ECOLE MAROCAINE DES SCIENCES DE L'INGENIEUR
Membre de
HONORIS UNITED UNIVERSITIES

COMPTE RENDU JEE DE LA PARTIE "COUPLAGE-COUPLAGE FAIBLE/FORT"

Préparé par : CHAARAOUI Oumaima 4IIR/G2

Année universitaire : 2022/2023

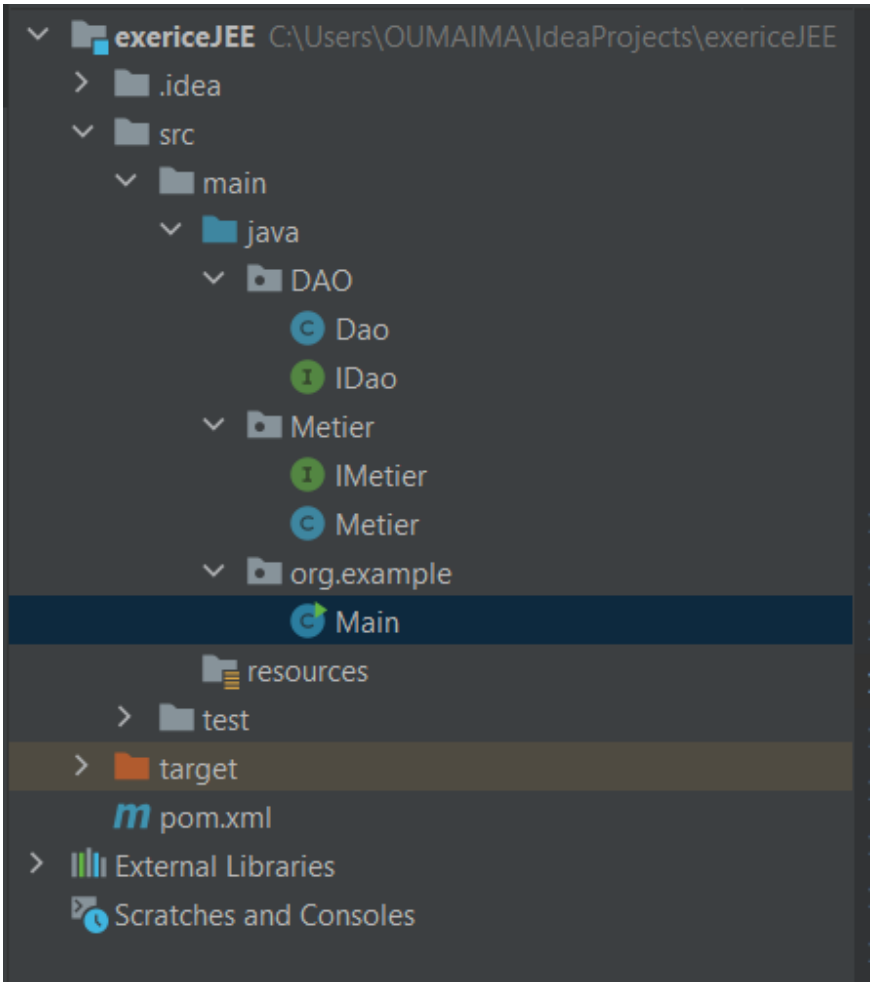
OBJECTIF DU TP :

“L'objectif de ce TP est de savoir le principe et le rôle du couplage faible pour avoir une application ouverte à l'extension et fermée à la modification. Pour cela on aura besoin d'utiliser des interfaces. Donc on notera bien lors de ce TP comment faire l'injection des dépendances avec et sans framework (spring).”

INJECTION DES DÉPENDANCES AVEC LE FRAMEWORK SPRING : EN UTILISANT LES ANNOTATIONS

L'injection des dépendances est souvent la base de tout programme moderne.
L'idée en résumé est de porter la responsabilité de la liaison des composants du programme dans un framework afin de pouvoir facilement changer ces composants ou leur comportement. Dans notre cas, on va appliquer cette injection avec les annotations. Donc on va créer un projet Maven et ajouter les dépendances (spring core, spring context et spring beans) dans le fichier pom.xml. Ensuite il faut installer Spring and Java Tools.

Structure du projet :



Les dépendances installées :

```xml
<dependencies>
    <!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>6.0.6</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/org.springframework/spring-core -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-core</artifactId>
        <version>6.0.6</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/org.springframework/spring-beans -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-beans</artifactId>
        <version>6.0.6</version>
    </dependency>
```

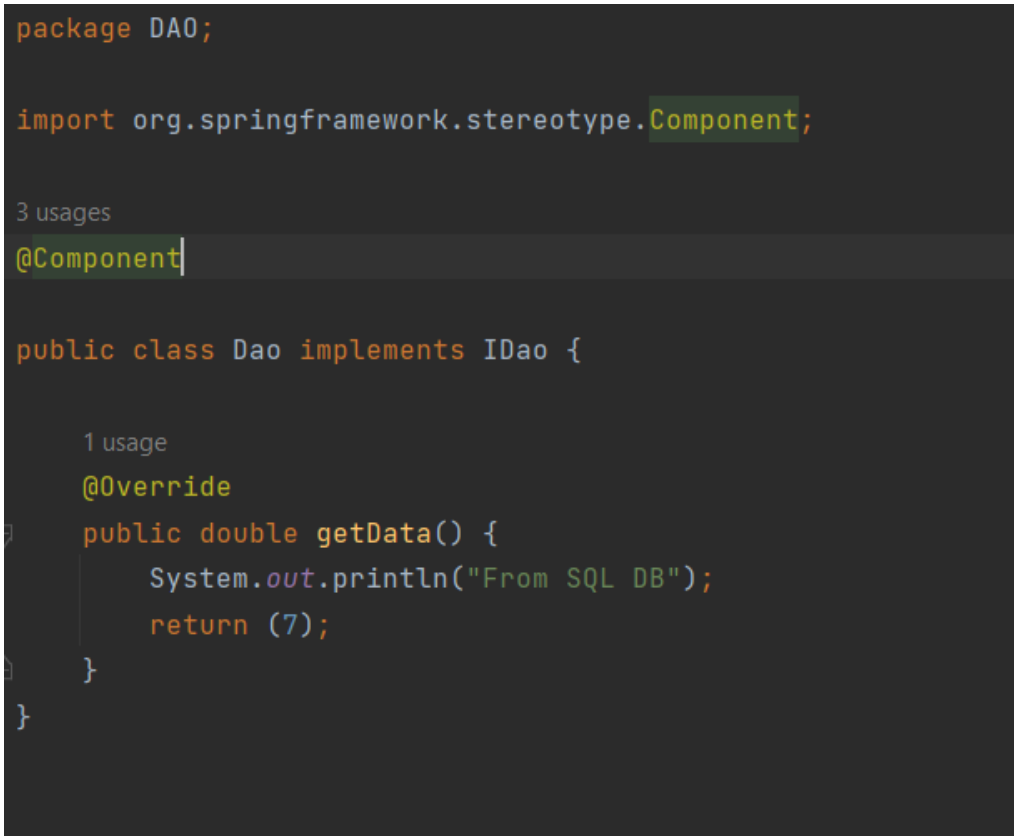
- NB : Les dépendances doivent être toutes avec la même version.

Les annotations utilisées :

- Autowired : L'annotation @Autowired permet d'activer l'injection automatique de dépendance. Cette annotation peut être placée sur un constructeur, une méthode setter ou directement sur un attribut (même privé).
- Component : @Component est un stéréotype générique puisqu'il indique simplement que cette classe doit être utilisée pour instancier un bean.

```java
package Metier;

import DAO.IDao;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

3 usages
@Component
public class Metier implements IMetier {
    2 usages
    @Autowired
    IDao dao;

    1 usage
    @Override
    public double calcul()
    {
        double data = dao.getData();
        return data*10;
    }

    1 usage
    public void setDao(IDao dao) { this.dao=dao; }
}
```

- Spring est capable d'injecter un bean de type IDao dans l'attribut dao. La dépendance n'est marquée que par la présence de cet attribut.



```java
package DAO;

import org.springframework.stereotype.Component;

3 usages
@Component

public class Dao implements IDao {

    1 usage
    @Override
    public double getData() {
        System.out.println("From SQL DB");
        return (7);
    }
}
```

Les interfaces :

```java
package Metier;

public interface IMetier {
    1 usage   1 implementation
    double calcul();
}
```

```java
package DAO;

public interface IDao {
    1 usage   1 implementation
    double getData();
}
```

La couche présentation :

```java
public class Presentation {

    public static void main(String[] args) {

        ApplicationContext context = new AnnotationConfigApplicationContext( ...basePackages: "dao", "metier");
        IMetier metier = context.getBean(IMetier.class);

        System.out.println("The result is :" + metier.calcul());
    }
}
```

- C'est la partie frontend qui est en relation avec le client. On fait appel aux objets ApplicationContext et AnnotationConfigApplicationContext.
- Le Spring Framework s'est enrichi de nouvelles classes implémentant l'interface **ApplicationContext** afin d'offrir des méthodes alternatives à la création d'un contexte d'application. La méthode la plus souvent recommandée consiste à ajouter des annotations sur les classes de notre application. Pour cela, nous devons utiliser une instance de la classe **AnnotationConfigApplicationContext**.
- L'utilisation du getBean consiste à charger un Bean qui implémente l'interface IMetier

L'exécution :

```
C:\Users\OUMAIMA\.jdks\openjdk-18.0.1.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ I
From SQL DB
The result is :70.0

Process finished with exit code 0
```

INJECTION DES DÉPENDANCES SANS LE FRAMEWORK SPRING : EN UTILISANT L'INSTANCIATION STATIQUE

Dans ce cas, on va créer un projet Java avec trois packages toujours; metier, dao et présentation.
L'instanciation statique consiste à l'utilisation des constructeurs, des setters etc.
Les interfaces sont les mêmes que dans le cas des annotations.

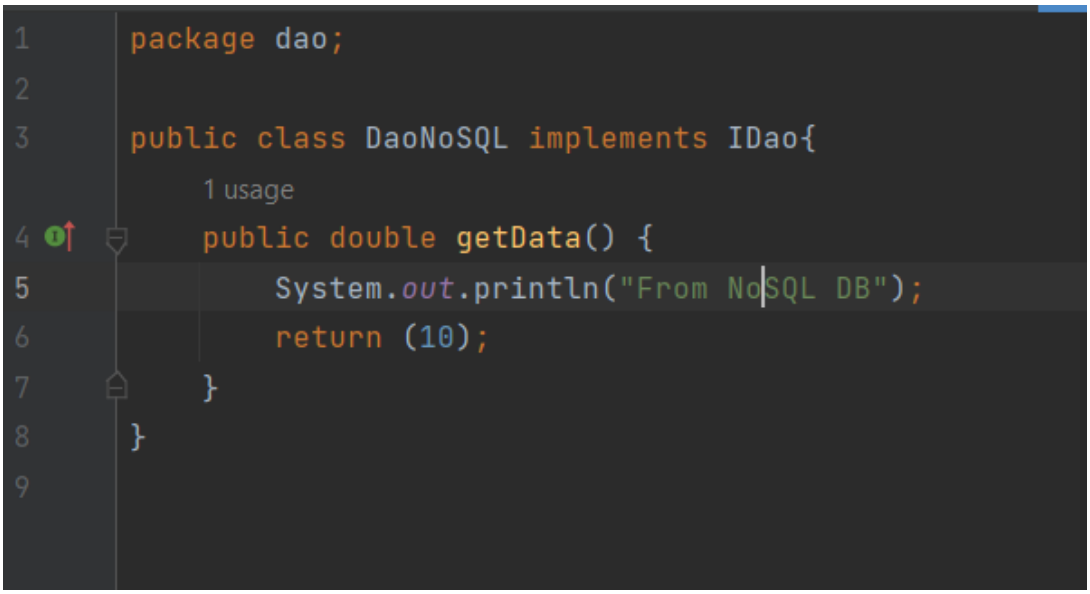
- La méthode 'calcul()' permet de récupérer les données depuis la couche DAO pour faire le calcul.
- On ajoute la classe DaoNoSQL dans la couche précédente pour éviter la modification.

La classe Metier :

```java
package Metier;

import dao.IDao;

3 usages
public class Metier implements IMetier{
    2 usages
    IDao dao;

    1 usage
    public double calcul()
    {
        double data = dao.getData();
        return data=10;
    }

    1 usage
    public void setDao(IDao dao) { this.dao=dao; }
```

- Dans le cas de l'injection sans framework, on va utiliser le setter pour associer deux objets.

La classe DaoNoSQL :



```java
package dao;

public class DaoNoSQL implements IDao{
    1 usage
    public double getData() {
        System.out.println("From NoSQL DB");
        return (10);
    }
}
```

- Elle implémente getData() de l'interface IDao et permet de se connecter à la base de données pour récupérer les données.

L'exécution :

```
C:\Users\OUMAIMA\.jdks\openjdk-18.0.1.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\
From NoSQL DB
result is 10.0

Process finished with exit code 0
```

CONCLUSION :

“

On peut différencier d'après ce TP facilement entre le couplage fort et le couplage faible; **le couplage fort** est le fait d'associer une classe à une autre ce qui engendre que cette classe ne peut pas fonctionner sans l'autre. **Le couplage faible** est simplement l'association aux interfaces au lieu à des classes.
Aussi d'après les injections sans framework (le cas de l'instanciation statique), on s'appuie sur l'utilisation des constructeurs, des setters...En revanche, l'injection avec **Spring** (les annotations), c'est à travers les fichiers xml qu'on communique avec le framework en spécifiant la partie technique souhaitée.

”