
        stmt.setInt( parameterIndex: 1,e.get_nuempl())
        stmt.setString( parameterIndex: 2,e.get_nomempl())
        stmt.setInt( parameterIndex: 3,e.get_hebdo())
        stmt.setInt( parameterIndex: 4,e.get_affect())
        stmt.setInt( parameterIndex: 5,e.get_salaire())
        stmt.executeUpdate()

    }catch(e: SQLException){
        println(e.errorCode)//numéro d'erreur
        println(e.message)// message d'erreur qui provient d'oracle, trigger ou procédure
    }
}
```

```
fun read(){
    //var essai = SessionOracle();
    var conn: Connection? = null
    conn= session?.getConnectionOracle()
    val requete: String="call lecture.liste_employes(?)"
    try {
        val stmt : CallableStatement= conn!!.prepareCall( sql: "call lecture.liste_employes(?)");
        stmt.registerOutParameter( parameterIndex: 1, OracleTypes.CURSOR);
        stmt.execute();
        val result: ResultSet= stmt.getObject( parameterIndex: 1) as ResultSet;
        /* Parcourir le résultat du select avec la fonction next();*/
        while (result!!.next()) {

            // getting the value of the id column
            val id = result.getInt( columnLabel: "nuempl")
            val nom=result.getString( columnLabel: "nomempl")
            println("$id $nom")
        }
        result.close()
    }
    catch(e: SQLException){
        println(e.errorCode)//numéro d'erreur
        println(e.message)// message d'erreur qui provient d'oracle, trigger ou procédure
    }
}
```

```
20 marcel
23 claude
37 michele
39 leon
41 jules
48 jean
51 paul
17 sophie
52 pierre
57 anne
14 alexandre
28 marie
30 edith
62 marcelle
68 casimir
71 gedeon
19 bruno
65 simone
67 bertrand
73 germaine
95 guillaume
69 matthis

Process finished with ex
```

Ensuite pour tester c'est exactement la même chose en remplacent DAOEmployeBis par DAOEmployeTer.

```
var essai = SessionOracle( username: "i2c08a", password: "i2c08a");

essai.getConnectionOracle()

var dd = DAOEmployeTer(essai)
var e = EMPLOYE( nuempl: 88, nomempl: "matthis", hebdo: 36, affect: 5, salaire: 1200)
var a = EMPLOYE( nuempl: 95, nomempl: "guillaume", hebdo: 36, affect: 5, salaire: 1200)
```